



UNIVERSITY OF IOANNINA
SCHOOL OF SCIENCE
DEPARTMENT OF MATHEMATICS



Athanasios L. Konstantinidis

ALGORITHMS AND COMPLEXITY OF GRAPH MODIFICATION PROBLEMS

PhD Dissertation

Ioannina, 2021

”This research is co-financed by Greece and the European Union (European Social Fund-ESF) through the Operational Programme «Human Resources Development, Education and Lifelong Learning» in the context of the project “Strengthening Human Resources Research Potential via Doctorate Research” (MIS-5000432), implemented by the State Scholarships Foundation (IKY)”



Operational Programme
Human Resources Development,
Education and Lifelong Learning
Co-financed by Greece and the European Union





ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΜΑΘΗΜΑΤΙΚΩΝ



Αθανάσιος Λ. Κωνσταντινίδης

Αλγόριθμοι και Πολυπλοκότητα σε Προβλήματα Επεξεργασίας Γραφημάτων

Διδακτορική Διατριβή

Ιωάννινα, 2021

«Το έργο συγχρηματοδοτείται από την Ελλάδα και την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) μέσω του Επιχειρησιακού Προγράμματος «Ανάπτυξη Ανθρώπινου Δυναμικού, Εκπαίδευση και Διά Βίου Μάθηση», στο πλαίσιο της Πράξης «Ενίσχυση του ανθρώπινου ερευνητικού δυναμικού μέσω της υλοποίησης διδακτορικής έρευνας» (MIS-5000432), που υλοποιεί το Ίδρυμα Κρατικών Υποτροφιών (ΙΚΥ)»



Επιχειρησιακό Πρόγραμμα
Ανάπτυξη Ανθρώπινου Δυναμικού,
Εκπαίδευση και Διά Βίου Μάθηση
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης



Η παρούσα Διδακτορική Διατριβή εκπονήθηκε στο πλαίσιο των σπουδών για την απόκτηση του Διδακτορικού Διπλώματος που απονέμει το Τμήμα Μαθηματικών της Σχολής Θετικών Επιστημών του Πανεπιστημίου Ιωαννίνων.

Εγκρίθηκε την 31/03/2021 από την εξεταστική επιτροπή:

Χάρης Παπαδόπουλος	Αναπληρωτής Καθηγητής (Επιβλέπων) Τμήμα Μαθηματικών Πανεπιστήμιο Ιωαννίνων
Σταύρος Δ. Νικολόπουλος	Καθηγητής (Τριμελή Συμβουλευτική Επιτροπή) Τμήμα Μηχανικών Η/Υ και Πληροφορικής Πανεπιστήμιο Ιωαννίνων
Λεωνίδας Παληός	Καθηγητής (Τριμελή Συμβουλευτική Επιτροπή) Τμήμα Μηχανικών Η/Υ και Πληροφορικής Πανεπιστήμιο Ιωαννίνων
Λουκάς Γεωργιάδης	Αναπληρωτής Καθηγητής Τμήμα Μηχανικών Η/Υ και Πληροφορικής Πανεπιστήμιο Ιωαννίνων
Χρήστος Ζαρολιάγκης	Καθηγητής Τμήμα Μηχανικών Η/Υ και Πληροφορικής Πανεπιστήμιο Πατρών
Δημήτριος Θηλυκός	Καθηγητής Τμήμα Μαθηματικών Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
Χρήστος Νομικός	Επίκουρος Καθηγητής Τμήμα Μηχανικών Η/Υ και Πληροφορικής Πανεπιστήμιο Ιωαννίνων

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ

”Δηλώνω υπεύθυνα ότι η παρούσα διατριβή εκπονήθηκε κάτω από τους διεθνείς ηθικούς και ακαδημαϊκούς κανόνες δεοντολογίας και προστασίας της πνευματικής ιδιοκτησίας. Σύμφωνα με τους κανόνες αυτούς, δεν έχω προβεί σε ιδιοποίηση ξένου επιστημονικού έργου και έχω πλήρως αναφέρει τις πηγές που χρησιμοποίησα στην εργασία αυτή.”

Αθανάσιος Α. Κωνσταντινίδης

ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor Charis Papadopoulos for his continuous support all these years. Without his encouragement and scientific guidance it wouldn't be possible to complete this dissertation. I should mention that he was always willing to listen to every thought and explain any question patiently. All these fruitful conversations and constructive meetings were a great lesson for me. I am glad you are my teacher. Also, I thank you for the opportunities that you gave me. One of them was when Prof. Pinar Heggernes invited us for the collaboration at the Department of Informatics, University of Bergen, Norway.

I would like to thank Pinar Heggernes for the invitation and for her hospitality in Bergen; it was an amazing and constructive week. Additionally, I want to thank the other two co-authors from the same Department, Petr Golovach and Paloma T. Lima. I really enjoyed working with you.

Moreover, I would like to thank Prof. Stavros D. Nikolopoulos (co-author and member of advisor committee) for the cooperation and valuable advices. It was a great experience working with you. I also want to thank Prof. Leonidas Palios (member of advisor committee) for the useful comments and advices.

Last but not least, a special thanks to my family for their support throughout my studies.

ABSTRACT

Graph modification problems play important role in both structural and algorithmic graph theory. These problems have been studied for decades and find a large number of practical applications in several different fields in real world. In this thesis, we study a famous edge deletion problem, known under the terms **CLUSTER DELETION** or P_3 -FREE EDGE DELETION, and we consider an edge labeling scheme that characterizes social networks in terms of an edge deletion problem, known as **MAXSTC**. Both problems are known to be NP-hard. We provide the first computational results of **MAXSTC** and we determine the computational complexity of **CLUSTER DELETION** on particular graph classes. Moreover, we generalize the **MAXSTC** problem and propose a relaxation of the classical F -FREE EDGE DELETION problem that we call **STRONG F -CLOSURE**. We study **STRONG F -CLOSURE** from the parameterized perspective and provide computational results with various natural parameterizations.

In social networks the **STRONG TRIADIC CLOSURE** is an assignment of the edges with strong or weak labels such that any two vertices that have a common neighbor with a strong edge are adjacent. The problem of maximizing the number of strong edges that satisfy the strong triadic closure is known as **MAXSTC** and it was recently shown to be NP-complete for general graphs. Here we initiate the study of graph classes for which **MAXSTC** is solvable in polynomial time or NP-complete. On the positive side, we show that the problem admits a polynomial-time algorithm for the following incomparable classes of graphs: proper interval graphs, cographs, graphs with maximum degree 3, and graphs with bounded treewidth. To complement our result, we show that the problem remains NP-complete on the following graphs: split graphs, and consequently also on chordal graphs, graphs with maximum degree at most 4, planar graphs, and $(3K_1, 2K_2)$ -free graphs. Thus, we contribute to define the first border between graph classes on which the problem is polynomially solvable and on which it remains NP-complete.

Inspired by the close relative of **MAXSTC**, we consider the **CLUSTER DELETION** problem. The goal is to remove the minimum number of edges of a given graph, such that every connected component of the resulting graph constitutes a clique. It is known that the decision version of **CLUSTER DELETION** is NP-complete on $(P_5$ -free) chordal graphs, whereas **CLUSTER DELETION** is solved in polynomial time on split graphs. However, the existence of a polynomial-time algorithm of **CLUSTER DELE-**

TION on interval graphs, a proper subclass of chordal graphs, remained a well-known open problem. Our main contribution is that we settle this problem in the affirmative, by providing a polynomial-time algorithm for CLUSTER DELETION on interval graphs. Moreover, despite the simple formulation of a polynomial-time algorithm on split graphs, we show that CLUSTER DELETION remains NP-complete on a natural and slight generalization of split graphs that constitutes a proper subclass of P_5 -free chordal graphs. To complement our results, we provide faster and simpler polynomial-time algorithms for CLUSTER DELETION on subclasses of such a generalization of split graphs.

Furthermore we introduce and initiate the parameterized study of the STRONG F -CLOSURE problem, for a fixed graph F , which is a natural generalization of MAXSTC. The goal is to select a maximum number of edges of the input graph G , and mark them as *strong edges*, in the following way: whenever a subset of the strong edges forms a subgraph isomorphic to F , then the corresponding induced subgraph of G is *not* isomorphic to F . Hence, the subgraph of G defined by the strong edges is not necessarily F -free, but whenever it contains a copy of F , there are additional edges in G to forbid that strong copy of F in G . Therefore STRONG F -CLOSURE is a generalization of MAXSTC, whereas it is a relaxation of F -FREE EDGE DELETION. We study STRONG F -CLOSURE from a parameterized perspective with various natural parameterizations. Our main focus is on the number k of strong edges as the parameter. We show that the problem is FPT with this parameterization for every fixed graph F , whereas it does not admit a polynomial kernel even when $F = P_3$. In fact, this latter case is equivalent to the MAXSTC problem, which motivates us to study this problem on input graphs belonging to well known graph classes. We show that MAXSTC does not admit a polynomial kernel even when the input graph is a split graph, whereas it admits a polynomial kernel when the input graph is planar, and even d -degenerate. Furthermore, on graphs of maximum degree at most 4, we show that MAXSTC is FPT with the above guarantee parameterization $k - \mu(G)$, where $\mu(G)$ is the maximum matching size of G . We conclude with some results on the parameterization of STRONG F -CLOSURE by the number of weak edges of G .

Περίληψη

Τα προβλήματα επεξεργασίας γραφημάτων παίζουν σημαντικό ρόλο τόσο στην δομική όσο και στην αλγοριθμική θεωρία γραφημάτων. Τα προβλήματα αυτά έχουν μελετηθεί για δεκαετίες και βρίσκουν πρακτικές εφαρμογές σε σημαντικές περιοχές έρευνας. Σε αυτήν την διατριβή μελετάμε ένα κλασικό πρόβλημα επεξεργασίας ακμών, γνωστό ως CLUSTER DELETION ή P_3 -FREE EDGE DELETION, και εξετάζουμε έναν κανόνα επιγραφής ακμών ο οποίος χαρακτηρίζει τα κοινωνικά δίκτυα, ως ένα πρόβλημα διαγραφής ακμών, γνωστό ως MAXSTC. Τα δύο αυτά προβλήματα είναι γνωστό ότι είναι NP-δύσκολα. Παρέχουμε τα πρώτα υπολογιστικά αποτελέσματα για το MAXSTC και καθορίζουμε την υπολογιστική πολυπλοκότητα για το CLUSTER DELETION σε κλάσεις γραφημάτων. Επιπλέον, γενικεύουμε το πρόβλημα MAXSTC προτείνοντας μία “χαλάρωση” του κλασικού προβλήματος επεξεργασίας ακμών F -FREE EDGE DELETION, το οποίο καλούμε STRONG F -CLOSURE. Μελετάμε το πρόβλημα STRONG F -CLOSURE από την σκοπιά της παραμετρικής πολυπλοκότητας και παρέχουμε τα πρώτα υπολογιστικά αποτελέσματα υπό διαφορετικούς παραμέτρους.

Στα κοινωνικά δίκτυα η Ισχυρή Τριαδική Κλειστότητα είναι μία ανάθεση επιγραφών στις ακμές ως ισχυρές ή ασθενείς ακμές, έτσι ώστε για οποιεσδήποτε δύο κορυφές οι οποίες έχουν ένα κοινό γείτονα με ισχυρή ακμή είναι γειτονικές μεταξύ τους (είτε με ισχυρή είτε με ασθενή ακμή). Το πρόβλημα μεγιστοποίησης του αριθμού των ισχυρών ακμών οι οποίες να ικανοποιούν την ισχυρή τριαδική κλειστότητα είναι γνωστό ως MAXSTC και έχειδειχθεί πρόσφατα ότι είναι NP-πλήρες σε γενικά γραφήματα. Σε αυτή διατριβή επικεντρωνόμαστε σε κλάσεις γραφημάτων με την συστηματική μελέτη για τις οποίες το πρόβλημα MAXSTC είναι επιλύσιμο σε πολυωνυμικό χρόνο ή παραμένει NP-πλήρες. Από την θετική σκοπιά των υπολογιστικών αποτελεσμάτων, δείχνουμε ότι το πρόβλημα επιδέχεται πολυωνυμικού χρόνου αλγόριθμο στις ακόλουθες μη-συγκρίσιμες κλάσεις γραφημάτων: proper interval γραφήματα, cographs, γραφήματα με μέγιστο βαθμό 3, και γραφήματα με φραγμένο δεντροπλάτος. Από την άλλη πλευρά, δείχνουμε ότι το πρόβλημα παραμένει NP-πλήρες ακόμα και όταν το γράφημα εισόδου ανήκει σε μια από τις ακόλουθες κλάσεις γραφημάτων: split γραφήματα (επομένως, και chordal γραφήματα), γραφήματα με μέγιστο βαθμό το πολύ 4, επίπεδα γραφήματα, και $(3K_1, 2K_2)$ -free γραφήματα. Συνεπώς, συμβάλλουμε να οριστούν τα πρώτα υπολογιστικά όρια μεταξύ κλάσεων γραφημάτων για τις οποίες το πρόβλημα είναι πολυωνυμικά επιλύσιμο και για τις οποίες παραμένει NP-πλήρες.

Εμπνευσμένοι από την στενή σχέση που έχει με το πρόβλημα MAXSTC, θεωρούμε το πρόβλημα CLUSTER DELETION. Σκοπός είναι να αφαιρέσουμε το ελάχιστο πλήθος ακμών από ένα δοθέν γράφημα, έτσι ώστε κάθε συνεκτική συνιστώσα του εναπομείναντος γραφήματος να σχηματίζει κλίκα. Είναι γνωστό ότι το CLUSTER DELETION ως πρόβλημα απόφασης είναι NP-πλήρες στα $(P_5\text{-free})$ chordal γραφήματα, ενώ το CLUSTER DELETION λύνεται σε πολυωνυμικό χρόνο στα split γραφήματα. Ωστόσο, η ύπαρξη ενός πολυωνυμικού χρόνου αλγορίθμου για το CLUSTER DELETION στα interval γραφήματα, μία γνήσια υποκλάση των chordal γραφημάτων, παρέμενε ανοιχτό πρόβλημα. Η κύρια συνεισφορά στο πρόβλημα αυτό είναι ότι απαντάμε θετικά, δίνοντας έναν πολυωνυμικό αλγόριθμο για το πρόβλημα CLUSTER DELETION στα interval γραφήματα. Επιπλέον, αν και ο πολυωνυμικός αλγόριθμος για τα split γραφήματα είναι σχετικά απλός, δείχνουμε ότι το πρόβλημα CLUSTER DELETION παραμένει NP-πλήρες σε μία φυσική και μικρή γενίκευση των split γραφημάτων που αποτελεί ταυτόχρονα μία γνήσια υποκλάση των $(P_5\text{-free})$ chordal γραφημάτων. Παρά την δυσκολία σε αυτή τη γενίκευση των split γραφημάτων, παρέχουμε γρηγορότερους και απλούστερους πολυωνυμικούς αλγορίθμους για το πρόβλημα CLUSTER DELETION σε υποκλάσεις της συγκεκριμένης γενίκευσης.

Επιπροσθέτως, εισάγουμε και μελετάμε την παραμετρική πολυπλοκότητα του προβλήματος STRONG F -CLOSURE, για σταθερό γράφημα F , που αποτελεί μία γενίκευση του προβλήματος MAXSTC. Ο στόχος είναι να επιλέξουμε ένα μέγιστο πλήθος από ακμές του δοθέντος γραφήματος G και να τις χαρακτηρίσουμε ως ισχυρές ακμές, με τον ακόλουθο τρόπο: όταν ένα υποσύνολο από ισχυρές ακμές σχηματίζει ένα υπογράφημα ισόμορφο με το F , τότε το αντίστοιχο επαγόμενο υπογράφημα του G δεν είναι ισόμορφο με το F . Συνεπώς, το υπογράφημα του G που ορίζεται από τις ισχυρές ακμές δεν είναι απαραίτητα F -free, αλλά όταν περιέχει ένα αντίγραφο του F , υπάρχουν επιπλέον ακμές στο G ώστε να απαγορεύουν αυτό το ισχυρό αντίγραφο του F στο G . Επομένως, το πρόβλημα STRONG F -CLOSURE αποτελεί μία γενίκευση του προβλήματος MAXSTC όταν $F = P_3$, ενώ αποτελεί ένα είδος “χαλάρωσης” του προβλήματος F -FREE EDGE DELETION. Μελετάμε την παραμετρική πολυπλοκότητα του προβλήματος υπό διάφορες φυσικές παραμέτρους. Επικεντρωνόμαστε κυρίως στο πλήθος k των ισχυρών ακμών ως παράμετρο. Δείχνουμε ότι το πρόβλημα είναι FPT με αυτή την παραμετροποίηση για κάθε σταθερό γράφημα F , ενώ δεν επιδέχεται πολυωνυμικό πυρήνα ακόμα και όταν το $F = P_3$. Αυτή η τελευταία περίπτωση είναι ισοδύναμη με το MAXSTC πρόβλημα, το οποίο μας παρακινεί να μελετήσουμε το πρόβλημα αυτό σε γνωστές κλάσεις γραφημάτων. Δείχνουμε ότι το MAXSTC δεν επιδέχεται πολυωνυμικό πυρήνα ακόμα και όταν το γράφημα που μας δίνεται είναι split, ενώ επιδέχεται ένα πολυωνυμικό πυρήνα αν το γράφημα είναι επίπεδο ή d -degenerate γράφημα. Επιπροσθέτως, στα γραφήματα με μέγιστο βαθμό το πολύ 4, δείχνουμε ότι το MAXSTC παραμένει FPT ακόμα και με τη παράμετρο $k - \mu(G)$, όπου $\mu(G)$ είναι το μέγεθος του

μέγιστου ταιριάσματος του G . Κλείνουμε με ορισμένα υπολογιστικά αποτελέσματα της παραμετροποίησης του προβλήματος **STRONG F -CLOSURE** υπό το πλήθος των ασθενών ακμών, όπου δείχνουμε ότι το πρόβλημα είναι FPT και επιδέχεται πολυωνυμικό πυρήνα με τη συγκεκριμένη παράμετρο.

CONTENTS

Abstract	i
Περίληψη	iii
1 Introduction	1
1.1 Graph modification problems	1
1.2 Previously known results	3
1.3 Our contribution	5
1.4 Road map	9
2 Definitions and Notations	11
2.1 Basic Concepts on Graph Theory	11
2.2 Graph Classes	15
2.3 Computational Complexity	19
2.4 Problem Definitions	24
3 MaxSTC on Split and Proper Interval Graphs	29
3.1 Introduction	29
3.2 Preliminaries	32
3.2.1 Basic Results	33
3.2.2 The line-incompatibility graph and twin vertices	33
3.3 MaxSTC on split graphs	36
3.4 Computing MaxSTC on proper interval graphs	43
4 MaxSTC on Cographs and graphs of low maximum degree	57
4.1 Introduction	57
4.2 Preliminaries	60

4.3	Computing MaxSTC on Cographs	61
4.3.1	Maximum independent set of the cartesian product of cographs	66
4.4	Graphs of Low Maximum Degree	68
5	Cluster Deletion on Interval graphs and Starlike graphs	73
5.1	Introduction	73
5.2	Preliminaries	76
5.3	Polynomial-time algorithm on interval graphs	77
5.3.1	Splitting into partial solutions	82
5.4	Cluster Deletion on starlike graphs	89
5.4.1	Polynomial-time algorithms on subclasses of starlike graphs .	94
6	Parameterized Aspects of Strong Subgraph Closure	103
6.1	Introduction	104
6.2	Preliminaries	107
6.3	Parameterized complexity of Strong F-closure	108
6.4	Parameterized complexity of MaxSTC	114
6.5	Further Results	127
7	Conclusion	135
7.1	Summary	135
7.2	Open Problems	138
	Bibliography	141
	Short CV	151
	List of Publications	153

CHAPTER 1

INTRODUCTION

In this chapter, we discuss graph modification problems and provide previous known results on such problems. We present the main motivation of our study and we highlight our results. Moreover, we outline the content of each forthcoming chapter.

1.1 Graph modification problems

Graph modification problems is one of the fundamental computational problems in structural and algorithmic graph theory. In such problems we want to modify the input graph with the minimum number of modifications such that the resulting graph satisfies a certain property. Completion, deletion, editing, contraction can be some of the type of modifications over the vertices or the edges of the graph. These problems have been studied over the last decades. In the classical book of Garey and Johnson, 18 different types of vertex and edge modification problems are mentioned [51]. Regarding the vertex deletion problem there is a negative result which states that for any non-trivial and hereditary property (on induced subgraphs) the problem is *NP*-complete [99]. However, there is no such generalized result for edge modification problems, despite the individual *NP*-completeness results on most of the interesting graph properties which are typically formalized into graph classes. In edge modification problems the researchers focus on graph classes with good structural properties and propose approximation or parameterized algorithms [7, 15, 16, 29, 107]. Apart from the algorithmic aspects of these problems, graph modification problems find a large number of practical applications in several different fields in real world. Molecular biology, numerical linear algebra, social network, and other fields of computer science are some of the application areas [6, 54, 111].

Our main aim in this thesis is to extend the algorithmic results on such problems. More precisely, we study a famous edge deletion problem, known under the terms *CLUSTER DELETION* or *P_3 -FREE EDGE DELETION*, and we consider an edge labeling scheme that characterizes social networks in terms of an edge deletion prob-

lem, known under the term MAXSTC. We provide the first computational results of MAXSTC and we determine the classical computational complexity of CLUSTER DELETION on particular graph classes. Moreover, we generalize the MAXSTC problem and propose a relaxation of the classical F -FREE EDGE DELETION problem that we call STRONG F -CLOSURE. We study STRONG F -CLOSURE from the parameterized perspective and provide computational results with various natural parameterizations. In classical computational complexity the goal is to show that a problem either can be solved by a polynomial-time algorithm depending on the size of the input or provide an NP-hardness reduction. In contrast to classical complexity, parameterized complexity splits the size of the input into an input size and another value, called the parameter. The main goal is to show whether or not a problem is fixed parametric tractable (FPT). A problem is FPT if can be solved in time $f(k) \cdot n^{O(1)}$, where k is the parameter, f some computational function and n is the size of the problem. In other words, the goal is to design an algorithm that are efficient when the parameter is small.

The principle of Strong Triadic Closure is an important concept in social networks [41]. Understanding the strength and nature of social relationships has found an increasing usefulness in the last years due to the explosive growth of social networks (see e.g., [5]). Towards such a direction the Strong Triadic Closure principle enables us to understand the structural properties of the underlying graph: it is not possible for two individuals to have a strong relationship with a common friend and not know each other [60]. Such a principle stipulates that if two people in a social network have a "strong friend" in common, then there is an increased likelihood that they will become friends themselves at some point in the future. Satisfying the Strong Triadic Closure (STC) is to characterize the edges of the underlying graph into weak and strong such that any two vertices that have a strong neighbor in common are adjacent. Since users interact and actively engage in social networks by creating strong relationships, it is natural to consider the MAXSTC problem: maximize the number of strong edges that satisfy the Strong Triadic Closure. The problem has been introduced recently by Sintos and Tsaparas and is known to be NP-hard [118].

Clustering is a general term that is considered on a set of elements with a relation between the given elements. The task is to partition the elements into subsets, named clusters, such that the elements of the same cluster have high similarities and elements of different clusters have low similarities to each other [68, 69]. Some application of clustering can be found in computational biology [117], image processing [124], VLSI design [66]. Here, we study the CLUSTER DELETION problem: given a graph we want to delete the minimum number of edges such that the resulting graph is a union of cliques. Equivalently, the goal is characterized by no induced path P_3 on three vertices on the resulting graph and, therefore, it is also known under the term P_3 -FREE EDGE DELETION problem. It is known to be NP-hard and has been studied extensively even

on restricted inputs formalized by graph classes.

Looking closer at the above definitions, one can realize a relationship between the two problems MAXSTC and CLUSTER DELETION. In particular, the edges inside the cliques of the resulting graph for CLUSTER DELETION can be labeled as strong edges for MAXSTC, whereas the rest of the (deleted) edges can be labeled as weak edges. It is not difficult to see that such a labeling scheme satisfies the strong triadic closure. Thus, the number of edges in an optimal solution for CLUSTER DELETION consists a lower bound for the number of weak edges in an optimal solution for MAXSTC. However, the opposite is not always true. Hence, an interesting point of research direction is to study and characterize graph classes in which both problems have the same size of solution.

Motivated by the role of triadic closure in social networks and the importance of finding a maximum subgraph avoiding a fixed pattern, we introduce STRONG F -CLOSURE which is a generalization of MAXSTC and, at the same time, it consists a relaxation of F -FREE EDGE DELETION. The task in STRONG F -CLOSURE is to select a maximum number of edges of the input graph G , and mark them as *strong edges*, in the following way: whenever a subset of the strong edges forms a subgraph isomorphic to F , then the corresponding induced subgraph of G is *not* isomorphic to F . Hence, the subgraph of G defined by the strong edges is not necessarily F -free, but whenever it contains a copy of F , there are additional edges in G to forbid that strong copy of F in G . Notice that whenever $F = P_3$, the STRONG F -CLOSURE problem is equivalent to the MAXSTC problem.

1.2 Previously known results

Motivated by social network analysis, Sintos and Tsaparas introduced MAXSTC and proved that it is an NP-complete problem on general graphs [118]. Also, a constant factor approximation ratio was proposed for its dual problem of minimizing the number of weak edges [118].

On the contrary, CLUSTER DELETION has attracted several researchers. Regarding its classical complexity, it is known to be NP-hard on general graphs [116]. With respect to the maximum degree of a graph, Komusiewicz and Uhlmann[85] have shown an interesting dichotomy result: CLUSTER DELETION remains NP-hard on C_4 -free graphs with maximum degree four, whereas it can be solved in polynomial time on graphs having maximum degree at most three. For particular graph classes characterized by forbidden induced subgraphs, Gao et al.[50] showed that CLUSTER DELETION is NP-hard on $(C_5, P_5, \text{bull}, \text{fork}, \text{co-gem}, 4\text{-pan}, \text{co-}4\text{-pan})$ -free graphs or $(3K_1, 2K_2)$ -free graphs. In the positive side, they provide a polynomial-time algorithm on cographs

(P_4 -free graphs) by iteratively picking a maximum clique as a cluster.

Moreover, Bonomo et al. [10] extended the result of CLUSTER DELETION on cographs for the class of P_4 -reducible graphs by using the same approach. On another paper, Bonomo et al. [11] studied CLUSTER DELETION on subclasses of chordal graphs. They showed that the problem remains NP-hard on P_5 -free chordal graphs and, thus, on chordal graphs. Moreover, they proposed polynomial-time algorithms on proper interval graphs and split graphs. In regards with the parameterized complexity, the brute-force approach proposed by Cai [16] shows that CLUSTER DELETION is FPT parameterized by the number of deleted edges. Furthermore, CLUSTER DELETION is APX-hard [116] and there is a 2-approximation (non-polynomial) algorithm [33]. A summary of the known results on classical complexity of MAXSTC and CLUSTER DELETION are given in Table 1.1.

During our research, there was an increasing interest on generalizations of the MAXSTC problem as well as the CLUSTER DELETION problem. Grüttemeier and Komusiewicz [62, 63] studied the parameterized complexity of MAXSTC and CLUSTER DELETION. They showed that MAXSTC admits a linear kernel parameterized by the weak edges, which implies that the particular parameterization problem is FPT. Further, they showed that MAXSTC and CLUSTER DELETION are FPT whenever the parameter is the number of strong edges, but both problems do not admit polynomial kernels. Moreover, a complexity dichotomy of both problems was proposed on H -free graphs for any fixed graph H of order at most four [62, 63].

Furthermore, Sintos and Tsaparas [118] had introduced a generalization of MAXSTC with c different types of strong edges, called MULTI-STC. In MULTI-STC an induced P_3 may receive two strong labels as long as they are different. Bulteau et al. [14] studied the classical and parameterized complexity of MULTI-STC and two variations under the terms VL-MULTI-STC and EL-MULTI-STC. In the first variation every vertex has a set of possible strong labels and the labeling of the incident edges must belong to this set. In the latter variation every edge has a set of possible strong labels. They showed that for all $c \geq 1$ MULTI-STC, VL-MULTI-STC, and EL-MULTI-STC are NP-hard problems [14]. In case of $c \geq 3$ they obtain NP-hardness even if the number of weak edges is equal to zero. Also, they showed that, assuming the ETH, there is no $2^{o(|V|^2)}$ -time algorithm for VL-MULTI-STC and EL-MULTI-STC even if the number of weak edges is equal to zero and $c \in O(|V|)$. Regarding the parameterized complexity, Bulteau et al. [14] proposed to use the number of weak edges of an optimal solution of MAXSTC, denoted by k_1 , as a parameter. They provided a linear kernel for MULTI-STC and a $2^{c+1} \cdot k_1$ -vertex kernel for VL-MULTI-STC and EL-MULTI-STC. On the other hand, parameterization only by k_1 leads to $W[1]$ -hardness for both variations [14]. Also, in [64] the authors studied the parameterized complexity of MULTI-STC and EL-MULTI-STC by considering the following parameter, denoted by ξ_{c-1} : the minimum number

of edges that need to be deleted from the input graph in order to obtain a graph with maximum degree $c-1$. They showed that MULTI-STC is FPT with this parameter when $c \leq 4$ [64]. Moreover, EL-MULTI-STC admits a linear kernel parameterized by ξ_2 for every fixed c [64].

Besides the above generalizations of MAXSTC, several other variations have been considered closely related to triadic closure. Rozenshtein et al. [113] introduced two such variations under the terms MINVIOL and its dual MAXTRI. In the MINVIOL problem, we are given a graph $G = (V, E)$, a set of communities $C_1, \dots, C_k \subseteq V$, and a set of strong edges $S \subseteq E$, and the task is to ensure that each $(C_i, S(C_i))$ is connected and the number of triadic violations, $viol(S)$, is minimized. The authors showed that both problems MINVIOL and MAXTRI are NP-hard. In the MAXTRI problem the goal is to maximize the number of non-violated triangles. Rozenshtein et al. [113] developed an algorithm for MAXTRI with an approximation guarantee. Moreover, Bevern et al. [122] showed that *uncapacitated facility location problem with matroid constraints* (UFLP-MC) is FPT. By using this problem, they proved that MAXTRI is FPT with respect to $r + k$, where r is the number of non-violated triangles and k the number of communities.

Another way of attacking MAXSTC is through Linear Programming (LP) approach, since MAXSTC can be formulated as maximization problem where: *i*) for each edge there is a variable which takes 0 or 1 whether its label is weak or strong respectively, *ii*) the objective function is the sum of all variables, and *iii*) the constraints can be constructed as follows: for every two variables that correspond to an induced P_3 at most one can be 1. Thus, MAXSTC has an Integer Programming (IP) formulation. Adriaens et al. [3] studied such a relaxation of MAXSTC. The first relaxation is on the IP variables, where instead of using binary values for the variables, they allow each variable to take value between 0 and 1. In case of $\{0, 1/2, 1\}$, the problem is half-integral and can be solved in polynomial time [3]. The second relaxation allows the variables that correspond to a triangle in the graph to take values larger than 1. Moreover, the authors proposed three types of relaxations by changing the objective function and the constraints: allow violations, allow negative values, and maximize the number of strong edges in all triangles [3].

1.3 Our contribution

Inspired by the NP-completeness result of MAXSTC on general graphs and in order to have a better understanding on the complexity behavior of the problem, we focus on graphs classes and provide new algorithmic results. We study MAXSTC on subclasses of chordal graphs, since the class of chordal graphs finds important appli-

cations in both theoretical and practical areas related to social networks [2, 83, 108]. We prove that MAXSTC remains NP-complete on split graphs and, thus, on chordal graphs, since the class of split graphs forms a proper subclass of chordal graphs. Moreover, we give a polynomial-time algorithm on trivially perfect graphs by characterizing an auxiliary graph that we call line-incompatibility. Surprisingly, our main result is a polynomial-time algorithm on proper interval graphs by using a sophisticated analysis and a rather technical dynamic programming approach which comes in contrast to other known computational problems on proper interval graphs. Moreover, we are able to express MAXSTC and CLUSTER DELETION in monadic second order logic of second type (MSO_2) which provide us to solve both problems in linear time on graphs of bounded treewidth by applying Courcelle’s machinery through MSO_2 . These results have led to the following publications [88, 90]:

- **Maximizing the strong triadic closure in split graphs and proper interval graphs.** Athanasios L. Konstantinidis, and Charis Papadopoulos. *In 28th International Symposium on Algorithms and Computation (ISAAC 2017), Leibniz International Proceedings in Informatics (LIPIcs), pages 53:1–53:12, 2017.*
- **Maximizing the strong triadic closure in split graphs and proper interval graphs.** Athanasios L. Konstantinidis and Charis Papadopoulos. *Discrete Applied Mathematics 285: 79-95, 2020.*

Pushing further algorithmic results for MAXSTC, we prove that the optimal value for MAXSTC matches the optimal value for CLUSTER DELETION on cographs. By doing so, we reveal an interesting vertex partitioning with respect to maximum cliques and maximum independent sets. This result enables us to give an $O(n^2)$ -time algorithm for MAXSTC on cographs. As a byproduct we characterize a maximum independent set of the cartesian product of two cographs, which implies a polynomial-time algorithm for computing such a maximum independent set. Furthermore, we work on graphs of bounded degree. We show an interesting complexity dichotomy result: for graphs of maximum degree four MAXSTC remains NP-complete, whereas for graphs of maximum degree three the problem is solved in polynomial time. The proof of hardness on graphs of maximum degree four implies that there is no subexponential-time algorithm for MAXSTC unless the Exponential-Time Hypothesis (ETH) fails. These results have led to the following publications [86, 87]:

- **Strong triadic closure in cographs and graphs of low maximum degree.** Athanasios L. Konstantinidis, Stavros D. Nikolopoulos, and Charis Papadopoulos. *23rd Annual International Computing and Combinatorics Conference, (COCOON 2017), Hong Kong, China, 2017. Springer Verlag, LNCS 10392: 346–358.*

- **Strong triadic closure in cographs and graphs of low maximum degree.** Athanasios L. Konstantinidis, Stavros D. Nikolopoulos, and Charis Papadopoulos. *Theoretical Computer Science* 740: 76 -84, 2018.

Next we consider the CLUSTER DELETION which is well-studied problem. Despite the intensively efforts of determining its complexity on proper subclasses of chordal graphs, it still remained unresolved of whether it can be solved in polynomial time on interval graphs. As already mentioned, Bonomo et al. [11] proved that CLUSTER DELETION is polynomial solvable on proper interval graphs, a subclass of interval graphs, and it is NP-complete on chordal graphs. Our main contribution is that we settle this problem in the affirmative, by providing a polynomial-time algorithm for CLUSTER DELETION on interval graphs. In particular, our algorithm for interval graphs suggests to consider a particular consecutiveness of a solution and apply a dynamic programming approach defined by two vertex orderings.

Moreover, there is a very simple characterization of an optimal solution for CLUSTER DELETION on split graphs: either a maximal clique constitutes the only non-edgeless cluster, or there are exactly two non-edgeless clusters whenever there is a vertex of the independent set that is adjacent to all the vertices of the clique except one. Due to the fact that true twins belong to the same cluster in an optimal solution, it is natural to consider true twins at the independent set, as they are expected not to influence the solution characterization. Surprisingly, we show that CLUSTER DELETION remains NP-complete even on such a slight generalization of split graphs. The class of graphs that are obtained from split graphs by adding true twins is known as the class of starlike graphs and, besides the NP-completeness, we provide interesting structural characterizations. Based on the obtained structural properties, we reveal subclasses of starlike graphs for which CLUSTER DELETION is polynomial time solvable. These results have led to the following publication [89, 91]:

- **Cluster deletion on interval graphs and split related graphs.** Athanasios L. Konstantinidis and Charis Papadopoulos. *44th International Symposium on Mathematical Foundations of Computer Science, (MFCS 2019), Aachen, Germany, 2019. Leibniz-Zentrum für Informatik, LIPIcs* 138: 12(1)-12(14), 2019.
- **Cluster deletion on interval graphs and split related graphs.** Athanasios L. Konstantinidis and Charis Papadopoulos. *Algorithmica*, 2021.

In Table 1.1 we summarize our results together with previously-known results concerning the complexity of CLUSTER DELETION and MAXSTC on graph classes.

Further, we introduce a generalization of MAXSTC, named STRONG F -CLOSURE problem, and we study parameterized aspects of such a problem. We consider three

Graphs	CLUSTER DELETION	MAXSTC
General	NP-hard [116]	NP-hard [118]
Chordal	NP-hard [11]	NP-hard
Interval	Poly-time [Th. 5.3.14]	?
Proper interval	Poly-time [11]	Poly-time [Th. 3.4.17]
Split	Poly-time [11]	NP-hard [Th. 3.3.4]
Starlike	NP-hard [[11], Th. 5.4.3]	NP-hard
Cograph	Poly-time [50]	Poly-time [Th. 4.3.6]
Trivially-perfect	Poly-time [50]	Poly-time [Th. 3.2.3]
Triangle free	Poly-time	Poly-time
Bounded treewidth	Poly-time	Poly-time
Planar	NP-hard [Cor. 6.5.4]	NP-hard [Th. 6.5.3]
$(3K_1, 2K_2)$ -free	NP-hard [50]	NP-hard [Th. 6.5.5]
$\Delta = 3$	Poly-time [85]	Poly-time [Th. 4.4.3]
$\Delta \geq 4$	NP-hard [85]	NP-hard [Th. 4.4.1]

Table 1.1: Complexity of CLUSTER DELETION and MAXSTC restricted on particular graph classes. Our results obtained within this thesis are presented in **bold**.

different natural parameters to study the parameterized complexity of STRONG F -CLOSURE: the number of strong edges, the number of strong edges above guarantee (maximum matching size) and the number of weak edges. We show that STRONG F -CLOSURE is FPT when parameterized by the number of strong edges for a fixed F , even when we allow the size of F to be a parameter. We also observe that STRONG F -CLOSURE parameterized by the previous parameterization admits a polynomial kernel if F has a component with at least three vertices and the input graph is restricted to be d -degenerate. Next, we focus on the special case of $F = P_3$ which coincides with the MAXSTC problem. We complement our FPT results by proving that MAXSTC does not admit polynomial kernel even on split graphs unless $NP \subseteq coNP/poly$. Since the size of a maximum matching consists a lower bound for a solution of MAXSTC, we study the parameterization above this bound. In particular, we show that MAXSTC is FPT on graphs of maximum degree at most 4, parameterized by $k - \mu(G)$, where $\mu(G)$ is the maximum matching size of G . Moreover, we prove that STRONG F -CLOSURE is FPT and admits a polynomial kernel when parameterized by the number of weak edges and F is a fixed graph. We conclude with some results related to classical computational complexity. We show that MAXSTC remains NP-hard on $(3K_1, 2K_2)$ -free graphs and on planar graphs. Our reduction for planar graphs also works for the CLUSTER DELETION problem and, thus, CLUSTER DELETION remains NP-hard on planar graphs. These results have led to the following publications [55, 56]:

- **Parameterized aspects of strong subgraph closure.** Petr A. Golovach, Pinar Heggernes, Athanasios L. Konstantinidis, Paloma T. Lima and Charis Papadopoulos. *16th Scandinavian Symposium and Workshops on Algorithm Theory, (SWAT 2018), Malmo, Sweden, 2018. Leibniz-Zentrum für Informatik, LIPIcs 101: 23(1)-23(13), 2018.*
- **Parameterized aspects of strong subgraph closure.** Petr A. Golovach, Pinar Heggernes, Athanasios L. Konstantinidis, Paloma T. Lima, and Charis Papadopoulos. *Algorithmica 82: 2006-2038, 2020.*

1.4 Road map

In Chapter 2, we give fundamental definitions and notations of graph theory and graph classes. Moreover, we provide definitions of classical and parameterized complexity theory. Finally, we give formal definitions of all decision problems that we consider within this thesis.

In Chapter 3, we present our results for MAXSTC on split graphs, proper interval graphs and trivially perfect graphs. For split graphs we prove NP-hardness, whereas for the other two graph classes we provide polynomial-time algorithms.

In Chapter 4, our main result is a polynomial-time algorithm for computing MAXSTC on cographs and the equivalency between MAXSTC and CLUSTER DELETION. Furthermore, we study MAXSTC on graphs with low maximum degree.

In Chapter 5, we give a polynomial-time algorithm for CLUSTER DELETION on interval graphs and we study the complexity of the problem on graphs related to a natural generalization of split graphs.

In Chapter 6, we study the parameterized complexity of STRONG F -CLOSURE and MAXSTC with three different natural parameter. Moreover, we present NP-hardness results for MAXSTC on planar and $(3K_1, 2K_2)$ -free graphs.

In Chapter 7, we summarize our results and we discuss possible future directions for further research.

CHAPTER 2

DEFINITIONS AND NOTATIONS

In this chapter we fix some basic notation and terminology, and briefly introduce the most fundamental concepts in computational complexity from both the classical as well as the parameterized viewpoints. We discuss several graph classes which are relevant to this thesis. Moreover we provide a catalogue with formal definitions of all considered problems discussed within this thesis.

2.1 Basic Concepts on Graph Theory

A *graph* is a pair $G = (V, E)$ such that $E \subseteq V \times V$. The elements of V are called *vertices* or *nodes* and the element of E are called *edges* of G . So, each edge is a pair of vertices. We consider all pairs are unordered. We denote the number of vertices of G by n and the number of edges of G by m ; that is $n = |V(G)|$ and $m = |E(G)|$.

A vertex v is *incident* to an edge e if $v \in e$ or the edge e is *incident* to v . The two vertices of an edge are called *end-vertices* or *endpoints* and the edge *joins* its end-vertices. We denote an edge e that joins the vertices u and v by $e = \{u, v\}$ or $e = uv$. For simplicity, we sometimes write $v \in G$, instead of $v \in V(G)$, and $e \in G$ instead of $e \in E(G)$.

Two vertices $u, v \in G$ are *adjacent* or *neighbors* if $e = \{u, v\}$ is an edge of G . Two edges are adjacent if they have an endpoint in common. The *neighborhood* of a vertex v of G is $N(v) = \{x | \{v, x\} \in E\}$ and the *closed neighborhood* of a vertex v of G is $N[v] = N(v) \cup \{v\}$. We extend this notion to vertex sets: for $S \subseteq V$, $N(S) = \bigcup_{v \in S} N(v) \setminus S$ and $N[S] = N(S) \cup S$. For two vertices u and v we say that u *sees* v if $\{u, v\} \in E(G)$; otherwise, we say that u *misses* v . We extend this notion to vertex sets: a set A *sees* (resp., *misses*) a vertex set B if every vertex of A is adjacent (resp., non-adjacent) to every vertex of B . Moreover, two adjacent vertices u and v are called *true twins* if $N[u] = N[v]$, whereas two non-adjacent vertices x and y are called *false twins* if $N(x) = N(y)$.

The *degree* of a vertex v is $d(v) = |N(v)|$, that is, the number of the incident edges

to v . We denote by Δ the maximum degree of G , so that $\Delta = \max\{d(v) \mid \forall v \in G\}$. A vertex of degree 0 is called *isolated*, a vertex of degree 1 is called *pendant*, and a vertex v is called *universal* if $d(v) = |V(G)| - 1$.

The *complement* of a graph G , is the graph $\bar{G} = (V', E')$ with $V' = V$ and $E' = \{\{u, v\} \mid \{u, v\} \notin E\}$.

A set of pairwise adjacent vertices of G is called *clique*. The size of maximum clique in G is called the *clique number*, denoted by $\omega(G)$. A maximal clique of G is a clique of G that is not properly contained in any clique of G .

A set of pairwise non-adjacent vertices of G is called *independent set*. The maximum independent set denoted by $\alpha(G)$.

A *vertex cover* of a graph is a set of vertices that includes at least one endpoint of every edge of the graph. It is not difficult to see that a set of vertices X is a vertex cover if and only if $V(G) \setminus X$ is an independent set.

A *matching* in G is a set of edges having no common endpoint. The *maximum matching number*, denoted by $\mu(G)$, is the maximum number of edges in any matching of G . We say that a vertex v is *covered* by a matching M if v is incident to an edge of M . We denote by $V(M)$ the set of vertices covered by a matching M . It is not difficult to see that the number of vertices in a maximum matching constitutes a lower bound for the size of a minimum vertex cover.

Operations over the graphs

Let $G = (V, E)$ be a graph. The following operations applied on G , result in a graph that is obtained from G by standard set operations on V and E . For a vertex $v \in V$ the *vertex deletion* of v from G has as result to delete the vertex v from G and all the incident edges to v . We denote this operation by $G - v$. For an edge $e \in E$ the *edge deletion* of e from G has as result to delete only the edge e from G . We denote this operation by $G - e$. We can extend these definitions for a set of vertices U and for a set of edges F . In the former case we delete all the vertices of U and all the incident edges to U from G , denoted by $G - U$. In the latter case we delete only the edges of F from G , denoted by $G \setminus F$. Let $\{x, y\}$ be an edge of G . The *contraction* of the edge $\{x, y\}$ is the operation consisting in deleting the vertices x and y from G and adding a new vertex which is adjacent to every vertex of $G \setminus \{x, y\}$ that is adjacent to x or y in G . More general, *contracting* a set of vertices S is the operation of substituting the vertices of S by a new vertex w with $N(w) = N(S)$.

Relations between graphs

Let $G = (V, E)$ and $H = (U, F)$ be two graphs. We have the following definitions:

- G and H are *disjoint* if $V \cap U = \emptyset$ (implying that $E \cap F = \emptyset$).

- H is a *subgraph* of G if $U \subseteq V$ and $F \subseteq E$, which means H can be obtained from G by deleting some vertices or edges from G .
- H is an *induced subgraph* of G if $U \subseteq V$ and $F \subseteq E$, but H can be obtained from G by deleting only some vertices from G .
- H is a *spanning subgraph* of G if $U = V$ and $F \subseteq E$ (implying that H can be obtained from G by deleting only some edges from G).
- H is a *minor* of G if H can be obtained from G by a (possibly empty) sequence of vertex deletions or edge deletions or edge contractions.
- G and H are *isomorphic* if there is a bijection $f: V \rightarrow U$ that preserves the adjacency, i.e. $\{u, v\} \in E$ if and only if $\{f(u), f(v)\} \in F$. If G and H are isomorphic then we write $G \simeq H$, whereas $G \not\simeq H$ stands for two non-isomorphic graphs.

For $X \subseteq V(G)$, the subgraph of G *induced* by X , denoted by $G[X]$, has vertex set X , and for each vertex pair u, v from X , $\{u, v\}$ is an edge of $G[X]$ if and only if $u \neq v$ and $\{u, v\}$ is an edge of G . Observe that $G[X]$ is the graph obtained from G by deleting the vertices of $V \setminus X$. We say that G *contains* a graph H (as an induced subgraph) if G has an induced subgraph isomorphic to H .

A graph G is *self-complementary* if G is isomorphic to its complement.

Operations between two graphs

Let G and H be two disjoint graphs G and H . We consider the following operations resulting on a new graph:

- the *disjoint union* of G and H : $G \oplus H = (V(G) \cup V(H), E(G) \cup E(H))$.
- the *complete join* of G and H : $G \otimes H = (V(G) \cup V(H), E(G) \cup E(H) \cup \{\{u, v\} \mid u \in E(G), v \in E(H)\})$.
- the *cartesian product* of G and H , denoted by $G \times H$, is the graph with the vertex set $V(G) \times V(H)$ and any two vertices (u, u') and (v, v') are adjacent in $G \times H$ if and only if either $u = v$ and u' is adjacent to v' in H , or $u' = v'$ and u is adjacent to v in G .

For a positive integer p , pG denotes the disjoint union of p copies of G . Moreover, the disjoint union of two graphs G and H may be simply denoted by $G + H$.

Special graphs

A graph with n vertices is called *complete graph* if there is an edge for every pair

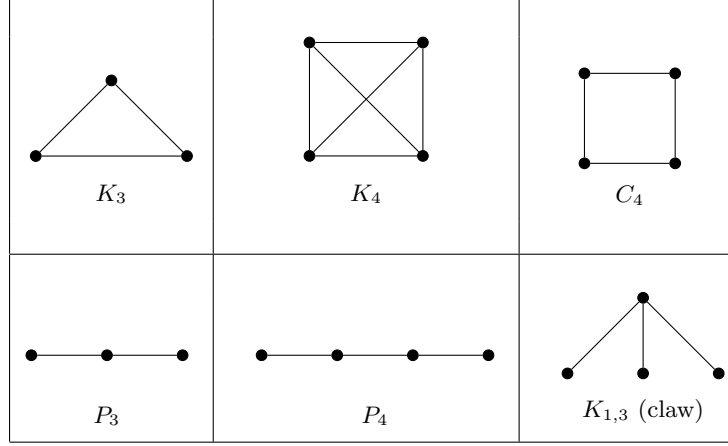


Figure 2.1: Some basic graphs.

of vertices in the graph, denoted by K_n . The complete graph with three vertices, K_3 , is called *triangle*.

A *chordless path* with n vertices, denoted by P_n , is a graph with $V(P_n) = \{v_1, \dots, v_n\}$ and $E(P_n) = \{\{v_i, v_{i+1}\} | 1 \leq i < n\}$.

A *chordless cycle* with n vertices, denoted by C_n , is a graph with $V(C_n) = \{v_1, \dots, v_n\}$ and $E(C_n) = \{\{v_i, v_{i+1}\} | 1 \leq i < n\} \cup \{\{v_n, v_1\}\}$.

A graph G is *connected* if every pair of distinct vertices is joined by a path. Otherwise, the graph is referred to as *disconnected*. A maximal connected subgraph of a graph is called a *connected component* of the graph.

For a fixed graph F , a graph G is said to be *F-free* if G has no induced subgraph isomorphic to F .

A *cluster graph* is a graph in which every connected component is a complete graph. Cluster graphs are characterized as exactly the graphs that do not contain a P_3 as an induced subgraph. That is, cluster graphs are exactly the P_3 -free graphs.

The *line graph* $L(G)$ of a graph G is the graph that has as vertex set the edges of G and two vertices of $L(G)$ are adjacent if and only if the corresponding edges in G have a common endpoint.

The *line-incompatibility* \widehat{G} of a graph G is the graph that has as vertex set the edges of G and two vertices of \widehat{G} are adjacent if and only if the corresponding edges induce a P_3 in G . The line-incompatibility graph has already been considered under the term *Gallai graph*, denoted by $\Gamma(G)$, in [96] or as an auxiliary graph in [24, 118]. Note that the line-incompatibility graph of G is a spanning subgraph of the line graph of G .

An *induced matching* in a graph G , denoted by qK_2 , is a set of q edges, no two of which have a common vertex or are joined by an edge of G .

An *acyclic* graph is a graph that does not contain any cycle and is called *forest*. A connected forest is called *tree*. In a tree, the vertices of degree 1 are called *leaves* whereas the rest of its vertices are called *internals*. A tree with one internal vertex and n leaves is called *star graph* and is denoted by $K_{1,n}$. We will often refer to a particular star graph: $K_{1,3}$ that is called *claw*.

In Figure 2.1 are given some examples of special graphs.

2.2 Graph Classes

A graph class is a set of graphs that share a common property. Graphs arising from applications may naturally fall into one such class. On the other hand, graph classes arise naturally within the scope of better understanding the tractability boundaries of a computational hard problem. For each pair of a graph problem that is intractable in general and a graph class, we wish to determine that either the problem becomes efficiently solvable when restricted to the class, or that it still remains hard. The body of literature upon this type of results is vast, as exposed by the online database [31] that currently lists 1623 graph classes with the corresponding 26763 algorithmic reference.

Let M be a family of nonempty sets. The *intersection graph* of M is obtained by representing each set in M by a vertex and two vertices are connected by an edge if and only if their corresponding sets have a non-empty intersection. We refer to the obtained graph as the *intersection graph* G of M and we call M the *intersection model* of G . Without imposing some restrictions on the elements and the family of M , the class of graphs obtained as intersection graphs is simply all undirected graphs. An interesting case is when M is taken over a set of geometric objects. In such sense, intersection graphs are a very popular “meta graph class”. An algorithmic vein towards this direction is to determine the complexity of recognizing the members of each graph class. In particular, the problem of characterizing the intersection graphs of families of sets having some specific topological or other pattern is often very interesting and frequently has applications to the real world.

A class \mathcal{G} of graphs containing with each graph G all induced subgraphs of G is called *hereditary*. In other words, \mathcal{G} is hereditary if and only if $G \in \mathcal{G}$ implies $G[X] \in \mathcal{G}$ for any $X \subseteq V(G)$. Some known hereditary graph classes are perfect graphs, cluster graphs, bipartite graphs, whereas simple examples showing the non-hereditary property are paths or trees.

Let \mathcal{F} be a set of graphs. A graph class \mathcal{G} is characterized as *\mathcal{F} -free* if and only if every graph $G \in \mathcal{G}$ does not contain any graph $F \in \mathcal{F}$ as an induced subgraph. In this

scope, we say that the graphs in \mathcal{F} are *forbidden induced subgraphs* for the class \mathcal{G} .

Looking closer at the above two definitions of a hereditary class \mathcal{G} and a set of forbidden induced subgraphs \mathcal{F} , an interesting characterization shows that a class of graphs \mathcal{G} is hereditary if and only if there is a set \mathcal{F} such that \mathcal{G} is \mathcal{F} -free.

Characterizations of graph classes by forbidden induced subgraphs do not only provide algorithmic tools, but also play a key role as the subject of profound mathematics. The most striking example is probably the case of perfect graphs.

Perfect graphs: A graph is *perfect* if for every induced subgraph H the chromatic number $\chi(H)$ equals the clique number $\omega(H)$. The class of perfect graphs is very important from both theoretical as well as practical point of view, since the two problems of determining the chromatic number and the clique number can be solved in polynomial time [61]. Moreover, there is another characterization for perfect graphs, the Strong Perfect Graph Theorem, which states that a graph G is perfect if and only if G and \bar{G} contain no induced C_{2k+1} for $k \geq 2$ [23]. By this theorem, it is easy to see that a perfect graph is self-complementary. Therefore a maximum independent set can be found in polynomial time on perfect graphs, as it is a maximum clique in the complement graph.

Chordal graphs: A well known subclass of perfect graphs is the class of chordal graphs, also known as triangulated graphs. A graph is *chordal* if each cycle in G of length at least 4 has at least one chord. Equivalently, G does not contain an induced subgraph isomorphic to C_n for $n > 3$. Moreover, a graph is chordal if and only if it is the intersection graph of a family of subtrees of a tree [52].

Interval graphs: A graph is an *interval* graph if there is a bijection between its vertices and a family of closed intervals of the real line such that two vertices are adjacent if and only if the two corresponding intervals intersect. Such a bijection is called an interval representation of the graph. An example of an interval graph and its interval representation can be seen in Table 2.1. Thus they correspond to the graphs that can be represented as the intersection graphs of intervals on the real line. Interval graphs form a proper subclass of chordal graphs. Moreover, it is known that an interval graph can be characterized by a set of forbidden induced subgraphs [98].

Proper interval graphs: The interval graphs in which no interval is properly contained in another interval form the class of *proper interval* graphs. An example of a proper interval graph and its interval representation can be seen in Table 2.1. The interval graphs in which all intervals are required to have unit length form the class of *unit interval* graphs. Roberts showed that the classes of unit interval graphs and proper interval graphs coincide [110]. Also, he characterized them as claw-free interval graphs

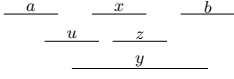
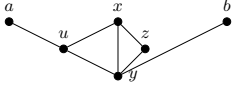
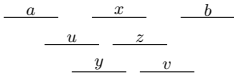
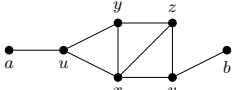
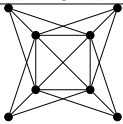
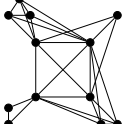
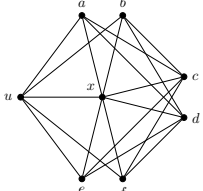
Interval graph	
	
Proper interval graph	
	
Split graph	
	
Starlike graph	
	
Cograph	
	

Table 2.1: Five known graph classes and an example for each class.

[110].

Split graphs: Another subclass of chordal graphs is the class of split graphs which are incomparable to interval and proper interval graphs. A graph is a *split* if its vertex set can be partitioned into a clique and an independent set. Such a partition is called *split partition*. A split graph with four vertices in the independent set and four vertices in the clique can be seen in Table 2.1. It is not difficult to see that the class of split graphs is self-complementary, that is, the complement of a split graph remains a split graph. Furthermore, they have been characterized by forbidden set of induced subgraphs. A graph is a split if and only if it contains no C_4 , C_5 and $2K_2$ as an induced subgraph, that is, they are exactly the $\{C_4, C_5, 2K_2\}$ -free graphs [45]. Moreover, in terms of the intersection model, split graphs form the intersection graphs of distinct subtrees of a star.

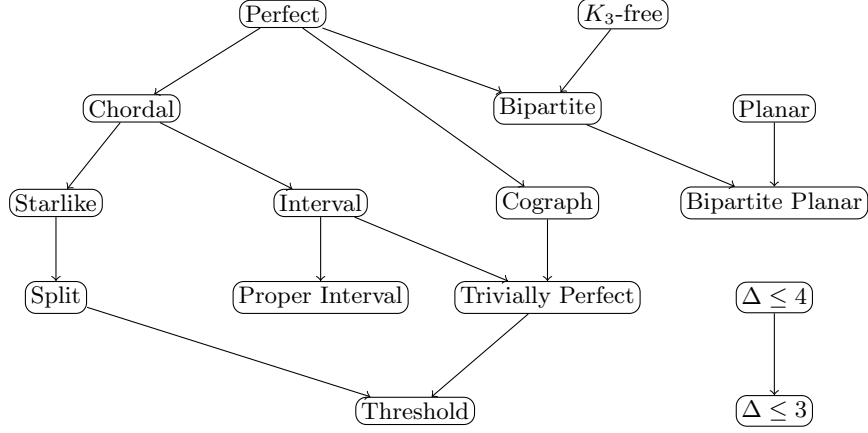


Figure 2.2: A Hasse diagram among the considered graph classes. An arrow from class X to class Y shows that Y is a proper subclass of X .

Starlike graphs: A graph $G = (V, E)$ is called starlike graph if its vertex set can be partitioned into C and I such that $G[C]$ is a clique and every two vertices of I are either non-adjacent or true twins in G . A starlike graph with eight vertices in I and four vertices in C can be seen in Table 2.1. By definition, split graphs form a proper subclass of starlike graphs. Their name comes from the corresponding characterization through intersection graphs: they are exactly the intersection graphs of subtrees of a star [19].

Bipartite graphs: A graph is *bipartite* if its vertex set can be partitioned into at most 2 independent sets. A bipartite graph G with a given bipartition $A \cup B$ will be denoted $G = (A, B, E)$. Moreover, a graph G is bipartite if and only if it contains no cycles of odd length. A bipartite graph $G = (A, B, E)$ is *complete bipartite* if every vertex of A is adjacent to every vertex of B . A complete bipartite graph with parts of size n and m is denoted $K_{n,m}$. Bipartite graphs can be seen as intersection bigraphs of two given families M, M' of subsets: the bipartite graph in which each set in M is a red vertex, each set in M' is a blue vertex, and a red vertex is adjacent to a blue vertex if and only if the corresponding sets intersect.

Cographs: A graph is *cograph* if it can be generated from a single vertex graph and recursively applying the disjoint union and complete join operations. Cographs are exactly the graphs that do not contain any chordless path on four vertices, that is, they are exactly the P_4 -free graphs [26]. An example of a cograph can be seen in Table 2.1. Cographs are unrelated to chordal graphs, as they contain chordless cycles, whereas any chordless path is a chordal graph.

Trivially Perfect graphs: A graph G is called *trivially-perfect* (also known as *quasi-threshold*) if for each induced subgraph H of G , the number of maximal cliques of H is equal to the maximum size of an independent set of H . The class of *trivial perfect* graphs coincides with the class of chordal graphs and cographs. In other words, this is the class of (C_4, P_4) -free graphs [57].

Threshold graphs: A graph is a *threshold* graph if there is a real number s (known as the *threshold*) and for every vertex v there is a real weight $w(v)$ such that: $\{u, v\}$ is an edge if and only if $w(v) + w(u) \geq s$. A more intuitive characterization is that threshold graphs are exactly the graphs that are both split graphs and cographs. Hence, they are exactly the $(P_4, C_4, 2K_2)$ -free graphs. However, since split graphs are also chordal, threshold graphs are actually a subclass of trivially perfect graphs. As they are split graphs, their vertex set can be partitioned into a clique and an independent set, but, in addition, the neighborhood of the independent set has the special property of being orderable by inclusion [103].

Non-Perfect graphs:

A graph is *triangle-free* if it contains no K_3 as an induced subgraph. Every bipartite graph is triangle-free, whereas every triangle-free graph is not necessarily bipartite.

For a non-negative integer k , a *k-degenerate* graph is a graph in which every subgraph has a vertex of degree at most k . To give an example, forests are exactly the 1-degenerate graphs.

A graph is *planar* if it can be drawn in the plane so that no two edges cross each other. However the most famous characterization of this graph class is through forbidden minors. Planar graphs are exactly the graphs that do not contain K_5 and $K_{3,3}$ as minors. A planar graph is 5-degenerate graph, as every planar graph has a vertex of degree five or less.

In Figure 2.2 we present the inclusion relations among the considered graph classes.

2.3 Computational Complexity

Classical complexity

The theory of computational complexity explores the amount of resources that a computational problem needs to be solved and classifies the problems according to the amount of resources required. For example, as amount of resources can be considered time and space. Here, we mainly focus on time. In other words, for given a computational problem we want to know if there is an algorithm that solves the problem in an efficient way or not. Hence, the *complexity* of an algorithm is intended to

measure exactly this type of efficiency.

A fundamental notion used to describe the complexity of an algorithm is the asymptotic upper bound $O(f(n))$. Big O -notation expresses the upper bound of operations (worst case running time) that an algorithm needs to solve a problem. More precisely, for a given function $g(n)$, that measures the worst case running time of an algorithm on an input of size n , and another function $f(n)$, we say that $g(n)$ is $O(f(n))$ if there exist constants $c > 0$ and $n_0 > 0$ so that for all $n \geq n_0$, we have $g(n) \leq c \cdot f(n)$.

Another useful notation is the little o -notation in which we say that $g(n)$ is $o(f(n))$, if for any constant $c > 0$ there exists a constant $n_0 > 0$ so that for all $n \geq n_0$, we have $g(n) < c \cdot f(n)$. Besides the asymptotic upper bound $O(f(n))$, one can consider the asymptotic lower bound big Ω -notation in which we say that $g(n)$ is $\Omega(f(n))$, if there exist constants $c > 0$ and $n_0 > 0$ so that for all $n \geq n_0$, we have $g(n) \geq c \cdot f(n)$.

The notation of big O helps us to order the algorithms based on their asymptotic running times. For example, an algorithm is linear if it is $O(n)$ and it is more efficient than an exponential algorithm $O(2^n)$. We say that an algorithm is efficient if it is polynomial, meaning $O(n^k)$ for some constant $k \geq 0$. Thus, we can classify the problems in classes based on the complexity of the algorithms that solve the problems. The classes are known as *Complexity Classes*.

The class that contains all the problems which can be solved in polynomial time is denoted by P . There are problems that are harder and, probably, they cannot be solved in polynomial time. One class that contains such problems is the class of NP . To define this class we must give the definition of *certifier*. A certifier is an algorithm that checks if a given possible solution for a problem is indeed a solution, that is whether the decision problem answers correctly “yes”. The class of NP contains all the problems that have a polynomial certifier. It is obvious that $P \subseteq NP$, but it is widely open whether $P = NP$ or $P \neq NP$. Also, there is the complement of the class NP . This class is called $coNP$ and is defined as the complexity class which contains the complements of problems found in NP (whether the decision problem answers correctly “no”).

In order to study these problems, we need a technique to compare the relative difficulty among them. In some sense, we want to know which problem is as least as hard as another. This is achieved through the technique of *reduction*. Suppose we have a black box that can solve instances of a problem Y . We say that a problem X is polynomial time reducible to Y , denoted by $X \leq_p Y$, if any instance of X can be solved using polynomial number of computational step (transformed to some instance of Y) plus a polynomial number of calls of the black box that solves Y .

There are some important results of this technique. If we know that the problem Y is polynomial solvable then we have an algorithm to solves problem X in polynomial time. On the other hand, if X cannot be solved in polynomial time then Y cannot be

solved in polynomial time. By taking advantage of the second property, we can define the hardest problem in class NP. A problem Y is **NP-complete** if: (i) $Y \in \text{NP}$ and (ii) every problem $X \in \text{NP}$ is polynomial reducible to Y . If we remove the first assumption of NP-completeness (membership in NP), then we say that the problem Y is **NP-hard**.

Parameterized complexity

In contrast to classical complexity, parameterized complexity is a two dimensional framework for studying the computational complexity of a problem. One dimension is the input size n and the other is a parameter k associated with the input. As a parameter in a problem more often is used the natural parameter (the size of the solution). On the other hand, it can be used anything that measures a structural property of the input. For example, in graph problems parameters such as maximum degree, treewidth or maximum matching may be widely chosen.

The basic idea of parameterized complexity is to refine the analysis of hard problems. It was developed by Downey and Fellows [36, 38]. The central part of the theory is fixed-parameter tractability. A problem with input size n and parameter k is fixed parameter tractable (FPT), if it can be solved in time $f(k) \cdot n^{O(1)}$ for some computable function f . Here, we want to design an algorithm that the non-polynomial part depends only on the parameter. This means that the problem is solvable in an efficient way whenever the parameter is small.

A classical NP-complete problem which is FPT parameterized by the size of the solution is VERTEX COVER. It is known that VERTEX COVER can be solved in $2^k \cdot n^{O(1)}$ time by a Bounded Search Tree technique. The best running time for VERTEX COVER is $O(1.2738^k + k \cdot n)$ [21]. On the other hand, it is not known whether INDEPENDENT SET can be solved in $f(k) \cdot n^{O(1)}$. It is believed that no such fixed-parameter algorithm for INDEPENDENT SET exists. For a fixed k , it can be solved in $O(n^k)$ time by enumerating all subsets of size at most k . Problems which can be solved in $f(k) \cdot n^{g(k)}$ for some computable functions f, g are called *slice-wise polynomial* and they are denoted by the complexity class XP. It holds $\text{FPT} \subseteq \text{XP}$. Hence, there are problems that have the same classical complexity, but they have different behavior in parameterized complexity.

Below we provide some formal definitions on parameterized complexity.

Definition 2.3.1 ([30]). *A parameterized problem is a language $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet. For an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, k is called the parameter.*

Definition 2.3.2 ([30]). *A parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is called parameter tractable (FPT) if there exist an algorithm A (called a fixed parameter algorithm), computable function $f : \mathbb{N} \rightarrow \mathbb{N}$, and a constant c such that, given $(x, k) \in \Sigma^* \times \mathbb{N}$, the algorithm A correctly decide whether $(x, k) \in L$ in time bounded by $f(k) \cdot |(x, k)|^c$. The complexity class containing all fixed parameter tractable problems is called FPT.*

Definition 2.3.3 ([30]). A parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is called slice-wise polynomial (**XP**) if there exists an algorithm A and two computable functions $f, g : \mathbb{N} \rightarrow \mathbb{N}$ such that, given $(x, k) \in \Sigma^* \times \mathbb{N}$, the algorithm A correctly decides whether $(x, k) \in L$ in time bounded by $f(k) \cdot |(x, k)|^{g(k)}$. The complexity class containing all slice-wise polynomial problems is called **XP**.

Moreover, there is an analogous notion of reduction for parameterized problems that transfers fixed-parameter tractability.

Definition 2.3.4 ([30]). Let $A, B \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. A parameterized reduction from A to B is an algorithm that, given an instance (x, k) of A , outputs an instance (x', k') of B such that:

- (i) (x, k) is a yes-instance of A if and only if $((x', k'))$ is a yes-instance of B ,
- (ii) $k' \leq g(k)$ for some computable function g , and
- (iii) the running time is $f(k) \cdot |x|^{O(1)}$ for some computable function f .

As already mentioned, there is an FPT algorithm for VERTEX COVER, but not for INDEPENDENT SET which is in XP. It is interesting to provide evidence which shows that a parameterized problem is not FPT. In this direction, Downey and Fellows introduced the W -Hierarchy [37]. They defined the complexity classes $W[t]$, for all $t \in \mathbb{N}$ such that $\text{FPT} \subseteq W[1] \subseteq W[2] \subseteq \dots \subseteq \text{XP}$. The level of W -hierarchy are defined by restricting the WEIGHTED CIRCUIT SATISFIABILITY problem. Furthermore, a parameterized problem Q is called $W[t]$ -hard if every problem in $W[t]$ is parameterized reducible to Q , and $W[t]$ -complete if it is also contained in $W[t]$. For example, it is known that INDEPENDENT SET is $W[1]$ -complete and DOMINATING SET is $W[2]$ -complete. Hence, we have evidence that it is unlikely for both problems to be fixed-parameter tractable.

There are problems that do not admit FPT algorithms and not even belong to XP. A typical example is the problem of k -COLORABILITY. It is NP-complete even if we use $k = 3$ colors. Thus any $f(k) \cdot n^c$ algorithm or $f(k) \cdot n^{g(k)}$, for any computable functions f, g and any constant c that solves k -COLORABILITY implies that $P = NP$. For these type of problems we have the following complexity class.

A parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is in *para-NP*, if there is a computable function $f : \mathbb{N} \rightarrow \mathbb{N}$, a constant c , and a nondeterministic algorithm that, given $x \in \Sigma^*$, decides if $(x, k) \in L$ in at most $f(k) \cdot |(x, k)|^c$ steps [44].

A parameterized problem Q is *para-NP-hard* if for any parameterized problem P in *para-NP* there is parameterized reduction from P to Q . If the parameterized problem Q is additionally contained in *para-NP* then Q is *para-NP-complete*. Observe, every parameterized problem that is already NP-hard for a constant parameter value

is para-NP-hard, i.e., para-NP-hard problems do not admit FPT-algorithms under the assumption that $P \neq NP$. Hence, the problem of k -COLORABILITY is known to be para-NP-complete.

In order to study a parameterized problem one either designs a FPT algorithm or shows that the problem is $W[t]$ for some t . Instead of explicitly design a FPT algorithm, there are some tools to implicitly show that a problem is FPT. One of them is kernelization, a preprocessing algorithm that solves the easy part of an instance (with polynomial rules) and shrinks it as much as possible (kernel) to its difficult part (solvable in brute force manner).

Definition 2.3.5. A generalized kernelization [8] (or bi-kernelization [4]) for a parameterized problem P is a polynomial algorithm that maps each instance (x, k) of P with the input x and the parameter k into to an instance (x', k') of some parameterized problem Q such that

- (i) (x, k) is a yes-instance of P if and only if (x', k') is a yes-instance of Q ,
- (ii) the size of x' is bounded by $f(k)$ for a computable function f , and
- (iii) k' is bounded by $g(k)$ for a computable function g .

The output (x', k') is called a *generalized kernel* of the considered problem. The function f defines the size of a generalized kernel and the *generalized kernel has polynomial size* if the function f is polynomial. If $Q = P$, then generalized kernel is called *kernel*. Note that if Q is in NP and P is NP-complete, then the existence of a polynomial generalized kernel implies that P has a polynomial kernel because there exists a polynomial reduction of Q to P .

Definition 2.3.6. A polynomial compression of a parameterized problem P into a (nonparameterized) problem Q is a polynomial algorithm that takes as an input an instance (x, k) of P and returns an instance x' of Q such that

- (i) (x, k) is a yes-instance of P if and only if x' is a yes-instance of Q ,
- (ii) the size of x' is bounded by $p(k)$ for a polynomial p .

Clearly, the existence of a (generalized) polynomial kernel implies that the problem admit a polynomial compression but not the other way around. It is well-known that every decidable parameterized problem is FPT if and only if it admits a kernel, but it is unlikely that every problem in FPT has a polynomial kernel or polynomial compression. In particular, the now standard *composition* and *cross-composition* techniques [8, 9] allow to show that certain problems have no polynomial compressions unless $NP \subseteq coNP/poly$.

It is common to build an FPT algorithm or a kernel for a parameterized problem by constructing a series of *reduction rules*, that is, polynomial algorithms that either solve the problem or produce instances of the problem that, typically, have small sizes or small values of the parameter. Respectively, a rule is *safe* or *sound* if it either correctly solves the problem or constructs an equivalent instance.

Lower Bounds and ETH

In order to prove lower bounds for a problem there is the conjecture of *Exponential-Time Hypothesis (ETH)*: it states that k -SAT, $k \geq 3$, cannot be solved in time $2^{o(n)}$ or $2^{o(m)}$ where n is the number of variables and m is the number of clauses in the given k -CNF formula (see for e.g., [74, 101, 123]). In this context, algorithms with running time $2^{o(p)}$ for some parameter p are called *subexponential-time* algorithms. Thus, for any fixed $k \geq 3$, k -satisfiability does not have a subexponential time algorithm, under ETH.

2.4 Problem Definitions

In this section, we provide formal statements of some well-known NP-complete problems, given in the following list. All problems are considered as decision problems in which the output is a yes- or a no-answer.

VERTEX COVER

Input: A graph $G = (V, E)$ and a non-negative integer k .

Task: Does G have a vertex cover of size k or less?

INDEPENDENT SET

Input: A graph $G = (V, E)$ and a non-negative integer k .

Task: Does G have an independent set of size k or more?

CLIQUE

Input: A graph $G = (V, E)$ and a non-negative integer k .

Task: Does G have a clique of size k or more?

DOMINATING SET

Input: A graph $G = (V, E)$ and a non-negative integer k .

Task: Does G have a dominating set of size k or less, (i.e a subset $V' \subseteq V$ with $|V'| \leq k$ such that for all $u \in V - V'$ there is a $v \in V'$ for which $\{u, v\} \in E$)?

K-COLORABILITY (CHROMATIC NUMBER)

Input: A graph $G = (V, E)$ and a non-negative integer k .

Task: Is G k -colorable, i.e., does there exist a function $f: V \rightarrow \{1, 2, \dots, k\}$ such that $f(u) \neq f(v)$ whenever $\{u, v\} \in E$?

F-FREE EDGE DELETION

Input: A graph $G = (V, E)$ and a non-negative integer k .

Task: Does there exist a set $E' \subset E$ of size at most k such that $G \setminus E'$ is an F -free graph?

INDUCED SUBGRAPH ISOMORPHISM

Input: Two graphs G and H .

Task: Does G have an induced subgraph isomorphic to H ?

 d -SET PACKING

Input: A universe \mathcal{U} , a family \mathcal{F} of sets over \mathcal{U} , where each set in \mathcal{F} is of size at most d , and a non-negative integer k .

Task: Does there exist a family $\mathcal{F}' \subseteq \mathcal{F}$ of k pairwise disjoint sets?

 d -HITTING SET

Input: A universe \mathcal{U} , a family \mathcal{F} of sets over \mathcal{U} , where each set in \mathcal{F} is of size at most d , and a non-negative integer k .

Task: Does there exist a set $X \subset \mathcal{U}$ of size at most k that has a nonempty intersection with every element of \mathcal{F} ?

EXACT COVER BY 3-SETS (X3C)

Input: A set X with $|X| = 3q$ elements and a collection C of triplets of X .

Task: Does C contain an exact cover for X , i.e., a subcollection $C' \subseteq C$ such that every element of X occurs in exactly one member of C' ?

PLANARX3C

Input: A bipartite planar graph $G = (A, B, E)$ where every vertex of B has degree 3.

Task: Is there an induced subgraph $H = (A', B', E')$ of G such that $A' = A$, $B' \subseteq B$, and every vertex of A' has degree one?

KNAPSACK

Input: Finite set U , for each $u \in U$ a size $s(u) \in \mathbb{Z}^+$ and a value $v(u) \in \mathbb{Z}^+$, and positive integers B and K .

Task: Is there a subset $U' \subseteq U$ such that $\sum_{u \in U'} s(u) \leq B$ and $\sum_{u \in U'} v(u) \geq K$?

In the CLUSTER DELETION problem the goal is to partition the vertices of a given graph G into vertex-disjoint cliques with the minimum number of edges outside the cliques, or, equivalently, with the maximum number of edges inside the cliques.

CLUSTER DELETION

Input: A graph $G = (V, E)$ and a non-negative integer k .

Task: Does there exist a disjoint union of cliques by deleting at most k edges?

Observe that CLUSTER DELETION coincides with the F -FREE EDGE DELETION problem by setting $F = P_3$.

Next we highlight and give formal statements of the main problems considered within this thesis. Given a graph $G = (V, E)$, a *strong-weak labeling* on the edges of G is a function λ that assigns to each edge of $E(G)$ one of the labels *strong* or *weak*; i.e., $\lambda : E(G) \rightarrow \{\text{strong}, \text{weak}\}$. An edge that is labeled strong (resp., weak) is simply called *strong* (resp. *weak*). The *strong triadic closure* of a graph G is a strong-weak labeling λ such that for any two strong edges $\{u, v\}$ and $\{v, w\}$ there is a (weak or strong) edge $\{u, w\}$. We say that such a labeling *satisfies* the strong triadic closure. In other words, in a strong triadic closure there is no pair of strong edges $\{u, v\}$ and $\{v, w\}$ such that $\{u, w\} \notin E(G)$.

The problem of computing a maximum strong triadic closure, denoted by MAXSTC, is stated as follows:

MAXSTC

Input: A graph $G = (V, E)$ and a non-negative integer k .

Task: Does there exist a strong-weak labeling of $E(G)$ that satisfies the strong triadic closure and has at least k strong edges?

Note that the MAXSTC problem can be considered as an edge modification problem where the task is to find a spanning subgraph H of G with at least k edges, such that for every induced P_3 of H the vertices of P_3 induce a K_3 in G . We formalize this observation in the following generalized problem.

In the STRONG F -CLOSURE problem, we have a fixed graph F , and we are given an input graph G , together with an integer k . The task is to decide whether we can select at least k edges of G and mark them as *strong*, in the following way: whenever the subgraph of G spanned by the strong edges contains an induced subgraph isomorphic to F , then the corresponding induced subgraph of G on the same vertex subset is not isomorphic to F . The remaining edges of G that are not selected as strong, will be called *weak*. Consequently, whenever a subset S of the strong edges form a copy of F , there must be an additional strong or weak edge in G with endpoints among the endpoints of

edges in S . A formal definition of the problem is easier to give via spanning subgraphs. If two graphs H and F are isomorphic then we write $H \simeq F$, whereas if they are not isomorphic then we simply write $H \not\simeq F$. Given a graph G and a fixed graph F , we say that a (not necessarily induced) subgraph H of G *satisfies the F -closure* if, for every $S \subseteq V(H)$ with $H[S] \simeq F$, we have that $G[S] \not\simeq F$. In this case, the edges of H form exactly the set of strong edges of G .

STRONG F -CLOSURE

Input: A graph G and a non-negative integer k .

Task: Decide whether G has a spanning subgraph H that satisfies the F -closure, such that $|E(H)| \geq k$.

Based on this definition and the above explanation, the terms “marking an edge as weak (in G)” and “removing an edge (of G to obtain H)” are equivalent, and we will use them interchangeably. In connection to the previously mentioned problems, observe that STRONG P_3 -CLOSURE is exactly the MAXSTC problem.

MAXSTC ON SPLIT AND PROPER INTERVAL GRAPHS

In this chapter, we initiate the study of graph classes for which MAXSTC is solvable. We show that the problem admits a polynomial-time algorithm for two incomparable classes of graphs: proper interval graphs and trivially-perfect graphs. To complement our result, we show that the problem remains NP-complete on split graphs, and consequently also on chordal graphs. Thus, we contribute to define the first border between graph classes on which the problem is polynomially solvable and on which it remains NP-complete.

The results of this chapter have led to the following publications [88, 90]:

- **Maximizing the strong triadic closure in split graphs and proper interval graphs.** Athanasios L. Konstantinidis, and Charis Papadopoulos. *In 28th International Symposium on Algorithms and Computation (ISAAC 2017), Leibniz International Proceedings in Informatics (LIPIcs), pages 53:1–53:12, 2017.*
- **Maximizing the strong triadic closure in split graphs and proper interval graphs.** Athanasios L. Konstantinidis and Charis Papadopoulos. *Discrete Applied Mathematics* 285: 79-95, 2020.

3.1 Introduction

Predicting the behavior of a network is an important concept in the field of social networks [41]. Understanding the strength and nature of social relationships has found an increasing usefulness in the last years due to the explosive growth of social networks (see e.g., [5]). Towards such a direction the STRONG TRIADIC CLOSURE principle enables us to understand the structural properties of the underlying graph: it is not possible for two individuals to have a strong relationship with a common friend and not know each other [60]. Such a principle stipulates that if two people in a social

network have a “strong friend” in common, then there is an increased likelihood that they will become friends themselves at some point in the future. Satisfying the **STRONG TRIADIC CLOSURE** is to characterize the edges of the underlying graph into weak and strong such that any two vertices that have a strong neighbor in common are adjacent. Since users interact and actively engage in social networks by creating strong relationships, it is natural to consider the **MAXSTC** problem: maximize the number of strong edges that satisfy the **STRONG TRIADIC CLOSURE**. The problem has been shown to be NP-complete for general graphs while its dual problem of minimizing the number of weak edges admits a constant factor approximation ratio [118]. More recently, interesting variations of the **MAXSTC** problem have been considered which impose additional information than the network structure (e.g., through a collection of strong subgraphs) [112, 113].

In this work, we initiate the computational complexity study of the **MAXSTC** problem in important classes of graphs. If the input graph is a P_3 -free graph (i.e., a graph having no induced path on three vertices which is equivalent with a graph that consists of vertex-disjoint union of cliques) then there is a trivial solution by labeling strong all the edges. Such an observation might falsely lead into a graph modification problem, known as **CLUSTER DELETION** problem (see e.g., [11, 70]), in which we want to remove the minimum number of edges that correspond to the weak edges, such that the resulting graph does not contain a P_3 as an induced subgraph. More precisely the obvious reduction would consist in labeling the deleted edges in the instance of **CLUSTER DELETION** as weak, and the remaining ones as strong. However, this reduction fails to be correct due to the fact that the graph obtained by deleting the weak edges in an optimal solution of **MAXSTC** may contain an induced P_3 , so long as those three vertices induce a triangle in the original graph (prior to deleting the weak edges). We stress that there are examples on split graphs (Figure 3.1) and proper interval graphs (Figure 3.4) showing such a difference.

To the best of our knowledge, no previous results were known prior to our work when restricting the input graph for the **MAXSTC** problem. It is not difficult to see that for bipartite graphs the **MAXSTC** problem has a simple polynomial-time solution by considering a maximum matching that represent the strong edges [72]. In fact such an argument regarding the maximum matching generalizes to the larger class of triangle-free graphs. Also notice that for triangle-free graphs a set of edges is a maximum matching if and only if it is formed by a solution for the **CLUSTER DELETION** problem. It is well-known that a maximum matching of a graph corresponds to a maximum independent set of its line graph that represents the adjacencies between the edges [42]. As previously noted, for general graphs it is not necessarily the case that a maximum matching corresponds to the optimal solution for **MAXSTC**. Here we show a similar characterization for **MAXSTC** by considering the adjacencies between the

edges of a graph that participate in induced P_3 's. Such a characterization allows us to exhibit structural properties towards an optimal solution of MAXSTC.

Due to the nature of the P_3 existence that enforce the labeling of weak edges, there is an interesting connection to problems related to the *square root* of a graph; a graph H is a *square root* of a graph G and G is the *square* of H if two vertices are adjacent in G whenever they are at distance one or two in H . Any graph does not have a square root (for example consider a simple path), but every graph contains a subgraph that has a square root. Although it is NP-complete to determine if a given chordal graph has a square root [95], there are polynomial-time algorithms when the input is restricted to bipartite graphs [94], or proper interval graphs [95], or trivially-perfect graphs [106]. Among several square roots that a graph may have, one can choose the square root with the maximum or minimum number of edges [24, 97]. The relationship between MAXSTC and to that of determining square roots can be seen as follows. In the MAXSTC problem we are given a graph G and we want to select the maximum possible number of edges, at most one from each induced P_3 in G . Thus we need to find the largest subgraph (in terms of the number of its edges) H of G such that the square of H is a subgraph of G . However the known results related to square roots were concerned with deciding if the whole graph has a (maximum or minimum) square root and there are no such equivalent formulations related to the largest square root.

Our main motivation in this chapter is to understand the complexity of the problem on subclasses of chordal graphs, since the class of chordal graphs (i.e., graphs having no chordless cycle of length at least four) finds important applications in both theoretical and practical areas related to social networks [2, 83, 108]. More precisely two famous properties can be found in social networks. For most known social and biological networks their diameter, that is, the length of the longest shortest path between any two vertices of a graph, is known to be a small constant [76]. On the other hand it has been shown that the most prominent social network subgraphs are cliques, whereas highly infrequent induced subgraphs are cycles of length four [121]. Thus it is evident that subclasses of chordal graphs are close related to such networks, since they have rather small diameter (e.g., split graphs or trivially-perfect graphs) and are characterized by the absence of chordless cycles (e.g., proper interval graphs). Towards such a direction we show that MAXSTC is NP-complete on split graphs and consequently also on chordal graphs. On the positive side, we present the first polynomial-time algorithm for computing MAXSTC on proper interval graphs. Proper interval graphs, also known as unit interval graphs or indifference graphs, form a subclass of interval graphs and they are unrelated to split graphs [110]. By our result they form the first graph class, other than triangle-free graphs, for which MAXSTC is shown to be polynomial time solvable. In order to obtain our algorithm, we take advantage of their clique path (consecutive arrangement of maximal cliques) and apply a dynamic programming on sub-



Figure 3.1: A split graph G is shown to the left side. The right side depicts a solution for MAXSTC on G where the weak edges are exactly the edges of G that are not shown.

problems defined by passing the clique path in its natural ordering. Our structural proofs together with its polynomial solution on proper interval graphs can be seen as useful tools towards settling the complexity of MAXSTC on interval graphs. Furthermore by considering the characterization of the induced P_3 's mentioned earlier, we show that MAXSTC admits a simple polynomial-time solution on trivially-perfect graphs (i.e., graphs having no induced P_4 or C_4). Thus we contribute to define the first borderline between graph classes on which the problem is polynomially solvable and on which it remains NP-complete.

3.2 Preliminaries

Before we start with our results on MAXSTC, it is convenient here to remind the formal definition of MAXSTC and provide a useful observation.

MAXSTC

Input: A graph $G = (V, E)$.

Task: Does there exist a strong-weak labeling of $E(G)$ that satisfies the strong triadic closure and has at least k strong edges?

Let G be a strong-weak labeled graph. We denote by (E_S, E_W) the partition of $E(G)$ into strong edges E_S and weak edges E_W . The graph spanned by E_S is the graph $G \setminus E_W$. For a vertex $v \in V(G)$ we say that the *strong neighbors* of v are the other endpoints of the strong edges incident to v . We denote by $N_S(v) \subseteq N(v)$ the strong neighbors of v . Similarly we say that a vertex u is *strongly adjacent* to v if u is adjacent to v and the edge $\{u, v\}$ is strong.

Observation 3.2.1. *Let G be a strong-weak labeled graph with edge partition (E_S, E_W) . Then G satisfies the strong triadic closure if and only if for every P_3 in $G \setminus E_W$, the vertices of P_3 induce a K_3 in G .*

Proof. Observe that $G \setminus E_W$ is the graph spanned by the strong edges. If for two strong edges $\{u, v\}$ and $\{v, w\}$, $\{u, w\} \notin E(G \setminus E_W)$ then $\{u, w\}$ is an edge in G and, thus, u, v, w induce a K_3 in G . On the other hand notice that any two strong edges of $G \setminus E_W$ are either non-adjacent or share a common vertex. If they share a common vertex then the vertices must induce a K_3 in G , implying that (E_S, E_W) satisfies the strong triadic closure. \square

Therefore in the MAXSTC problem we want to minimize the number of the removal (weak) edges E_W from G such that every three vertices that induce a P_3 in $G \setminus E_W$ form a clique in G . Then it is not difficult to see that G satisfies the strong triadic closure if and only if for every vertex v , $N_S[v]$ induces a clique in G .

3.2.1 Basic Results

It is interesting to settle the complexity of MAXSTC on graphs of small structural parameter, such as treewidth. Towards such a direction observe that every problem expressible in monadic second order logic of second type (MSO₂) with quantification over vertex sets and edge sets can be solved in linear time for graphs of bounded treewidth [28]. Indeed, MAXSTC can be formulated in MSO₂: (i) the edges are partitioned into two subsets E_S, E_W (i.e., a strong-weak labeling), (ii) the endpoints of every path of length two spanned by the edges of E_S have an edge in G (i.e., satisfy the strong triadic closure), and (iii) $|E_S|$ is as large as possible. In particular, the following two expressions correspond to properties (i) and (ii), respectively:

- $\forall x \forall y \text{ adj}(x, y) \rightarrow (E_W(x, y) \vee E_S(x, y)) \wedge (\neg E_W(x, y) \vee \neg E_S(x, y))$
- $\forall x \forall y \forall z (E_S(x, y) \wedge E_S(y, z)) \rightarrow \text{adj}(x, z)$

Therefore there is a linear-time algorithm for MAXSTC on graphs of bounded treewidth [28]. Notice that a similar observation holds for the CLUSTER DELETION problem in which the only difference is that the endpoints considered in case (ii) must have an edge of E_S (i.e., the strong edges span a P_3 -free graph where the strong edges reflect the edges of the graph resulting after removing the weak edges). We remark that the graphs considered hereafter, such as split graphs, proper interval graphs, and trivially perfect graphs do not have bounded treewidth.

3.2.2 The line-incompatibility graph and twin vertices

We next provide some interesting characterizations related to the MAXSTC problem that might be of independent interest with respect to our main results concerning

split graphs and proper interval graphs. First, we give an equivalent transformation of MAXSTC related to the independent set of an auxiliary graph and, then, we show how to contract twin vertices. As a consequence of the former characterization, we show that its application leads to a polynomial solution for the class of trivially-perfect graphs.

Instead of maximizing the strong edges of the original graph G , we will look at the maximum independent set of the following graph that we call the *line-incompatibility graph* \widehat{G} of G : for every edge of G there is a node in \widehat{G} and two nodes of \widehat{G} are adjacent if and only if the vertices of the corresponding edges induce a P_3 in G . In a different context the notion of line-incompatibility has already been considered under the term *Gallai graph* in [96] or as an auxiliary graph in [24, 118]. Note that the line-incompatibility graph of G is a subgraph of the line graph of G . Moreover, observe that for a graph G , its line graph and its line-incompatibility graph coincide if and only if G is a triangle-free graph.

Proposition 3.2.2. *A subset S of edges $E(G)$ is an optimal solution for MAXSTC of G if and only if S is a maximum independent set of \widehat{G} .*

Proof. By Observation 3.2.1 for every P_3 in G at least one of its two edges must be labeled weak in S . This means that these two edges are adjacent in \widehat{G} and they cannot belong to an independent set of \widehat{G} . On the other hand, by construction two nodes of \widehat{G} are adjacent if and only if there is a P_3 in G . Thus the nodes of an independent set of \widehat{G} can be labeled strong in G satisfying the strong triadic closure. \square

Therefore, for the optimal solution of G one may look at a solution for a maximum independent set of \widehat{G} . Also note that for any graph G , computing a maximum independent set of \widehat{G} is an NP-complete problem [96]. As a byproduct, if we are interested in minimizing the number of weak edges then we ask for the minimum vertex cover of \widehat{G} . To distinguish the vertices of \widehat{G} with those of G , we refer to the former as nodes and to the latter as vertices. For an edge $\{u, v\}$ of G we denote by uv the corresponding node of \widehat{G} . Figure 3.2 shows a graph and its line-incompatibility graph.

Due to Proposition 3.2.2 it is natural to characterize the graphs for which their line-incompatibility graph is perfect. Such a characterization will lead to polynomial cases of MAXSTC, since the problem of finding a maximum independent set of perfect graphs admits a polynomial solution [61]. A typical example is the class of bipartite graphs for which their line graph coincides with their line-incompatibility graph and it is known that the line graph of a bipartite graph is perfect (see for e.g., [12]). As we show next, another paradigm of this type is the class of trivially-perfect graphs.

We remind here a graph G is called *trivially-perfect* (also known as *quasi-threshold*) if for each induced subgraph H of G , the number of maximal cliques of H is equal to

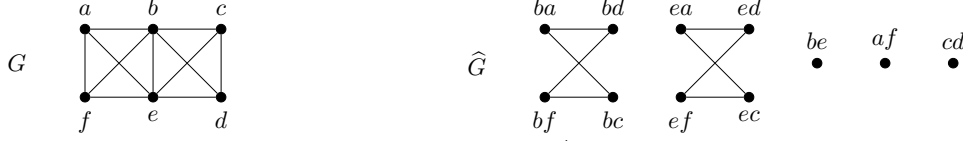


Figure 3.2: A graph G on the left side and its line-incompatibility graph \widehat{G} on the right side. Notice that if the nodes of an independent set in \widehat{G} are labeled strong and the rest are labeled weak then such a strong-weak labeling satisfies the strong triadic closure.

the maximum size of an independent set of H . It is known that the class of trivially-perfect graphs coincides with the class of (P_4, C_4) -free graphs, that is every trivially-perfect graph has no induced P_4 or C_4 [57]. Notice that the graph given in Figure 3.2 is trivially-perfect. A *cograph* is a graph without an induced P_4 , that is a cograph is a P_4 -free graph. Hence trivially-perfect graphs form a subclass of cographs.

Theorem 3.2.3. *The line-incompatibility graph of a trivially-perfect graph is a cograph.*

Proof. Let G be a trivially-perfect graph, that is G is a (P_4, C_4) -free graph. We will show that the line-incompatibility graph \widehat{G} of G is a P_4 -free graph. Consider any P_3 in \widehat{G} . Due to the construction of \widehat{G} , the P_3 has one of the following forms: (i) v_1v_2, v_2v_3, v_3v_4 or (ii) v_1x, v_2x, v_3x . We prove that the P_3 has the second form because G has no induced P_4 or C_4 . If (i) applies then $\{v_1, v_3\}, \{v_2, v_4\} \notin E(G)$ and $\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\} \in E(G)$ which implies that $v_4 \neq v_1$. Thus G contains a P_4 or a C_4 depending on whether there is the edge $\{v_1, v_4\}$ in G . Hence every P_3 in \widehat{G} has the form v_1x, v_2x, v_3x where v_1, v_2, v_3, x are distinct vertices of G . Now, assume for contradiction that \widehat{G} contains a P_4 . Then the P_4 is of the form v_1x, v_2x, v_3x, v_4x because it contains two induced P_3 's. The structure of the P_4 implies that $\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\} \notin E(G)$ and $\{v_1, v_3\}, \{v_2, v_4\}, \{v_4, v_1\} \in E(G)$. This however shows that the vertices v_3, v_1, v_4, v_2 induce a P_4 in G leading to a contradiction that G is a (P_4, C_4) -free graph. Therefore \widehat{G} is a P_4 -free graph. \square

By Theorem 3.2.3 and the fact that a maximum independent set of a cograph can be computed in linear time [26], MAXSTC can be solved in polynomial time on trivially-perfect graphs. We would like to note that the line-incompatibility graph of a cograph or a split graph or a proper interval graph is not necessarily a perfect graph. It is also interesting to note that the line-incompatibility graph of the complement of a bipartite graph (co-bipartite graph) is bipartite.

A natural contraction for several graph problems is to group twin vertices since they play the same role on the given graph. For the CLUSTER DELETION problem, such an approach has already been used [11, 109]. With the next result, we show that this is

indeed the case for the MaxSTC problem. We denote by $I_{\widehat{G}}$ a maximum independent set of \widehat{G} .

Lemma 3.2.4. *Let x and y be twin vertices of a graph G . Then there is an optimal solution $I_{\widehat{G}}$ such that $xy \in I_{\widehat{G}}$ and for every vertex $u \in N(x)$, $xu \in I_{\widehat{G}}$ if and only if $yu \in I_{\widehat{G}}$.*

Proof. First, we show that xy is an isolated node in \widehat{G} . If xy is adjacent to xu then y is non-adjacent to u in G which contradicts the fact that x and y are twins. Thus xy is an isolated node in \widehat{G} which implies $xy \in I_{\widehat{G}}$. For the second argument observe that for every vertex $u \in N(x)$, xu and yu are non-adjacent in $I_{\widehat{G}}$. Let $u \in N(x)$. Then notice that $u \in N(y)$. This means that if $xu \in I_{\widehat{G}}$ (resp., $yu \in I_{\widehat{G}}$) then yu (resp., xu) is a node of \widehat{G} . We define the following sets of nodes in \widehat{G} :

- Let A_x be the set of nodes xa such that $xa \in I_{\widehat{G}}$ and $ya \notin I_{\widehat{G}}$ and let A_y be the set of nodes ya such that $xa \in A_x$.
- Let B_y be the set of nodes yb such that $yb \in I_{\widehat{G}}$ and $xb \notin I_{\widehat{G}}$ and let B_x be the set of nodes xb such that $yb \in B_y$.

It is clear that $A_x \subseteq I_{\widehat{G}}$, $B_y \subseteq I_{\widehat{G}}$, and $A_x \cap B_y = \emptyset$. Also note that $|A_x| = |A_y|$ and $|B_y| = |B_x|$, since $N[x] = N[y]$.

Let $I_{xy} = I_{\widehat{G}} \setminus (A_x \cup B_y)$ so that $I_{\widehat{G}} = A_x \cup B_y \cup I_{xy}$. We show that every node of A_y is non-adjacent to any node of $I_{\widehat{G}} \setminus B_y$. Let ya be a node of A_y . If there is a node $az \in I_{\widehat{G}} \setminus B_y$ that is adjacent to ya then z and y are non-adjacent in G which implies that z and x are non-adjacent in G . This however leads to a contradiction because $xa, az \in I_{\widehat{G}}$ and xa is adjacent to az in \widehat{G} . If there is a node $yb \in I_{\widehat{G}}$ that is adjacent to ya then a is non-adjacent to b in G so that xa is also adjacent to xb in \widehat{G} . This means that $xb \notin I_{\widehat{G}}$ implying that $yb \in B_y$. Thus every node of A_y is non-adjacent to any node of $I_{\widehat{G}} \setminus B_y$ and with completely symmetric arguments, every node of B_x is non-adjacent to any node of $I_{\widehat{G}} \setminus A_x$. Hence both sets $I_1 = A_x \cup A_y \cup I_{xy}$ and $I_2 = B_x \cup B_y \cup I_{xy}$ form independent sets in \widehat{G} . By the facts that $|A_x| = |A_y|$ and $|B_y| = |B_x|$ we have $|I_1| \geq |I_{\widehat{G}}|$ whenever $|A_x| \geq |B_y|$ and $|I_2| \geq |I_{\widehat{G}}|$ whenever $|A_x| < |B_y|$. Therefore we can safely replace one of the sets A_x or B_y by B_x or A_y and obtain the solutions I_2 or I_1 , respectively. Now observe that in both solutions I_1 and I_2 we have $xu \in I_i$ if and only if $yu \in I_i$, for $i \in \{1, 2\}$, and this completes the proof. \square

3.3 MaxSTC on split graphs

Here we provide an NP-hardness result for MaxSTC on split graphs. In fact, our main result stated in Theorem 3.3.4 implies that computing a maximum independent

set of the line-incompatibility graph of a split graph is NP-hard, improving a result in [96] which states that computing a maximum independent set of the Gallai graph of a graph is NP-hard. A graph $G = (V, E)$ is a *split graph* if V can be partitioned into a clique C and an independent set I , where (C, I) is called a *split partition* of G . Split graphs form a subclass of the larger and widely known graph class of *chordal graphs*. It is known that split graphs are self-complementary, that is, the complement of a split graph remains a split graph. First we need the following two results.

Lemma 3.3.1. *Let $G = (V, E)$ be a split graph with a split partition (C, I) . Let E_S be the set of strong edges in a solution for MAXSTC on G and let I_W be the vertices of I that are incident to at least one edge of E_S . Furthermore, for every vertex w_i of I we denote by A_i its strong neighbors in C and we denote by B_i the set of vertices in C that are non-adjacent to w_i . Then,*

$$|E_S| \leq |E(C)| + \sum_{w_i \in I_W} |A_i| \left(1 - \left\lfloor \frac{|B_i|}{2} \right\rfloor\right).$$

Proof. Let w_i be a vertex of I . For the edges of the clique, there are $|A_i||B_i|$ weak edges due to the strong triadic closure. Moreover any vertex w_j of $I \setminus \{w_i\}$ cannot have a strong neighbor in A_i . This means that $A_i \cap A_j = \emptyset$. Notice, however, that both sets $B_i \cap B_j$ and $A_i \cap B_j$ are not necessarily empty.

Observe that I_W contains the vertices of I that are incident to at least one strong edge. Let $E(A, B)$ be the set of weak edges that have one endpoint in A_i and the other endpoint in B_i , for every $1 \leq i \leq |I_W|$. We show that $2|E(A, B)| \geq \sum_{w_i \in I_W} |A_i||B_i|$. Let $\{a, b\} \in E(A, B)$ such that $a \in A_i$ and $b \in B_i$. Assume that there is a pair A_j, B_j such that $\{a, b\}$ is an edge between A_j and B_j , for $j \neq i$. Then a cannot belong to A_j since $A_i \cap A_j = \emptyset$. Thus, $a \in B_j$ and $b \in A_j$. Therefore, for every edge $\{a, b\} \in E(A, B)$ there are at most two pairs (A_i, B_i) and (A_j, B_j) for which $a \in A_i \cap B_j$ and $b \in B_i \cap A_j$. This means that every edge of $E(A, B)$ is counted at most twice in $\sum_{w_i \in I_W} |A_i||B_i|$.

For any two edges $\{u, v\}, \{v, z\} \in E(C) \setminus E(A, B)$, observe that they satisfy the strong triadic closure since there is the edge $\{u, z\}$ in G . Thus, the strong edges of the clique are exactly the set of edges $E(C) \setminus E(A, B)$. In total by counting the number of strong edges between the independent set and the clique, we have $|E_S| = |E(C) \setminus E(A, B)| + \sum_{w_i \in I_W} |A_i|$. Since $2|E(A, B)| \geq \sum_{w_i \in I_W} |A_i||B_i|$, we get the stated formula. \square

Lemma 3.3.2. *Let $G = (V, E)$ be a split graph with a split partition (C, I) . Let E_S be the set of strong edges in an optimal solution for MAXSTC on G and let I_W be the vertices of I that are incident to at least one edge of E_S .*

1. *If every vertex of I_W misses at least three vertices of C in G , then $E_S = E(C)$.*

2. If every vertex of I_W misses exactly one vertex of C in G , then $|E_S| \leq |E(C)| + \lfloor \frac{|I_W|}{2} \rfloor$.

Proof. We apply Lemma 3.3.1 in each case. The first claim of the lemma holds because $|B_i| \geq 3$ so that $I_W = \emptyset$. For the second claim we show that for every vertex of I_W , $|A_i| = 1$. Let $w_i \in I_W$ such that $|A_i| \geq 2$ and let $B_i = \{b_i\}$. Recall that no other vertex of I_W has strong neighbors in A_i . Also note that there is at most one vertex w_j in I_W that has b_i as a strong neighbor. If such a vertex w_j exists and for the vertex b_j of the clique that misses w_j it holds $b_j \in A_i$, then we let $v = b_j$; otherwise we choose v as an arbitrary vertex of A_i . Observe that no vertex of $I \setminus \{w_i\}$ has a strong neighbor in $A_i \setminus \{v\}$ and only $w_j \in I_W$ is strongly adjacent to b_i . Then we label weak the $|A_i| - 1$ edges between w_i and the vertices of $A_i \setminus \{v\}$ and we label strong the $|A_i| - 1$ edges between b_i and the vertices of $A_i \setminus \{v\}$. Making strong the edges between b_i and the vertices of $A_i \setminus \{v\}$ does not violate the strong triadic closure since every vertex of $C \cup \{w_j\}$ is adjacent to every vertex of $A_i \setminus \{v\}$. Therefore for every vertex $w_i \in I_W$, $|A_i| = 1$ and by substituting $|B_i| = 1$ in the formula for $|E_S|$ we get the claimed bound. \square

In order to give the reduction, we introduce the following problem that we call *maximum disjoint non-neighborhood* (MAXDISJOINTNN):

MAXDISJOINTNN —

- Input:* A split graph (C, I) where every vertex of I misses three vertices from C and a nonnegative integer k .
Task: Find a subset S_I of I such that the non-neighborhoods of the vertices of S_I are pairwise disjoint and $|S_I| \geq k$.

The polynomial-time reduction to MAXDISJOINTNN is given from the classical NP-complete problem 3-SET PACKING [80]: given a universe \mathcal{U} of n elements, a family \mathcal{F} of triplets of \mathcal{U} , and an integer k , the problem asks for a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ with $|\mathcal{F}'| \geq k$ such that all triplets of \mathcal{F}' are pairwise disjoint.

Corollary 3.3.3. MAXDISJOINTNN is NP-complete.

Proof. Given a split graph $G = (C, I)$ and $S_I \subseteq I$, checking whether S_I is a solution for MAXDISJOINTNN amounts to checking whether every pair of vertices of S_I have common neighborhood. As this can be done in polynomial time the problem is in NP. We will give a polynomial-time reduction to MAXDISJOINTNN from the classical NP-complete problem 3-SET PACKING [80]: given a universe \mathcal{U} of n elements, a family \mathcal{F} of triplets of \mathcal{U} , and an integer k , the problem asks for a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ with $|\mathcal{F}'| \geq k$ such that all triplets of \mathcal{F}' are pairwise disjoint.

Let $(\mathcal{U}, \mathcal{F}, k)$ be an instance of the 3-SET PACKING. We construct a split graph $G = (C, I)$ as follows. The clique of G is formed by the n elements of \mathcal{U} . For every triplet F_i of \mathcal{F} we add a vertex v_i in I that is adjacent to every vertex of C except the three vertices that correspond to the triplet F_i . Thus every vertex of I misses exactly three vertices from C and sees the rest of C . Now it is not difficult to see that there is a solution \mathcal{F}' for 3-SET PACKING $(\mathcal{U}, \mathcal{F}, k)$ of size at least k if and only if there is a solution S_I for MAXDISJOINTNN (G, k) of size at least k . For every pair (F_i, F_j) of \mathcal{F}' we know that $F_i \cap F_j = \emptyset$ which implies that the vertices v_i and v_j have disjoint non-neighborhood since F_i corresponds to the non-neighborhood of v_i . By the one-to-one mapping between the sets of \mathcal{F} and the vertices of I , every set F_i belongs to \mathcal{F}' if and only if v_i belongs to S_I . \square

Now we turn to our original problem MAXSTC. The decision version of MAXSTC takes as input a graph G and an integer k and asks whether there is strong-weak labeling of the edges of G that satisfies the strong triadic closure with at least k strong edges.

Theorem 3.3.4. *The decision version of MAXSTC is NP-complete on split graphs.*

Proof. Given a strong-weak labeling (E_S, E_W) of a split graph $G = (C, I)$, checking whether (E_S, E_W) satisfies the strong triadic closure amounts to check in $G \setminus E_W$ whether there is a non-edge in G between the endvertices of every P_3 according to Observation 3.2.1. Thus by listing all P_3 's of $G \setminus E_W$ the problem belongs to NP. Next we give a polynomial-time reduction to MAXSTC from the MAXDISJOINTNN problem on split graphs which is NP-complete by Corollary 3.3.3. Let (G, k) be an instance of MAXDISJOINTNN where $G = (C, I)$ is a split graph such that every vertex of the independent set I misses exactly three vertices from the clique C . For a vertex $w_i \in I$, we denote by B_i the set of the three vertices in C that are non-adjacent to w_i . Let $n = |C|$. We extend G and construct another split graph G' as follows (see Figure 3.3):

- We add n vertices y_1, \dots, y_n in the clique that constitute the set C_Y .
- We add n vertices x_1, \dots, x_n in the independent set that constitute the set I_X .
- For every $1 \leq i \leq n$, y_i is adjacent to all vertices of $(C \cup C_Y \cup I \cup I_X) \setminus \{x_i\}$.
- For every $1 \leq i \leq n$, x_i is adjacent to all vertices of $(C \cup C_Y) \setminus \{y_i\}$.

Thus w_i misses only the vertices of B_i from the clique. By construction it is clear that G' is a split graph with a split partition $(C \cup C_Y, I \cup I_X)$. Notice that the clique $C \cup C_Y$ has $2n$ vertices and $G = G'[I \cup C]$.

We next prove that G has a solution for MAXDISJOINTNN of size at least k if and only if G' has a strong triadic closure with at least $n(2n-1) + \lfloor \frac{n}{2} \rfloor + \lceil \frac{k}{2} \rceil$ strong edges.

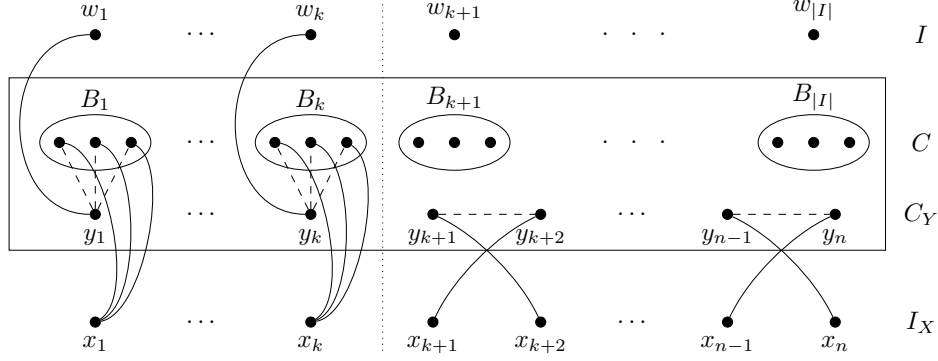


Figure 3.3: The split graph $(C \cup C_Y, I \cup I_X)$ given in the polynomial-time reduction. Every vertex w_i misses the vertices of B_i and sees the vertices of $(C \cup C_Y) \setminus B_i$. Every vertex x_i misses y_i and sees the vertices of $(C \cup C_Y) \setminus \{y_i\}$. The sets B_1, \dots, B_k are pairwise disjoint whereas for every set B_j , $k < j \leq |I|$, there is a set B_i , $1 \leq i \leq k$, such that $B_i \cap B_j \neq \emptyset$. The drawn edges correspond to the strong edges between the independent set and the clique, and the dashed edges are the only weak edges in the clique $C \cup C_Y$.

(\Rightarrow) Assume that $\{w_1, \dots, w_k\} \subseteq I$ is a solution for MAXDISJOINTNN on G of size at least k . Since the sets B_1, \dots, B_k are pairwise disjoint, there are k distinct vertices y_1, \dots, y_k in C_Y such that $k \leq n$. We will give a strong-weak labeling for the edges of G' that fulfills the strong triadic closure and has at least the claimed number of strong edges. For simplicity, we describe only the strong edges; the edges of G' that are not given are all labeled weak. We label the edges incident to each vertex w_i, y_i, x_i and the three vertices of each set B_i , for $1 \leq i \leq k$ as follows:

- The edges of the form $\{y_i, v\}$ are labeled strong if $v \in (C \cup C_Y) \setminus B_i$ or $v = w_i$.
- The edges between x_i and the three vertices of B_i are labeled strong.

Next we label the edges incident to the rest of the vertices. No edge incident to a vertex of $I \setminus \{w_1, \dots, w_k\}$ is labeled strong. For every vertex $u \in C \setminus (B_1 \cup \dots \cup B_k)$ we label the edge $\{u, v\}$ strong if $v \in (C \cup C_Y)$. Let $C'_Y = \{y_{k+1}, \dots, y_n\}$ and let $I'_X = \{x_{k+1}, \dots, x_n\}$. Recall that every vertex x_{k+j} is adjacent to every vertex of $C'_Y \setminus \{y_{k+j}\}$. Let $\ell = \lfloor \frac{n-k}{2} \rfloor$. Let $M = \{e_1, \dots, e_\ell\}$ be a maximal set of pairwise non-adjacent edges in $G'[C'_Y]$ where $e_j = \{y_{k+2j-1}, y_{k+2j}\}$, for $j \in \{1, \dots, \ell\}$; note that M is a maximal matching of $G'[C'_Y]$. For every vertex $y \in C'_Y$, we label the edge $\{y, v\}$ strong if $v \in (C \cup C_Y) \setminus \{y'\}$ such that $\{y, y'\} \in M$. Moreover, for $j \in \{1, \dots, \ell\}$, the edges $\{x_{k+2j-1}, y_{k+2j}\}$ and $\{x_{k+2j}, y_{k+2j-1}\}$ are labeled strong. Note that if $n - k$ is odd then

no edge incident to the unique vertex y_n belongs to M and all edges between y_n and the vertices of $C \cup C_Y$ are labeled strong; in such a case also note that no edge incident to x_n is strong.

Let us show that such a labeling fulfills the strong triadic closure. Any labeling for the edges inside $G'[C \cup C_Y]$ is satisfied since $G'[C \cup C_Y]$ is a clique. Also note that there are no two adjacent strong edges that have a common endpoint in the clique $C \cup C_Y$ and the two other endpoints in the independent set $I \cup I_X$. If there are two strong edges incident to the same vertex v of the independent set then $v \in \{x_1, \dots, x_k\}$ and $N_S[v] = B_i$ which is a clique. Assume that there are two adjacent strong edges $\{u, v\}$ and $\{v, z\}$ such that $u \in I \cup I_X$, and $v, z \in C \cup C_Y$.

- If $u \in \{w_1, \dots, w_k\}$ then $\{u, z\} \in E(G')$ since every w_i misses only the vertices of B_i .
- If $u \in \{x_1, \dots, x_k\}$ then $v \in B_i$ and $\{u, z\} \in E(G')$ since every vertex x_i misses only y_i .
- If $u \in I_X \setminus \{x_1, \dots, x_k\}$ then the strong neighbors of v in $C \cup C_Y$ are adjacent to u in G' since for the only non-neighbor of u in $C \cup C_Y$ there is a weak edge incident to v .

Recall that there is no strong edge incident to the vertices of $I \setminus \{w_1, \dots, w_k\}$. Therefore the given strong-weak labeling fulfills the strong triadic closure.

Observe that the number of vertices in $C \cup C_Y$ is $2n$. There are exactly $3k + \ell$ weak edges in $G'[C \cup C_Y]$. Thus the number of strong edges in $G'[C \cup C_Y]$ is $n(2n - 1) - 3k - \ell$. There are k strong edges incident to $\{w_1, \dots, w_k\}$, $3k$ strong edges incident to $\{x_1, \dots, x_k\}$, and 2ℓ strong edges incident to $I_X \setminus \{x_1, \dots, x_k\}$. Thus the total number of strong edges is $n(2n - 1) - 3k - \ell + k + 3k + 2\ell = n(2n - 1) + \ell + k$ and by substituting $\ell = \lfloor \frac{n-k}{2} \rfloor$ we get the claimed bound.

(\Leftarrow) For the opposite direction, assume that G' has a strong triadic closure with at least $n(2n - 1) + \lfloor \frac{n}{2} \rfloor + \lceil \frac{k}{2} \rceil$ strong edges. Let E_S be the set of strong edges in such a strong-weak labeling. Observe that the number of edges in $G'[C \cup C_Y]$ is $n(2n - 1)$ which implies that E_S contains edges between the independent set $I \cup I_X$ and the clique $C \cup C_Y$. If no vertex of I_X is incident to an edge of E_S then the first statement of Lemma 3.3.2 implies that $|E_S| = |E(C \cup C_Y)| = n(2n - 1)$. And if no vertex of I is incident to an edge of E_S then the second statement of Lemma 3.3.2 shows that $|E_S| \leq |E(C \cup C_Y)| + \lfloor \frac{n}{2} \rfloor$. Therefore, E_S contains edges that are incident to a vertex of I and edges that are incident to a vertex of I_X .

In the graph spanned by E_S we denote by S_W the set of vertices of I that have strong neighbors in $C \cup C_Y$. A *nice solution* is a strong-weak labeling that satisfies the strong

triadic closure with a set of strong edges E'_S such that $|E_S| = |E'_S|$. We will show that the non-neighborhoods of the vertices of S_W in $C \cup C_Y$ are disjoint in G' and, since G is an induced subgraph of G' , their non-neighborhoods are also disjoint in G .

Claim 3.3.5. *There exists a nice solution such that for every $w_i \in S_W$, (i) $N_S(w_i) \subseteq C_Y$ holds and (ii) there is a unique vertex $x \in I_X$ with $N_S(x) = B_i$.*

Proof: Let w_i be a vertex of S_W . We first show that $N_S(w_i) \subseteq C_Y$. Let W_i be the strong neighbors of w_i in C and let Y_i be the strong neighbors of w_i in C_Y . Observe that no other vertex of S_W has a strong neighbor in $W_i \cup Y_i$. Furthermore, notice that there are $(|W_i| + |Y_i|)|B_i|$ weak edges since w_i is non-adjacent to the vertices of B_i . We show that $W_i = \emptyset$, for every vertex $w_i \in S_W$. For all vertices w_i for which $W_i \neq \emptyset$ we replace in E_S the strong edges between w_i and the vertices of W_i by the edges between the vertices of B_i and W_i . Notice that making strong the edges between the vertices of B_i and W_i does not violate the strong triadic closure since no vertex from S_W has a strong neighbor in B_i and every vertex of I_X is adjacent to all the vertices of W_i . Let $E(W, B)$ be the set of edges that have one endpoint in W_i and the other endpoint in B_i , for every $w_i \in S_W$. Notice that the difference between the two described solutions is $|E(W, B)| - \sum |W_i|$. By Lemma 3.3.1 and $|B_i| = 3$, we know that $|E(W, B)| \geq 3/2 \sum |W_i|$. Thus, such a replacement is safe for the number of edges of E_S and every vertex $w_i \in S_W$ has strong neighbors only in C_Y .

Let X_i be the set of vertices of I_X that have at least one non-neighbor in Y_i . By construction every vertex of Y_i is non-adjacent to exactly one vertex of I_X , and thus $|X_i| = |Y_i|$. Since w_i has strong neighbors in Y_i , every edge between X_i and Y_i is weak. By the previous argument every vertex of S_W has strong neighbors only in C_Y so that $N_S(B_i) \cap I = \emptyset$. Also notice that no two vertices of the independent set have a common strong neighbor in the clique, which means that there are at most $|B_i|$ strong neighbors between the vertices of B_i and I_X . Choose an arbitrary vertex $x \in X_i$. We replace all strong edges in E_S between B_i and I_X by $|B_i|$ strong edges between x and the vertices of B_i . Notice that such a replacement is safe since the unique non-neighbor of x belongs to Y_i and there are weak edges already in the solution between B_i and Y_i because of the strong edges between w_i and Y_i . Thus $B_i \subseteq N_S(x)$. We focus on the edges between the vertices of $(C \cup C_Y) \setminus (B_i \cup Y_i)$ and x . If a vertex u of X_i has a strong neighbor u in $(C \cup C_Y) \setminus B_i$ then the edge $\{u, y\}$ is weak where $y \in Y_i$ is the unique non-neighbor of x . Also notice that $N_S(u) \cap (I \cup I_X) = \{x\}$, $N_S(y) \cap (I \cup I_X) = \{w_i\}$, and w_i is adjacent to u . Then we can safely replace the strong edge $\{x, u\}$ by the edge $\{u, y\}$ and keep the same size of E_S . Hence $N_S(x) = B_i$. \diamond

Claim 3.3.6. *There exists a nice solution such that for every $w_i \in S_W$, $N_S(w_i) = \{y\}$ where $y \in C_Y$ is the non-neighbor of x with $N_S(x) = B_i$.*

Proof: Let $Y_i = N_S(w_i)$. By Claim 3.3.5 we know that $Y_i \subseteq C_Y$ and there exists $x \in I_X$ such that $N_S(x) = B_i$. Both w_i and x are vertices of the independent set and, thus, no other vertex of $I \cup I_X$ has strong neighbors in $B_i \cup Y_i$. This means that if we remove w_i from S_W by making weak the edges incident to w_i and the vertices of Y_i then the edges between the vertices of B_i and $Y_i \setminus \{y\}$ are safely turned into strong. Let E'_S be the set of strong edges in a nice solution such that all edges incident to w_i are weak. Then $|E_S| - |E'_S| = |Y_i| + |B_i| - |Y_i||B_i|$ and $|E_S| > |E'_S|$ only if $|Y_i| = 1$ because $|B_i| > 1$. Thus $N_S(w_i)$ contains exactly one vertex $y \in C_Y$. \diamond

We claim that for every pair of vertices $w_i, w_j \in S_W$, $B_i \cap B_j = \emptyset$. Assume for contradiction that $B_i \cap B_j \neq \emptyset$. Applying Claim 3.3.5 for w_i shows that there exists $x \in I_X$ that has strong neighbors in every vertex of $B_i \cap B_j$. With a similar argument for w_j we deduce that there exists $x' \in I_X$ that has strong neighbors in every vertex of $B_i \cap B_j$. If $x \neq x'$ then a vertex from $B_i \cap B_j$ has two distinct strong neighbors in I_X which is not possible due to the strong triadic closure. Thus $x = x'$. Claim 3.3.6 implies that the unique non-neighbor y of x is strongly adjacent to both w_i and w_j . This however violates the strong triadic closure for the edges of E_S since w_i, w_j are non-adjacent and we reach a contradiction. Thus $B_i \cap B_j = \emptyset$. This means that the number of edges in E_S is at least $n(2n - 1) + \lfloor \frac{n}{2} \rfloor + \lceil \frac{|S_W|}{2} \rceil$ which is maximized for $k = |S_W|$. Therefore there are at least $|S_W|$ vertices from the independent set which form a solution for MAXDISJOINTNN on G , since G is an induced subgraph of G' . \square

3.4 Computing MaxSTC on proper interval graphs

Due to the NP-completeness on split graphs given in Theorem 3.3.4, it is natural to consider interval graphs that form another well-studied subclass of chordal graphs. However besides few observations of this section that may be applied for interval graphs, we found several unresolved technical cases. Moreover we would like to note that, to the best of our knowledge, the complexity of the close-related CLUSTER DELETION problem still remains unresolved on interval graphs [11]. Therefore we further restrict the input to the class of proper interval graphs that form a proper subclass of interval graphs. Our polynomial solution for MaxSTC on proper interval graphs can be seen as a first step towards determining its complexity on interval graphs.

A graph is a *proper interval graph* if there is a bijection between its vertices and a family of closed intervals of the real line such that two vertices are adjacent if and only if the two corresponding intervals overlap and no interval is properly contained in another interval. A vertex ordering σ is a linear arrangement $\sigma = \langle v_1, \dots, v_n \rangle$ of the vertices of G . For a vertex pair x, y we write $x \preceq y$ if $x = v_i$ and $y = v_j$ for some indices $i \leq j$; if $x \neq y$ which implies $i < j$ then we write $x \prec y$. We extend \prec to support sets

of vertices, as follows: for two sets of vertices A and B we write $A \prec B$ if for any $a \in A$ and $b \in B$, $a \prec b$ holds. The first position in σ will be referred to as the *left end* of σ , and the last position as the *right end*. We will use the expressions *to the left of*, *to the right of*, *leftmost*, and *rightmost* accordingly.

A vertex ordering σ for G is called a *proper interval ordering* if for every vertex triple x, y, z of G with $x \prec y \prec z$, $\{x, z\} \in E(G)$ implies $\{x, y\}, \{y, z\} \in E(G)$. Proper interval graphs are characterized as the graphs that admit such orderings, that is, a graph is a proper interval graph if and only if it has a proper interval ordering [102]. We only consider this vertex ordering characterization for proper interval graphs. Moreover it can be decided in linear time whether a given graph is a proper interval graph, and if so, a proper interval ordering can be generated in linear time [102]. It is clear that a vertex ordering σ for G is a proper interval ordering if and only if the reverse of σ is a proper interval ordering. A connected proper interval graph without twin vertices has a unique proper interval ordering σ up to reversal [32, 73]. Figure 3.4 shows a proper interval graph with its proper interval ordering.

Before reaching the details of our algorithm for proper interval graphs, let us highlight the difference between the optimal solution for MAXSTC and the optimal solution for the CLUSTER DELETION. As already explained in the Introduction a solution for CLUSTER DELETION satisfies the strong triadic closure, though the converse is not necessarily true. In fact such an observation carries out for the class of proper interval graphs as shown in the example given in Figure 3.4. For the CLUSTER DELETION problem twin vertices can be grouped together following a similar characterization with Lemma 3.2.4, as proved in [11]. This means that the P_3 -free graph depicted in the lower part of Figure 3.4 that is obtained by removing its weak edges (i.e., the dashed drawn lines) is an optimal solution for CLUSTER DELETION problem on the given proper interval graph. Therefore when restricted to proper interval graphs the optimal solution for CLUSTER DELETION does not necessarily imply an optimal solution for MAXSTC.

Let G be a proper interval graph and let σ be a proper interval ordering for G . Hereafter we denote by E_S the set of strong edges in an optimal solution for MAXSTC on G . Moreover a labeled edge $\{x, y\}$ (weak or strong) is simply denoted by xy . By Proposition 3.2.2 and the strong triadic closure, in the forthcoming arguments we will apply the following observations:

- If $xy \in E_S$, then for every strong edge $yz \in E_S$, $\{x, z\} \in E(G)$.
- If $xy \notin E_S$, then there is a strong edge $yz \in E_S$ such that $\{x, z\} \notin E(G)$.

We say that a solution E_S has the *consecutive strong property* with respect to σ if for any three vertices x, y, z of G with $x \prec y \prec z$ the following holds: $xz \in E_S$ implies $xy, yz \in E_S$. Our task is to show that such an ordering exists for an optimal solution.

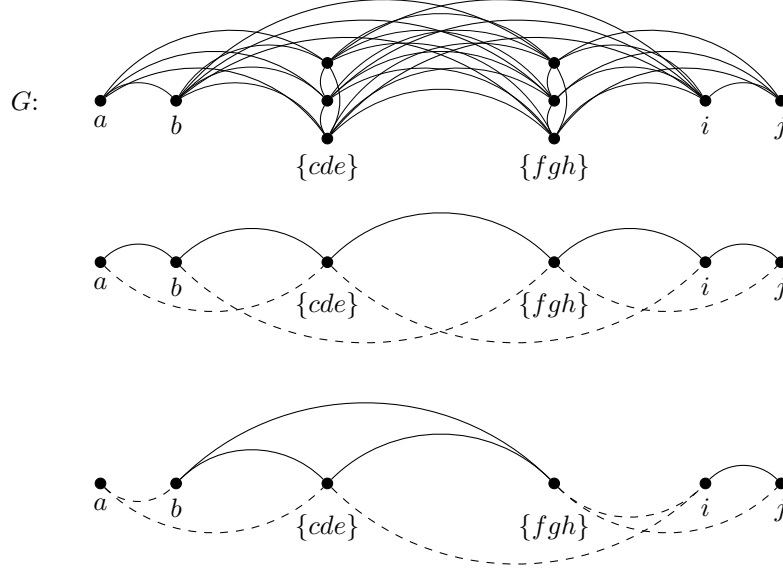


Figure 3.4: A proper interval graph G and its proper interval ordering. The vertices $\{c, d, e\}$ and $\{f, g, h\}$ form twin sets in G . The two lower orderings depict two solutions for MAXSTC on G . A solid edge corresponds to a strong edge, whereas a dashed edge corresponds to a weak edge. Observe that the upper solution contains larger number of strong edges than the lower one. Also note that the lower solution consists an optimal solution for the CLUSTER DELETION problem on G .

We start by characterizing the optimal solution E_S with respect to the proper interval ordering σ .

Lemma 3.4.1. *Let x, y, z be three vertices of a proper interval graph G such that $x \prec y \prec z$. If $xz \in E_S$ then $xy \in E_S$ or $yz \in E_S$.*

Proof. We show that at least one of xy or yz belongs to E_S . Assume towards a contradiction that neither xy nor yz belong to E_S . Consider the edge xy in G . Since $xy \notin E_S$, there is a strong edge of E_S that share a common vertex with xy . Suppose that there is an edge $xx_\ell \in E_S$ such that $\{x_\ell, y\} \notin E(G)$. Then observe that $x_\ell \prec y$ because $x \prec y$ and $\{x_\ell, y\} \notin E(G)$. Since both xx_ℓ and xz belong to E_S , $\{x_\ell, z\} \in E(G)$. This, however, contradicts the proper interval ordering because $x_\ell \prec y \prec z$, $\{x_\ell, z\} \in E(G)$ and y is non-adjacent to x_ℓ . Thus, there is no strong edge xx_ℓ with $\{x_\ell, y\} \notin E(G)$ and, in analogous fashion, there is no strong edge zz_r with $\{y, z_r\} \notin E(G)$.

Now, if all strong neighbors of y are adjacent to x (resp., z) then the edge xy (resp., yz) can be made strong. So, let us assume that there are edges $yy_r \in E_S$ and $y_\ell y \in E_S$ such that $\{x, y_r\} \notin E(G)$ and $\{z, y_\ell\} \notin E(G)$. Since $\{x, z\} \in E(G)$, y_ℓ and y_r cannot be ordered between x and y in the proper interval ordering. In particular, by the facts $\{x, y_r\} \notin E(G)$ and $\{z, y_\ell\} \notin E(G)$ we have $y_\ell \prec x \prec y \prec z \prec y_r$. Then, notice that $\{y_\ell, y_r\} \in E(G)$, because both $yy_r, yy_\ell \in E_S$. By the proper interval ordering we know that both x and z are adjacent to y_ℓ, y_r , leading to a contradiction to the assumptions $\{x, y_r\} \notin E(G)$ and $\{z, y_\ell\} \notin E(G)$. Therefore, at least one of xy or yz belongs to E_S . \square

Thus, by Lemma 3.4.1 we have two symmetric cases to consider. The next characterization suggests that there is a fourth vertex with important properties in each corresponding case.

Lemma 3.4.2. *Let x, y, z be three vertices of a proper interval graph G such that $x \prec y \prec z$ and $xz \in E_S$.*

- *If $xy \notin E_S$ and $yz \in E_S$ then for every edge $x_\ell x \in E_S$, $\{x_\ell, y\} \in E(G)$ holds and there is a vertex w such that $yw \in E_S$, $\{x, w\} \notin E(G)$, and $z \prec w$.*
- *If $xy \in E_S$ and $yz \notin E_S$ then for every edge $z z_r \in E_S$, $\{y, z_r\} \in E(G)$ holds and there is a vertex w such that $wy \in E_S$, $\{w, z\} \notin E(G)$ and $w \prec x$.*

Proof. Let $xy \notin E_S$ and $yz \in E_S$. The case for $xy \in E_S$ and $yz \notin E_S$ is completely symmetric. First, we prove that there is no strong edge $x_\ell x$ such that $\{x_\ell, y\} \notin E(G)$. Suppose that there is an edge $x_\ell x \in E_S$ with y being non-adjacent to x_ℓ . Notice that $x_\ell \prec x$ because y is adjacent to x and $x \prec y$. Due to the fact that $x_\ell x, xz \in E_S$, we have $\{x_\ell, z\} \in E(G)$. Since $x_\ell \prec x \prec y \prec z$ and $\{x_\ell, z\} \in E(G)$, by the proper interval ordering we get $\{x_\ell, y\} \in E(G)$ leading to a contradiction. Thus there is no strong edge $x_\ell x \in E_S$ with $\{x_\ell, y\} \notin E(G)$.

Now to prove the statement, assume towards a contradiction that there is no vertex w such that $yw \in E_S$, $\{x, w\} \notin E(G)$, and $z \prec w$. Thus, for all edges $yy_r \in E_S$ either $\{x, y_r\} \notin E(G)$ and $y_r \prec z$, or $\{x, y_r\} \in E(G)$. If $\{x, y_r\} \notin E(G)$ we know that $y_r \prec x$ or $x \prec y_r \prec z$ by the proper interval ordering. However, both cases lead to a contradiction to $\{x, y_r\} \notin E(G)$ since in the former case we have $\{y_r, y\} \in E(G)$ and $y_r \prec x \prec y$, and in the latter case we know that $\{x, z\} \in E(G)$. If $\{x, y_r\} \in E(G)$ then putting xy in E_S does not violate the strong triadic closure, contradicting the assumption $xy \notin E_S$ of the statement. Therefore, the corresponding statement holds. \square

Now we are ready to show that there is an optimal solution that has the described properties with respect to the given proper interval ordering.

Lemma 3.4.3. *There exists an optimal solution E_S that has the consecutive strong property with respect to σ .*

Proof. Let σ be a proper interval ordering for G . Assume towards a contradiction that E_S does not have the consecutive strong property. Then there exists a *conflict* with respect to σ , that is, there are three vertices x, y, z with $x \prec y \prec z$ and $xz \in E_S$ such that $xy \notin E_S$ or $yz \notin E_S$. We will show that as long as there are conflicts in σ , we can reduce the number of conflicts in σ without affecting the value of the optimal solution E_S . Consider such a conflict formed by the three vertices $x \prec y \prec z$ with $xz \in E_S$. By Lemma 3.4.1 we know that $xy \in E_S$ or $yz \in E_S$. Without loss of generality, assume that $yz \in E_S$. Then, clearly $xy \notin E_S$, for otherwise there is no conflict. Moreover, by Lemma 3.4.2 there is a vertex w such that $yw \in E_S$, $\{x, w\} \notin E(G)$, and $x \prec y \prec z \prec w$. Notice that both triples x, y, z and y, z, w create conflicts in σ .

We start by choosing an appropriate such conflict that is formed by four vertices x, y, z, w so that $x \prec y \prec z \prec w$, $xz, yz, yw \in E_S$, and $\{x, w\} \notin E(G)$. Fix y and z in σ with y, z being the leftmost and the rightmost vertices, respectively, such that for every vertex v with $y \prec v \prec z$, $yv, vz \in E_S$ holds. Recall that $yz \in E_S$. We choose x as the leftmost vertex such that $xz \in E_S$, $xy \notin E_S$ and we choose w as the rightmost vertex such that $yw \in E_S$, $zw \notin E_S$. Observe that $\{x, w\} \notin E(G)$, since y and z participate in a conflict. Due to the properties of the considered conflicts, all such vertices exist (see for e.g., Figure 3.5). Our task is to create a new set of strong edges E'_S from E_S such that (i) E'_S does not violate the strong triadic closure, (ii) $|E'_S| = |E_S|$, and (iii) E'_S has strictly less conflicts than E_S . We do so, with a series of claims that we prove next.

We define the following sets of vertices with respect to the strong neighbors of z and y :

- Let $X(z)$ be the set of vertices x_j such that $x_jz \in E_S$ and $\{x_j, w\} \notin E(G)$.
- Let $W(y)$ be the set of vertices w_i such that $yw_i \in E_S$ and $\{x, w_i\} \notin E(G)$.

Claim 3.4.4. *$X(z) \prec y \prec z \prec W(y)$ holds. Moreover, for any vertex of $x_j \in X(z)$, $x_jy \notin E_S$ holds, and for any vertex of $w_i \in W(y)$, $zw_i \notin E_S$ holds.*

Proof: Let $x_j \in X(z)$. If $w \prec x_j$ then $\{x_j, w\} \in E(G)$ because $z \prec w$ and $\{x_j, z\} \in E(G)$, and if $y \prec x_j \prec w$ then $\{x_j, w\} \in E(G)$ because $\{y, w\} \in E(G)$. By the definition of $X(z)$ we know that $\{x_j, w\} \notin E(G)$ which means $x_j \prec y$. Thus, $\{x_j, y\} \in E(G)$ holds, since $\{x_j, z\} \in E(G)$. With symmetric arguments for the vertices of $W(y)$, we conclude that $X(z) \prec y \prec z \prec W(y)$. Moreover, if $x_jy \in E_S$ then by the fact that $yw \in E_S$, we have $\{x_j, w\} \in E(G)$, contradicting the definition of $X(z)$. \diamond

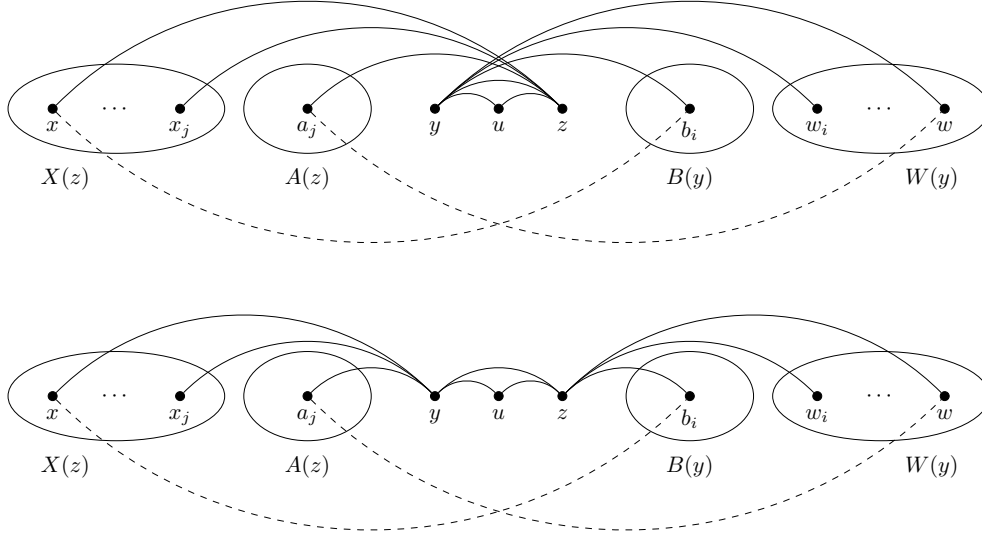


Figure 3.5: A proper interval ordering for a graph G with two different solutions considered in the proof of Lemma 3.4.3. A solid edge corresponds to a strong edge in an optimal strong-weak labeling, whereas a dashed edge corresponds to a weak edge in an optimal strong-weak labeling. Observe that the lowest ordering contains less *conflicts* than the topmost, that is, triple of vertices that violate the consecutive strong property.

Notice that the previous claim and the choices for x and w imply that x is the leftmost vertex of $X(z)$ and w is the rightmost vertex of $W(y)$. We further partition the strong neighbors of z that lie to the left of y , and the strong neighbors of y that lie to the right of z , as follows (see Figure 3.5):

- Let $A(z)$ be the set of vertices a_j such that $a_j \prec y$, $a_j z \in E_S$, $\{a_j, w\} \in E(G)$, and $a_j y \notin E_S$.
- Let $B(y)$ be the set of vertices b_i such that $z \prec b_i$, $y b_i \in E_S$, $\{x, b_i\} \in E(G)$, and $z b_i \notin E_S$.

Claim 3.4.5. $X(z) \prec A(z) \prec y \prec z \prec B(y) \prec W(y)$ holds. Moreover, every vertex of $A(z)$ is adjacent to all the vertices of $W(y)$ and every vertex of $B(y)$ is adjacent to all the vertices of $X(z)$.

Proof: By Claim 3.4.4 and the definition of $A(z)$, we know that $X(z) \prec y$ and $A(z) \prec y$. Let $x_j \in X(z)$ and $a_j \in A(z)$. Since $\{x_j, w\} \notin E(G)$, $\{a_j, w\} \in E(G)$, and σ is a proper

interval ordering, we deduce that $X(z) \prec A(z) \prec y$. Moreover, a_j is adjacent to w by definition, and w is the rightmost vertex of $W(y)$. Thus, a_j is adjacent to every vertex of $W(y)$. Symmetric arguments for the vertices of $B(y)$ and $W(y)$ show the rest of the claims. \diamond

Claim 3.4.6. *Let v be a vertex of G . For any vertex $x_j \in X(z)$, we have $vx_j \in E_S$ implies $\{v, y\} \in E(G)$. Moreover, for any vertex $a_j \in A(z)$, we have $va_j \in E_S$ implies $\{v, y\} \in E(G)$.*

Proof: For the first claim, notice that $v \prec w$, since $\{x_j, w\} \notin E(G)$. If $x_j \prec v \prec w$ then $\{v, y\} \in E(G)$, since $x_j \prec y \prec w$ and y is adjacent to x_j and w ; and if $v \prec x_j$ then, due to the fact that $x_j z, vx_j \in E_S$ and $\{z, v\} \in E(G)$, we have again $\{v, y\} \in E(G)$, because $v \prec y \prec z$. For the latter, if $y \prec v$ then $\{v, y\} \in E(G)$ because $a_j \prec y \prec v$ and $\{a_j, v\} \in E(G)$; and if $v \prec y$ then $\{v, z\} \in E(G)$ because $a_j z, va_j \in E_S$, so that $\{v, y\} \in E(G)$ by the proper interval ordering $v \prec y \prec z$. \diamond

With symmetric arguments, we get the following for the vertices of $W(y)$ and $B(y)$:

Claim 3.4.7. *Let v be a vertex of G . For any vertex $w_i \in W(y)$, we have $vw_i \in E_S$ implies $\{v, z\} \in E(G)$. Moreover, for any vertex $b_i \in B(y)$, we have $vb_i \in E_S$ implies $\{v, z\} \in E(G)$.*

The next claim shows that for every vertex v that is strongly adjacent to y and lies to the right of y , either v belongs to $B(y) \cup W(y)$ or v is strongly adjacent to z , as well.

Claim 3.4.8. *Let $yv \in E_S$ such that $y \prec v$ and $v \notin B(y) \cup W(y)$. Then, $zv \in E_S$.*

Proof: If $y \prec v \prec z$ then according to the choice of y and z we have $yv, zv \in E_S$. Assume for contradiction that $z \prec v$ and $zv \notin E_S$. Then, v is either adjacent to x or non-adjacent to x . In the former case we have $v \in B(y)$ and in the latter case we have $v \in W(y)$. Thus, in all cases we get $zv \in E_S$. \diamond

Symmetrically, for the left strong neighbors of z we have the following:

Claim 3.4.9. *Let $zv \in E_S$ such that $v \prec z$ and $v \notin X(z) \cup A(z)$. Then, $yv \in E_S$.*

We define the following set of weak and strong edges:

- Let $S(y)$ be the set of weak edges of the form yv where $v \in X(z) \cup A(z)$.
- Let $E(y)$ be the set of strong edges of the form yv where $v \in B(y) \cup W(y)$.
- Let $S(z)$ be the set of weak edges of the form zv where $v \in B(y) \cup W(y)$.
- Let $E(z)$ be the set of strong edges of the form zv where $v \in X(z) \cup A(z)$.

Now we are ready to modify E_S as follows:

$$E'_S = (E_S \setminus (E(y) \cup E(z))) \cup (S(y) \cup S(z)).$$

We say that a labeled edge e is *removed* from E_S if $e \in E_S \setminus E'_S$, whereas e is *added* in E'_S if $e \in E'_S \setminus E_S$. Under these terms, observe that an edge uy is added only if $u \prec y$, and uy is removed only if $y \prec u$; symmetrically, uz is added only if $z \prec u$, and uz is removed only if $u \prec z$.

Claim 3.4.10. *Labeling strong the edges of E'_S does not violate the strong triadic closure.*

Proof: Observe that we only replace labeled edges incident to y and z . Consider the edges incident to y . Those new strong edges are formed by the set $S(y)$. Since E_S satisfies the strong triadic closure, the edges of $E_S \setminus E(y)$ do not violate the strong triadic closure. Let $uy \in E'_S$. By the definition of $S(y)$, we know that $u \in X(z) \cup A(z)$ and $u \prec y$. There are three cases to consider:

- (i) there is an edge $vy \in E'_S$ and $v \in X(z) \cup A(z)$.
- (ii) there is an edge $vy \in E'_S$ and $v \notin X(z) \cup A(z)$.
- (iii) there is an edge $uv \in E'_S$.

In the former case (i), observe that $uz, vz \in E_S$ according to the definition of $X(z)$ and $A(z)$. Thus, $\{u, v\} \in E(G)$ holds. For the second case (ii), if $y \prec v$ then $vz \in E_S$ by Claim 3.4.8, which means that $\{u, v\} \in E(G)$ holds, since $uz \in E_S$. If $v \prec y$ then $x \prec v$ because $yw, vy \in E_S$ and $\{x, w\} \notin E(G)$. Thus, $x \prec \{u, v\} \prec y \prec z$ and, since $\{x, z\} \in E(G)$, we get $\{u, v\} \in E(G)$. For the third case (iii), Claim 3.4.6 shows that $\{y, v\} \in E(G)$. Therefore, the edges incident to y in E'_S do not violate the strong triadic closure.

Completely symmetric arguments hold for the edges incident to z in E'_S . Hence, labeling strong the edges of E'_S does not violate the strong triadic closure. \diamond

Claim 3.4.11. $|E'_S| = |E_S|$ holds.

Proof: Only edges incident to y and z are modified from E_S with respect to E'_S . By definition, the number of edges incident to y that are removed from E_S is $|B(y)| + |W(y)|$, whereas the number of edges incident to z that are added in E'_S is $|B(y)| + |W(y)|$. Symmetrically, the number of edges incident to z that are removed from E_S is $|X(z)| + |A(z)|$, whereas the number of edges incident to y that are added in E'_S is $|X(z)| + |A(z)|$. Since the rest of the edges in E_S and E'_S remain the same, we have $|E'_S| = |E_S|$. \diamond

Claim 3.4.12. *The number of conflicts in E'_S is strictly smaller than the number of conflicts in E_S .*

Proof: Observe that conflicts of the form $X(z) \prec y \prec z$ and $y \prec z \prec W(y)$ in E_S do not create conflicts in E'_S by the construction of E'_S . Since $X(z)$ and $W(y)$ are non-empty sets, such already existed conflicts in E_S do not appear in E'_S . We show that for every other conflict in E'_S , there is a unique conflict in E_S . First we show that both y and z do not participate in a conflict in E'_S . Assume for contradiction that y and z form a conflict in E'_S with another vertex v . If $v \prec y \prec z$ then, since $yz \in E_S \cap E'_S$, we have $vy \notin E'_S$ and $vz \in E'_S$ in order to create a conflict. Then, observe that $vz \in E_S$, because no added edge is incident to the left of z in E'_S . Moreover, no edge incident to the left of y is removed from E_S which means that $vy \notin E_S$. This, however, contradicts Claim 3.4.9, since $vz \in E_S$, $v \notin X(z) \cup A(z)$, and $vy \notin E_S$. If $y \prec z \prec v$ then symmetric arguments show that there is no such conflict. Moreover, if $y \prec v \prec z$ holds, then by the choice of y and z , we know that $yv, vz \in E_S \cap E'_S$. Thus, there is no conflict in E'_S that is formed by y and z .

Consider the edges incident to y . Note that with respect to the proper interval ordering, we only add edges incident to y and a vertex to the left of y , and we only remove edges incident to y and a vertex to the right of y . This means that for any vertex $v \prec y$ with $vy \notin E'_S$, we have $vy \notin E_S$, and for any vertex $y \prec v$ with $yv \in E'_S$, we have $yv \in E_S$. Observe also that the (added or removal) edges of $E'_S \setminus E_S$ and $E_S \setminus E'_S$ are incident to y or z . Thus, for any two vertices u, v different than y and z , we have $uv \in E'_S$ if and only if $uv \in E_S$.

Assume that there is a conflict in E'_S formed by the vertices u, v, y . Note that $u, v \neq z$, since y and z do not participate in a conflict. If both edges of the conflict belong to $E_S \cap E'_S$ and the non-edge of the conflict does not belong to $E_S \cup E'_S$, then such a conflict formed by the vertices u, v, y was already a conflict in E_S . So, let us assume next that at least one of the two edges in the conflict belongs to $E'_S \setminus E_S$ and is incident to y (i.e., $uy \in E'_S \setminus E_S$ or $vy \in E'_S \setminus E_S$) or the non-edge of the conflict belongs to $E_S \setminus E'_S$ and is incident to y (i.e., $uy \in E_S \setminus E'_S$).

- Suppose that $u \prec v \prec y$. Then, we have either $uy, uv \in E'_S$ and $vy \notin E'_S$, or $uy, vy \in E'_S$ and $uv \notin E'_S$. In the former case, we have the following:
 - Let $uy \in E'_S$, $uv \in E'_S$, and $vy \notin E'_S$. Notice that $uv \in E_S$ and $vy \notin E_S$, which means that $uy \in E'_S \setminus E_S$. Then, by the definition of E'_S , we have $u \in X(z) \cup A(z)$ and $v \notin X(z) \cup A(z)$. This means that $vz \notin E_S$ by Claim 3.4.9. Thus, $u \prec v \prec z$ form a conflict in E_S , because $uz \in E_S$ and $vz \notin E_S$. Moreover, observe that u, v, z is not a conflict in E'_S , because $uz \notin E'_S$ by the definition of E'_S .

In the latter case, we distinguish between uy being an added edge:

- Let $uy \in E'_S \setminus E_S$, $uv \notin E'_S$, and $vy \in E'_S$. Notice that $uv \notin E_S$. If $vy \in E'_S \setminus E_S$ then $u \prec v \prec z$ form a conflict in E_S , because $u, v \in X(z) \cup A(z)$ and $uz, vz \in E_S$. Moreover, by the definition of E'_S observe that $uz, vz \notin E'_S$, so that u, v, z is not a conflict in E'_S . If $vy \in E_S$ then, since $uz \in E_S$, v is strongly adjacent to u or z in E_S by Lemma 3.4.1. By the fact that $uv \notin E_S$, we get $vz \in E_S$. This shows that $u \prec v \prec z$ form a conflict in E_S . Moreover, observe that u, v, z is not a conflict in E'_S , because $uz \notin E'_S$ by the definition of E'_S .
- Let $uy \in E'_S \cap E_S$, $uv \notin E'_S$, and $vy \in E'_S \setminus E_S$. Notice that $uy \in E_S$, $uv \notin E_S$, $vy \notin E_S$. This, however, is not possible due to Lemma 3.4.1, since $u \prec v \prec y$ and v is not strongly adjacent to u and y .
- Suppose that $u \prec y \prec v$. Then, we have either $uv, uy \in E'_S$ and $yv \notin E'_S$, or $uv, yv \in E'_S$ and $uy \notin E'_S$. In the former case, we distinguish between uy being an added edge:
 - Let $uy \in E'_S \setminus E_S$, $uv \in E'_S$, and $yv \notin E_S \cup E'_S$. Notice that $uy \notin E_S$ and $uv \in E_S$. This, however, is not possible due to Lemma 3.4.1, since $uy, yv \notin E_S$.
 - Let $uy \in E'_S \setminus E_S$, $uv \in E'_S$, and $yv \in E_S \setminus E'_S$. Notice that $uy \notin E_S$, $uv \in E_S$, and $yv \in E_S$. Thus $u \prec y \prec v$ form a conflict in E_S .
 - Let $uy \in E_S \cap E'_S$, $uv \in E'_S$, and $yv \in E_S \setminus E'_S$. Notice that $uv \in E_S$ and $yv \in E_S$. Then, by the definition of E'_S , we have $v \in B(y) \cup W(y)$ which implies $zv \notin E_S$ and $y \prec z \prec v$. Moreover, by Lemma 3.4.1 we have $uz \in E_S$, since $u \prec z \prec v$ and $uv \in E_S$. Thus, $u \prec z \prec v$ form a conflict in E_S , because $uv, uz \in E_S$, and $zv \notin E_S$. Further, notice that u, z, v is not a conflict in E'_S , because $zv \in E'_S$ by $v \in B(y) \cup W(y)$ and $uz \in E'_S$ by $u \notin X(z) \cup A(z)$.

In the latter case, we have the following:

- Let $uy \notin E'_S$, $uv \in E'_S$, and $yv \in E'_S$. Notice that $uy \notin E_S$ because $u \prec y$, $uv \in E_S$, and $yv \in E_S$ because $y \prec v$. Thus, $u \prec y \prec v$ form a conflict in E_S .
- Suppose that $y \prec u \prec v$. Then, we have either $yv, yu \in E'_S$ and $uv \notin E'_S$, or $yv, uv \in E'_S$ and $yu \notin E'_S$. We distinguish between the two cases:
 - Let $yv \in E'_S$, $yu \in E'_S$ and $uv \notin E'_S$. Notice that $yv \in E_S$, $yu \in E_S$, because $y \prec \{u, v\}$, and $uv \notin E_S$. Thus, $y \prec u \prec v$ form a conflict in E_S .

- Let $yu \notin E'_S$, $yv \in E'_S$, and $uv \in E'_S$. Notice that $yv \in E_S$ because $y \prec v$, and $uv \in E_S$. Thus, yu is a removal edge, so that $yu \in E_S \setminus E'_S$. Then, by the definition of E'_S , we have $u \in B(y) \cup W(y)$ and $v \notin B(y) \cup W(y)$. This means that $zu \notin E_S$ and by Claim 3.4.8 we have $zv \in E_S$. Thus, $z \prec u \prec v$ form a conflict in E_S , because $zu \notin E_S$, $zv \in E_S$, and $uv \in E_S$. Moreover, observe that z, u, v is not a conflict in E'_S , because $zu, zv \in E'_S$ by the definition of E'_S .

Putting together, for all edges incident to y that create new conflict in E'_S there is a unique conflict in E_S . With completely symmetric arguments, we get that edges incident to z that create new conflict in E'_S there is a unique conflict in E_S . Therefore, E'_S has strictly smaller number of conflicts than E_S , since the conflicts formed by y and z in E_S are not conflicts in E'_S . \diamond

Thus, we replace appropriate set of strong edges in E_S and obtain an optimal solution by Claims 3.4.10 and 3.4.11 having a smaller number of conflicts by Claim 3.4.12. Therefore, by applying such a replacement in every possible conflict, we get an optimal solution that has no conflicts and, thus, it satisfies the consecutive strong property. \square

Lemma 3.4.3 suggests to find an optimal solution that has the consecutive strong property with respect to σ . In fact by the proper interval ordering, this reduces to computing the largest proper interval subgraph H of G such that the vertices of every P_3 of H induce a clique in G .

Let G be a proper interval graph and let $\sigma = \langle v_1, \dots, v_n \rangle$ be its proper interval ordering. For a vertex v_i we denote by $\ell(i)$ and $r(i)$ the positions of its leftmost and rightmost neighbors, respectively, in σ . Observe that for any two vertices $v_i \prec v_j$ in σ , $v_{\ell(i)} \preceq v_{\ell(j)}$ and $v_{r(i)} \preceq v_{r(j)}$ [32]. For $1 \leq i \leq n$, let $V_i = \{v_i, \dots, v_n\}$ and let $G_i = G[V_i]$. In what follows, we assume that E_S is a solution of G that has the consecutive strong property with respect to σ . Given E_S , we denote by $r_S(i)$ the position of the rightmost strong neighbor of v_i , that is,

$$r_S(i) = \begin{cases} n & \text{if } i = n \text{ or } v_i v_k \in E_S \text{ for every vertex } v_k \in V_{i+1}, \\ i & \text{if } v_i v_k \notin E_S \text{ for every vertex } v_k \in V_{i+1}, \\ k & \text{if } v_i v_k \in E_S \text{ and } v_i v_{k+1} \notin E_S \text{ for a vertex } v_k \in V_{i+1}. \end{cases}$$

Observe that if $r_S(i) = i$, then v_i has no strong neighbor in V_i . Moreover, notice that $i \leq r_S(i) \leq r(i)$. It is not difficult to see that given $r_S(i)$ for every vertex v_i , we can describe all strong edges of a solution E_S , since, by the consecutive strong property, for $i < j$, we have $v_i v_j \in E_S$ if and only if $i < j \leq r_S(i)$ holds.

We are now ready to define our subproblems with respect to σ . Let A_i be the value of an optimal solution E_S of G_i . Clearly A_1 corresponds to the value of an optimal solution of G .

Definition 3.4.13. For $i \leq r \leq r(i)$, we denote by $A[i, r]$ the value of an optimal solution of G_i such that $r_S(i) = r$.

As a base case, observe that $A_n = A[n, n] = 0$, because G_n contains exactly one vertex. By the consecutive strong property, we get the following equation, for any $1 \leq i \leq n$:

$$A_i = \max_{i \leq r \leq r(i)} A[i, r]. \quad (3.1)$$

For $i \leq r$, we denote by $r_{S(i,r)}(j)$ the position of the rightmost strong neighbor of every vertex $v_j \in V_i$ that is described by the value $A[i, r]$. For $1 \leq i < r \leq r' \leq n$, let $B_{i,r,r'}$ be the number of strong edges incident to v_{i+1}, \dots, v_{r-1} (i.e., to the vertices of $V_{i+1} \setminus V_r$) that belong to G_{i+1} in an optimal solution of G_i such that

- $r_S(i) = r$ and
- $r_S(j) = r_{S(r,r')}(j)$, for every $r \leq j \leq n$.

Observe that if $r = i + 1$ then $B_{i,r,r'} = 0$, as there is no vertex between v_i and v_r (i.e., the set $V_{i+1} \setminus V_r$ is empty). Moreover, notice that $r_{S(r,r')}(j)$ corresponds to the position of the rightmost strong neighbor of v_j described by $A[r, r']$.

In order to recursively compute $A[i, r]$, the key idea is that we try every allowed position r' of the rightmost strong neighbor of r . This is achieved through the consecutive strong property. Then, we split the solution into two parts among the vertices of $V_i \setminus V_r$ and V_r , respectively.

Lemma 3.4.14. Let $1 \leq i \leq r \leq n$. Then $A_n = A[n, n] = 0$, $A[i, i] = A_{i+1}$, and

$$A[i, r] = \max_{r \leq r' \leq r(i)} (A[r, r'] + B_{i,r,r'} + r - i).$$

Proof. The base cases follow by definition. Assume that $v_i v_r \in E_S$ with $i < r$ and v_r being the rightmost strong neighbor of v_i . It is clear that $r \leq r(i)$, as $\{v_i, v_r\} \in E(G)$. Moreover, for the rightmost strong neighbor $v_{r'}$ of v_r , we know that $\{v_i, v_{r'}\} \in E(G)$ since both strong edges $v_i v_r$ and $v_r v_{r'}$ satisfy the strong triadic closure. Thus we have $r \leq r' \leq r(i)$. Furthermore, v_i is strongly adjacent to every vertex v_{i+1}, \dots, v_r by the consecutive strong property, implying that there are $r - i$ strong edges incident to v_i in G_i .

Let $S_{i,r}$ and $S_{r,r'}$ be the strong edges described by the values $A[i, r]$ and $A[r, r']$, respectively. Suppose that $\{x, y\}$ is an edge of G_r . We claim that $xy \in S_{i,r}$ if and only if $xy \in S_{r,r'}$. If $xy \in S_{i,r}$ then $xy \in S_{r,r'}$, since $V_r \subseteq V_i$. Assume for contradiction that $xy \in S_{r,r'}$ and $xy \notin S_{i,r}$. Then there is a vertex $w \in V_i \setminus V_r$ such that $wx \in S_{i,r}$ and $\{w, y\} \notin E(G)$. If $w \neq v_i$ then wx is not a strong edge described by the definition of $B_{i,r,r'}$. Thus $w = v_i$. Since v_r is the only strong neighbor of v_i in G_r , we know that $x = v_r$. For any strong neighbor y of v_r in $S_{r,r'}$ we know that $v_r \preceq y \preceq v_{r'}$ by the consecutive strong property. Then, however, by the fact that $r' \leq r(i)$ we reach a contradiction to $\{v_i, y\} \notin E(G)$. Therefore the given formula describes the considered solutions. \square

Next we focus on computing $B_{i,r,r'}$. For doing so, we define a recursive formulation on subsolutions that take into account the position $r_S(i+1)$ of the rightmost strong neighbor of v_{i+1} . Observe that $r \leq r_S(i+1) \leq r'$ holds, since $v_i v_r \in E_S$ and $v_r v_{r'} \in E_S$ by the consecutive strong property. In fact, it is not difficult to verify that, for every $i < k < r$, $r \leq r_S(k) \leq r'$ holds, because $r_S(i) = r$ and $r_S(r) = r'$.

Definition 3.4.15. For $i < r \leq t \leq r'$, we denote by $B[i, r, t, r']$ the number of strong edges incident to v_{i+1}, \dots, v_{r-1} (i.e., to the vertices of $V_{i+1} \setminus V_r$) that belong to G_{i+1} in an optimal solution of G_i such that

- $r_S(i) = r$,
- $t \leq r_S(k)$, for every $i < k < r$, and
- $r_S(j) = r_{S(r,r')}(j)$, for every $r \leq j \leq n$.

By the definition of the describes values and the previous discussion on the consecutive strong property, we get the following equation:

$$B_{i,r,r'} = \max_{r \leq t \leq r'} B[i, r, t, r'] \quad (3.2)$$

For every $i < k < r$, let $x_{r'}(k)$ be the rightmost position j such that $j \leq r'$ and $r_{S(r,r')}(j) \leq r(k)$. Observe that $r \leq x_{r'}(k)$, because $r' \leq r(i) \leq r(k)$ for every $v_i \prec v_k$.

Lemma 3.4.16. Let $1 \leq i < r \leq t \leq r' \leq n$. Then $B[r-1, r, t, r'] = 0$ and

$$B[i, r, t, r'] = \max_{t \leq j \leq x_{r'}(i+1)} (B[i+1, r, j, r'] + j - (i+1)).$$

Proof. Let v_j be the rightmost strong neighbor of v_{i+1} in G_{i+1} . Then $t \leq j$ holds, by the restriction described in $B[i, r, t, r']$. By the fact that $r_{S(r,r')}(r) = r'$ and $v_{i+1} \prec v_r$ we deduce that $j \leq r'$. Suppose that $x_{r'}(i+1) < j$. By the choice of $x_{r'}(i+1)$ we know that

either $r' < j$ which is not possible, or $j \leq r'$ and there is a strong edge $v_j v \in S(r, r')$ with $\{v_{i+1}, v\} \notin E(G)$ which violates the strong triadic closure. Thus we conclude $t \leq j \leq x_{r'}(i+1)$ and the consecutive strong property implies that there are $j - i - 1$ strong edges incident to v_{i+1} in G_{i+1} .

Let S_i and S_{i+1} be the strong edges described by $B[i, r, t, r']$ and $B[i+1, r, j, r']$, respectively. Suppose that $\{v_{i+2}, v\}$ is an edge of G_{i+2} . We claim that $v_{i+2}v \in S_i$ if and only if $v_{i+2}v \in S_{i+1}$. If $v_{i+2}v \in S_i$ then it is clear that $v_{i+2}v \in S_{i+1}$, since $V_{i+1} = V_{i+2} \cup \{v_{i+1}\}$. Assume for contradiction that $v_{i+2}v \in S_{i+1}$ and $v_{i+2}v \notin S_i$. Observe that $v_r \preceq v \preceq v_{r'}$. Let z be the rightmost strong neighbor of v in S_{i+1} . Then, by definition, we have $vz \in S_i$. This means that there is a strong edge wv_{i+2} in $S_i \setminus S_{i+1}$ such that $\{w, v\} \notin E(G)$. By the definition of $B[i, r, t, r']$, we have that $w = v_{i+1}$. Then, however, we reach a contradiction to $\{v_{i+1}, v\} \notin E(G)$, since $v \preceq v_{r'}$ and $r' \leq r(i) \leq r(i+1)$. Therefore we conclude the described formula. \square

Now we are equipped with our necessary tools in order to obtain our main result, namely a polynomial-time algorithm that solves the MAXSTC problem on proper interval graphs.

Theorem 3.4.17. *There is a polynomial-time algorithm that computes the MAXSTC of a proper interval graph.*

Proof. Let G be a proper interval graph on n vertices and m edges. We first compute its proper interval ordering σ in linear time [102]. In order to compute an optimal solution A_1 for G described in Equations 3.1 and 3.2, we use a dynamic programming approach based on their recursive formulation given in Lemmas 3.4.14 and 3.4.16, respectively. For doing so, we store two tables \mathcal{T}_A and \mathcal{T}_B that correspond to the values $A[i, r]$ and $B[i, r, t, r']$. Whenever we compute the value $\mathcal{T}_A[i, r]$, we perform a second sweep to backtrack the actual rightmost strong neighbor of each vertex in V_i . More precisely, it suffices to backtrack the rightmost strong neighbors of each vertex of $V_i \setminus V_r$. Correctness follows from Lemmas 3.4.3, 3.4.14, and 3.4.16.

Regarding the running time, notice that all instances of $\mathcal{T}_A[i, r]$ and $\mathcal{T}_B[i, r, t, r']$ can be computed as follows. At each vertex v_i , we compute all possible vertex pairs $v_r, v_{r'}$ with $i \leq r \leq r' \leq r(i)$ which are bounded by n^2 . For each vertex v_k with $i < k < r$, computing $x_{r'}(k)$ takes $O(n)$ time by scanning the rightmost strong neighbors of the vertices of $V_r \setminus V_{r'}$. Moreover, for each index t with $r \leq t \leq r'$, there are at most n choices. Thus the number of instances $\mathcal{T}_A[i, r]$ and $\mathcal{T}_B[i, r, t, r']$ generated by v_i is $O(n^3)$. Because we visit n vertices, the total running time of the algorithm is $O(n^4)$. \square

MAXSTC ON COGRAPHS AND GRAPHS OF LOW MAXIMUM DEGREE

Here in Chapter 4, we further explore the complexity of the MAXSTC problem on graph classes. Also, we conduct the first systematic study that reveals graph families for which the optimal solutions for MAXSTC and CLUSTER DELETION coincide. We first show that MAXSTC coincides with CLUSTER DELETION on cographs and, thus, MAXSTC is solvable in polynomial time on cographs. As a side result, we give an interesting computational characterization of the maximum independent set on the cartesian product of two cographs. Furthermore, we address the influence of the low degree bounds to the complexity of the MAXSTC problem. We show that this problem is polynomial-time solvable on graphs of maximum degree three, whereas MAXSTC becomes NP-complete on graphs of maximum degree four. The proof of the latter result implies that there is no subexponential-time algorithm for MAXSTC unless the Exponential-Time Hypothesis fails.

The results of this chapter have led to the following publications [86, 87]:

- **Strong triadic closure in cographs and graphs of low maximum degree.** Athanasios L. Konstantinidis, Stavros D. Nikolopoulos, and Charis Papadopoulos. *23rd Annual International Computing and Combinatorics Conference, (COCOON 2017), Hong Kong, China, 2017. Springer Verlag, LNCS 10392: 346–358.*
- **Strong triadic closure in cographs and graphs of low maximum degree.** Athanasios L. Konstantinidis, Stavros D. Nikolopoulos, and Charis Papadopoulos. *Theoretical Computer Science 740: 76–84, 2018.*

4.1 Introduction

The principle of strong triadic closure is an important concept in social networks [41]. It states that it is not possible for two individuals to have a strong relationship with

a common friend and not know each other [60]. The strong triadic closure is satisfied if the edges of the underlying graph are characterized into weak and strong such that any two vertices that have a strong neighbor in common are adjacent. Towards the investigation of the behavior of a network, such a principle has been recently proposed as a maximization problem, called MAXSTC, in which the goal is to assign each edge as strong or weak so that to maximize the number of strong edges of the underlying graph that satisfy the strong triadic closure [118]. Closely related to the MAXSTC problem is the CLUSTER DELETION problem which finds important applications in areas involving clustering [6]. In the latter problem the goal is to remove the minimum number of edges such that the resulting graph consists of vertex-disjoint union of cliques.

The relation between MAXSTC and CLUSTER DELETION arises from the fact that the edges inside the cliques in the resulting graph for CLUSTER DELETION can be seen as strong edges for MAXSTC which satisfy the strong triadic closure. Thus, the number of edges in an optimal solution for CLUSTER DELETION consists a lower bound for the number of strong edges in an optimal solution for MAXSTC. However there are graphs (see for e.g., Figure 4.1) showing that an optimal solution for MAXSTC contains larger number of edges than an optimal solution for CLUSTER DELETION. Interestingly, there are also families of graphs in which their optimal value for MAXSTC matches such a lower bound. For instance, any maximum matching on graphs that do not contain triangles constitutes a solution for both problems. Here we initiate a systematic study on other non-trivial classes of graphs for which the optimal solutions for both problems have exactly the same value.

Our main motivation is to further explore the complexity of the MAXSTC problem when restricted to graph classes. As MAXSTC has been recently introduced, there are few results concerning its complexity. The problem has been shown to be NP-complete for general graphs [118] and we showed in Chapter 3 that MAXSTC is NP-complete for split graphs (Theorem 3.3.4) whereas it becomes polynomial-time tractable on proper interval graphs (Theorem 3.4.17) and trivially perfect graphs (Theorem 3.2.3). The NP-completeness for MAXSTC on split graphs shows an interesting algorithmic difference between the two problems, since CLUSTER DELETION on such graphs can be solved in polynomial time [11]. It is known that CLUSTER DELETION is NP-complete on general graphs [116] and remains NP-complete on chordal graphs and, also, on graphs of maximum degree four [11, 85]. On the positive side CLUSTER DELETION admits polynomial-time algorithms on proper interval graphs [11], graphs of maximum degree three [85], and cographs [50]. In fact for cographs a greedy algorithm that finds iteratively maximum cliques gives an optimal solution, although no running time was explicitly given in [50].

Such a greedily approach is also proposed for computing a maximal independent set of the cartesian product of general graphs. Summing the partial products between

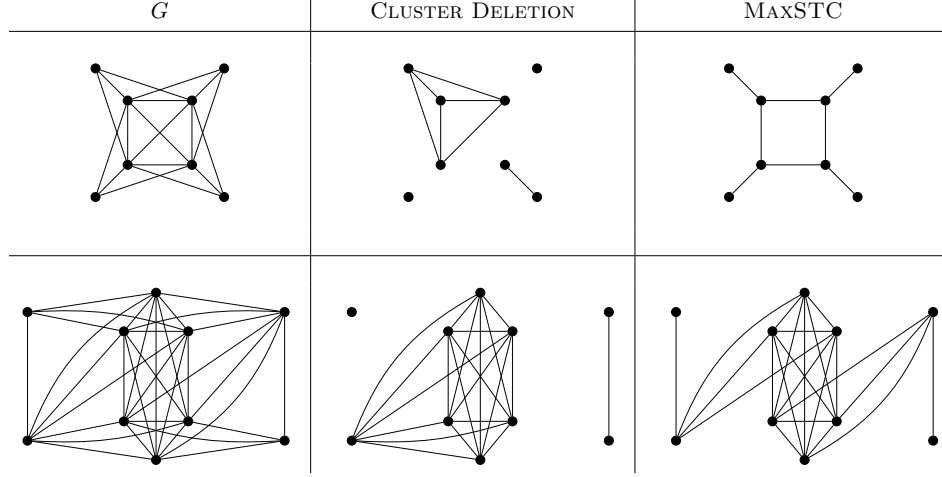


Figure 4.1: Two examples of graphs with their corresponding optimal solutions for CLUSTER DELETION and MAXSTC, respectively. For the MAXSTC problem the edges of G that are not drawn in the solution correspond to the weak edges.

iteratively maximum independent sets consists a lower bound for the cardinality of the maximum independent set of the cartesian product [75, 77]. Here we prove that a maximum independent set of the cartesian product of two cographs matches such a lower bound. We would like to note that a polynomial-time algorithm for computing such a maximum independent set is already claimed [71]. However neither a characterization is given, nor an explicit running time of the algorithm is reported.

In the following sections, we further explore the complexity of the MAXSTC problem. We consider two unrelated families of graphs, namely, cographs and graphs of bounded degree and we settle the complexity of the MAXSTC problem on both graph families. Cographs are characterized by the absence of a chordless path on four vertices. For such graphs we prove that the optimal value for MAXSTC matches the optimal value for CLUSTER DELETION. For doing so, we reveal an interesting vertex partitioning with respect to their maximum clique and maximum independent set. This result enables us to give an $O(n^2)$ -time algorithm for MAXSTC on cographs. As a byproduct we characterize a maximum independent set of the cartesian product of two cographs which implies a polynomial-time algorithm for computing such a maximum independent set. Moreover we study the influence of low maximum degree for the MAXSTC problem. We show an interesting complexity dichotomy result: for graphs of maximum degree four MAXSTC remains NP-complete, whereas for graphs of maximum degree three the problem is solved in polynomial time. Our reduction for the NP-completeness on graphs of maximum degree four implies that, under the Exponential-

Time Hypothesis, there is no subexponential time algorithm for MAXSTC.

4.2 Preliminaries

We remind here the definitions of MAXSTC and CLUSTER DELETION as well as some notations.

In the MAXSTC problem the goal is to find a strong-weak labeling on the edges of $E(G)$ that satisfies the strong triadic closure and has the maximum number of strong edges. We denote by (E_S, E_W) the partition of $E(G)$ into strong edges E_S and weak edges E_W . The graph *spanned by* E_S is the graph $G \setminus E_W$; notice that the graph spanned by E_S consists of the whole vertex set $V(G)$ and it may contain vertices with degree equal to zero. For a strong edge $\{u, v\}$, we say that u (resp., v) is a strong neighbor of v (resp., u). We denote by $N_S(v) \subseteq N(v)$ the strong neighbors of v . Given an optimal solution for MAXSTC that consists of the strong edges E_S , the graph spanned by the edges of E_S is denoted by $E_S(G)$. Whenever we write $|E_S(G)|$ we refer to its number of edges, that is $|E_S(G)| = |E_S|$.

In the CLUSTER DELETION problem the goal is to partition the vertices of a given graph G into vertex-disjoint cliques with the minimum number of edges outside the cliques, or, equivalently, with the maximum number of edges inside the cliques. A *cluster graph* is a graph in which every connected component is a clique. Cluster graphs are characterized as exactly the graphs that do not contain a P_3 as an induced subgraph. Given an optimal solution for CLUSTER DELETION, the cluster graph spanned by the edges that are inside the cliques is denoted by $E_C(G)$. We write $|E_C(G)|$ to denote the number of edges in the cluster graph. Notice that if we assign strong labels to all the edges of a cluster graph then such a labeling satisfies the strong triadic closure of the given graph. Thus $|E_C(G)| \leq |E_S(G)|$ holds for any graph G .

Figure 4.1 shows two graphs in which the optimal solution of CLUSTER DELETION contains strictly less edges than the optimal solution for MAXSTC. In terms of $E_C(G)$ and $E_S(G)$ notice that in such cases we have $|E_C(G)| < |E_S(G)|$, though in general $|E_C(G)| \leq |E_S(G)|$ holds. In the first example, from top to bottom, an optimal solution of CLUSTER DELETION consists of 7 edges whereas there is a solution of MAXSTC that contains 8 edges. The second example shows an optimal solution of CLUSTER DELETION with 22 edges, whereas there is a solution of MAXSTC with 23 edges. Notice that in the second example the 6 vertices drawn in the middle induce a clique on 6 vertices.

4.3 Computing MaxSTC on Cographs

A graph is a *cograph* if it can be generated from single-vertex graphs and recursively applying the disjoint union and complete join operations. The complement of a cograph is also a cograph. Cographs are exactly the graphs that do not contain any chordless path on four vertices [26], and they can be recognized in linear time [27].

Let G be the given cograph. Our main goal is to show that there is an optimal solution for MAXSTC on G that coincides with an optimal solution for CLUSTER DELETION. The strong edges that belong to an optimal solution for MAXSTC span the graph $E_S(G)$. An optimal solution for CLUSTER DELETION consists of a cluster graph $E_C(G)$ by removing a minimum number of edges of G . Labeling all edges of a cluster graph as strong, results in a strong-weak labeled graph that satisfy the strong triadic closure. Thus, our goal is to show that there is an optimal solution $E_S(G)$ for MAXSTC that is a cluster graph.

A clique (resp. independent set) of G having the maximum number of vertices is denoted by $C_{\max}(G)$ (resp., $I_{\max}(G)$). A *greedy clique partition* of G , denoted by \mathcal{C} , is the ordering of cliques (C_1, C_2, \dots, C_p) in G such that

- $C_1 = C_{\max}(G)$ and
- $C_i = C_{\max}\left(G - \bigcup_{j=1}^{i-1} C_j\right)$ for $i = 2, 3, \dots, p$.

Similarly, a *greedy independent set partition* of G , denoted by \mathcal{I} , is the ordering of independent sets (I_1, I_2, \dots, I_q) in G such that

- $I_1 = I_{\max}(G)$ and
- $I_i = I_{\max}\left(G - \bigcup_{j=1}^{i-1} I_j\right)$ for $i = 2, 3, \dots, q$.

Observe that the subgraph spanned by the edges of \mathcal{C} does not contain any P_3 and, thus, consists a solution for CLUSTER DELETION. Although in general a greedy clique partition does not necessarily imply an optimal solution for CLUSTER DELETION, when restricted to cographs the optimal solution is characterized by the greedy clique partition.

Lemma 4.3.1 ([50]). *Let G be a cograph with a greedy clique partition \mathcal{C} . Then the edges of \mathcal{C} span an optimal solution $E_C(G)$ for CLUSTER DELETION.*

We will use such a characterization of CLUSTER DELETION in order to give its equivalence with the MAXSTC problem. Notice, however, that due to the freedom of the adjacencies between the cliques of a greedy clique partition, it is not sufficient to consider

such a partition of the vertices. For doing so, we will further decompose the cliques of a greedy clique partition. It is known that a graph G is a cograph if and only if for any maximal clique C and any maximal independent set I of every induced subgraph of G , $|C \cap I| = 1$ holds (also known as the *clique-kernel intersection property*) [26]. Thus, we state the following lemma.

Lemma 4.3.2 ([26]). *Let G be a cograph. Then $C_{\max}(G) \cap I_{\max}(G) = \{v\}$ for some vertex v .*

We recursively apply Lemma 4.3.2 to obtain the following result.

Lemma 4.3.3. *Let G be a cograph with a greedy clique partition $\mathcal{C} = (C_1, \dots, C_p)$ and a greedy independent set partition $\mathcal{I} = (I_1, \dots, I_q)$. For every i, j with $1 \leq i \leq p$ and $1 \leq j \leq q$, if $|C_i| \geq j$ or $|I_j| \geq i$ then $C_i \cap I_j \neq \emptyset$.*

Proof. We prove that if $|C_i| \geq j$ or $|I_j| \geq i$ then $C_i \cap I_j \neq \emptyset$. Assume for contradiction that there exist C_i and I_j such that $C_i \cap I_j = \emptyset$. Let i and j be the smallest integers for which $C_i \cap I_j = \emptyset$. By the choice of j we know that for every $j' < j$, $C_i \cap I_{j'} \neq \emptyset$ holds because $|C_i| \geq j > j'$. This means that there are $j - 1$ vertices u_1, \dots, u_{j-1} such that $C_i \cap I_1 = \{u_1\}, \dots, C_i \cap I_{j-1} = \{u_{j-1}\}$. Similarly, for every $i' < i$ we have $|C_{i'}| \geq |C_i| \geq j$, by the greedy choice of C_1, \dots, C_i . Thus there are $i - 1$ vertices v_1, \dots, v_{i-1} such that $C_1 \cap I_j = \{v_1\}, \dots, C_{i-1} \cap I_j = \{v_{i-1}\}$. Let $G_{i,j}$ be the graph obtained from G by removing the sets of vertices C_1, \dots, C_{i-1} and I_1, \dots, I_{j-1} . Notice that $G_{i,j}$ contains at least one vertex because $|C_i| \geq j$ or $|I_j| \geq i$. We will prove that $C_{\max}(G_{i,j}) = C_i \setminus \{u_1, \dots, u_{j-1}\}$ and $I_{\max}(G_{i,j}) = I_j \setminus \{v_1, \dots, v_{i-1}\}$.

Let C' be the vertices of C_1, \dots, C_{i-1} and let I' be the vertices of I_1, \dots, I_{j-1} . By the greedy independent set partition, the vertices of $G_{i,j}$ can be partitioned into $|C_i| - j + 1$ independent sets $I_j \setminus C', \dots, I_{|C_i|} \setminus C'$. This implies that a maximum clique of $G_{i,j}$ has size at most $|C_i| - j + 1$. As $G_{i,j}$ is an induced subgraph of G , $C_i \setminus \{u_1, \dots, u_{j-1}\}$ is a clique of size $|C_i| - j + 1$ of $G_{i,j}$. Thus, we have $C_{\max}(G_{i,j}) = C_i \setminus \{u_1, \dots, u_{j-1}\}$. Following symmetric arguments, the vertices of $G_{i,j}$ can be partitioned into $|I_j| - i + 1$ cliques $C_i \setminus I', \dots, C_{|I_j|} \setminus I'$. This implies that a maximum independent set of $G_{i,j}$ has size at most $|I_j| - i + 1$. Thus $I_{\max}(G_{i,j}) = I_j \setminus \{v_1, \dots, v_{i-1}\}$.

Notice that $\{u_1, \dots, u_{j-1}\} \cap \{v_1, \dots, v_{i-1}\} = \emptyset$, due to the choice of i and j . Then Lemma 4.3.2 applies to $G_{i,j}$, which shows that

$$(C_i \setminus \{u_1, \dots, u_{j-1}\}) \cap (I_j \setminus \{v_1, \dots, v_{i-1}\}) \neq \emptyset.$$

Therefore $C_i \cap I_j \neq \emptyset$, leading to a contradiction that proves the desired statement. \square

Lemma 4.3.3 suggests a partition of the vertices of G with respect to \mathcal{C} and \mathcal{I} as follows. We call *greedy canonical partition* a pair $(\mathcal{C}, \mathcal{I})$ with elements $\langle v_{i,j} \rangle$, where

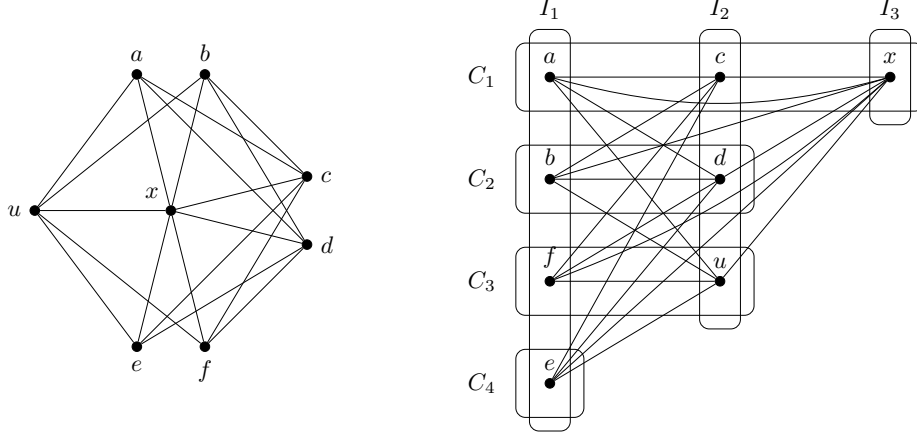


Figure 4.2: A cograph and its greedy canonical partition $(\mathcal{C}, \mathcal{I})$ where $\mathcal{C} = (C_1, C_2, C_3, C_4)$ and $\mathcal{I} = (I_1, I_2, I_3)$.

$1 \leq i \leq p$ and $1 \leq j \leq |C_i|$, such that $V(G) = \{v_{1,1}, \dots, v_{p,|C_p|}\}$ and $v_{i,j} \in C_i \cap I_j$. Figure 4.2 shows such a greedy canonical partition of a given cograph. Observe that such a partition corresponds to a 2-dimensional representation of G . By Lemma 4.3.3 it follows that a cograph admits a greedy canonical partition.

Let us turn our attention back to the initial MaxSTC problem. We first consider the disjoint union of cographs.

Lemma 4.3.4. *Let G and H be vertex-disjoint cographs. Then $E_S(G \oplus H) = E_S(G) \oplus E_S(H)$ and $E_C(G \oplus H) = E_C(G) \oplus E_C(H)$.*

Proof. There are no edges between G and H so that a strong edge of G and a strong edge of H have no common endpoint. Thus the union of the solutions for G and H satisfy the strong triadic closure. By Lemma 4.3.1, $E_C(G \oplus H)$ contains the edges of a greedy clique partition which is obtained from the corresponding cliques of G and H . \square

We next consider the complete join of cographs. Given two vertex-disjoint cographs G and H with greedy clique partitions $\mathcal{C} = (C_1, \dots, C_p)$ and $\mathcal{C}' = (C'_1, \dots, C'_{p'})$, respectively, we denote by $C_i(G, H)$ the edges that have one endpoint in C_i and the other endpoint in C'_j , for every $1 \leq i \leq \min\{p, p'\}$.

Lemma 4.3.5. *Let G and H be vertex-disjoint cographs with greedy clique partitions $\mathcal{C} = (C_1, \dots, C_p)$ and $\mathcal{C}' = (C'_1, \dots, C'_{p'})$, respectively. Then,*

- $E_S(G \otimes H) = (E_S(G) \oplus E_S(H)) \cup E(G, H)$ and
- $E_C(G \otimes H) = (E_C(G) \oplus E_C(H)) \cup E(G, H)$,

where $E(G, H) = C_1(G, H) \cup \dots \cup C_k(G, H)$ and $k = \min\{p, p'\}$.

Proof. For the edges of $E_C(G \otimes H)$ we know that a greedy clique partition of $G \otimes H$ forms an optimal solution by Lemma 4.3.1. A greedy clique partition of $G \otimes H$ is obtained from the cliques $C_i \cup C'_i$, for every $1 \leq i \leq k$, since all the vertices of G are adjacent to all the vertices of H . The edges of $C_i \cup C'_i$ can be partitioned into the sets $E(C_i)$, $E(C'_i)$, and $C_i(G, H)$ giving the desired formulation for $E_C(G \otimes H)$.

We consider the optimal solution for MaxSTC described by the edges of $E_S(G \otimes H)$. Let us show that any solution on the edges of G satisfy the strong triadic closure in the graph $G \otimes H$. Consider a strong edge $\{x, y\}$ of G . If the resulting labeling does not satisfy the strong triadic closure then there is a strong edge $\{x, w\}$ such that y and w are non-adjacent. As G and H are vertex-disjoint graphs, $w \in V(G)$ or $w \in V(H)$. If $w \in V(G)$ then we already know that the labeling of $E_S(G)$ satisfies the strong triadic closure so that y and w are adjacent. If $w \in V(H)$ then by the complete join operation w is adjacent to y . Thus maximizing the number of strong edges that belong in G and H results in an optimal solution for $G \otimes H$.

We next consider the edges that have one endpoint in G and the other in H , denoted by $E(G, H)$. Our goal is to show that edges of $C_1(G, H) \cup \dots \cup C_k(G, H)$ belong to an optimal solution. Let $(\mathcal{C}, \mathcal{I})$ and $(\mathcal{C}', \mathcal{I}')$ be the greedy canonical partitions of G and H , respectively, where

- $\mathcal{C} = (C_1, \dots, C_p), \mathcal{I} = (I_1, \dots, I_q)$, and
- $\mathcal{C}' = (C'_1, \dots, C'_{p'}), \mathcal{I}' = (I'_1, \dots, I'_{q'})$.

Now observe that $|C_1(G, H) \cup \dots \cup C_k(G, H)| = \sum_{i=1}^k |C_i||C'_i|$. Notice that the edges of $C_1(G, H) \cup \dots \cup C_k(G, H)$ satisfy the strong triadic closure, since every two strong edges incident to a vertex of G belong to $C_i(G, H)$ which implies that the endpoints of H belong to a clique C'_i and, thus, are adjacent in $G \otimes H$. Therefore, we have $|E_S(G \otimes H)| \geq |E_C(G \otimes H)|$ and

$$|E(G, H)| \geq \sum_{i=1}^k |C_i||C'_i|.$$

In the forthcoming arguments we prove that $|E(G, H)| \leq \sum_{i=1}^k |C_i||C'_i|$.

We consider the vertices of I_j , $1 \leq j \leq q$, and count the number of strong edges that have one endpoint in I_j and the other endpoint on a vertex of H . Without loss

of generality assume that $|I_1| \leq |I'_1|$. Then, $k = |I_1|$ since $p = |I_1|$ and $p' = |I'_1|$ by Lemma 4.3.3. For a subset W of vertices of G , we denote by $s(W)$ the number of strong edges of $E(G, H)$ that are incident to the vertices of W . By the strong triadic closure principle, any vertex of H has at most one strong neighbor in I_j and any vertex of G has at most one strong neighbor in $I'_{j'}$, $1 \leq j' \leq q'$. Thus, for any $I'_{j'}$ of H there are at most $\min\{|I_j|, |I'_{j'}|\}$ strong edges between the vertices of I_j and $I'_{j'}$. Let r_j be the largest index of $\{1, \dots, q'\}$ for which $|I'_{r_j}| \geq |I_j|$; notice that r_j exists, since $|I_j| \leq |I_1| \leq |I'_1|$. Then, since $|I'_1| \geq \dots \geq |I'_{q'}|$, it is clear that $|I_j|$ is smaller than or equal to any of $|I'_1|, \dots, |I'_{r_j}|$ and greater than to any of $|I'_{r_j+1}|, \dots, |I'_{q'}|$. Thus, we get the following inequality:

$$s(I_j) \leq \sum_{j'=1}^{q'} \min\{|I_j|, |I'_{j'}|\} = \sum_{j'=1}^{r_j} |I_j| + \sum_{j'=r_j+1}^{q'} |I'_{j'}|.$$

We next describe the vertices of $I'_1, \dots, I'_{r_j}, I'_{r_j+1}, \dots, I'_{q'}$ by the cliques of H . In particular, for every $1 \leq i \leq |I_j|$, we consider a clique C'_i of H . By Lemma 4.3.3 we know that C'_i contains exactly one vertex from each of $I'_1, \dots, I'_{r_j}, I'_{r_j+1}, \dots, I'_{|C'_i|}$. This means that all previously described vertices are contained in the disjoint union of cliques $C'_1, \dots, C'_{|I_j|}$. Thus, the previous inequality can be written as follows.

$$s(I_j) \leq \sum_{j'=1}^{r_j} |I_j| + \sum_{j'=r_j+1}^{q'} |I'_{j'}| = \sum_{i=1}^{|I_j|} |C'_i|.$$

Summing up each of $s(I_j)$ for every I_j , $1 \leq j \leq q$, we obtain:

$$|E(G, H)| = \sum_{j=1}^q s(I_j) \leq \sum_{j=1}^q \sum_{i=1}^{|I_j|} |C'_i|.$$

Observe that, in the described sum, each $|C'_i|$ is counted for all $1 \leq j \leq q$ for which $|I_j| \geq i$. For such $|I_j|$ and i , by Lemma 4.3.3 we have $C_i \cap I_j \neq \emptyset$. Thus, the number that $|C'_i|$ appears in the formula is exactly $|C_i|$. Moreover, by the greedy canonical partition we know that $\sum_{j=1}^q |I_j| = \sum_{i=1}^p |C_i|$ and $p = |I_1|$. Hence, we get the desired upper bound for the number of strong edges in $E(G, H)$:

$$|E(G, H)| \leq \sum_{j=1}^q \sum_{i=1}^{|I_j|} |C'_i| = \sum_{i=1}^{|I_1|} |C_i| |C'_i|.$$

Therefore, the claimed formula holds for the strong edges of $E_S(G \otimes H)$ and this concludes the proof. \square

We are now ready to state our claimed result, namely that the solutions for MAXSTC and CLUSTER DELETION coincide for the class of cographs.

Theorem 4.3.6. *Let G be a cograph. There is an optimal solution for MAXSTC on G that is a cluster graph. Moreover MAXSTC on G can be solved in $O(n^2)$ time.*

Proof. An optimal solution for MAXSTC coincides with an optimal solution for CLUSTER DELETION trivially for graphs that consist of a single vertex. If G is a non-trivial cograph then it is constructed by the disjoint union or the complete join operation. In the former case Lemma 4.3.4 applies, whereas in the later Lemma 4.3.5 applies showing that in all cases $E_S(G) = E_C(G)$.

Regarding the running time, a maximum clique C_1 of G can be found in $O(n)$ time [26], due to a suitable data structure called *cotree*. We first construct the cotree of G which takes time $O(n + m)$ [27]. Removing a vertex v from a cograph G and updating the cotree takes $O(d(v))$ time, where $d(v)$ is the degree of v in G [115]. Thus, after removing all vertices from G we can maintain the cotree in an overall $O(n + m)$ time. In every intermediate step, we first remove the set of vertices C_i in $O(d(C_i))$ time where $d(C_i)$ is the sum of the degree of the vertices of C_i , and then spend $O(n)$ time to compute a maximum clique by using the resulting cotree. Therefore, since there are at most n such cliques in \mathcal{C} , a greedy clique partition of G can be found in total $O(n^2)$ time. \square

4.3.1 Maximum independent set of the cartesian product of cographs

In this section we apply the characterization of Theorem 4.3.6 in order to show an interesting computational characterization of the cartesian product of cographs. Towards such a characterization we take advantage of the equivalent transformation of an optimal solution for MAXSTC in terms of a maximum independent set of the *line-incompatibility* graph. The line-incompatibility (also known under the term *Gallai graph* [24, 96]), denoted by $\Gamma(G)$, graph has a node uv in $\Gamma(G)$ for every edge $\{u, v\}$ of G , and two nodes uv, vw of $\Gamma(G)$ are adjacent if and only if the vertices u, v, w induce a P_3 in G . The connection between a maximum independent set in $\Gamma(G)$ and a solution for MAXSTC in G was proved in Chapter 3, Proposition 3.2.2 and it is the following:

Proposition 3.2.2: For any graph G , a subset E_S of edges span $E_S(G)$ if and only if the nodes corresponding to E_S form $I_{\max}(\Gamma(G))$.

Let G and H be two vertex-disjoint graphs. The *cartesian product* of G and H , denoted by $G \times H$, is the graph with the vertex set $V(G) \times V(H)$ and any two vertices (u, u') and (v, v') are adjacent in $G \times H$ if and only if either $u = v$ and u' is adjacent to v' in H , or $u' = v'$ and u is adjacent to v in G . We are interested in computing a maximum independent set of $G \times H$ whenever G and H are cographs. We first characterize the graph $\Gamma(G \otimes H)$ in terms of $G \times H$.

Lemma 4.3.7. *Let G and H be two vertex-disjoint cographs. Then, $\Gamma(G \otimes H) = \Gamma(G) \oplus \Gamma(H) \oplus (\overline{G} \times \overline{H})$.*

Proof. Notice that $G \otimes H$ is a connected cograph, as every vertex of G is adjacent to every vertex of H . The edges of $G \otimes H$ can be partitioned into the following sets of edges: $E(G)$, $E(H)$, and $E(G, H)$ where $E(G, H)$ is the set of edges between G and H in $G \otimes H$. By definition the nodes of $\Gamma(G)$ and $\Gamma(H)$ correspond to the sets $E(G)$ and $E(H)$. Moreover since G and H are vertex-disjoint graphs, $\Gamma(G)$ and $\Gamma(H)$ are also node-disjoint. This means that there are no common endpoints in the edges inside G and H . Hence every node of $\Gamma(G)$ is non-adjacent to all nodes of $\Gamma(H)$.

Next we show that every node of $\Gamma(G \otimes H)$ that corresponds to an edge of $E(G, H)$ is non-adjacent to the nodes of $\Gamma(G)$ and $\Gamma(H)$. If a node xy of $\Gamma(G)$ is adjacent to a node xa of $E(G, H)$ then a is a vertex of H and $\{y, a\}$ is not an edge of $G \otimes H$ contradicting the adjacency between the vertices of G and H . Symmetric arguments show that any node of $\Gamma(H)$ is non-adjacent to any node of $E(G, H)$. Thus no node that corresponds to an edge of $E(G, H)$ is adjacent to any node of $\Gamma(G) \oplus \Gamma(H)$.

To complete the proof we need to show that graph of $\Gamma(G \otimes H)$ induced by the nodes of $E(G, H)$ is exactly the graph $\overline{G} \times \overline{H}$. Let x, y be two vertices of G and let w, z be two vertices of H . By the definition of $\Gamma(G \otimes H)$, two nodes xw, yz are adjacent if and only if either $x = y$ and w is non-adjacent to z in H (so that w is adjacent to z in \overline{H}), or $w = z$ and x is non-adjacent to y in G (so that x is adjacent to y in \overline{G}). Such an adjacency corresponds exactly to the definition of the cartesian product of \overline{G} and \overline{H} . Therefore the graph of $\Gamma(G \otimes H)$ induced by the nodes of $E(G, H)$ is exactly the graph $\overline{G} \times \overline{H}$. \square

Now we are ready to give the characterization of a maximum independent set of the cartesian product of cographs, in terms of their greedy independent set partition. Although a polynomial-time algorithm for computing such a maximum independent set has already been claimed earlier [71], no characterization is proposed nor an explicit bound on the running time is reported.

Theorem 4.3.8. *Let G and H be two vertex-disjoint cographs with greedy independent set partitions $\mathcal{I} = (I_1, \dots, I_q)$ and $\mathcal{I}' = (I'_1, \dots, I'_{q'})$, respectively. Then the vertices of $(I_1 \times I'_1) \oplus \dots \oplus (I_\ell \times I'_\ell)$ form a maximum independent set of $G \times H$, where $\ell = \min\{q, q'\}$. Moreover $I_{\max}(G \times H)$ can be computed in $O(n^2)$ time, where $n = \max\{|V(G)|, |V(H)|\}$.*

Proof. Let (C_1, \dots, C_p) and $(C'_1, \dots, C'_{p'})$ be greedy clique partitions of G and H , respectively. By Lemma 4.3.5, we know that $E_S(G \otimes H) = E_S(G) \oplus E_S(H) \cup E(G, H)$, where $E(G, H) = C_1(G, H) \cup \dots \cup C_k(G, H)$ and $k = \min\{p, p'\}$. Notice that if

(C_1, \dots, C_p) is a greedy clique partition for G then (C_1, \dots, C_p) is a greedy independent set partition for \overline{G} . Moreover, by Proposition 3.2.2, we know that the edges of $E_S(G \otimes H)$ correspond to the nodes of $I_{\max}(\Gamma(G \otimes H))$. Since $\Gamma(G \otimes H) = \Gamma(G) \oplus \Gamma(H) \oplus (\overline{G} \times \overline{H})$ by Lemma 4.3.7, we get $E(G, H) = I_{\max}(\overline{G} \times \overline{H})$. Therefore, the vertices of $(I_1 \times I'_1) \oplus \dots \oplus (I_\ell \times I'_\ell)$ consist a $I_{\max}(\Gamma(G \otimes H))$.

For the running time, we need to compute two greedy independent set partitions (I_1, \dots, I_q) and $(I'_1, \dots, I'_{q'})$ for G and H , respectively, and then combine each of I_j with I'_j , for $1 \leq j \leq \ell$. Computing a greedy independent set partition for a cograph G can be done in $O(n^2)$ time by applying the algorithm on \overline{G} given in the proof of Theorem 4.3.6. Therefore, the total running time is bounded by $O(|V(G)|^2 + |V(H)|^2)$. \square

4.4 Graphs of Low Maximum Degree

Here we study the influence of the bounded degree in a graph for the MAXSTC problem. We show an interesting complexity dichotomy result: for graphs of maximum degree four MAXSTC remains NP-complete, whereas for graphs of maximum degree three the problem has a polynomial solution.

We prove the hardness result even on a proper subclass of graphs with maximum degree four. A graph G is a *4-regular K_4 -free graph*, if every vertex of G has degree four and there is no K_4 in G . The decision version of MAXSTC takes as input a graph G and an integer k and asks whether there is a strong-weak labeling of G that satisfies the strong triadic closure with at least k strong edges. Similarly the decision version of CLUSTER DELETION takes as input a graph G and an integer k and asks whether G has a spanning cluster subgraph by removing at most k edges. It is known that the decision version of CLUSTER DELETION on connected 4-regular K_4 -free graphs is NP-complete [85].

Theorem 4.4.1. *The decision version of MAXSTC is NP-complete on connected 4-regular K_4 -free graphs.*

Proof. We give a polynomial-time reduction to MAXSTC from the CLUSTER DELETION problem on connected 4-regular K_4 -free graphs which is already known to be NP-complete [85]. Let $G = (V, E)$ be a connected 4-regular K_4 -free graph with $n = 3q$ and $2n$ edges. Let $E_C(G)$ be a solution for the CLUSTER DELETION with $k = n$ edges. It is not difficult to see that every connected component of $E_C(G)$ is a triangle, since the graph is 4-regular and K_4 is a forbidden graph [85]. Then $E_C(G)$ is a solution for MAXSTC with at least n strong edges.

For the opposite direction, assume that $E_S(G)$ is a solution for MAXSTC with at least n strong edges. We show that the graph spanned by the strong edges of $E_S(G)$ is a

two-regular graph. That is, every vertex of G has exactly two strong neighbors. Assume that there is a vertex v that has at least three strong neighbors. By the strong triadic closure all its strong neighbors must induce a clique in G . Then $N[v]$ induces a K_4 which is a forbidden subgraph. Thus every vertex has at most two strong neighbors. Furthermore if there is a vertex having only one strong neighbor then $|E_S(G)| < n$ which contradicts the assumption of n strong edges. Hence every vertex has exactly two strong neighbors in $E_S(G)$.

Since $E_S(G)$ is a 2-regular graph we know that the graph spanned by the strong edges is the disjoint union of triangles or chordless cycles C_p , with $4 \leq p \leq n$. Let us also rule out that a connected component of $E_S(G)$ is a chordless cycle on four vertices C_4 . To see this, observe that if there is a C_4 in $E_S(G)$ then the four vertices of the C_4 induce a K_4 in G . Now assume that there is a connected component of $E_S(G)$ that is a chordless cycle C_p with $4 < p < n$. In such a connected component, every vertex belongs to two distinct P_3 's as an endpoint. More precisely, let v_1, \dots, v_p be the vertices of C_p such that $\{v_i, v_{i+1}\}$ and $\{v_p, v_1\}$ are strong edges with $1 \leq i < p$. Then, for every vertex v_i of C_p there two P_3 's v_{i-2}, v_{i-1}, v_i and v_i, v_{i+1}, v_{i+2} such that $v_{i-2} \neq v_{i+2}$. By the strong triadic closure, we know that v_i is adjacent to both v_{i-2} and v_{i+2} in G . Since G is a 4-regular graph, there are no more edges incident to any vertex of C_p . Thus, every vertex of C_p is non-adjacent to any other vertex of $G - C_p$ which contradicts the original connectivity of G . Therefore, either every connected component of $E_S(G)$ is a triangle or $E_S(G)$ is connected and $E_S(G) = C_n$.

If every connected component of $E_S(G)$ is a triangle then clearly $E_S(G)$ spans a cluster graph. Suppose that $E_S(G) = C_n$. Since $n = 3q$, we can partition the vertices of C_n into q triangles with the same number of strong edges as follows. For every triplet of vertices v_i, v_{i+1}, v_{i+2} , $1 \leq i \leq n - 2$, we further label the edge $\{v_i, v_{i+2}\}$ strong and we label both edges $\{v_{i+2}, v_{i+3}\}$ and $\{v_n, v_1\}$ weak. Observe that $\{v_i, v_{i+2}\}$ is an edge of G , since both $\{v_i, v_{i+1}\}$, $\{v_{i+1}, v_{i+2}\}$ are strong edges. Such a labeling satisfies the strong triadic closure property and maintain the same number of strong edges. Therefore in every case a solution for MAXSTC with n edges can be equivalently transformed into a solution for CLUSTER DELETION with n edges. \square

We can also obtain lower bounds for the running time of MAXSTC with respect to the integer k (size of the solution) or the number of vertices n . For that purpose, we make use of the *exponential-time hypothesis*: it states that k -SAT, $k \geq 3$, cannot be solved in time $2^{o(n)}$ or $2^{o(m)}$ where n is the number of variables and m is the number of clauses in the given k -CNF formula (see for e.g., [74, 101, 123]). In this context, algorithms with running time $2^{o(p)}$ for some parameter p are called *subexponential-time* algorithms.

A subexponential-time algorithm for MAXSTC would imply an algorithm for solv-

ing CLUSTER DELETION that has running time subexponential in the size of the solution k or the number of vertices n . However, CLUSTER DELETION does not admit such subexponential-time algorithms even if we restrict to graphs of maximum degree four [85]. Since we can reduce CLUSTER DELETION to MAXSTC instances on the same graph with $k = n$, we arrive at the following.

Corollary 4.4.2. *MAXSTC cannot be solved in $2^{o(k)} \cdot \text{poly}(n)$ time or in $O(2^{o(n)})$ time unless the exponential-time hypothesis fails.*

Due to Proposition 3.2.2, we stress that MAXSTC reduces to finding a minimum vertex cover of $\Gamma(G)$ corresponding to the weak edges in an optimal solution. Thus MAXSTC admits algorithms with running times $2^{\Omega(k)} \text{poly}(n)$ or $O^*(c^n)$ ¹ where k is the minimum number of weak edges and $c < 2$ is a constant [30, 46].

Now let us show that if we restrict to graphs of maximum degree three then MAXSTC becomes polynomial-time solvable. Our goal is to show that there is an optimal solution for MAXSTC that is a cluster graph, since CLUSTER DELETION is solved in polynomial time on such graphs [85].

Theorem 4.4.3. *Let G be a graph with maximum degree three. Then, there is an optimal solution for MAXSTC on G that is a cluster graph.*

Proof. Observe that if there is a K_4 in G then the vertices of the K_4 form a connected component in G since no vertex can have degree more than three. Let $E_S(G)$ be the graph spanned by the strong edges in an optimal solution for MAXSTC. For a vertex v , we denote by $N_S(v)$ the strong neighbors of v . Clearly $|N_S(v)| \leq 3$. If $|N_S(v)| = 3$ then the vertices of $N[v]$ form a K_4 since the strong neighbors of v are adjacent in G , which implies that all edges of $G[N[v]]$ are strong. In what follows we assume that for every vertex v , $|N_S(v)| \leq 2$ holds.

If every connected component of $E_S(G)$ is a clique then $E_S(G)$ is a cluster graph. Assume that there is a connected component of $E_S(G)$ that is not a clique. Then, there is a $P_3 = x, y, z$ in $E_S(G)$ so that $\{x, y\}$ and $\{y, z\}$ are strong edges. Notice that y has no other strong neighbor in $E_S(G)$. We distinguish cases according to the strong neighbors of x and z .

- Let $N_S(x) = \{y\}$ and $N_S(z) = \{y\}$. Observe that $\{x, z\}$ is an edge of G by the strong triadic closure. Then, we reach a contradiction to the optimality of $E_S(G)$ since labeling the edge $\{x, z\}$ as strong does not violate the strong triadic closure.

¹The O^* notation suppresses polynomial factors of n .

- Let $N_S(x) = \{x', y\}$ and $N_S(z) = \{y\}$. Then, observe that the edge $\{y, x'\}$ is weak. We show that we can label the edge $\{y, x'\}$ as strong and the edge $\{y, z\}$ as weak without violating the strong triadic closure. Assume for contradiction that labeling the edge $\{y, x'\}$ as strong violates the strong triadic closure. Then, since there is no other strong edge incident to y , there is a strong edge $\{x', a\}$. Since $N_S(z) = \{y\}$, $a \neq z$. Then, however, we reach a contradiction to the degree of x since x is adjacent to a, x', y , and z in G . Hence, we can safely label the $\{y, x'\}$ as strong so that x, y, x' does not induce a P_3 in $E_S(G)$.
- Let $N_S(x) = \{x', y\}$ and $N_S(z) = \{z', y\}$. If $x' \neq z'$ then y must be adjacent in G to all four vertices x, z, x', z' which contradicts its degree in G . Thus $x' = z'$ and the vertices x, y, z, x' induce a K_4 in G by the strong triadic closure. This means that all edges of the K_4 are strong.

Since $|N_S(x)| \leq 2$ and $|N_S(z)| \leq 2$, we have considered all cases for the strong neighbors of x and z . Thus, we can reform the solution $E_S(G)$ for MAXSTC into a union of cliques and keep the same size. Therefore, there is a solution for CLUSTER DELETION having the same size with an optimal solution for MAXSTC. \square

By combining Theorem 4.4.3 with the fact that CLUSTER DELETION can be solved in $O(n^{1.5} \cdot \log^2 n)$ on graphs with maximum degree three [85], we get the following result.

Corollary 4.4.4. *MAXSTC can be solved in $O(n^{1.5} \cdot \log^2 n)$ time when the input graph has maximum degree three.*

CLUSTER DELETION ON INTERVAL GRAPHS AND STARLIKE GRAPHS

It is known that the decision version of CLUSTER DELETION is NP-complete on (P_5 -free) chordal graphs, whereas CLUSTER DELETION is solved in polynomial time on split graphs. The existence of a polynomial-time algorithm of CLUSTER DELETION on interval graphs, a proper subclass of chordal graphs, remained a well-known open problem. Our main contribution, in this chapter, is that we settle this problem in the affirmative, by providing a polynomial-time algorithm for CLUSTER DELETION on interval graphs. Moreover, we show that CLUSTER DELETION remains NP-complete on a natural and slight generalization of split graphs that constitutes a proper subclass of P_5 -free chordal graphs. Although the later result arises from the already-known reduction for P_5 -free chordal graphs, we give an alternative proof showing an interesting connection between edge-weighted and vertex-weighted variations of the problem. To complement our results, we provide faster and simpler polynomial-time algorithms for CLUSTER DELETION on subclasses of such a generalization of split graphs.

The results of this chapter have led to the following publications [89, 91]:

- **Cluster deletion on interval graphs and split related graphs.** Athanasios L. Konstantinidis and Charis Papadopoulos. *44th International Symposium on Mathematical Foundations of Computer Science, (MFCS 2019), Aachen, Germany, 2019. Leibniz-Zentrum für Informatik, LIPIcs 138: 12(1)-12(14), 2019.*
- **Cluster deletion on interval graphs and split related graphs.** Athanasios L. Konstantinidis and Charis Papadopoulos. *Algorithmica (to appear).*

5.1 Introduction

In graph theoretic terms, *clustering* is the task of partitioning the vertices of the graph into subsets, called *clusters*, in such a way that there should be many edges within

each cluster and relatively few edges between the clusters. In many applications, the clusters are restricted to induced cliques, as the represented data of each edge corresponds to a similarity value between two objects [68, 69]. Under the term cluster graph, which refers to a disjoint union of cliques, one may find a variety of applications that have been extensively studied [6, 20, 114]. Here we consider the CLUSTER DELETION problem which asks for a minimum number of edge deletions from an input graph, so that the resulting graph is a disjoint union of cliques. In the decision version of the problem, we are also given an integer k and we want to decide whether at most k edge deletions are enough to produce a cluster graph.

Although CLUSTER DELETION is NP-hard on general graphs [116], settling its complexity status restricted on graph classes has attracted several researchers. CLUSTER DELETION remains NP-hard on C_4 -free graphs with maximum degree four, whereas it can be solved in polynomial time on graphs having maximum degree at most three. Quite recently, Golovach et al. [55] have shown that it remains NP-hard on planar graphs. For graph classes characterized by forbidden induced subgraphs, Gao et al. [50] showed that CLUSTER DELETION is NP-hard on $(C_5, P_5, \text{bull}, \text{fork}, \text{co-gem}, 4\text{-pan}, \text{co-4-pan})$ -free graphs and on $(2K_2, 3K_1)$ -free graphs. Regarding H -free graphs, Grüttemeier et al. [62], showed a complexity dichotomy result for any graph H consisting of at most four vertices. In particular, for any graph H on four vertices with $H \notin \{P_4, \text{paw}\}$, CLUSTER DELETION is NP-hard on H -free graphs, whereas it can be solved in polynomial time on P_4 - or paw-free graphs [62]. Interestingly, CLUSTER DELETION remains NP-hard on P_5 -free chordal graphs [11].

On the positive side, CLUSTER DELETION has been shown to be solved in polynomial time on cographs [50], proper interval graphs [11], split graphs [11], and P_4 -reducible graphs [10]. More precisely, iteratively picking maximum cliques defines a clustering on the graph which actually gives an optimal solution on cographs (i.e., P_4 -free graphs), as shown by Gao et al. in [50]. In fact, the greedy approach of selecting a maximum clique provides a 2-approximation algorithm, though not necessarily in polynomial-time [33]. As the problem is already NP-hard on chordal graphs [11], it is natural to consider subclasses of chordal graphs such as interval graphs and split graphs. Although for split graphs there is a simple polynomial-time algorithm, restricted to interval graphs only the complexity on proper interval graphs was determined by giving a solution that runs in polynomial-time [11]. Settling the complexity of CLUSTER DELETION on interval graphs, was left open [11, 10, 50].

For proper interval graphs, Bonomo et al. [11] characterized their optimal solution by consecutiveness of each cluster with respect to their natural ordering of the vertices. Based on this fact, a dynamic programming approach led to a polynomial-time algorithm. It is not difficult to see that such a consecutiveness does not hold on interval graphs, as potential clusters might require to break in the corresponding vertex order-

ing. Here, we characterize an optimal solution of interval graphs whenever a cluster is required to break. In particular, we take advantage of their consecutive arrangement of maximal cliques and describe subproblems of maximal cliques containing the last vertex. One of our key observations is that the candidate clusters containing the last vertex can be enumerated in polynomial time given two vertex orderings of the graph. We further show that each such candidate cluster separates the graph in a recursive way with respect to optimal subsolutions, that enables to define our dynamic programming table to keep track about partial solutions. Thus, our algorithm for interval graphs suggests to consider a particular consecutiveness of a solution and apply a dynamic programming approach defined by two vertex orderings. The overall running time of our algorithm is $O(n^6)$ for an interval graph on n vertices and, thus, exploiting the first polynomial-time such algorithm.

Furthermore, we complement the previously-known NP-hardness of CLUSTER DELETION on P_5 -free chordal graphs, by providing a proper subclass of such graphs for which we prove that the problem remains NP-hard. This result is inspired and motivated by the very simple characterization of an optimal solution on split graphs: either a maximal clique constitutes the only non-edgeless cluster, or there are exactly two non-edgeless clusters whenever there is a vertex of the independent set that is adjacent to all the vertices of the clique except one [11]. Due to the fact that true twins belong to the same cluster in an optimal solution, it is natural to extend split graphs by allowing two vertices that do not belong to the clique to be adjacent only if they are true twins, as they are expected not to influence the solution characterization. Surprisingly, we show that CLUSTER DELETION remains NP-complete even on such a slight generalization of split graphs. This is achieved by observing that the constructed graphs given in the reduction for P_5 -free graphs [11], constitute such split-related graphs. However, here we give a different reduction that highlights an interesting connection between edge-weighted and vertex-weighted split graphs. In fact, the resulting split-related graphs are known as *starlike graphs* which are exactly the intersection graphs of subtrees of a star [19]. We then study two different classes of starlike graphs that can be viewed as the parallel of split graphs that admit disjoint clique-neighborhood (that we call *stable-like graphs*) and nested clique-neighborhood (that we call *threshold-like graphs*). For CLUSTER DELETION we provide polynomial-time algorithms on both classes of graphs. In particular, for the former case, a polynomial-time algorithm is already known and is achieved through computing a minimizer of submodular functions [11]. Here we provide a simpler and faster (linear-time) algorithm for CLUSTER DELETION on such graphs that avoids the usage of submodular minimization. In order to unify both classes, we also consider the starlike graphs that are obtained from disjoint threshold-like graphs with a common clique (that we call *laminar-like graphs*). Our general approach that uses both subroutines on stable-like

and threshold-like graphs, results in a quadratic-time algorithm for CLUSTER DELETION on laminar-like graphs.

5.2 Preliminaries

We remind here the problem of CLUSTER DELETION. Given a graph $G = (V, E)$, the goal is to compute the minimum set $F \subseteq E(G)$ of edges such that every connected component of $G \setminus F$ is a clique. Also, cluster graph is a P_3 -free graph, or equivalently, any of its connected components is a clique. Thus, the task of CLUSTER DELETION is to turn the input graph G into a cluster graph by deleting the minimum number of edges. Let $S = C_1, \dots, C_k$ be a solution of CLUSTER DELETION such that $G[C_i]$ is a clique. In such terms, the problem can be viewed as a vertex partition problem into C_1, \dots, C_k . Each C_i is simply called *cluster*. Edgeless clusters, i.e., clusters containing exactly one vertex, are called *trivial clusters*. The edges of G are partitioned into *internal* and *external* edges: an internal edge uv has both its endpoints $u, v \in C_i$ in the same cluster C_i , whereas an external edge uv has its endpoints in different clusters $u \in C_i$ and $v \in C_j$, for $i \neq j$. Then, the goal of CLUSTER DELETION is to minimize the number of external edges which is equivalent to maximize the number of internal edges. We write $S(G)$ to denote an optimal solution for CLUSTER DELETION of the graph G , that is, a cluster subgraph of G having the maximum number of edges. Given a solution $S(G)$, the number of edges incident only to the same cluster, that is the number of internal edges, is denoted by $|S(G)|$.

Definition 5.2.1. For a clique C , we say that a vertex x is C -compatible if $C \setminus \{x\} \subseteq N(x)$.

We start with few preliminary observations regarding twin vertices. Notice that for true twins x and y , if x belongs to any cluster C then y is C -compatible.

Lemma 5.2.2 ([11]). Let x and y be true twins in G . Then, in any optimal solution x and y belong to the same cluster.

The above lemma shows that we can contract true twins and look for a solution on a vertex-weighted graph that does not contain true twins. Even though false twins cannot be grouped into the same cluster as they are non-adjacent, we can actually disregard one of the false twins whenever their neighborhood forms a clique.

Lemma 5.2.3. Let x and y be false twins in G such that $N(x) = N(y)$ is a clique. Then, there is an optimal solution such that x constitutes a trivial cluster.

Proof. Let C_x and C_y be the clusters of x and y , respectively, in an optimal solution such that $|C_x| \geq 2$ and $|C_y| \geq 2$. We construct another solution by replacing both clusters by $C_x \cup C_y \setminus \{y\}$ and $\{y\}$, respectively. To see that this indeed is a solution, first observe that x is adjacent to all the vertices of $C_y \setminus \{y\}$ because $N(x) = N(y)$, and $C_x \cup C_y \setminus \{y\} \subseteq N[x]$ forms a clique by the assumption. Moreover, since $|C_x| \geq 2$ and $|C_y| \geq 2$, we know that $|C_x| + |C_y| \leq |C_x||C_y|$, implying that the number of internal edges in the constructed solution is at least as the number of internal edges of the optimal solution. \square

Moreover, we prove the following generalization of Lemma 5.2.2.

Lemma 5.2.4. *Let C and C' be two clusters of an optimal solution and let $x \in C$ and $y \in C'$. If y is C -compatible then x is not C' -compatible.*

Proof. Let S be an optimal solution such that $C, C' \in S$. Assume for contradiction that x is C' -compatible. We show that S is not optimal. Since y is C -compatible, we can move y to C and obtain a solution S_y that contains the clusters $C \cup \{y\}$ and $C' \setminus \{y\}$. Similarly, we construct a solution S_x from S , by moving x to C' so that $C \setminus \{x\}, C' \cup \{x\} \in S_x$. Notice that the S_x forms a clustering, since x is C' -compatible. We distinguish between the following cases, according to the values $|C|$ and $|C'|$.

- If $|C| \geq |C'|$ then $|S_y| > |S|$, because $\binom{|C|+1}{2} + \binom{|C'|-1}{2} > \binom{|C|}{2} + \binom{|C'|}{2}$.
- If $|C| < |C'|$ then $|S_x| > |S|$, because $\binom{|C|-1}{2} + \binom{|C'|+1}{2} > \binom{|C|}{2} + \binom{|C'|}{2}$.

In both cases we reach a contradiction to the optimality of S . Therefore, x is not C' -compatible. \square

Corollary 5.2.5. *Let C be a cluster of an optimal solution and let $x \in C$. If there is a vertex y that is C -compatible and $N[y] \subseteq N[x]$, then y belongs to C .*

Proof. Assume for contradiction that y belongs to a cluster C' different than C . Then, observe that x is C' -compatible. Indeed, for any vertex u of C' , we know $xu \in E(G)$, since u is adjacent to y and $N[y] \subseteq N[x]$. Thus, by Lemma 5.2.4 we reach a contradiction, so that $y \in C$. \square

5.3 Polynomial-time algorithm on interval graphs

Here we present a polynomial-time algorithm for the CLUSTER DELETION problem on interval graphs. A graph is an *interval graph* if there is a bijection between its vertices and a family of closed intervals of the real line such that two vertices are adjacent

if and only if the two corresponding intervals intersect. Such a bijection is called an *interval representation* of the graph, denoted by \mathcal{I} . We identify the intervals of the given representation with the vertices of the graph, interchanging these notions appropriately. Whether a given graph is an interval graph can be decided in linear time and if so, an interval representation can be generated in linear time [48]. Notice that every induced subgraph of an interval graph is an interval graph.

Let G be an interval graph. Instead of working with the interval representation of G , we consider its sequence of maximal cliques. It is known that a graph G with p maximal cliques is an interval graph if and only if there is an ordering K_1, \dots, K_p of the maximal cliques of G , such that for each vertex v of G , the maximal cliques containing v appear consecutively in the ordering (see e.g., [12]). A path $\mathcal{P} = K_1 \cdots K_p$ following such an ordering is called a *clique path* of G . Notice that a clique path is not necessarily unique for an interval graph. Also note that an interval graph with n vertices contains at most n maximal cliques. By definition, for every vertex v of G , the maximal cliques containing v form a connected subpath in \mathcal{P} .

Given a vertex v , we denote by $K_{a(v)}, \dots, K_{b(v)}$ the maximal cliques containing v with respect to \mathcal{P} , where $K_{a(v)}$ and $K_{b(v)}$ are the *first (leftmost)* and *last (rightmost)* maximal cliques containing v . Notice that $a(v) \leq b(v)$ holds. Moreover, for every edge of G there is a maximal clique K_i of \mathcal{P} that contains both endpoints of the edge. Thus, two vertices u and v are adjacent if and only if $a(v) \leq a(u) \leq b(v)$ or $a(v) \leq b(u) \leq b(v)$.

For a set of vertices $U \subseteq V$, we write $a\text{-min } U$ and $a\text{-max } U$ to denote the minimum and maximum value, respectively, among all $a(u)$ with $u \in U$. Similarly, $b\text{-min } U$ and $b\text{-max } U$ correspond to the minimum and maximum value, respectively, with respect to $b(u)$.

With respect to the CLUSTER DELETION problem, observe that for any cluster C of a solution, we know that $C \subseteq K_i$ where $K_i \in \mathcal{P}$, as C forms a clique. A vertex y is called *guarded* by two vertices x and z if

$$\min\{a(x), a(z)\} \leq a(y) \text{ and } b(y) \leq \max\{b(x), b(z)\}.$$

For a clique C , observe that y is C -compatible if and only if there exists a maximal clique K_i such that $C \subseteq K_i$ with $a(y) \leq i \leq b(y)$.

Lemma 5.3.1. *Let x, y, z be three vertices of G such that y is guarded by x and z . If x and z belong to the same cluster C of an optimal solution and y is C -compatible then $y \in C$.*

Proof. To ease the presentation, for three non-negative numbers i, j, k we write $i \in [j, k]$ if $j \leq i \leq k$ holds. Without loss of generality, assume that $a(y) \in [a(x), a(z)]$. Assume for contradiction that y belongs to another cluster C' . We apply Lemma 5.2.4

to either x and y or z and y . To do so, we need to show that x is C' -compatible or z is C' -compatible, as y is already C -compatible. Since C' is a cluster that contains y , there is a maximal clique K_i such that $C' \subseteq K_i$ with $i \in [a(y), b(y)]$.

We show that $i \in [a(x), b(x)]$ or $i \in [a(z), b(z)]$. If $i \notin [a(x), b(x)]$ then $b(x) < i \leq b(y)$, because $a(x) \leq a(y) \leq i$. As y is guarded by x and z , we know that $i \leq b(y) \leq b(z)$. Now observe that if $i < a(z)$ then $b(x) < a(z)$, implying that x and z are non-adjacent, reaching a contradiction to the fact that $x, z \in C$. Thus, $a(z) \leq i \leq b(z)$ which shows that $i \in [a(z), b(z)]$. This means that $i \in [a(x), b(x)]$ or $i \in [a(z), b(z)]$.

Hence, x or z belong to the maximal clique K_i for which $C' \subseteq K_i$. Therefore, at least one of x or z is C' -compatible and by Lemma 5.2.4 we conclude that $y \in C$. \square

Let v_1, \dots, v_n be an ordering of the vertices such that $b(v_1) \leq \dots \leq b(v_n)$. For every v_i, v_j with $b(v_i) \leq b(v_j)$, we define the following set of vertices:

$$V_{i,j} = \{v \in V(G) : \min\{a(v_i), a(v_j)\} \leq a(v) \text{ and } b(v) \leq b(v_j)\}.$$

That is, $V_{i,j}$ contains all vertices that are guarded by v_i and v_j . We write $a(i, j)$ to denote the value of $\min\{a(v_i), a(v_j)\}$ and we simply write $K_{a(j)}$ and $K_{b(j)}$ instead of $K_{a(v_j)}$ and $K_{b(v_j)}$. Notice that for a neighbor u of v_j with $u \in V_{i,j}$, we have either $a(v_j) \leq a(u)$ or $a(v_i) \leq a(u) \leq a(v_j)$. This means that all neighbors of v_j that are totally included (i.e., all vertices u such that $a(v_j) \leq a(u) \leq b(u) \leq b(v_j)$) belong to $V_{i,j}$ for any v_i with $b(v_i) \leq b(v_j)$. To distinguish such neighbors of v_j , we define the following sets:

- $U(j)$ contains the neighbors $u \in V_{i,j}$ of v_j such that $a(u) < a(v_j) \leq b(u) \leq b(v_j)$ (neighbors of v_j in $V_{i,j}$ that partially overlap v_j).
- $M(j)$ contains the neighbors $w \in V_{i,j}$ of v_j such that $a(v_j) \leq a(w) \leq b(w) \leq b(v_j)$ (neighbors of v_j that are totally included within v_j).

In the forthcoming arguments, we restrict ourselves to the graph induced by $V_{i,j}$. It is clear that the first maximal clique that contains a vertex of $V_{i,j}$ is $K_{a(i,j)}$, whereas the last maximal clique is $K_{b(j)}$.

We now explain the necessary sets that our dynamic programming algorithm uses in order to compute an optimal solution of G .

Definition 5.3.2 (Optimal solutions $A_{i,j}$). *For two vertices v_i, v_j with $b(v_i) \leq b(v_j)$,*

- $A_{i,j}$ is the value of an optimal solution for CLUSTER DELETION of the graph $G[V_{i,j}]$.

To ease the notation, when we say a cluster of $A_{i,j}$ we mean a cluster of an optimal solution of $G[V_{i,j}]$. Notice that $A_{1,n}$ is the desired value for the whole graph G , since $V_{1,n} = V(G)$.

Our task is to construct the values for $A_{i,j}$ by taking into account all possible clusters that contain v_j . To do so, we show that (i) the number of clusters containing v_j in $A_{i,j}$ is polynomial and (ii) each such candidate cluster containing v_j separates the graph in a recursive way with respect to optimal subsolutions.

Observe that if $v_i v_j \in E(G)$ then $v_i \in U(j)$ if and only if $a(v_i) < a(v_j)$, whereas $v_i \in M(j)$ if and only if $a(v_j) \leq a(v_i)$; in the latter case, it is not difficult to see that $V_{i,j} = M(j) \cup \{v_j\}$, according to the definition of $V_{i,j}$. Thus, whenever $v_i \in M(j)$ holds, we have $V_{i,j} = V_{j,j}$. The candidates of a cluster of $A_{i,j}$ containing v_j lie among $U(j)$ and $M(j)$. Let us show with the next two lemmas that we can restrict ourselves into a polynomial number of such candidates. To avoid repeating ourselves, in the forthcoming statements we let v_i, v_j be two vertices with $b(v_i) \leq b(v_j)$.

Lemma 5.3.3. *Let C be a cluster of $A_{i,j}$ containing v_j . If there is a vertex $w \in M(j)$ such that $w \in C$ then there is a maximal clique K_t with $a(v_j) \leq t \leq b(v_j)$ such that $K_t \cap M(j) \subseteq C$ and $C \cap M(j) \subseteq K_t$.*

Proof. Since $v_j, w \in C$, we know that there is a maximal clique K_t for which $C \subseteq K_t$ with $a(v_j) \leq a(w) \leq t \leq \min\{b(v_j), b(w)\}$. We show that all other vertices of $K_t \cap M(j)$ are guarded by v_j and w . Notice that for every vertex $y \in M(j)$ we already know that $a(v_j) \leq a(y)$ and $b(y) \leq b(v_j)$. Thus, for every vertex $y \in M(j)$ we have $a(v_j) = \min\{a(v_j), a(w)\} \leq a(y)$ and $b(y) \leq \max\{b(v_j), b(w)\}$. This means that all vertices of $K_t \cap M(j) \setminus \{w\}$ are guarded by v_j and w . Moreover, since $C \subseteq K_t$, we know that all vertices of $K_t \cap M(j)$ are C -compatible. Therefore, we apply Lemma 5.3.1 to every vertex of $K_t \cap M(j)$, showing that $K_t \cap M(j) \subseteq C$. Furthermore, there is no vertex of $M(j) \setminus K_t$ that belongs to C , because $C \subseteq K_t$. \square

By Lemma 5.3.3, we know that we have to pick the entire set $K_t \cap M(j)$ for constructing candidates to form a cluster that contains v_j and some vertices of $M(j)$. As there are at most n choices for K_t , we get a polynomial number of such candidate sets. We next show that we can construct polynomial number of candidate sets that contain v_j and vertices of $U(j)$. For doing so, we consider the vertices of $U(j)$ increasingly ordered with respect to their first maximal clique. More precisely, let $U(j)_{\leq a} = (u_1, \dots, u_{|U(j)|})$ be an increasingly order of the vertices of $U(j)$ such that $a(u_1) \leq \dots \leq a(u_{|U(j)|})$. The right part of Figure 5.1 illustrates the corresponding case.

Lemma 5.3.4. *Let C be a cluster of $A_{i,j}$ containing v_j and let $u_q \in U(j)_{\leq a}$. If $u_q \in C$ then every vertex of $\{u_{q+1}, \dots, u_{|U(j)|}\}$ that is C -compatible belongs to C .*

Proof. Let u be a vertex of $\{u_{q+1}, \dots, u_{|U(j)|}\}$. We show that u is guarded by u_q and v_j . By the definition of $U(j)_{\leq a}$, we know that $a(u_q) < a(u) < a(v_j)$. Moreover, observe that $b(u) \leq b(v_j)$ holds by the fact that $u \in V_{i,j}$ and $b(u_q) \leq b(v_j)$. Thus, we apply

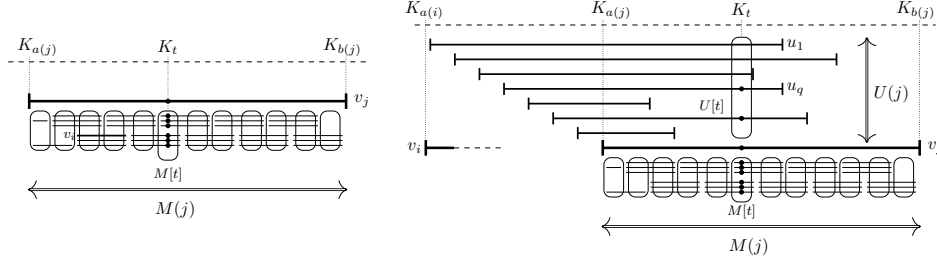


Figure 5.1: Illustrating the sets $M(j)$ and $U(j)$ for v_j . The left part shows the case in which $v_i \in M(j)$ (or, equivalently, $V_{i,j} = V_{j,j}$), whereas the right part corresponds to the case in which $a(v_i) < a(v_j)$.

Lemma 5.3.1 to u , because $u_q, v_j \in C$ and u is C -compatible, showing that $u \in C$ as desired. \square

For $a(v_j) \leq t \leq b(v_j)$, let $M[t] = K_t \cap M(j)$. Observe that each $M[t]$ may be an empty set. On the part $M(j)$, all vertices are grouped into the sets $M[a(v_j)], \dots, M[b(v_j)]$. Similar to $M[t]$, let $U[t] = U(j) \cap K_t$. Then, all vertices of $U[t]$ are $\{v_j, M[t]\}$ -compatible and all vertices of $M[t]$ are $\{v_j, U[t]\}$ -compatible. Figure 5.1 depicts the corresponding sets.

Lemma 5.3.5. *Let C be a cluster of $A_{i,j}$ containing v_j . Then, there is $a(v_j) \leq t \leq b(v_j)$ such that $M[t] \subseteq C$.*

Proof. Assume for contradiction that no set $M[t]$ is contained in C . Let $U_C = U(j) \cap C$ and let $i' = b - \min(U_C)$. Notice that $C = \{v_j\} \cup U_C$ because of the assumption as there are no other neighbors of v_j in $V_{i,j}$. Then, $a(v_j) \leq i' \leq b(v_j)$ holds, because $v_j \in C$. We show that $M[i'] \subseteq C$. Observe that $C \subseteq K_{i'}$. If $M[i'] = \emptyset$ then clearly $M[i'] \subseteq C$. Assume that $M[i'] \neq \emptyset$ and let C' be a non-empty subset of $M[i']$ that forms a cluster in $A_{i,j}$. Then, all vertices of C are C' -compatible and all vertices of C' are C -compatible, because $C, C' \in K_{i'}$. Thus, we reach a contradiction by Lemma 5.2.4 to the optimality of $A_{i,j}$. This means that there is a vertex $w \in M(j)$ that is contained in C together with v_j . Therefore, by Lemma 5.3.3, there is a set $M[t] = K_t \cap M(j)$ that is included in C . \square

All vertices of a cluster C containing v_j belong to $U(j) \cup M(j)$. Thus, $C \setminus \{v_j\}$ can be partitioned into $C \cap U(j)$ and $C \cap M(j)$. Also notice that $C \subseteq K_t$ for some $a(v_j) \leq t \leq b(v_j)$. Combined with the previous lemmas, we can enumerate all such subsets C of $U(j) \cup M(j)$ in polynomial-time. In particular, we first build all candidates for $C \cap M(j)$, which are exactly the sets $M[t]$ by Lemma 5.3.3 and Lemma 5.3.5. Then, for

each of such candidate $M[t]$, we apply Lemma 5.3.4 to construct all subsets containing the last q vertices of $U[t]_{\leq a}$. Thus, there are at most n^2 number of candidate sets from the vertices of $U(j) \cup M(j)$ that belong to the same cluster with v_j .

5.3.1 Splitting into partial solutions

We further partition the vertices of $M(j)$. Given a pivot group $M[t]$, we consider the vertices that lie on the right part of $M[t]$. More formally, for $a(v_j) \leq t < b(v_j)$, we define the set

$$B_j(t) = ((K_{t+1} \cup \dots \cup K_{b(j)}) \setminus K_t) \cap M(j).$$

The reason of breaking the vertices of the part $M(j)$ into sets $B_j(t)$ is the following.

Lemma 5.3.6. *Let C be a cluster of $A_{i,j}$ such that $\{v_j\} \cup M[t] \subseteq C$, for $a(v_j) \leq t \leq b(v_j)$. Then, for any two vertices $x \in V_{i,j} \setminus B_j(t)$ and $y \in B_j(t)$, there is no cluster of $A_{i,j}$ that contains both of them.*

Proof. First observe that $y \in (M[t+1] \cup \dots \cup M[b(j)]) \setminus M[t]$. We consider two cases for x , depending on whether $x \in M(j)$ or not. Assume that $x \in M(j)$. Then, we show that $xy \notin E(G)$. To see this, observe that by the definition of each group $M[t] = K_t \cap M(j)$, there is no maximal clique that contains both x and y . Thus, there is no cluster that contains both of them.

Now assume that $x \in U(j)$. If $x \in C$, then y does not belong to K_t , so that $y \notin C$. If $x \notin C$, then we show that x does not belong to a cluster with any vertex of $B_j(t)$. Assume for contradiction that x belongs to a cluster C' such that $C' \cap B_j(t) \neq \emptyset$. This means that $x \in K_{t'}$ with $t < t' \leq b(v_j)$ and $C' \subseteq K_{t'}$. Then v_j is C' -compatible and x is C -compatible, as both x and v_j belong to $K_t \cap K_{t'}$. Therefore, by Lemma 5.2.4 we reach a contradiction to x and v_j belonging to different clusters. \square

Definition 5.3.7 (Optimal solution $A(S)$). *For a non-empty set $S \subseteq V(G)$, we write $A(S)$ to denote the following solutions:*

- $A(S) = A_{i',j'}$, where $v_{i'}$ is the vertex of S having the smallest $a(v_{i'})$ and $v_{j'}$ is the vertex of S having the largest $b(v_{j'})$.

Having this notation, observe that $A_{i,j} = A(V_{i,j})$, for any v_i, v_j with $b(v_i) \leq b(v_j)$. However, it is important to notice that $A(S)$ does not necessarily represent the optimal solution of $G[S]$, since the vertices of S may not be consecutive with respect to $V_{i',j'}$, so that S is only a subset of $V_{i',j'}$ in the corresponding solution $A_{i',j'}$ for $A(S)$. Under the following assumptions, with the next result we show that for the chosen sets we have $S = V_{i',j'}$.

Observation 5.3.8. Let v_i, v_j be two vertices with $b(v_i) \leq b(v_j)$ and let $V_t = K_t \cap V_{i,j}$, for any maximal clique K_t of \mathcal{P} with $a(v_j) \leq t \leq b(v_j)$.

- (i) If $S_L = (V_{a(i,j)} \cup \dots \cup V_{t-1}) \setminus V_t$ then $S_L = V_{i',j'}$, where $i' = a\text{-min}(S_L)$ and $j' = b\text{-max}(S_L)$.
- (ii) If $S_R = (V_{t+1} \cup \dots \cup V_{b(v_j)}) \setminus V_t$ then $S_R = V_{i',j'}$, where $i' = a\text{-min}(S_R)$ and $j' = b\text{-max}(S_R)$.

Proof. We prove the case for $S_L = (V_{a(i,j)} \cup \dots \cup V_{t-1}) \setminus V_t$. As each V_t contains vertices of $V_{i,j}$, we have $V_{i',j'} \subseteq V_{i,j}$. Observe that either $a(v_{i'}) < a(v_{j'})$ or $a(v_{j'}) \leq a(v_{i'})$. In both cases we show that $b(v_{j'}) = t - 1$. Assume that there is a vertex $w \in S_L$ with $t - 1 < b(w)$. Then $a(w) \leq t - 1$ as $w \in S_L$, and $w \in K_t$ by the consecutiveness of the clique path. This shows that $w \notin S_L$ because $w \in V_t$. Thus, $b(v_{j'}) = t - 1$. We show that $a(v_{i'}) = \min\{a(v_i), a(v_j)\}$. If there is a vertex w in S_L with $a(w) < \min\{a(v_i), a(v_j)\}$ then $w \notin V_{i,j}$ leading to a contradiction that $V_{i',j'} \subseteq V_{i,j}$. Hence we have $a(v_{i'}) = \min\{a(v_i), a(v_j)\}$ and $b(v_{j'}) = t - 1$. Moreover, observe that by the definition of S_L , we already know that $S_L \subseteq V_{i',j'}$. Now it remains to notice that for every vertex w with $\min\{a(v_i), a(v_j)\} \leq a(w)$ and $b(w) \leq t - 1$ we have $w \in S_L$. This follows from the fact that $w \in V_{a(w)} \cup \dots \cup V_{b(w)}$ and $w \notin V_t$. Therefore we get $S_L = V_{i',j'}$. Completely symmetric arguments along the previous lines, shows the case for S_R . \square

Given the clique path $\mathcal{P} = K_1 \dots K_p$, a *clique-index* t is an integer $1 \leq t \leq p$. Let $\ell(j), r(j)$ be two clique-indices such that $a(i, j) \leq \ell(j) \leq a(v_j)$ and $a(v_j) \leq r(j) \leq b(v_j)$. We denote by $\ell_r(j)$ the minimum value of $a(v)$ among all vertices of $v \in K_{r(j)} \cap V_{i,j}$ having $\ell(j) \leq a(v)$. Clearly, $\ell(j) \leq \ell_r(j) \leq r(j)$ holds.

Definition 5.3.9 (Admissible pair and crossing). A pair of clique-indices $(\ell(j), r(j))$ is called *admissible pair* for a vertex v_j , if both

- $a(i, j) \leq \ell(j) \leq a(v_j)$ and
- $a(v_j) \leq r(j) \leq b(v_j)$ hold.

Given an admissible pair $(\ell(j), r(j))$, we define the following set of vertices:

- $C(\ell(j), r(j)) = \{z \in V_{i,j} : \ell_r(j) \leq a(z) \leq r(j) \leq b(z)\}$.

We say that a vertex u crosses the pair $(\ell(j), r(j))$ if we have $a(u) < \ell_r(j)$ and $r(j) \leq b(u)$.

Observe that all vertices of $C(\ell(j), r(j))$ induce a clique in G , because $C(\ell(j), r(j)) \subseteq K_{r(j)}$. It is not difficult to see that for a vertex u that crosses $(\ell(j), r(j))$, we have $u \notin C(\ell(j), r(j))$. We prove the following properties of $C(\ell(j), r(j))$.

Lemma 5.3.10. *Let $v_{i'}, v_{j'}$ be two vertices with $b(v_{i'}) \leq b(v_{j'})$ and let (ℓ, r) be an admissible pair for $v_{j'}$. Moreover, let v_i, v_j be the vertices of $V_{i', j'} \setminus C(\ell, r)$ having the smallest $a(v_i)$ and largest $b(v_j)$, respectively. If the vertices of $C(\ell, r)$ form a cluster in $A_{i', j'}$ then the following statements hold:*

1. $V_{i,j} = V_{i', j'} \setminus C(\ell, r)$.
2. If $a(x) \leq r \leq b(x)$ holds for a vertex $x \in V_{i,j}$, then x crosses (ℓ, r) .
3. Every vertex of $B_j(r)$ does not belong to the same cluster with any vertex of $V_{i,j} \setminus B_j(r)$.
4. Every vertex that crosses (ℓ, r) does not belong to the same cluster with any vertex $y \in V_{i,j}$ having $\ell_r \leq a(y)$.

Proof. First we show that $V_{i,j} = V_{i', j'} \setminus C(\ell, r)$. Assume that there is a vertex $v \in V_{i,j} \setminus V_{i', j'}$. Then $v \notin C(\ell, r)$ and v is distinct from v_i, v_j because, by definition, $v_i, v_j \in V_{i', j'}$. Also notice that $v \in V_{i,j}$ implies $a(i, j) \leq a(v)$ and $b(v) \leq b(v_j)$. By the second inequality, we get $b(v) \leq b(v_j) \leq b(v_{j'})$. Suppose that $a(v) < a(i', j')$. As we already know that $a(i, j) \leq a(v)$, we conclude that $a(i, j) < a(i', j')$ leading to a contradiction that $v_i, v_j \in V_{i', j'}$. Thus we have $a(i', j') \leq a(v)$ and $b(v) \leq b(v_{j'})$, showing that $v \in V_{i', j'}$. This means that $V_{i,j} \subset V_{i', j'}$, so that $V_{i,j} = V_{i', j'} \setminus C(\ell, r)$.

For the second statement, observe that if $\ell_r \leq a(x)$ then $x \in C(\ell, r)$. Since $x \in V_{i,j}$, we conclude that $x \notin C(\ell, r)$ by the first statement. Thus $a(x) < \ell_r$ holds, implying that x crosses (ℓ, r) .

With respect to the third statement, observe that no vertex of $B_j(r)$ belongs to the clique K_r . This means that all vertices of $B_j(r)$ belong to both sets $V_{i,j}$ and $V_{i', j'}$. Thus Lemma 5.3.6 and the first statement show that no two vertices $x \in V_{i,j} \setminus B_j(r)$ and $y \in B_j(r)$ belong to the same cluster.

For the fourth statement, let x be a vertex that crosses (ℓ, r) . By the first statement we know that $x \in V_{i,j}$. If $r < a(y)$ then $y \in B_j(r)$ and the third statement show that x and y do not belong to the same cluster. Suppose that $\ell_r \leq a(y) \leq r$. If $r \leq b(y)$ then $y \in C(\ell, r)$ contradicting the fact that $y \in V_{i,j}$. Putting together, we have $\ell_r \leq a(y) \leq b(y) < r$. Now assume for contradiction that x and y belong to the same cluster C_{xy} . By the fact that $a(x) < a(y)$, observe that $a(y) \leq a - \min(C_{xy}) \leq b - \min(C_{xy}) \leq \min\{b(v_j), b(y)\}$. We consider the graph induced by $V_{i', j'}$. We show that there is a vertex of C_{xy} that is $C(\ell, r)$ -compatible and there is a vertex of $C(\ell, r)$ that

is C_{xy} -compatible. Notice that x is $C(\ell, r)$ -compatible, because x crosses (ℓ, r) so that $x \in K_r$. To see that there is a vertex of $C(\ell, r)$ that is C_{xy} -compatible, choose z to be the vertex of $C(\ell, r)$ having the smallest $a(z)$. This means that $a(z) = \ell_r$. Then z is adjacent to every vertex of C_{xy} because $a(z) \leq a(y)$ and $b(y) < r \leq b(z)$. Thus, $z \in C(\ell, r)$ is C_{xy} -compatible. Therefore, Lemma 5.2.4 shows the desired contradiction, implying that x and y do not belong to the same cluster. \square

Notice that the number of admissible pairs $(\ell(j), r(j))$ for v_j is polynomial because there are at most n choices for each clique-index. Moreover, if $v_i \in M(j)$ then we have $\ell(j) = a(v_j)$.

Definition 5.3.11 (Bounding pair). *A pair of clique-indices (ℓ, r) with $\ell \leq r$ is called bounding pair for v_j if either $b(v_j) < r$ holds, or v_j crosses (ℓ, r) . Given an bounding pair (ℓ, r) for v_j , we write $(\ell(j), r(j)) \prec (\ell, r)$ to denote the set of admissible pairs $(\ell(j), r(j))$ for v_j such that*

- $r(j) \leq b(v_j)$, whenever $b(v_j) < r$ holds, and
- $r(j) < \ell$, otherwise.

Observe that if $b(v_j) < r$ holds, then $(\ell(j), r(j)) \prec (\ell, r)$ describes all admissible pairs for v_j with no restriction, regardless of ℓ . On the other hand, if $\ell < a(v_j)$ and $r \leq b(v_j)$ hold, then (ℓ, r) is not a bounding pair for v_j . In fact, we will show that the latter case will not be considered in our partial subsolutions. For any admissible pair $(\ell(j), r(j))$ and any bounding pair (ℓ, r) for v_j , observe that $v_j \in C(\ell(j), r(j))$ and $v_j \notin C(\ell, r)$. Intuitively, an admissible pair $(\ell(j), r(j))$ corresponds to the cluster containing v_j , whereas a bounding pair (ℓ, r) forbids v_j to select certain vertices as they have already formed a cluster that does not contain v_j .

Our task is to construct subsolutions over all admissible pairs for v_j with the property that the vertices of $C(\ell(j), r(j))$ form a cluster. To do so, we consider a vertex $v_{j'}$ with $b(v_j) \leq b(v_{j'})$ and a cluster containing $v_{j'}$. Let (ℓ, r) be an admissible pair for $v_{j'}$ such that $a(v_j) \leq r \leq b(v_j)$. The previous results suggest to consider solutions in which the vertices of $C(\ell, r)$ form a cluster in an optimal solution. It is clear that if $\ell \leq a(v_j)$ then $v_j \in C(\ell, r)$. Moreover, if $b(v_j) < r$, then no vertex of $V_{i,j}$ belongs to $C(\ell, r)$. Thus, we need to construct solutions for $A_{i,j}$, whenever (ℓ, r) is a bounding pair for v_j and the vertices of $C(\ell, r)$ form a cluster. Such an idea is formally described in the following restricted solutions.

Definition 5.3.12 ((ℓ, r) -restricted solution). *Let (ℓ, r) be a bounding pair for v_j . We call the following solution, (ℓ, r) -restricted solution:*

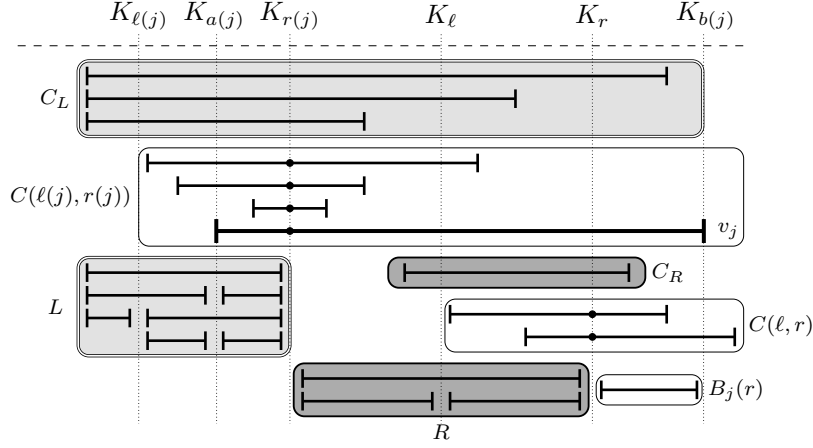


Figure 5.2: A partition of the set of vertices given in $A_{i,j}[\ell, r]$, where $V_L = C_L \cup L$ and $V_R = C_R \cup R$. Observe that $B_j(r(j)) = R \cup C_R \cup (C(\ell, r) \cap V_{i,j}) \cup B_j(r)$.

- $A_{i,j}[\ell, r]$ is the value of an optimal solution for CLUSTER DELETION of the graph $G[V_{i,j} - (C(\ell, r) \cup B_j(r))]$ such that the vertices of $C(\ell, r)$ form a cluster.

Hereafter, we assume that $B_j(t)$ with $t \geq b(v_j)$ corresponds to an empty set. Figure 5.2 illustrates a partition of the vertices with respect to $A_{i,j}[\ell, r]$. Notice that an optimal solution $A_{i,j}$ without any restriction is described in terms of $A_{i,j}[\ell, r]$ by $A_{i,j}[1, b(v_j) + 1]$, since no vertex of $V_{i,j}$ belongs to $C(1, b(v_j) + 1)$. Therefore, $A_{1,n}[1, n + 1]$ corresponds to the optimal solution of the whole graph G . As base cases, observe that if $V_{i,j}$ contains at most one vertex then $A_{i,j}[\ell, r] = 0$ for all bounding pairs (ℓ, r) , since there are no internal edges. For a set C , we write $|C|_2$ to denote the number $\binom{|C|}{2}$. With the following result, we describe a recursive formulation for the optimal solution $A_{i,j}[\ell, r]$, which is our central tool for our dynamic programming algorithm.

Lemma 5.3.13. *Let (ℓ, r) be a bounding pair for v_j . Then,*

$$A_{i,j}[\ell, r] = \max_{(\ell(j), r(j)) \prec (\ell, r)} (A(V_L)[\ell(j), r(j)] + |C(\ell(j), r(j))|_2 + A(V_R)[\ell, r]),$$

where $V_L = V_{i,j} \setminus (C(\ell(j), r(j)) \cup B_j(r(j)))$ and $V_R = B_j(r(j)) \setminus (C(\ell, r) \cup B_j(r))$.

Proof. We first argue that $C(\ell(j), r(j))$ corresponds to the correct cluster C containing v_j . Observe that $v_j \notin C(\ell, r)$, because (ℓ, r) is a bounding pair for v_j , so that $a(v_j) < \ell$ whenever $a(v_j) \leq r \leq b(v_j)$ holds. By Lemmas 5.3.4 and 5.3.3, there are $r(j) = t$ and $\ell(j) = k$, where $a(v_j) \leq t \leq b(v_j)$ and $k = a - \min(K_t \cap C)$, such that $C = C(\ell(j), r(j))$.

We show that such a set $C(\ell(j), r(j))$ is obtained from a correct choice among the described $(\ell(j), r(j))$. Assume first that $b(v_j) < r$. Then $A_{i,j}[\ell, r] = A_{i,j}$, because for every vertex u of $C(\ell, r)$ we know the $b(v_j) < b(u)$, so that $V_{i,j} \cap C(\ell, r) = \emptyset$. This means that $a(v_j) \leq r(j) \leq b(v_j)$ for every admissible pair $(\ell(j), r(j))$, as described in the given formula. Now assume that $r \leq b(v_j)$. Since v_j crosses (ℓ, r) , Lemma 5.3.10 (4) shows that v_j is not contained in a cluster with a vertex y having $\ell < a(y)$. Thus, for any vertex $y \in C$ we know that $y \in K_t$ where $a(v_j) \leq t < \ell$. This means that there is a set $C(\ell(j), r(j))$ that contains exactly the vertices of C such that $a(v_j) \leq r(j) < \ell$. Therefore, $(\ell(j), r(j)) \prec (\ell, r)$ holds, as desired.

Next, we consider the sets V_L and V_R . We show that $A(V_L)[\ell(j), r(j)]$ and $A(V_R)[\ell, r]$ correctly store the optimal values of each part. To do so, we show first that the vertex sets of each part correspond to the correct sets and, then, each pair $(\ell(j), r(j))$ and (ℓ, r) is indeed a bounding pair for the last vertex of V_L and V_R , respectively. We start with some preliminary observations. Notice that $B_j(r) \subseteq B_j(r(j))$, because $r(j) < r$, which means that every vertex $B_j(r)$ does not belong to $V_L \cup V_R$. Since $C(\ell(j), r(j))$ contains only vertices of $K_{r(j)}$ and $r(j) < \ell$, no vertex of $B_j(r)$ is considered in the described formula, as required in $A_{i,j}[\ell, r]$. By the properties of $C(\ell(j), r(j))$ and $C(\ell, r)$, we have the following:

- Let $x \in K_{r(j)} \cap V_{i,j}$. Then, either $x \in C(\ell(j), r(j))$ or x crosses the pair $(\ell(j), r(j))$. Moreover, if a vertex v crosses $(\ell(j), r(j))$ then $v \in V_L$.
- Let $y \in K_r \cap V_{i,j}$. Then, either $y \in C(\ell, r)$ or y crosses the pair (ℓ, r) . Moreover, if a vertex v crosses (ℓ, r) but does not cross $(\ell(j), r(j))$ then $v \in V_R$.

Let C_L be the set of vertices of $V_{i,j}$ that cross $(\ell(j), r(j))$ and let C_R be the set of vertices of $V_{i,j} \setminus C_L$ that cross (ℓ, r) . The previous properties imply that we can partition V_L to the vertices of C_L and the vertices of $V_{i,j}$ that belong to $L = (K_{a(i,j)} \cup \dots \cup K_{r(j)-1}) \setminus K_{r(j)}$. Similarly, V_R is partitioned to the vertices of C_R and the vertices of $V_{i,j}$ that belong to $R = (K_{r(j)+1} \cup \dots \cup K_{r-1}) \setminus (K_{r(j)} \cup K_r)$. See Figure 5.2 for an exposition of the corresponding sets. Thus, we have the following partitions for V_L and V_R :

- $V_L = C_L \cup L$, where $L = ((K_{a(i,j)} \cup \dots \cup K_{r(j)-1}) \setminus K_{r(j)}) \cap V_{i,j}$.
- $V_R = C_R \cup R$, where $R = ((K_{r(j)+1} \cup \dots \cup K_{r-1}) \setminus (K_{r(j)} \cup K_r)) \cap V_{i,j}$.

Let $v_{i'}$, $v_{j'}$ be the vertices of V_L with $i' = a - \min(V_L)$ and $j' = b - \max(V_L)$. We now show that $A(V_L)[\ell(j), r(j)]$ corresponds to the optimal solution of the graph $G[V_{i',j'}] - (B_{j'}(r(j)) \cup C(\ell(j), r(j)))$ such that the vertices of $C(\ell(j), r(j))$ form a cluster. Assume for contradiction that there is a vertex x of $V_{i',j'} \setminus (C(\ell(j), r(j)) \cup B_{j'}(r(j)))$ that does not belong to $V_L = V_{i,j} \setminus (B_j(r) \cup C(\ell, r))$. First notice that $K_{r(j)} \cap V_{i,j} = C(\ell(j), r(j))$

if and only if C_L is an empty set. In such a case, by Observation 5.3.8, we have $V_{i',j'} = V_{i,j} \setminus (K_{r(j)} \cup \dots \cup K_{b(j)})$, contradicting the existence of such a vertex x . Suppose that $v_{i'} \neq v_i$. Then $v_i \in M(j)$ or $v_i \in C(\ell(j), r(j))$, because $\min\{a(v_i), a(v_j)\}$ is the first maximal clique of all vertices of $V_{i,j}$. If $v_i \in M(j)$ then $U(j) = \emptyset$ and $\ell(j) = a(j)$. This means that for every $a(v_j) \leq r(j) \leq b(v_j)$, we have $K_{r(j)} \cap V_{i,j} = C(\ell(j), r(j))$, reaching a contradiction. If $v_i \in C(\ell(j), r(j))$ then $\ell(j) = a(v_i)$ and C_L is empty, reaching again a contradiction. Suppose now that $i' = i$. It is clear that $x \neq v_{j'}$. If $v_{j'} \in L$ then $C_L = \emptyset$, so that $K_{r(j)} \cap V_{i,j} = C(\ell(j), r(j))$. Assume that $v_{j'} \in C_L$. Now observe that if $x \in L \cup C_L$, then x is a vertex of $V_{i,j} \setminus (B_j(r) \cup C(\ell, r))$. Thus, $x \notin L \cup C_L$. If $b(x) < r(j)$ then $x \in L$ because $a(v_i) \leq a(x)$. This means that $r(j) \leq b(x)$. If $\ell(j) \leq a(x) \leq r(j)$ then $x \in C(\ell(j), r(j))$, leading to a contradiction that $x \in V_L$, and if $a(x) < \ell(j)$ then $x \in C_L$, leading to a contradiction that $x \notin L \cup C_L$. Thus, we know that $r(j) < a(x)$ and $b(x) \leq b(v_{j'})$. This, however, implies that $x \in B_{j'}(r(j))$, reaching a contradiction to the fact that $x \in V_{i',j'} \setminus B_{j'}(r(j))$. Therefore, we have shown that an optimal solution of the vertices of $V_{i',j'} \setminus (B_{j'}(r(j)) \cup C(\ell(j), r(j)))$ corresponds to an optimal solution of the vertices of V_L .

Furthermore, we argue that $(\ell(j), r(j))$ is a bounding pair for $v_{j'}$ in $A(V_L)[\ell(j), r(j)]$. Assume that $r(j) \leq b(v_{j'})$. If $r(j) \leq a(v_{j'})$ then $v_{j'} \in B_j(r(j))$, because $a(v_j) \leq r(j)$. As $v_{j'} \in V_L$, we have $a(v_{j'}) < r(j) \leq b(v_{j'})$. Then, if $\ell(j) \leq a(v_{j'})$, we get $v_{j'} \in C(\ell(j), r(j))$, which implies that $a(v_{j'}) < \ell(j)$, showing that $(\ell(j), r(j))$ is a bounding pair for $v_{j'}$. Assume next that $b(v_{j'}) < r(j)$. Then, $v_{j'} \notin C_L$, implying that $C_L = \emptyset$. Thus, for any value of $\ell(j)$ we know that $(\ell(j), r(j))$ is a bounding pair for $v_{j'}$. Therefore, $A(V_L)[\ell(j), r(j)]$ corresponds to the optimal solution of the graph $G[V_{i',j'}] - (B_{j'}(r(j)) \cup C(\ell(j), r(j)))$.

Next we consider the vertices of V_R , in order to show that $A(V_R)[\ell, r]$ corresponds to an optimal solution of the graph $G[V_R]$. Let $v_{i''}, v_{j''}$ be the vertices of V_R with $i'' = a\text{-min}(V_R)$ and $j'' = b\text{-max}(V_R)$. Assume for contradiction that there is a vertex x of $V_{i'',j''} \setminus (C(\ell, r) \cup B_{j''}(r))$ that does not belong to $V_R = B_j(r(j)) \setminus (C(\ell, r) \cup B_j(r))$. Every vertex of $R \cup C_R$ belongs to V_R , so that $x \notin R \cup C_R$. This means that $b(x) > r$, since $x \notin R$, and $a(x) > r$, since $x \notin C_R \cup C(\ell, r)$. Then we obtain $r < a(x) \leq b(x) \leq b(v_{j''})$, showing that $x \in B_{j''}(r)$. Thus we reach a contradiction, because $B_{j''}(r) \subseteq B_j(r)$. Hence, the vertices described in $A(V_R)[\ell, r]$ correspond to the vertices of V_R , as desired.

With respect to $A(V_R)[\ell, r]$, it remains to show that (ℓ, r) is a bounding pair for $v_{j''}$. If $b(v_{j''}) < r$ then $C_R = \emptyset$, which means that (ℓ, r) is a bounding pair for $v_{j''}$. Next suppose that $r \leq b(v_{j''})$. If $r \leq a(v_{j''})$ then $v_{j''} \in B_j(r)$, contradicting the fact that $v_{j''} \in V_R$. Thus, we know that $a(v_{j''}) < r \leq b(v_{j''})$. If further $\ell \leq a(v_{j''})$, then $v_{j''} \in C(\ell, r)$, contradicting $v_{j''} \in V_R$. Hence, we conclude that $v_{j''}$ crosses (ℓ, r) , showing that (ℓ, r) is indeed a bounding pair for $v_{j''}$.

To complete the proof, observe that no vertex of V_L belongs to the same cluster with a vertex of V_R by Lemma 5.3.10 (3). Thus, the optimal solutions described by $A(V_L)[\ell(j), r(j)]$ and $A(V_R)[\ell, r]$ do not overlap in $A_{i,j}[\ell, r]$. Therefore, the claimed formula holds. \square

Now we are ready to obtain our main result, namely a polynomial-time algorithm for CLUSTER DELETION on interval graphs.

Theorem 5.3.14. *CLUSTER DELETION is polynomial-time solvable on interval graphs.*

Proof. We describe a dynamic programming algorithm that computes $A_{1,n}$ based on Lemma 5.3.13. In a preprocessing step, we first compute two orderings of the vertices according to their first $a(v)$ and last $b(v)$ maximal cliques. Then we visit all vertices in ascending order with respect to $b(v_j)$ and for each such vertex v_j we consider the vertices v_i with $b(v_i) \leq b(v_j)$ in descending order with respect to $b(v_i)$. In such a way, we construct the sets $V_{i,j}$. We use a table $\mathcal{T}[i, j, \ell, r]$ to store the values of each $A_{i,j}[\ell, r]$. At the end, we output the maximum value of $\mathcal{T}[1, n, n+1, n+1]$ that corresponds to $A_{1,n}[n+1, n+1]$, as already explained. Regarding the running time, observe that the number of our table entries is at most n^4 , as each table index is bounded by n . Moreover, computing a single table entry requires $O(n^2)$ time, since we take the maximum of at most (ℓ, r) table entries. Therefore, the overall running time of the algorithm is $O(n^6)$. \square

5.4 Cluster Deletion on starlike graphs

A graph $G = (V, E)$ is a *split graph* if V can be partitioned into a clique C and an independent set I , where (C, I) is called a *split partition* of G . Split graphs are characterized as $(2K_2, C_4, C_5)$ -free graphs [45]. They form a subclass of the larger and widely known graph class of *chordal graphs*, which are the graphs that do not contain induced cycles of length 4 or more as induced subgraphs. In general, a split graph can have more than one split partition and computing such a partition can be done in linear time [67].

Hereafter, for a split graph G , we denote by (C, I) a split partition of G in which C is a maximal clique. It is known that CLUSTER DELETION is polynomial-time solvable on split graphs [11]. In fact, the algorithm given in [11] is characterized by its simplicity due to the following elegant characterization of an optimal solution: if there is a vertex $v \in I$ such that $N(v) = C \setminus \{w\}$ and w has a neighbor v' in I then the non-trivial clusters of an optimal solution are $C \setminus \{w\} \cup \{v\}$ and $\{w, v'\}$; otherwise, the only non-trivial cluster of an optimal solution is C [11]. Here we study whether such a simple characterization can be extended into more general classes of split graphs. Due to Lemma 5.2.2,

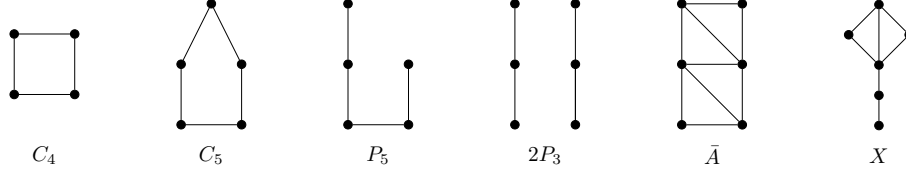


Figure 5.3: The list of forbidden induced subgraph characterization for starlike graphs.

it is natural to consider true twins of $V \setminus C$, as they are grouped together in an optimal solution and they are expected not to influence the solution characterization¹. Surprisingly, we show that CLUSTER DELETION remains NP-complete even on such a slight generalization of split graphs. Before presenting our NP-completeness proof, let us first show that such graphs form a proper subclass of P_5 -free chordal graphs. We start by giving the formal definition of such graphs that were introduced in [19].

Definition 5.4.1. A graph $G = (V, E)$ is called *starlike graph* if its vertex set can be partitioned into C and I such that $G[C]$ is a clique and every two vertices of I are either non-adjacent or true twins in G .

It is clear that in a starlike graph G with vertex partition (C, I) the following holds:

- (i) each connected component of $G[I]$ is a clique and forms a true-twin set in G , and
- (ii) contracting the connected components of $G[I]$ results in a split graph, denoted by G^* .

Starlike graphs are exactly the intersection graphs of subtrees of a star [19]. In fact, a forbidden induced subgraph characterization was already given in [19]; figure 5.3 illustrates the induced subgraphs that are forbidden in a starlike graph. Despite the known forbidden subgraph characterization, here we give a shorter and different proof of such a characterization.

Proposition 5.4.2. A graph G is starlike if and only if it does not contain any of the graphs $C_4, C_5, P_5, 2P_3, \bar{A}, X$ as induced subgraphs.

Proof. Let \mathcal{F} be the list of such subgraphs, i.e., $\mathcal{F} = \{C_4, C_5, P_5, 2P_3, \bar{A}, X\}$. We show that starlike graphs are exactly the \mathcal{F} -free graphs. It is clear that any subgraph of \mathcal{F} does not contain true twins. Moreover, each subgraph of $\mathcal{F} \setminus \{C_4, C_5\}$ contains an

¹Note that the class of split graphs is closed under the addition of true twins in the clique.

induced $2K_2$, which implies that all such subgraphs of F are not starlike graphs. Thus, if a graph G contains one of the subgraphs of \mathcal{F} then G is not a starlike graph.

We show that any \mathcal{F} -free graph G is starlike. If G is a split graph then, by definition, G is starlike. Assume that G is not a split graph. Since G does not contain C_4 or C_5 and split graphs are exactly the $(2K_2, C_4, C_5)$ -free graphs, there is an induced $2K_2$ in G . Let x_1x_2 and y_1y_2 be the two edges of an induced $2K_2$. We show that the endpoints of at least one of the two edges are true twins. Assume for contradiction that neither x_1, x_2 nor y_1, y_2 are true twins in G . Let a be a neighbor of x_1 that is non-adjacent to x_2 , and let b be a neighbor of y_1 that is non-adjacent to y_2 . We show that the vertices of $\{a, x_1, x_2, b, y_1, y_2\}$ induce one of the subgraphs of \mathcal{F} , contradicting the fact that no pair of vertices form true twins. If $b \notin N(\{x_1, x_2\})$ and $a \notin N(\{y_1, y_2\})$ then there is an induced P_5 or $2P_3$ depending on whether a and b are adjacent or not. Thus, $b \in N(\{x_1, x_2\})$ or $a \in N(\{y_1, y_2\})$. Observe that if a is adjacent to at least one of y_1 or y_2 then a is adjacent to both y_1 and y_2 ; otherwise, $\{x_1, x_2, a, y_1, y_2\}$ induce a P_5 . By symmetric arguments we know that either b is adjacent to both x_1, x_2 or to none. Without loss of generality, assume that $bx_1, bx_2 \in E(G)$.

- Suppose that a and b are non-adjacent. If $a \notin N(\{y_1, y_2\})$ then there is a P_5 induced by $\{a, x_1, b, y_1, y_2\}$. Moreover, by the previous argument, we know that if $a \in N(\{y_1, y_2\})$ then $ay_1, ay_2 \in E(G)$, which implies a C_4 in G induced by $\{a, x_1, b, y_1\}$. Thus if $ab \notin E(G)$ we obtain a induced subgraph of F .
- Suppose that a and b are adjacent. If $a \notin N(\{y_1, y_2\})$, then all six vertices induce an X graph. Otherwise, we know that $ay_1, ay_2 \in E(G)$, showing that all six vertices induce a graph \bar{A} , where a and b are the degree four vertices.

Thus in all cases we obtain an induced subgraph of \mathcal{F} , reaching to a contradiction that G being an \mathcal{F} -free graph. This means that for any $2K_2$ we know that at least one of the two edges contains true twin vertices in G . By iteratively picking such true twins and contracting them into a new vertex, results in a graph G^* that does not contain $2K_2$. Therefore G^* is a split graph, implying that G is a starlike graph. \square

Thus by Proposition 5.4.2, starlike graphs form a proper subclass of P_5 -free chordal graphs, i.e., of (C_4, C_5, P_5) -free graphs. Now let us show that decision version of CLUSTER DELETION is NP-complete on starlike graphs. This is achieved by observing that the constructed graphs given in the reduction for P_5 -free graphs [11], constitute such split-related graphs. In particular, the reduction shown in [11] comes from the X3C problem: given a universe X of $3q$ elements and a collection $C = \{C_1, \dots, C_{|C|}\}$ of 3-element subsets of X , asks whether there is a subset $C' \subseteq C$ such that every element of X occurs in exactly one member of C' . The constructed graph G is obtained by identifying the elements of X as a clique K_X and there are $|C|$ disjoint cliques $K_1, \dots, K_{|C|}$

each of size $3q$ corresponding to the subsets of C and a vertex x of K_X is adjacent to all the vertices of K_i if and only if x belongs to the corresponding subset C_i of K_i . Then, it is not difficult to see that the vertices of each K_i are true twins and the contracted graph G^* is a split graph, showing that G is indeed a starlike graph. Therefore, by the NP-completeness given in [11], we have:

Theorem 5.4.3. *CLUSTER DELETION is NP-complete on starlike graphs.*

However, here we give a different reduction that highlights an interesting connection between edge-weighted and vertex-weighted split graphs. In the EDGE WEIGHTED CLUSTER DELETION problem, each edge of the input graph is associated with a weight and the objective is to construct a clustered graph having the maximum total (cumulative) weight of edges. As already explained, we can contract true twins and obtain a vertex-weighted graph as input for the corresponding CLUSTER DELETION. Similarly, it is known that for edge-weighted graphs the corresponding EDGE WEIGHTED CLUSTER DELETION remains NP-hard even when restricted to particular variations on special families of graphs [11]. In fact, it is known [11] that EDGE WEIGHTED CLUSTER DELETION remains NP-hard on split graphs even when

- (i) all edges inside the clique have weight one,
- (ii) all edges incident to a vertex $w \in I$ have the same weight q , and
- (iii) $q = |C|$.

We abbreviate the latter problem by EWCD and denote by (C, I, k) an instance of the problem where (C, I) is a split partition of the vertices of G and k is the total weight of the edges in a cluster solution for G . With the following result, we show an interesting connection between the two variations of the problem when restricted to starlike graphs.

Theorem 5.4.4. *There exists a polynomial time algorithm that, given an instance (C, I, k) for EWCD, produces an equivalent instance for CLUSTER DELETION on starlike graphs.*

Proof. Let (C, I, k) be an instance of EWCD, where $G = (C \cup I, E)$ is a split graph. From G , we build a starlike graph $G' = (C' \cup I', E')$ by keeping the same clique $C' = C$, and for every vertex $w_j \in I$ we apply the following:

- We replace w_j by $q = |C|$ true twin vertices I'_j (i.e., by a q -clique) such that for any vertex $w' \in I'_j$ we have $N_{G'}(w') = N_G(w_j) \cup (I'_j \setminus \{w'\})$. That is, their neighbors outside I'_j are exactly $N_G(w_j)$. Moreover, the set of vertices $I'_1, \dots, I'_{|I|}$ form I' .

By the above construction, it is not difficult to see that G' is a starlike graph, since the graph induced by I' is a disjoint union of cliques and two adjacent vertices of I' are true twins in G' . Also observe that the construction takes polynomial time because q is at most $n = |V(G)|$. We claim that there is an edge weighted cluster solution for G with total weight at least k if and only if there is a cluster solution for G' having at least $k + |I| \cdot \binom{q}{2}$ edges.

Assume that there is a cluster solution S for G with total weight at least k . From S , we construct a solution S' for G' . There are three types of clusters in S :

- (a) Cluster formed only by vertices of the clique C , i.e., $Y \in S$, where $Y \subseteq C$. We keep such clusters in S' . We denote by t_a the total weight of clusters of type (a). Notice that since the weight of edges having both endpoints in C are all equal to one, t_a corresponds to the number of edges in Y .
- (b) Cluster formed only by one vertex $w_j \in I$, i.e., $\{w_j\} \in S$. In S' we replace such cluster by the corresponding clique I'_j having exactly $\binom{q}{2}$ edges. It is clear that the total weight of such clusters do not contribute to the value of S .
- (c) Cluster formed by the vertices y_1, \dots, y_p, w_j , where $y_i \in C$ and $w_j \in I$. As the weights of the edges between the vertices of y_i is one, the total number of weights in such a cluster is $\binom{p}{2} + p \cdot q$. Let t_c be the total weight of clusters of type (c). In S' we replace w_j by the vertices of I'_j and obtain a cluster S' having $\binom{p}{2} + p \cdot q + \binom{q}{2}$ number of edges.

Now observe that in S we have $t_a + t_c$ total weight, which implies $t_a + t_c \geq k$. Thus, in S' we have at least $t_a + t_c + |I| \cdot \binom{q}{2}$ edges, giving the desired bound.

For the opposite direction, assume that there is a cluster solution S' of G' having at least $k + |I| \cdot \binom{q}{2}$ edges. All vertices of I'_j are true twins and, by Lemma 5.2.2, we know that they belong to the same cluster in S' . Thus, any cluster of S' has one of the following forms:

- (i) Y' , where $Y' \subseteq C'$,
- (ii) I'_j ,
- (iii) $I'_j \cup \{y'_1, \dots, y'_p\}$, where $y'_i \in C'$.

This means that all internal edges having both endpoints in I' contribute to the value of S' by $|I| \cdot \binom{q}{2}$. Moreover, observe that for any internal edge of S' of the form $y'w'$ with $y' \in C'$ and $w' \in I'_j$, we know that there are exactly q internal edges incident to y' and the q vertices of I'_j . Thus, internal edges $y'w'$ of S' correspond to exactly one

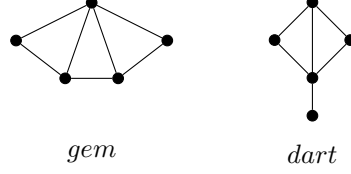


Figure 5.4: Forbidden induced subgraphs of stable-like graphs that are starlike graphs.

internal edge yw_j of S having weight q , where $y = y'$ (recall that $C = C'$) and w_j is the vertex of I associated with I_j . Hence, all internal edges outside each I_j' in S' correspond to either a weighted internal edge in S or to the same unweighted edge of the clique C in S . Therefore, there is an edge weighted solution S having weight at least k . \square

5.4.1 Polynomial-time algorithms on subclasses of starlike graphs

Due to the hardness result given in Theorem 5.4.4, it is natural to consider subclasses of starlike graphs related to their analogue subclasses of split graphs. We consider two such subclasses. The first one corresponds to the starlike graphs in which the vertices of I have no common neighbor in the clique, unless they are true or false twins. The second one is related to the true twin extension of *threshold graphs* (i.e., split graphs in which the vertices of the independent set have nested neighborhood) and form the starlike graphs in which the vertices of the I have nested neighborhood. The third one comprises a generalization of the formers and consists of the starlike graphs that are obtained from vertex-disjoint threshold graphs with a common clique. We formally define such graphs and give polynomial-time algorithms for CLUSTER DELETION on the considered graph classes. For a vertex $x \in I$ we write $N_C(x)$ to denote the set $N(x) \cap C$ and for a vertex $a \in C$ we write $N_I(a)$ to denote the set $N(a) \cap I$.

Definition 5.4.5. A starlike graph G with partition (C, I) on its vertices is called *stable-like graph* if

- $\forall x, y \in I$: either $N_C(x) \cap N_C(y) = \emptyset$ or $N_C(x) = N_C(y)$.

It is not difficult to see that in a stable-like graph, any two vertices of I having a common neighbor in C have exactly the same neighborhood in C . Before presenting our linear-time algorithm, we first give a forbidden induced subgraph characterization for the class of stable-like graphs. The graphs *gem* and *dart* are shown in Figure 5.4.

Proposition 5.4.6. A graph G is stable-like if and only if it does not contain any of the graphs C_4 , C_5 , P_5 , $2P_3$, *gem*, *dart* as induced subgraphs.

Proof. We first show that if G is stable-like then it does not contain any graph of the given list as an induced subgraph. Since G is a starlike graph, by Proposition 5.4.2 G does not contain any of $C_4, C_5, P_5, 2P_3$ as induced subgraphs. Moreover, it is not difficult to see that in any proper partition (C, I) of a *gem* or a *dart* there are no two vertices $x, y \in I$ for which $N_C(x) \cap N_C(y) = \emptyset$ or $N_C(x) = N_C(y)$. Thus, the claimed list is indeed forbidden for stable-like graphs.

For the opposite direction, we show that any starlike graph that is not stable-like contains a *gem* or a *dart* as an induced subgraph. Then, by Proposition 5.4.2 we obtain the claimed list of forbidden induced subgraphs. Let G be a starlike graph that is not stable-like, with partition (C, I) such that $|C|$ is maximum. By definition, we know that there are two vertices $x, y \in I$ such that $N_C(x) \cap N_C(y) \neq \emptyset$ and $N_C(x) \neq N_C(y)$.

- Assume that $N_C(x) \not\subseteq N_C(y)$ and $N_C(y) \not\subseteq N_C(x)$. Let $a \in N_C(x) \setminus N_C(y)$, $b \in N_C(y) \setminus N_C(x)$, and $c \in N_C(x) \cap N_C(y)$. Notice that all three vertices exist because of our assumptions. Then, $xy \notin E(G)$ because there is no C_4 in a starlike graph which means that the vertices of $\{x, y, a, b, c\}$ induce a *gem* in G .
- Assume that $N_C(x) \subsetneq N_C(y)$. There are two vertices $b, c \in C$ such that $b \in N_C(y) \setminus N_C(x)$ and $c \in N_C(x) \cap N_C(y)$. We show that y is non-adjacent to all the vertices of C . For this, observe that if $C \subseteq N_C(y)$ then $(C \cup y, I \setminus \{y\})$ is a partition of the vertices of G that respects Definition 5.4.1 and properties (i) and (ii). By the maximality of C , we obtain that there is a vertex $z \in C$ such that $z \notin N_C(y)$. Since $N_C(x) \subsetneq N_C(y)$, we know that $z \notin N_C(x)$. Thus the vertices of $\{x, y, b, c, z\}$ induce a *gem* whenever $xy \in E(G)$ or a *dart* whenever $xy \notin E(G)$.

Therefore, in all cases we obtain a *gem* or a *dart* as an induced subgraph. \square

Theorem 5.4.7. CLUSTER DELETION can be solved in time $O(n + m)$ for a stable-like graph on n vertices and m edges.

Proof. Let G be a stable-like graph with partition (C, I) . First observe that if G is disconnected then I contains isolated cliques, i.e., true twins having no neighbor in C . Thus we can restrict ourselves to a connected graph G , since by Lemma 5.2.2 each isolated clique is contained in exactly one cluster of an optimal solution. We now show that all vertices of C that have a common neighbor in I are true twins. Let u and v be two vertices of C such that $x \in N(u) \cap N(v) \cap I$. All vertices of $C \setminus \{u, v\}$ are adjacent to both u and v . Assume that there is a vertex $y \in I$ that is adjacent to u and non-adjacent to v . If $xy \in E(G)$ then by the definition of starlike graphs x and y are true twins which contradicts the assumption of $xv \in E(G)$ and $yv \notin E(G)$. Otherwise, x and y are non-adjacent and since $N_C(x) \cap N_C(y) \neq \emptyset$ we reach a contradiction to the

definition of stable-like graphs. Thus, all vertices of C that have a common neighbor in I are true twins.

We partition the vertices of C into true twin classes C_1, \dots, C_k , such that each C_i contains true twins of C . From the previous discussion, we know that any vertex of I is adjacent to all the vertices of exactly one class C_i ; otherwise, there are vertices of different classes in C that have common neighbor. For a class C_i , we partition the vertices of $N(C_i) \cap I$ into true twin classes I_i^1, \dots, I_i^q such that $|I_i^1| \geq \dots \geq |I_i^q|$.

We claim that in an optimal solution S , the vertices of each class I_i^j with $j \geq 2$ constitute a cluster. To see this, observe first that the vertices of I_i^j , $1 \leq j \leq q$, are true twins, and by Lemma 5.2.2 they all belong to the same cluster of S . Also, by Lemma 5.2.2 we know that all the vertices of C_i belong to the same cluster of S . Moreover, all vertices between different classes $I_i^j, I_i^{j'}$ are non-adjacent and are C_i -compatible. Since every vertex of I_i^j is non-adjacent to all the vertices of $V(G) \setminus (I_i^j \cup C_i)$, we know that any cluster of S that contains I_i^j is of the form either $I_i^j \cup C_i$ or I_i^j . Assume that there is a cluster that contains $I_i^j \cup C_i$ with $j \geq 2$. Then, we substitute the vertices of I_i^j by the vertices of I_i^1 and obtain a solution of at least the same size, because $|I_i^1| \geq |I_i^j|$ implies $\binom{|C_i|+|I_i^1|}{2} \geq \binom{|C_i|+|I_i^j|}{2}$. Thus, all vertices of each class I_i^j with $j \geq 2$ constitute a cluster in an optimal solution S .

This means that we can safely remove the vertices of I_i^j with $j \geq 2$, by constructing a cluster that contains only I_i^1 . Hence, we construct a graph G^* from G , in which there are only matched pair of k classes (C_i, I_i) such that (i) all sets C_i, I_i are non-empty except possibly the set I_k , (ii) $N(C_i) \cap I = I_i$, (iii) $N(I_i) = C_i$, (iv) $G^*[C_i \cup I_i]$ is a clique, and (v) $G^*[C_1 \cup \dots \cup C_k]$ is a clique. Our task is to solve CLUSTER DELETION on G^* , since for the rest of the vertices we have determined their cluster. By Lemma 5.2.2, observe that if the vertices of $C_i \cup C_j$ belong to the same cluster then the vertices of each I_i and I_j constitute two respectively clusters. Thus, for each set of vertices I_i we know that either one of $C_i \cup I_i$ or I_i constitutes a cluster in S . This boils down to compute a set M of matched pairs (C_i, I_i) from the k classes, having the maximum value

$$\sum_{(C_i, I_i) \in M} \binom{|C_i| + |I_i|}{2} + \left(\sum_{C_j \notin M} |C_j| \right) + \sum_{I_j \notin M} \binom{|I_j|}{2}.$$

Let (C_i, I_i) and (C_j, I_j) be two pairs of classes such that $|C_i| + |I_i| \leq |C_j| + |I_j|$. We show that if $(C_j, I_j) \notin M$ then $(C_i, I_i) \notin M$. Assume for contradiction that $(C_j, I_j) \notin M$ and $(C_i, I_i) \in M$. Observe that $|I_j| < \sum_{C_t \notin M \setminus C_j} |C_t|$, because I_j is C_j -compatible. Similarly, we know that $\sum_{C_t \notin M \setminus C_j} |C_t| + |C_j| \leq |I_i|$. This however, shows that $|C_j| + |I_j| < |I_i|$, contradicting the fact that $|C_i| + |I_i| \leq |C_j| + |I_j|$. Thus $(C_j, I_j) \notin M$ implies $(C_i, I_i) \notin M$.

This means that we can consider the k pair of classes (C_i, I_i) in a decreasing order

according to their number of vertices $|C_i| + |I_i|$. With a simple dynamic programming algorithm, starting from the largest ordered pair (C_1, I_1) we know that either (C_1, I_1) belongs to M or not. In the former, we add $\binom{|C_1|+|I_1|}{2}$ to the optimal value of $(C_2, I_2), \dots, (C_k, I_k)$ and in the latter we know that no pair belongs to M giving a total value of $\binom{\sum_2 |C_i|}{2} + \sum \binom{|I_i|}{2}$. By choosing the maximum between the two values, we construct a table of size k needed for the dynamic programming. Computing the twin classes and the partition (C, I) takes linear time in the size of G and sorting the pair of classes can be done $O(n)$ time, since $\sum(|C_i| + |I_i|)$ is bounded by n . Thus, the total running time is $O(n + m)$, as the dynamic programming for computing M requires $O(n)$ time. Therefore, all steps can be carried out in linear time for a stable-like graph G . \square

We next define the analogue of threshold graphs in terms of starlike graphs.

Definition 5.4.8. A starlike graph G with partition (C, I) on its vertices is called *threshold-like graph* if

- $\forall x, y \in I: N_C(x) \subseteq N_C(y)$.

It is not difficult to see that the class of threshold-like graphs and stable-like graphs are unrelated. Threshold-like graphs are also known as *starlike-threshold graphs* under the notions of intersection graphs [19]. Although the absence of an induced P_4 follows from the results of [19], we give the following short proof for completeness.

Lemma 5.4.9. Let G be a threshold-like graph. Then G is a P_4 -free graph.

Proof. We show that there is no induced path on four vertices, P_4 , in G . Assume for contradiction that there is a $P_4 = v_1v_2v_3v_4$ in G . Since $G[C]$ is a clique and $G[I]$ is a disjoint union of cliques, at least one of v_1, v_4 , say v_1 , belongs to I . If $v_4 \in C$ then $v_2 \in I$ because $v_4v_2 \notin E(G)$, which gives a contradiction as $v_1v_2 \in E(G)$ and v_1, v_2 are not true twins. Otherwise, we have $v_4 \in I$, so that $v_2, v_3 \in C$ because v_1, v_2 and v_3, v_4 are not true twins G . The latter, results again in a contradiction because $N_C(v_1) \not\subseteq N_C(v_4)$ and $N_C(v_4) \not\subseteq N_C(v_1)$. Therefore, G is a P_4 -free graph. \square

By Lemma 5.4.9 and the $O(n^2)$ -time algorithm on P_4 -free graphs (also known as *cographs*) [50, 87], CLUSTER DELETION is polynomial-time solvable on threshold-like graphs.

Next we proceed with a subclass of starlike graphs that generalizes the previous two classes, as it contains both the class of stable-like graphs and the class of threshold-like graphs.

Definition 5.4.10. A starlike graph G with partition (C, I) on its vertices is called *laminar-like graph*² if

1. $\forall x, y \in I$: either $N_C(x) \cap N_C(y) = \emptyset$ or $N_C(x) \subseteq N_C(y)$, and
2. $\forall a, b \in C$: either $N_I(a) \cap N_I(b) = \emptyset$ or $N_I(a) \subseteq N_I(b)$.

We start by characterizing the laminar-like graphs in terms of disjoint threshold-like graphs.

Lemma 5.4.11. A graph $G = (V, E)$ is a laminar-like graph with partition (C, I) if and only if $V(G)$ can be partitioned into vertex-disjoint threshold-like graphs $G_i = (C_i \cup I_i, E_i)$ such that $C = \cup C_i$, $I = \cup I_i$, and $E(G) = E(C) \cup (\cup E_i)$.

Proof. Given a laminar-like graph G with partition (C, I) , we partition the vertices of G according to whether the vertices of I have a common neighbor in C . Let I_i be a subset of I that contains all vertices $x, y \in I$ such that $N_C(x) \cap N_C(y) \neq \emptyset$ or $N_C(x) = N_C(y)$. Let also $C_i = N_C(I_i)$. We claim that $G_i = G[C_i \cup I_i]$ is a threshold-like graph. By the first property of Definition 5.4.10, for any two vertices $x, y \in I_i$ we have $N_C(x) \subseteq N_C(y)$. Let $z \in I_i$. Then $N_C(z) \subseteq N_C(y)$ by the construction of I_i . Assume for contradiction that $N_C(x) \not\subseteq N_C(z)$ and $N_C(z) \not\subseteq N_C(x)$. Let $a \in N_C(x) \setminus N_C(z)$ and $b \in N_C(z) \setminus N_C(x)$. Then observe that $y \in N_I(a) \cap N_I(b)$. Thus by the second property of Definition 5.4.10 we reach a contradiction to $N_I(a) \subseteq N_I(b)$. Therefore the vertices of I_i can be ordered as $w_1, \dots, w_{|I_i|}$ such that $N_C(w_1) \subseteq \dots \subseteq N_C(w_{|I_i|})$ which means that G_i is indeed a threshold-like graph. Moreover consider any two subgraphs $G_i = (C_i, I_i)$ and $G_j = (C_j, I_j)$ that are constructed as explained above. Then it is clear that $I_i \cap I_j = \emptyset$ and $C_i \cap C_j = \emptyset$, since the construction partitions I into equivalent classes of I . In particular, for every two vertices $w \in I_i$ and $w' \in I_j$ we have $N_C(w) \cap N_C(w') = \emptyset$. What is left to show is that there are no edges between the vertices of I_i and I_j . For this, observe that if there is an edge between $w \in I_i$ and $w' \in I_j$ then w and w' are true twins, as G is a starlike graph. Therefore we have $N_C(w) = N_C(w')$ which means that both w, w' belong to the same set I_i .

For the opposite direction, assume that we are given vertex-disjoint threshold-like graphs $G_i = (C_i \cup I_i, E_i)$. We consider the graph G obtained from the union of G_i by adding all edges among the vertices of $\cup C_i$. As there are all the edges among the vertices of C_i and C_j , we have that $C = \cup C_i$ is a clique. Moreover, each class of true twins of I_i remains a class of true twins in G . Thus G is starlike graph. We show that G is indeed a laminar-like graph by verifying the two properties of Definition 5.4.10.

²The term *laminar* comes from the notion of laminar family of sets: a family of sets is called *laminar* if any two of its sets are either disjoint or one includes the other.

For any two vertices $x, y \in I_i$ we have $N_C(x) \subseteq N_C(y)$ by Definition 5.4.8. If $x \in I_i$ and $y \in I_j$ then $N_C(x) \cap N_C(y) = \emptyset$, since there are no edges between the vertices of I_i and I_j . Similarly, for any two vertices $a, b \in C_i$ we have $N_{I_i}(a) \subseteq N_{I_i}(b)$ which means that $N_I(a) \subseteq N_I(b)$ because every vertex of $G - G_i$ is either adjacent to both a and b or non-adjacent to both a and b . Moreover, for two vertices $a \in C_i$ and $b \in C_j$, we have $N_I(a) \cap N_I(b) = \emptyset$ because $I_i \cap I_j = \emptyset$ and both a and b are adjacent to every vertex of C . Therefore G is a laminar-like graph. \square

We next show a polynomial-time algorithm for solving CLUSTER DELETION on laminar-like graphs which form the more general subclass of the considered subclasses of starlike graphs. Towards this, we apply Lemma 5.4.11, obtain an optimal solution in each G_i , and then apply the algorithm given in Theorem 5.4.7.

Theorem 5.4.12. *CLUSTER DELETION can be solved in time $O(n^2)$ for a laminar-like graph on n vertices.*

Proof. Let G be a laminar-like graph. We first compute the true twin classes and the partition (C, I) of G which can be done in linear time. By the true twin classes and Lemma 5.4.11, we compute the threshold-like induced subgraphs G_i of G . For doing so, all vertices of I , denoted by I_i , having a common neighbor in C belong to the same graph G_i , whereas all vertices of I having no neighbor in C belong to the same graph, that we denote by G_0 . Observe that the vertices of I_i define the set $C_i = N_C(I_i)$. Moreover, all adjacent vertices of I_0 are true twins in G and $N_C(I_0) = \emptyset$. Thus each connected component of G_0 is already a clique in G and forms a cluster in any optimal solution.

Consider a threshold-like graph G_i with partition (C_i, I_i) . To ease the notation, we let $H = G_i$ and (A, B) be the partition (C_i, I_i) . By Lemma 5.4.9, H is a P_4 -free graph. For P_4 -free graphs, it is known that greedily selecting maximum cliques results in an optimal solution for CLUSTER DELETION [50]. Let $S(H) = (S_1, \dots, S_k)$ be the clusters of an optimal solution of H such that S_i is a maximum clique of the graph $H - (S_1 \cup \dots \cup S_{i-1})$, for $1 \leq i \leq k$ with $S_0 = \emptyset$. We call $S(H)$ a *greedy-optimal* solution of H . Observe that all true twins of H belong to the same cluster S_i by the greedy choice of a maximum clique. We partition the vertices of each cluster $S_i \in S(H)$ with respect to (A, B) . In particular, for every $1 \leq i \leq k$, we define $A_i = S_i \cap A$ and $B_i = S_i \cap B$. Due to the construction of H , in which $N_C(B) = A$, notice that all sets B_i are non-empty, whereas a set A_i may be empty. We prove the following claim.

Claim 5.4.13. *Let $S(H) = (S_1, \dots, S_k)$ be a greedy-optimal solution of the threshold-like graph H . For every $S_i = (A_i, B_i)$, $1 \leq i \leq k$, the following hold:*

1. B_i constitutes a class of true twins.

2. For every $i < j \leq k$ with $S_j = (A_j, B_j)$, we have $|A_i| + |B_i| \geq |W| + |B_j|$, where $W = (A_i \cup \dots \cup A_k) \cap N_H(B_j)$.
3. Removing all edges with one endpoint in A_i and the other endpoint in B_j , for $j \neq i$, results in a stable-like graph.

Proof: Let $C(H) = A_1 \cup \dots \cup A_k$ and $I(H) = B_1 \cup \dots \cup B_k$. For the first statement, observe that every pair of adjacent vertices in $I(H)$ are true twins since H is a starlike graph. Thus, by Lemma 5.2.2, every set B_i constitutes a class of true twins.

For the second statement, let H' be the graph obtained from H by removing the vertices of $(A_1, B_1), \dots, (A_{i-1}, B_{i-1})$. As (A_i, B_i) is a maximum clique in H' by the greedy choice for S_i , $|A_i| + |B_i|$ is greater or equal than the size of any other (maximal) clique in H' . Any maximal clique containing B_j in H' , consists of B_j together with adjacent vertices of $A_i \cup \dots \cup A_k$. Therefore the second statement follows.

For the third statement we consider the graph H'' with vertex set $C(H) \cup I(H)$ and edge set formed by making $C(H)$ and each (A_i, B_i) a clique, for $1 \leq i \leq k$. In order to show that H'' is a stable-like graph, observe that every pair of adjacent vertices in $I(H)$ are true twins since they belong to the same set B_i . Thus H'' is a starlike graph. To conclude, we need to prove that for any two vertices x, y of $I(H)$ either $N_{C(H)}(x) \cap N_{C(H)}(y) = \emptyset$ or $N_{C(H)}(x) = N_{C(H)}(y)$. If $x, y \in B_i$ then $N_{C(H)}(x) = N_{C(H)}(y) = A_i$ by construction, and if $x \in B_i$ and $y \in B_j$ then $N_{C(H)}(x) = A_i$ and $N_{C(H)}(y) = A_j$ so that $A_i \cap A_j = \emptyset$. Therefore, H'' is indeed a stable-like graph. \diamond

Next we show that there is an optimal solution for G that *respects* the internal clusters of a greedy-optimal solution of H . That is, every cluster (A_i, B_i) in H remains a cluster in G , or is split into two clusters $A_i \cup Z$ and B_i where Z is a set of vertices of C .

Claim 5.4.14. *Let $S(H) = (S_1, \dots, S_k)$ be a greedy-optimal solution of H and let $S_i = (A_i, B_i)$ be a cluster of $S(H)$, $1 \leq i \leq k$. There is an optimal solution $S(G)$ of G such that either $A_i \cup B_i \in S(G)$, or $A_i \cup Y, B_i \in S(G)$, where $Y \subseteq C$.*

Proof: Let (C, I) be the partition of the vertices of G into a clique C and a union of cliques I . Let also $C(H) = A_1 \cup \dots \cup A_k$ and $I(H) = B_1 \cup \dots \cup B_k$. Recall that every vertex of B_i is non-adjacent to any vertex of $G - H$, whereas every vertex of A_i is adjacent to every vertex of C . Thus for any vertex $z \in V(G) \setminus V(H)$ we know that the vertices $\{z\} \cup A_i \cup B_i$ do not induce a clique in G . If there is no cluster of $S(G)$ that contains vertices of both $G - H$ and H , then every cluster (A_i, B_i) of H is a cluster of G , since $S(H)$ is an optimal solution of H . In what follows, we assume that a set Z of vertices of $G - H$ together with a set X of vertices of H constitutes a cluster in $S(G)$. It is clear $Z \subseteq C \setminus C(H)$ and $X \subseteq C(H)$. Observe also that all the vertices of Z are adjacent to every vertex of $C(H) = A_1 \cup \dots \cup A_k$ and non-adjacent to any vertex of

$I(H) = B_1 \cup \dots \cup B_k$ by Lemma 5.4.11. First we claim that there is no cluster $Z' \cup X'$ in $S(G)$ with $Z' \subseteq C \setminus (V(H) \cup Z)$ and $X' \subseteq C(H) \setminus X$. To see this, notice that all the vertices of Z are $(Z' \cup X')$ -compatible and all the vertices of Z' are $(Z \cup X)$ -compatible which by Lemma 5.2.4 is not possible.

We consider the graph $F = G[Z \cup V(H)]$. It is not difficult to see that F is a threshold-like graph, since H is a threshold-like graph and all the vertices of Z are adjacent to every vertex of $C(H)$ and non-adjacent to any vertex of $I(H)$. In particular, $(C(H), I(H) \cup Z)$ is a partition of the vertices of F where the vertices of Z are true twins. Let S_ℓ be the cluster of $S(H)$ with the smallest $1 \leq \ell \leq k$ such that $|A_\ell| + |B_\ell| < |Z| + |A_\ell| + \dots + |A_k|$. We show that there is a greedy-optimal solution (S'_1, \dots, S'_{k+1}) of F such that:

- $S'_i = S_i$, for every $1 \leq i \leq \ell - 1$,
- $S'_\ell = (A_\ell \cup \dots \cup A_k, Z)$, and
- $S'_{j+1} = (\emptyset, B_j)$, for every $\ell \leq j \leq k$.

For this, observe that for any S_i , $1 \leq i \leq \ell - 1$, we have $|A_i| + |B_i| \geq |Z| + |A_i| + \dots + |A_k|$ by the choice of S_ℓ . Thus, by Claim 5.4.13 (2), S_i is a maximum clique of $F - (S_1 \cup \dots \cup S_{i-1})$, so that $S'_i = S_i$. Due to the greedy choice and Claim 5.4.13 (2), we know that the described S'_ℓ is the maximum clique of $F - (S_1 \cup \dots \cup S_{\ell-1})$. Thus S'_ℓ is indeed a cluster of a greedy-optimal solution of F . Moreover all vertices of B_j , $\ell \leq j \leq k$, form true twins by Claim 5.4.13 (1). Since the vertices of each B_j have no neighbors in $F - (S'_1 \cup \dots \cup S'_\ell \cup B_j)$, every B_j constitutes a cluster. Therefore there is a greedy-optimal solution of F with the claimed properties. As the clusters in F remain clusters in G , we conclude the claim. \diamond

Let us now describe the remaining steps of our algorithm. Assume that every graph G_i is an induced threshold-like subgraph of G as given in Lemma 5.4.11.

1. For every G_i , compute a greedy-optimal solution $S(G_i) = (S_1^i, \dots, S_{k_i}^i)$.
2. Construct the graph G' from G by removing all edges among the vertices of S_p^i and S_q^i , for every G_i and $1 \leq p, q \leq k_i$ with $p \neq q$.
3. Run the algorithm described in Theorem 5.4.7 on G' and return the obtained solution.

For the correctness, observe that Claim 5.4.13 (3) shows that every induced subgraph of G' on the vertices of $V(G_i)$ is indeed a stable-like graph. Since the vertices of each set I_i and I_j of G_i and G_j , respectively, have no common neighbor in G' , we conclude that G' is indeed a stable-like graph. Moreover Claim 5.4.14 implies that the

constructed solution is an optimal solution of G , as required. Regarding the running time, observe that a greedy-optimal solution on each P_4 -free graph G_i can be computed in $O(n_i^2)$ time where $n_i = |V(G_i)|$ [50, 87]. The removal of the described edges and the algorithm given in Theorem 5.4.7 takes linear time. Therefore the total running of the algorithm is $O(n^2)$. \square

PARAMETERIZED ASPECTS OF STRONG SUBGRAPH CLOSURE

In this chapter, motivated by the role of triadic closures in social networks, and the importance of finding a maximum subgraph avoiding a fixed pattern, we introduce and initiate the parameterized study of the STRONG F -CLOSURE problem, where F is a fixed graph. This is a generalization of MAXSTC, whereas it is a relaxation of F -FREE EDGE DELETION. We study STRONG F -CLOSURE from a parameterized perspective with various natural parameterizations. Our main focus is on the number k of strong edges as the parameter. We show that the problem is FPT with this parameterization for every fixed graph F , whereas it does not admit a polynomial kernel even when $F = P_3$. In fact, this latter case is equivalent to the MAXSTC problem, which motivates us to study this problem on input graphs belonging to well known graph classes. We show that MAXSTC does not admit a polynomial kernel even when the input graph is a split graph, whereas it admits a polynomial kernel when the input graph is planar, and even d -degenerate. Furthermore, on graphs of maximum degree at most 4, we show that MAXSTC is FPT with the above guarantee parameterization $k - \mu(G)$, where $\mu(G)$ is the maximum matching size of G . We conclude with some results on the parameterization of STRONG F -CLOSURE by the number of edges of G that are not selected as strong.

The results of this chapter have led to the following publications [55, 56]:

- **Parameterized aspects of strong subgraph closure.** Petr A. Golovach, Pinar Heggernes, Athanasios L. Konstantinidis, Paloma T. Lima and Charis Papadopoulos. *16th Scandinavian Symposium and Workshops on Algorithm Theory, (SWAT 2018), Malmo, Sweden, 2018. Leibniz-Zentrum für Informatik, LIPIcs 101: 23(1)-23(13), 2018.*
- **Parameterized aspects of strong subgraph closure.** Petr A. Golovach, Pinar Heggernes, Athanasios L. Konstantinidis, Paloma T. Lima, and Charis Papadopoulos. *Algorithmica 82: 2006-2038, 2020.*

6.1 Introduction

Graph modification problems are at the heart of parameterized algorithms. In particular, the problem of deleting as few edges as possible from a graph so that the remaining graph satisfies a given property has been studied extensively from the viewpoint of both classical and parameterized complexity for the last four decades [38, 30, 125]. For a fixed graph F , a graph G is said to be F -free if G has no induced subgraph isomorphic to F . The F -FREE EDGE DELETION problem asks for the removal of a minimum number of edges from an input graph G so that the remaining graph is F -free. Here, we introduce a relaxation of this problem, which we call STRONG F -CLOSURE. Our problem is also a generalization of the MAXSTC problem, which asks to select as many edges as possible of a graph as *strong*, so that whenever two strong edges uv and vw share a common endpoint v , the edge uw is also present in the input graph (not necessarily strong). This problem is well studied in the area of social networks [5, 41], and its classical computational complexity has been studied recently on general graphs [118].

In the STRONG F -CLOSURE problem, we have a fixed graph F , and we are given an input graph G , together with an integer k . The task is to decide whether we can select at least k edges of G and mark them as *strong*, in the following way: whenever the subgraph of G spanned by the strong edges contains an induced subgraph isomorphic to F , then the corresponding induced subgraph of G on the same vertex subset is not isomorphic to F . The remaining edges of G that are not selected as strong, will be called *weak*. Consequently, whenever a subset S of the strong edges form a copy of F , there must be an additional strong or weak edge in G with endpoints among the endpoints of edges in S . A formal definition of the problem is easier to give via spanning subgraphs. If two graphs H and F are isomorphic then we write $H \simeq F$, and if they are not isomorphic then we write $H \not\simeq F$. Given a graph G and a fixed graph F , we say that a (not necessarily induced) subgraph H of G *satisfies the F -closure* if, for every $S \subseteq V(H)$ with $H[S] \simeq F$, we have that $G[S] \not\simeq F$. In this case, the edges of H form exactly the set of strong edges of G .

STRONG F -CLOSURE

Input: A graph G and a nonnegative integer k .

Task: Decide whether G has a spanning subgraph H that satisfies the F -closure, such that $|E(H)| \geq k$.

Based on this definition and the above explanation, the terms "marking an edge as weak (in G)" and "removing an edge (of G to obtain H)" are equivalent, and we will use them interchangeably. An induced path on three vertices is denoted by P_3 . Relating STRONG F -CLOSURE to the already mentioned problems, observe that STRONG

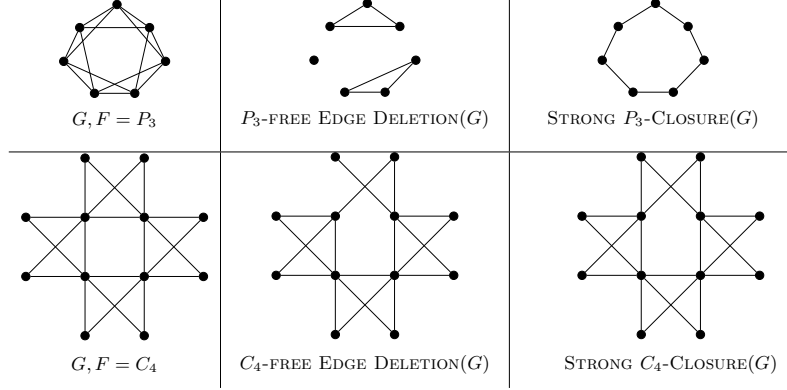


Figure 6.1: Two examples where the optimal solution of STRONG F -CLOSURE is slightly larger than the optimal solution of H -FREE EDGE DELETION.

P_3 -CLOSURE is exactly MAXSTC. Observe also that a solution for F -FREE EDGE DELETION is a solution for STRONG F -CLOSURE, since the removed edges in the first problem can simply be taken as the weak edges in the second problem. However it is important to note that the reverse is not always true. For instance, consider the square of a chordless cycle on seven vertices, denoted by C_7^2 (i.e., the graph obtained from C_7 by adding edges between vertices that are in distance two in C_7). An optimal solution for the P_3 -FREE EDGE DELETION consists of two vertex-disjoint triangles and a singleton vertex spanned by 6 edges. For the STRONG P_3 -CLOSURE, an optimal solution is spanned by the 7 edges of the C_7 . Such observations arise from the fact that any edge removal of a P_3 in C_7^2 results in a new P_3 which needs to be handled for the P_3 -FREE EDGE DELETION, whereas for the STRONG P_3 -CLOSURE we cannot create new forbidden structure by the removal of edges. Figure 6.1 shows the above instance of the square of a chordless cycle on seven vertices as well as another instance where $F = C_4$ and the optimal solution of STRONG C_4 -CLOSURE is larger than the optimal solution of C_4 -FREE EDGE DELETION.

All of the mentioned problems are known to be NP-hard. The parameterized complexity of F -FREE EDGE DELETION has been studied extensively when parameterized by ℓ , the number of removed edges. With this parameter, the problem is fixed parameter tractable (FPT) if F is of constant size [16], whereas it becomes W[1]-hard when parameterized by the size of F even for $\ell = 0$ [81]. Moreover, there exists a small graph F on seven vertices for which F -FREE EDGE DELETION does not admit a polynomial kernel [92] when the problem is parameterized by ℓ . In Table 6.1 we summarize the parameterized complexity of F -FREE EDGE DELETION. To our knowledge, MAXSTC has not been studied with respect to parameterized complexity before our work.

We study the parameterized complexity of **STRONG F -CLOSURE** with three different natural parameters: the number of strong edges, the number of strong edges above guarantee (maximum matching size), and the number of weak edges. In follow we explain the content of every section.

- In Section 6.3, we show that **STRONG F -CLOSURE** is FPT when parameterized by $k = |E(H)|$ for a fixed F . Moreover, we prove that the problem is FPT even when we allow the size of F to be a parameter, that is, if we parameterize the problem by $k + |V(F)|$, except if F has at most one edge. In the latter case **STRONG F -CLOSURE** is co-W[1]-hard when parameterized by $|V(F)|$ even if $k \leq 1$. We also observe that **STRONG F -CLOSURE** parameterized by $k + |V(F)|$ admits a polynomial kernel if F has a component with at least three vertices and the input graph is restricted to be d -degenerate.
- In Section 6.4, we focus on the case $F = P_3$, that is, we investigate the parameterized complexity of **MAXSTC**. We complement the FPT results of the previous section by proving that **MAXSTC** does not admit a polynomial kernel even on split graphs unless $\text{NP} \subseteq \text{coNP}/\text{poly}$. It is straightforward to see that if F has a connected component on at least three vertices, then a matching in G gives a feasible solution for **STRONG F -CLOSURE**. Thus the maximum matching size $\mu(G)$ provides a lower bound for the maximum number of edges of H . Consequently, parameterization above this lower bound becomes interesting. Motivated by this, we study **STRONG F -CLOSURE** parameterized by $|E(H)| - \mu(G)$. It is known that **MAXSTC** can be solved in polynomial time on subcubic graphs, but it is NP-complete on graphs of maximum degree at most d for every $d \geq 4$ [86]. As a first step in the investigation of the parameterization above lower bound, we show that **MAXSTC** is FPT on graphs of maximum degree at most 4, parameterized by $|E(H)| - \mu(G)$.
- Finally, in Section 6.5, we consider **STRONG F -CLOSURE** parameterized by $\ell = |E(G)| - |E(H)|$, that is, by the number of weak edges. We show that the problem is FPT and admits a polynomial generalized kernel if F is a fixed graph. Notice that, contrary to the parameterization by $k + |V(F)|$, we cannot hope for FPT results when the problem is parameterized by $\ell + |V(F)|$. This is because, when $\ell = 0$, **STRONG F -CLOSURE** is equivalent to asking whether G is F -free, which is equivalent to solving **INDUCED SUBGRAPH ISOMORPHISM** that is well known to be W[1]-hard [39, 81]. We also state some additional results and open problems. Our findings are summarized in Table 6.2.

Independently from our work, Grüttemeier and Komusiewicz [62] very recently studied **MAXSTC** and showed that the problem parameterized by $|E(H)| = k$, the

Parameter	Restriction	Parameterized Complexity	Reference
$ E(H) + V(F) $	$ E(F) \leq 1$	W[1]-hard	[81]
$ E(G) - E(H) $	None	FPT	[16]
		no polynomial kernel	[92]

Table 6.1: Summary of known results: parameterized complexity analysis of F -FREE EDGE DELETION.

number of strong edges, is fixed-parameter tractable but has no polynomial kernel unless $\text{NP} \subseteq \text{coNP/poly}$. Also, they showed that MAXSTC parameterized by $\ell = |E(G)| - |E(H)|$, the number of weak edges, admits a linear kernel.

Parameter	Restriction	Parameterized Complexity	Result
$ E(H) + V(F) $	$ E(F) \leq 1$	co-W[1]-hard	Propositions 6.3.2, 6.3.3
	$ E(F) \geq 2$	FPT	Theorem 6.3.8
	F has a component with ≥ 3 vertices, G is d -degenerate	polynomial kernel	Proposition 6.3.10
$ E(H) $	F has no isolated vertices	FPT	Corollary 6.3.9
	$F = P_3$, G is split	no polynomial kernel	Theorem 6.4.1
$ E(H) - \mu(G)$	$F = P_3$, $\Delta(G) \leq 4$	FPT	Theorem 6.4.9
	$F = K_{1,t}$, $t \geq 3$	FPT	Theorem 6.5.6
$ E(G) - E(H) $	None	FPT	Theorem 6.5.1
		polynomial generalized kernel	Theorem 6.5.2

Table 6.2: Summary of our results: parameterized complexity analysis of STRONG F -CLOSURE.

6.2 Preliminaries

Let us give a couple of observations on the nature of our problem. An F -graph of a subgraph H of G is an induced subgraph $H[S] \simeq F$ such that $G[S] \simeq F$. Clearly, if H is a solution for STRONG F -CLOSURE on G , then there is no F -graph in H , even though H might have induced subgraphs isomorphic to F . For F -FREE EDGE DELETION, note that the removal of an edge that belongs to a forbidden subgraph might generate a new forbidden subgraph. However, for STRONG F -CLOSURE problem, it is not difficult to see that the removal of an edge that belongs to an F -graph cannot create a new

critical subgraph.

Observation 6.2.1. *Let G be a graph, and let H and H' be spanning subgraphs of G such that $E(H') \subseteq E(H)$. If H satisfies the F -closure for some F , then H' satisfies the F -closure.*

In particular, Observation 6.2.1 immediately implies that if an instance of STRONG F -CLOSURE has a solution, it has a solution with *exactly* k edges.

6.3 Parameterized complexity of Strong F-closure

In this section we give a series of lemmata, which together lead to the conclusion that STRONG F -CLOSURE is FPT when parameterized by $k = |E(H)|$. Observe that in our definition of the problem, F is a fixed graph of constant size. However, the results of this section allow us to also take the size of F as a parameter, making the results more general. We start by making some observations that will rule out some simple types of graphs as F .

Observation 6.3.1. *Let p be a positive integer. A graph G has a spanning subgraph H satisfying the pK_1 -closure if and only if G is pK_1 -free, and if G is pK_1 -free, then every spanning subgraph H of G satisfies the pK_1 -closure.*

Recall that the INDEPENDENT SET problem asks, given a graph G and a positive integer k , whether G has an independent set of size at least k . By combining Observation 6.3.1 and the well known result that INDEPENDENT SET is $W[1]$ -hard when parameterized by the size of the independent set [39], we obtain the following:

Proposition 6.3.2. *For a positive integer p , STRONG pK_1 -CLOSURE can be solved in time $n^{O(p)}$, and it is co- $W[1]$ -hard for $k \geq 0$ when parameterized by p .*

Using Proposition 6.3.2, we assume throughout the remaining parts of the paper that every considered graph F has at least one edge. We have another special case $F = pK_1 + K_2$.

Proposition 6.3.3. *For a nonnegative integer p , STRONG $(pK_1 + K_2)$ -CLOSURE can be solved in time $n^{O(p)}$, and it is co- $W[1]$ -hard for $k \geq 1$ when parameterized by p .*

Proof. Let $F = pK_1 + K_2$. If $p = 0$, then (G, k) is a yes-instance of STRONG F -CLOSURE if and only if $k = 0$. Assume that $p \geq 1$. Let H be a spanning subgraph of G . Notice that H satisfies the F -closure if and only if for every edge uv of H , $G - N[\{u, v\}]$ has no independent set of size p .

This observation implies that to find a spanning subgraph H of G satisfying the F -closure, we can use the following procedure: for every edge $uv \in E(G)$, we check whether $G - N[\{u, v\}]$ has an independent set with p vertices, and then if this holds, we discard uv , and we include uv in the set of edges of H otherwise. Clearly, it can be done in time $n^{O(p)}$.

To show hardness, we reduce INDEPENDENT SET. For simplicity, we prove the claim for $k = 1$. Let (G, p) be an instance of INDEPENDENT SET. Let Q be the graph obtained from two copies of the star $K_{1,p}$ by making their central vertices u and v adjacent. We define $G' = G + Q$. We claim that G' has a spanning subgraph H satisfying the F -closure that has exactly one edge if and only if G has no independent set with p vertices. Suppose that G has no independent set with p vertices. Then the spanning subgraph H of G' with $E(H) = \{uv\}$ satisfies the F -closure. Assume now that H is a spanning subgraph of G' with $E(H) = \{xy\}$. We show that $xy = uv$. Suppose this is not the case. If u (resp. v) is not an endpoint of xy , then $G' - N[\{x, y\}]$ contains an independent set of size at least p , namely the one formed by the p vertices of degree one adjacent to u (resp. v) in G' . This contradicts the property that H satisfies the F -closure. Hence, $xy = uv$. Then $G = G' - N[\{u, v\}]$ has no independent set with p vertices. By Observation 6.2.1, we have that (G, p) is a no-instance of INDEPENDENT SET if and only if (G', k) is a yes-instance of STRONG F -CLOSURE. \square

From now on we assume that $F \neq pK_1$ and $F \neq pK_1 + K_2$. We show that STRONG F -CLOSURE is FPT when parameterized by k and $|V(F)|$ in this case. We will consider separately the case when F has a connected component with at least 3 vertices and the case $F = pK_1 + qK_2$ for $p \geq 0$ and $q \geq 2$.

Lemma 6.3.4. *Let F be a graph that has a connected component with at least 3 vertices. Then STRONG F -CLOSURE can be solved in time $2^{O(k^2)}(|V(F)| + k)^{O(k)} + n^{O(1)}$.*

Proof. We show the claim by proving that the problem has a kernel with at most $2^{2k-2}(|V(F)| + k) + 2k - 2$ vertices. Let (G, k) be an instance of STRONG F -CLOSURE. We recursively apply the following reduction rule in G :

Rule 6.3.4.1. *If there are at least $|V(F)| + k + 1$ false twins in G , then remove one of them.*

To show that the rule is sound, let v_1, \dots, v_p be false twins of G for $p = |V(F)| + k + 1$ and assume that G' is obtained from G by deleting v_p . We claim that (G, k) is a yes-instance of STRONG F -CLOSURE if and only if (G', k) is a yes-instance.

Let (G, k) be a yes-instance. By Observation 6.2.1, there is a solution H for (G, k) such that $|E(H)| = k$. Since $|E(H)| = k$, there is $i \in \{1, \dots, p\}$ such that v_i is an isolated vertex of H . Since v_1, \dots, v_p are false twins we can assume without loss of

generality that $i = p$. Then $H' = H - v_p$ is a solution for (G', k) , that is, this is a yes-instance. Assume that (G', k) is a yes-instance of STRONG F -CLOSURE. Let H' be a solution for the instance with k edges. Denote by H the spanning subgraph of G with $E(H) = E(H')$. We show that H satisfies the F -closure with respect to G . To obtain a contradiction, assume that there is a set of vertices S of G such that $H[S] \simeq F$ and $G[S] \simeq F$. Since H' satisfies the F -closure with respect to G , $S \not\subseteq V(H')$. Thus, $v_p \in S$. Note that v_p is an isolated vertex of H . Because $p = |V(F)| + k + 1$, there is $i \in \{1, \dots, p-1\}$ such that v_i is an isolated vertex of H and $v_i \notin S$. Let $S' = (S \setminus \{v_p\}) \cup \{v_i\}$. Since v_i and v_p are false twins, $H[S'] = H'[S'] \simeq F$ and $G[S'] \simeq F$; a contradiction. Therefore, we conclude that H satisfies the F -closure with respect to G , that is, H is a solution for (G, k) .

It is straightforward to see that the rule can be applied in polynomial time. To simplify notations, assume that (G, k) is the instance of STRONG F -CLOSURE obtained by the exhaustive application of Rule 6.3.4.1. We greedily find an inclusion maximal matching M in G . Notice that the spanning subgraph H of G with $E(H) = M$ satisfies the F -closure because every component of H has at most two vertices and by the assumption of the lemma F has a component with at least 3 vertices. Therefore, if $|M| \geq k$, we have that H is a solution for the instance. Respectively, we return H and stop.

Assume that $|M| \leq k-1$. Let X be the set of end-vertices of the edges of M . Clearly, $|X| \leq 2k-2$ and X is a vertex cover of G . Let $Y = V(G) \setminus X$. We have that Y is an independent set, since M is an inclusion-wise maximal matching. Every vertex in Y has its neighbors in X . Hence, there are at most $2^{|X|}$ vertices of Y with pairwise distinct neighborhoods. Hence, the vertices of Y can be partitioned into at most $2^{|X|}$ classes of false twins. After applying Rule 6.3.4.1, each class of false twins has at most $|V(F)| + k$ vertices. It follows that $|Y| \leq 2^{|X|}(|V(F)| + k)$ and

$$|V(G)| = |X| + |Y| \leq |X| + 2^{|X|}(|V(F)| + k) \leq (2k-2) + 2^{2k-2}(|V(F)| + k).$$

Now we can find a solution for (G, k) by brute force checking all subsets of edges of size k by Observation 6.2.1. This can be done in time $|V(G)|^{O(k)}$. Hence, the total running time is $2^{O(k^2)}(|V(F)| + k)^{O(k)} + n^{O(1)}$. \square

Now we consider the case $F = pK_1 + qK_2$ for $p \geq 0$ and $q \geq 2$. First, we explain how to solve STRONG qK_2 -CLOSURE for $q \geq 2$. We use the *random separation technique* proposed by Cai, Chen and Chan [18] (see also [30]). To avoid dealing with randomized algorithms and subsequent standard derandomization we use the following lemma stated in [22].

Lemma 6.3.5 ([22]). *Given a set U of size n and integers $0 \leq a, b \leq n$, one can construct in time $2^{O(\min\{a,b\} \log(a+b))} \cdot n \log n$ a family \mathcal{S} of at most $2^{O(\min\{a,b\} \log(a+b))} \cdot \log n$*

subsets of U such that the following holds: for any sets $A, B \subseteq U$, $A \cap B = \emptyset$, $|A| \leq a$, $|B| \leq b$, there exists a set $S \in \mathcal{S}$ with $A \subseteq S$ and $B \cap S = \emptyset$.

Lemma 6.3.6. For $q \geq 2$, STRONG qK_2 -CLOSURE can be solved in time $2^{O(k \log k)} \cdot n^{O(1)}$.

Proof. Let (G, k) be an instance of STRONG qK_2 -CLOSURE. If $k < q$, then every spanning subgraph H of G with k edges satisfies the F -closure, that is, (G, k) is a yes-instance of STRONG F -CLOSURE if $k \leq |E(G)|$. Assume from now that $q \leq k$.

Suppose that G has a vertex v of degree at least k . Let X be the set of edges of G incident to v and consider the spanning subgraph H of G with $E(H) = X$. Since $F = qK_2$ and $q \geq 2$, H satisfies the F -closure. Hence, H is a solution for (G, k) . We assume that this is not the case and $\Delta(G) \leq k - 1$.

Suppose that (G, k) is a yes-instance. Then by Observation 6.2.1, there is a solution H with exactly k edges. Let $A = E(H)$ and denote by X the set of end-vertices of the edges of A . Denote by B the set of edges of $E(G) \setminus A$ that have at least one end-vertex in $N[X]$. Clearly, $A \cap B = \emptyset$. We have that $|A| = k$ and because the maximum degree of G is at most $k - 1$, $|B| \leq 2k(k - 1)(k - 2)$. Applying Lemma 6.3.5 for the universe $U = E(G)$, $a = k$ and $b = 2k(k - 1)(k - 2)$, we construct in time $2^{O(k \log k)} \cdot n^{O(1)}$ a family \mathcal{S} of at most $2^{O(k \log k)} \cdot \log n$ subsets of $E(G)$ such that there exists a set $S \in \mathcal{S}$ with $A \subseteq S$ and $B \cap S = \emptyset$. For every $S \in \mathcal{S}$, we find (if it exists) a spanning subgraph H of G with k edges such that (i) $E(H) \subseteq S$ and (ii) for every $e_1, e_2 \in S$ that are adjacent or have adjacent end-vertices, it holds that either $e_1, e_2 \in E(H)$ or $e_1, e_2 \notin E(H)$. Property (ii) ensures that the set of edges of $S \setminus E(H)$ do not belong to B . By Lemma 6.3.5, we have that if (G, k) is a yes-instance of STRONG F -CLOSURE, then it has a solution satisfying (i) and (ii). Hence, if we find a solution for some $S \in \mathcal{S}$, we return it and stop and, otherwise, if there is no solution satisfying (i) and (ii) for some $S \in \mathcal{S}$, we conclude that (G, k) is a no-instance.

Assume that $S \in \mathcal{S}$ is given. We describe the algorithm for finding a solution H with k edges satisfying (i) and (ii). Let R be the set of end-vertices of the edges of S . Consider the graph $G[R]$ and denote by C_1, \dots, C_r its components. Let $A_i = E(C_i) \cap S$ for $i \in \{1, \dots, r\}$.

Observe that if H is a solution with k edges satisfying (i) and (ii), then for each $i \in \{1, \dots, r\}$, either $A_i \subseteq E(H)$ or $A_i \cap E(H) = \emptyset$. It means that we are looking for a solution H such that $E(H)$ is union of some sets A_i , that is, $E(H) = \cup_{i \in I} A_i$ for $I \subseteq \{1, \dots, r\}$. Let $c_i = |A_i|$ for $i \in \{1, \dots, r\}$. Clearly, we should have that $\sum_{i \in I} c_i = k$. In particular, it means that if $|A_i| > k$, then the edges of A_i are not in any solution. Therefore, we discard such sets and assume from now that $|A_i| \leq k$ for $i \in \{1, \dots, r\}$. For $i \in \{1, \dots, r\}$, denote by w_i the maximum number of edges in A_i that form an induced matching in C_i . Since each $|A_i| \leq k$, the values of w_i can be computed in time

$2^k \cdot n^{O(1)}$ by brute force. Observe that for distinct $i, j \in \{1, \dots, r\}$, the vertices of C_i and C_j are at distance at least two in G and, therefore, the end-vertices of edges of A_i and A_j are not adjacent. It follows, that the problem of finding a solution H is equivalent to the following problem: find $I \subseteq \{1, \dots, r\}$ such that $\sum_{i \in I} c_i = k$ and $\sum_{i \in I} w_i \leq q$. It is easy to see that we obtain an instance of a variant of the well known KNAPSACK problem (see, e.g., [82]); the only difference is that we demand $\sum_{i \in I} c_i = k$ instead of $\sum_{i \in I} c_i \geq k$ as in the standard version. This problem can be solved by the standard dynamic programming algorithm (again see, e.g., [82]) in time $O(kn)$.

Since the family \mathcal{S} is constructed in time $2^{O(k \log k)} \cdot n^{O(1)}$ and we consider $2^{O(k \log k)} \cdot \log n$ sets S , we obtain that the total running time is $2^{O(k \log k)} \cdot n^{O(1)}$. \square

We use Lemma 6.3.6 to solve STRONG $(pK_1 + qK_2)$ -CLOSURE.

Lemma 6.3.7. *For $p \geq 0$ and $q \geq 2$, STRONG $(pK_1 + qK_2)$ -CLOSURE can be solved in time $2^{O((k+p) \log(k+p))} \cdot n^{O(1)}$.*

Proof. Let $F = pK_1 + qK_2$. If $p = 0$, we can apply Lemma 6.3.6 directly. Assume that $p \geq 1$. Let (G, k) be an instance of STRONG F -CLOSURE. If $k < q$, then every spanning subgraph H of G with k edges satisfies the F -closure, that is, (G, k) is a yes-instance of STRONG F -CLOSURE if $k \leq |E(G)|$. Assume from now that $q \leq k$.

Suppose that G has a vertex v of degree at least k . Then we argue in exactly the same way as in the proof of Lemma 6.3.6. We consider the set of edges X incident to v and define H be the spanning subgraph of G with $E(H) = X$. Since $q \geq 2$, H satisfies the F -closure and we have that H is a solution for (G, k) . We assume from now that this is not the case and $\Delta(G) \leq k - 1$.

Suppose that $|V(G)| < 2k(k - 1) + pk$. In this case we solve STRONG F -CLOSURE by brute force trying all possible subsets X of k edges and checking whether the spanning subgraph H with $E(H) = X$ is a solution. By Observation 6.2.1, it is sufficient to solve the problem. To check whether H is a solution, we have to verify whether H satisfies the F -closure. We do it by brute force in time $n^{O(|V(F)|)}$. Since $n \leq 2k(k - 1) + pk$ and $|V(F)| = p + 2q \leq p + 2k$, this can be done in time $2^{O((k+p) \log(k+p))}$. Since the number of sets X is $2^{O((k+p) \log(k+p))}$, the total running time is $2^{O((k+p) \log(k+p))}$.

Assume now that $|V(G)| \geq 2k(k - 1) + pk$.

We claim that in this case a spanning subgraph H of G satisfies the $pK_1 + qK_2$ -closure if and only if H satisfies the qK_2 -closure. It is straightforward to see that if H satisfies the qK_2 -closure, then H satisfies the $pK_1 + qK_2$ -closure. Suppose that H does not satisfy the qK_2 -closure. Then there is $S \subseteq V(G)$ of size $2q$ such that $G[S] = H[S]$ is a matching with q edges. Let $X = V(G) \setminus N[S]$. Since $\Delta(G) \leq k - 1$, $|N[S]| \leq 2k(k - 1)$ and, therefore, $|X| \geq pk$. It implies that $G[X]$ has an independent set S' of size at least p because the

maximum degree is bounded by $k - 1$. We have that $G[S \cup S'] = H[S \cup S'] \simeq pK_1 + qK_2$. It means that H does not satisfy the $pK_1 + qK_2$ -closure.

By the proved claim, we have to solve STRONG qK_2 -CLOSURE and this can be done in time $2^{O(k \log k)} \cdot n^{O(1)}$ by Lemma 6.3.6. \square

Combining Lemmata 6.3.4, 6.3.6, and 6.3.7, we obtain the following theorem.

Theorem 6.3.8. *If $F \neq pK_1$ for $p \geq 1$ and $F \neq pK_1 + K_2$ for $p \geq 0$, then STRONG F-CLOSURE is FPT when parameterized by $|V(F)| + k$.*

Notice that if $|E(F)| > k$, then (G, k) is a yes-instance of STRONG F-CLOSURE. This immediately implies the following corollary.

Corollary 6.3.9. *If F has no isolated vertices, then STRONG F-CLOSURE is FPT when parameterized by k , even when F is given as a part of the input.*

We conclude this section with a kernel result. It can be observed that if the input graph G is restricted to be a graph from a sparse graph class and is closed under taking subgraphs, then the kernel constructed in Lemma 6.3.4 becomes polynomial in some cases. We demonstrate this for d -degenerate graphs ¹.

Proposition 6.3.10. *If F has a connected component with at least 3 vertices, then STRONG F-CLOSURE has a kernel with $k^{O(d)}d(|V(F)| + k)$ vertices on d -degenerate graphs.*

Proof. Let (G, k) be an instance of STRONG F-CLOSURE and G is d -degenerate. First, we exhaustively apply Rule 6.3.4.1. To simplify notations, assume that (G, k) is the obtained instance. Then we find an inclusion maximal matching M in G . If $|M| \geq k$, we have that H is a solution for the instance. Respectively, we return H and stop. Assume that this is not the case, that is, $|M| \leq k - 1$. Let X be the set of end-vertices of the edges of M . Clearly, $|X| \leq 2k - 2$ and X is a vertex cover of G . Let $Y = V(G) \setminus X$. We have that Y is an independent set.

Observe that if Y contains at least $\binom{|X|}{d+1}d + 1$ vertices of degree at least $d + 1$, then G contains the complete bipartite graph $K_{d+1, d+1}$ as a subgraph contradicting d -degeneracy. We conclude that Y contains $d \cdot k^{O(d)}$ vertices of degree at least $d + 1$. The number of vertices of degree at most d with pairwise distinct neighborhoods is $k^{O(d)}$. This immediately implies that G has $k^{O(d)}d(|V(F)| + k)$ vertices. \square

In particular, we have a polynomial kernel when $F = P_3$. Similar results can be obtained for some classes of dense graphs. For example, if G is dK_1 -free, then $V(G) \setminus X$ has at most $d - 1$ vertices and we obtain a kernel with $2k + d - 3$ vertices.

¹NP-completeness result for $F = P_3$ restricted to planar graphs (and, thus, 5-degenerate graphs) is given in Section 6.5.

6.4 Parameterized complexity of MAXSTC

In this section we study the parameterized complexity of STRONG P_3 -CLOSURE, which is more famously known as MAXSTC.

Note that MAXSTC is FPT and admits an algorithm with running time $2^{\mathcal{O}(k^2)} \cdot n^{\mathcal{O}(1)}$ by Lemma 6.3.4. We complement this result by showing that MAXSTC does not admit a polynomial kernel, even when the input graph is a split graph. A graph is a *split graph* if its vertex set can be partitioned into an independent set and a clique. MAXSTC is known to be NP-hard on split graphs (Theorem 3.3.4).

Theorem 6.4.1. *MAXSTC has no polynomial compression unless $NP \subseteq coNP/poly$, even when the input graph is a split graph.*

Proof. The reduction comes from the SET PACKING problem: given a universe \mathcal{U} of t elements and subsets B_1, \dots, B_p of \mathcal{U} decide whether there are at least k subsets which are pairwise disjoint. SET PACKING (also known as RANK DISJOINT SET problem), parameterized by $|\mathcal{U}|$, does not admit a polynomial compression unless $NP \subseteq coNP/poly$ [35]. Clearly, it can be assumed that $k \leq t$ as, otherwise, we have a trivial no-instance. Given an instance $(\mathcal{U}, B_1, \dots, B_p, k)$ for the SET PACKING, we construct a split graph G with a clique $U \cup Y$ and an independent set $W \cup X$ as follows:

- The vertices of U correspond to the elements of \mathcal{U} .
- For every B_i there is a vertex $w_i \in W$ that is adjacent to all the vertices of $(U \cup Y) \setminus B_i$.
- X and Y contain additional $2t$ vertices with $X = \{x_1, \dots, x_t\}$ and $Y = \{y_1, \dots, y_t\}$ such that y_i is adjacent to all the vertices of $(W \cup X) \setminus \{x_i\}$ and x_i is adjacent to all the vertices of $(U \cup Y) \setminus \{y_i\}$.

Notice that the clique of G contains $2t$ vertices. We will show that there are at least k pairwise disjoint sets in $\{B_1, \dots, B_p\}$ if and only if there is a solution for STRONG P_3 -CLOSURE on G with at least $k' = |E(U \cup Y)| + \lceil k/2 \rceil + \lfloor t/2 \rfloor$ edges. Since $k \leq t = |U|$, this means that $k' = O(t^2)$ and, therefore, the existence of a polynomial compression for MAXSTC would imply the same result for SET PACKING parameterized by t .

Assume that \mathcal{B}' is a family of k pairwise disjoint sets of B_1, \dots, B_p . For every $B'_i \in \mathcal{B}'$ we choose three vertices w_i, y_i, x_i from W, Y , and X , respectively, such that x_i is non-adjacent to y_i with the following strong edges: w_i is strongly adjacent to y_i and x_i is strongly adjacent to the vertices of B'_i in U . We also make weak the edges inside the clique between the vertices of B'_i and y_i . All other edges incident to w_i and x_i are weak.

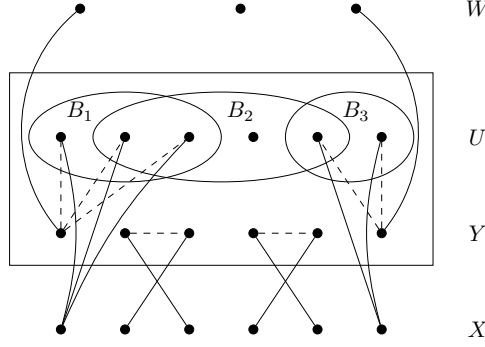


Figure 6.2: Illustrating the split graph G given in the construction in the proof of Theorem 6.4.1, where $U \cup Y$ is a clique and $W \cup X$ is an independent set. Given an instance $(U, B_1, B_2, B_3, 2)$ for the SET PACKING, the labeled edges correspond to a solution for STRONG P_3 -CLOSURE on G . To keep the figure clean, we only draw the strong edges between the independent set $W \cup X$ and the clique $U \cup Y$; the dashed edges of the clique $U \cup Y$ correspond to its weak edges. Notice that the dashed edges span a union of star graphs.

Let W', Y', X' be the set of vertices that are chosen from the family \mathcal{B}' according to the previous description. Every vertex of $W \setminus W'$ is not incident to a strong edge and, thus, it is isolated in H . For the $t - k$ vertices of $Y \setminus Y'$ we choose a maximum matching of $\lfloor \frac{t-k}{2} \rfloor$ edges. For each matched pair $y_j, y_{j'}$ we make the following edges strong: $x_j y_{j'}$ and $x_{j'} y_j$ where x_j and $x_{j'}$ are non-adjacent to y_j and $y_{j'}$, respectively. Moreover each edge $y_j y_{j'}$ of the clique is weak and all other edges incident to x_j and $x_{j'}$ are weak. The rest of the edges inside the clique $U \cup Y$ are strong. Figure 6.2 illustrates such a labeling on the edges of G .

Let us now show that the described subgraph H satisfies the P_3 -closure with the claimed number of strong edges. Observe that if there is a P_3 -graph in H then it must contain a vertex of the independent set incident to a strong edge. Also notice that no vertex of the clique $U \cup Y$ is strongly adjacent to more than one vertex of the independent set $W \cup X$. By construction for each $B'_i \in \mathcal{B}'$ the vertices w_i, x_i of the independent set are incident to a strong edge. The vertices of the clique that are non-adjacent to w_i constitute B'_i , and x_i is non-adjacent only to vertex y_i . Since all edges of $E(B'_i, \{y_i\})$ are weak, both vertices w_i and x_i cannot induce a P_3 -graph. The rest of the vertices of the independent set that are incident to at least one strong edge belong to $X \setminus X'$. Every vertex x_j of $X \setminus X'$ is adjacent to all vertices of $(U \cup Y) \setminus \{y_j\}$. For the strong edge $x_j y_{j'}$ there is a weak edge $y_j y_{j'}$ implying that x_j does not participate in any P_3 -graph of H . Thus for any vertex v of the independent set that is strongly adjacent to a vertex v' of

the clique there are weak edges between v' and the non-neighbors of v in the clique. Consequently there is no P_3 -graph in H . For the number of edges in H notice that for every weak edge inside the clique $U \cup Y$ there is a unique matched strong edge incident to a vertex of X . Furthermore every vertex of W' is incident to an unmatched strong edge and each of the $\lfloor \frac{|X \setminus X'|}{2} \rfloor$ vertices is incident to an additional unmatched strong edge. Hence $|E(H)| = |E(U \cup Y)| + k + \lfloor \frac{t-k}{2} \rfloor$, which gives the claimed bound k' .

For the opposite direction, assume that H is a subgraph of G that satisfies the P_3 -closure with at least k' edges. For a vertex $v \in W \cup X$, let $S(v)$ be the strong neighbors of v in H and let $B(v)$ be the non-neighbors of v in $U \cup Y$. Our task is to show that for any two vertices u, v of $W \cup X$ with non-empty sets $S(u), S(v)$, we have $B(u) \cap B(v) = \emptyset$. Since there is no P_3 -graph in H , it is clear that all edges of $E(S(v), B(v))$ are weak. Also observe that for any two vertices $u, v \in W \cup X$, $S(u) \cap S(v) = \emptyset$.

A spanning subgraph H of G that satisfies the P_3 -closure is called *nice solution* if for any weak edge uv of the clique $U \cup Y$ the following property holds:

- (W1) there are two vertices u', v' in the independent set $W \cup X$ such that $u \in S(u') \cap B(v')$ and $v \in S(v') \cap B(u')$.

We first prove that every solution can be transformed into an equivalent nice solution.

Claim 6.4.2. *For any spanning subgraph H of G that satisfies the P_3 -closure with at least k' edges, there is a nice solution H' with at least k' edges.*

Proof: We assume that H is not a nice solution. This means that there is a weak edge uv with $u, v \in U \cup Y$ that does not admit property (W1). We will show that we can safely make the edge uv strong and maintain the same number of strong edges. If $u, v \notin S(x)$ for every vertex x of $W \cup X$ then we can make the edge uv strong without violating the P_3 -closure. Thus there is at least one vertex u' that is strongly adjacent to u so that $u \in S(u')$. Moreover if $v \in S(u')$ then both u and v have no other strong neighbor in the independent set which means that we can safely make the edge uv strong. This implies that $v \notin S(u')$. Now assume that $v \notin B(u')$, meaning that v is a neighbor of u' in G but not a neighbor of u' in H . Observe that v has at most one strong neighbor in the independent set. If there is such a strong neighbor v' of v in $W \cup X$ then we make vv' weak and uv strong. Such a replacement is safe, since u has exactly one strong neighbor u' in $W \cup X$ and all other strong neighbors of u or v belong to the clique. Hence $v \in B(u')$.

Suppose next that v has no strong neighbor in the independent set. Then we replace the strong edge $u'u$ by the edge uv ; such a replacement is safe since v has no strong neighbors in the independent set and u' is the only strong neighbor of u in the independent set. Thus there is a strong neighbor v' of v such that $v' \in W \cup X$. Summarizing,

there are $u', v' \in W \cup X$ such that $u \in S(u')$, $v \in S(v')$, $v \in B(u')$, and by symmetry for v' we get $u \in B(v')$. Therefore $u \in S(u') \cap B(v')$ and $v \in S(v') \cap B(u')$. \diamond

In what follows we assume that H is a nice solution. We next consider the vertices of X from the independent set.

Claim 6.4.3. *Let H be a nice solution in which no vertex of W is incident to a strong edge. Then $|E(H)| \leq |E(U \cup Y)| + \lfloor t/2 \rfloor$.*

Proof: We first show that for every vertex x_i of X , $S(x_i)$ contains at most one vertex. Recall that $B(x_i)$ contains exactly one vertex. Assume for contradiction that $S(x_i)$ contains at least two vertices. Let $u, v \in S(x_i)$ and let $B(x_i) = z$. By the P_3 -closure, both edges uz and vz of the clique must be weak. Then by property (W1) and Claim 6.4.2, there is a vertex $x_j \in X$ such that $z \in S(x_j)$ and $\{u, v\} \subseteq B(x_j)$. This however is not possible since by construction we know that $B(x_j)$ contains exactly one vertex. Thus $|S(x_i)| \leq 1$ for every vertex $x_i \in X$.

Let E_W be the set of weak edges that have both their endpoints in the clique. If there are two edges of E_W incident to the same vertex u then by property (W1) and Claim 6.4.2 the unique vertex $u' \in X$ that is strongly adjacent to u has two non-adjacent vertices in the clique. Since every vertex of X is non-adjacent to exactly one vertex, there are no two edges of E_W incident to the same vertex. This means that the edges of E_W form a matching in $E(U \cup Y)$. Moreover property (W1) and the fact that H is nice solution, imply that for every edge of E_W there are exactly two strong edges between the vertices of the independent set and the clique. Thus $E_W \subseteq E(Y)$ and $|E_W| \leq \lfloor \frac{t}{2} \rfloor$, since $|Y| = t$. For the same reason, also observe that $|S(X)| = 2|E_W|$ where $S(X)$ are the strong edges with one endpoint in X . Therefore $E(H) = (E(U \cup Y) \setminus E_W) \cup S(X)$ which implies $|E(H)| \leq |E(U \cup Y)| + \lfloor \frac{t}{2} \rfloor$. \diamond

Thus by Claim 6.4.3 and the fact that a nice solution H contains $k' > |E(U \cup Y)| + \lfloor t/2 \rfloor$ edges, we know that some vertices of W are incident to strong edges in H . We next show that these type of vertices of W must have disjoint non-neighborhood in G . To do so, we consider the *weak components* of $E(H)$ in the clique. A *weak component* is a connected component of the clique spanned by the weak edges, that is, by the edges of $E(G) - E(H)$.

Let C_w be a weak component with $n(C_w)$ its number of vertices and $m(C_w)$ its number of weak edges. Denote by $E_S(C_w)$ the set of strong edges between C_w and $W \cup X$. By property (W1) and Claim 6.4.2, $E_S(C_w)$ is non-empty. Notice that every vertex of C_w has exactly one strong neighbor in the independent set, since H satisfies the P_3 -closure. This means that $|E_S(C_w)| = n(C_w)$. Then the number of edges in H can be

described as follows:

$$|E(H)| = |E(U \cup Y)| + \sum_{C_w} (n(C_w) - m(C_w)). \quad (6.1)$$

We say that a nice solution H is a *nice sparse solution* if every weak component of H is a tree.

Claim 6.4.4. *For every nice solution H , there is a nice sparse solution H' such that $|E(H)| = |E(H')|$.*

Proof: Consider a weak component C_w of H . If we make strong all edges among the vertices of C_w and remove the edges of $E_S(C_w)$ from H then the resulting graph H' satisfies the P_3 -closure. Thus if $m(C_w) \geq n(C_w)$ then we can safely ignore such a component in the sum of $|E(H)|$ in Equation 6.1 by replacing all its weak edges by the strong edges of $E_S(C_w)$. This means that $m(C_w) = n(C_w) - 1$ because the weak edges of C_w span a connected component. Therefore every weak component C_w is a tree in H' . \diamond

In fact we will prove that there is a nice solution in which every weak component is a tree of height one (*star graph*). Before that, let us first show the following property with respect to the nested non-neighborhood of vertices of $W \cup X$. For a vertex $v \in W \cup X$, observe that all edges between $S(v)$ and $B(v)$ are weak. Thus all vertices of $S(v)$ belong to the same weak component of H .

We say that a nice sparse solution H is *nice disjoint solution* if for any $v_i, v_j \in W \cup X$ with non-empty $S(v_i)$ and $S(v_j)$, we have $B(v_i) \not\subseteq B(v_j)$ and $B(v_j) \not\subseteq B(v_i)$.

Claim 6.4.5. *For every nice sparse solution H , there is a nice disjoint solution H' such that $|E(H)| = |E(H')|$.*

Proof: By the P_3 -closure of H , we know that no vertex of the clique has more than one strong neighbor in the independent set, which implies $S(v_i) \cap S(v_j) = \emptyset$. Assume that there are two vertices v_i, v_j in $W \cup X$ such that $B(v_i) \subseteq B(v_j)$. This means that the vertices of $S(v_i) \cup S(v_j)$ belong to the same weak component C_w . We show that there is an optimal solution H' for which $S(v_j) = \emptyset$ and $|E(H')| = |E(H)|$. There is no weak edge with the endpoints in $S(v_i)$ and $S(v_j)$, respectively, since C_w is a tree. Thus all edges between the vertices of $S(v_i)$ and $S(v_j)$ are strong. This means that v_i is adjacent to every vertex of $S(v_j)$. We construct H' by replacing all strong edges incident to v_j by strong edges incident to v_i . Remove all strong edges incident to v_j and let $S(v_i) \cup S(v_j)$ be the strong neighbors of v_i in H' . Notice that $|E(H')| = |E(H)|$. Since we only added strong edges incident to v_i and $B(v_i) \subseteq B(v_j)$, all edges between $B(v_i)$ and $S(v_i) \cup S(v_j)$ are weak and, thus, H' satisfies the P_3 -closure. Therefore applying the same replacement

for every pair of vertices with nested non-neighborhood, results in an optimal solution as desired. \diamond

Claim 6.4.6. *Every weak component of a nice disjoint solution H is a star graph.*

Proof. Let u_1, u_2, \dots, u_r be a path of a weak component C_w of H where u_1 is a leaf vertex of C_w . Since u_1u_2 is a weak edge, property (W1) implies that there is a vertex v_i in the independent set that is strongly adjacent to u_1 such that $B(v_i) = \{u_2\}$. If C_w is not a star then $r \geq 4$. For $r \geq 4$, the weak edge u_3u_4 implies that there is a vertex v_j in the independent set that is strongly adjacent to u_3 such that $\{u_2, u_4\} \subseteq B(v_j)$. Then we reach a contradiction since $B(v_i) \subset B(v_j)$ which is not possible by the definition of H . Therefore we have $r \leq 3$, which implies that C_w is a tree of height one. \square

Claim 6.4.7. *For any two vertices $v_i, v_j \in W \cup X$ of a nice disjoint solution H with non-empty $S(v_i)$ and $S(v_j)$, we have $B(v_i) \cap B(v_j) = \emptyset$.*

Proof. Recall that $S(v_i) \cap S(v_j) = \emptyset$ and notice that all edges of $E(S(v_i), B(v_i))$ and $E(S(v_j), B(v_j))$ are weak. If the vertices of $S(v_i)$ belong to a different weak component than the vertices of $S(v_j)$ then $B(v_i)$ and $B(v_j)$ are disjoint. Suppose that the vertices of $S(v_i)$ and $S(v_j)$ belong to the same weak component C_w . By Claim 6.4.6, C_w is a star graph. Let u be the non-leaf vertex of the star C_w . If both v_i and v_j are strongly adjacent to leaf vertices of C_w , then $B(v_i) = B(v_j) = \{u\}$. Thus by the definition of H , v_i is strongly adjacent to u so that $B(v_i) = V(C_w) \setminus \{u\}$ and v_j is strongly adjacent to all leaf vertices of C_w so that $B(v_j) = \{u\}$. Consequently $B(v_i)$ and $B(v_j)$ are disjoint sets. \diamond

Claim 6.4.8. *Let H be a nice disjoint solution. Then, the following hold:*

- (i) *The number of weak components in H is at least $\lceil k/2 \rceil + \lfloor t/2 \rfloor$.*
- (ii) *Every vertex of $W \cup X$ has strong neighbors in at most one weak component of H .*
- (iii) *For every weak component C_w of H , there are exactly two vertices of $W \cup X$ that have strong neighbors in C_w .*

Proof. Let C_w be a weak component of H . Since H is a nice disjoint solution, C_w is a tree which means $m(C_w) = n(C_w) - 1$. From Equation 6.1 we get $|E(H')| = |E(U \cup Y)| + c$, where c is the number of weak components in H' . Thus we have $c = \lceil k/2 \rceil + \lfloor t/2 \rfloor$.

For (ii), let v be a vertex of $W \cup X$ that has strong neighbors in a weak component C_w . Property (W1) implies that $B(v) \cap C_w \neq \emptyset$. This means that for any vertex $v' \in S(v)$ all edges between v' and $B(v)$ are weak from the P_3 -closure. Thus we have $S(v) \subset V(C_w)$.

For a weak component C_w , we know that there are at least two vertices v_1, v_2 of $W \cup X$ that have strong neighbors in C_w by property (W1). By Claim 6.4.6, C_w is a star graph. Let u be the non-leaf vertex of C_w . If $u \notin S(v_1) \cup S(v_2)$ then $u \in B(v_1) \cap B(v_2)$ which is not possible by Claim 6.4.7. Without loss of generality assume that $u \in B(v_1)$. Then, by property (W1) we have $u \in S(v_2)$. Recall that $S(v) \cap S(v') = \emptyset$ for any two vertices $v, v' \in W \cup X$. Assume that there is a vertex $v \in (W \cup X) \setminus \{v_1, v_2\}$ that is strongly adjacent to C_w . Then $u \notin S(v)$ and $S(v)$ contains a non-leaf vertex of C_w . Thus we reach a contradiction to Claim 6.4.7 because $u \in B(v)$ by property (W1) and $u \in B(v_1)$. Therefore the third statement follows. \diamond

Now we are equipped with our necessary tools to show our claimed result. Given a solution H of G with at least k' edges, Claims 6.4.2, 6.4.4, and 6.4.5 imply that there is a nice disjoint solution H' with $|E(H')| = k'$. By Claim 6.4.8 (i) there are at least $\lceil k/2 \rceil + \lfloor t/2 \rfloor$ weak components in H' . Moreover, Claim 6.4.8 (ii) and (iii) imply that there are at least $k + t$ vertices of $W \cup X$ that have a strong neighbor in $U \cup Y$. Recall that $|X| = t$ and, by construction, $B(x)$ with $x \in X$ is disjoint with any $B(v)$ of a vertex $v \in (W \cup X) \setminus \{x\}$. Thus there are at least k vertices in W that have a strong neighbor in $U \cup Y$. Claim 6.4.7 shows that all vertices of W that are incident to at least one strong edge in H' must have disjoint non-neighborhood. Since $B(w_i) = B_i$, there are k pairwise disjoint sets in $\{B_1, \dots, B_p\}$ for the k vertices of W that are incident to at least one strong edge in H' . Therefore there is a solution for the SET PACKING problem for $(\mathcal{U}, B_1, \dots, B_p, k)$. \square

Let F be a graph that has at least one component with at least three vertices. If M is a matching in a graph G , then the spanning subgraph H of G with $E(H) = M$ satisfies the F -closure. Hence, if G has a matching of size at least k , then (G, k) is a yes instance of STRONG F -CLOSURE. Such instances that admit a solution that is given by a matching can be detected in polynomial time, since the size of a maximum matching of a graph can be computed in polynomial time [105]. This gives rise to the question about the parameterized complexity of STRONG F -CLOSURE with the parameter $r = k - \mu(G)$. We show that MAXSTC is FPT with this parameter for the instances where $\Delta(G) \leq 4$. Note that MAXSTC is NP-complete on graphs G with $\Delta(G) \leq d$ for every $d \geq 4$ by Theorem 4.4.1.

Theorem 6.4.9. *MAXSTC can be solved in time $2^{O(r)} \cdot n^{O(1)}$ on graphs of maximum degree at most 4, where $r = k - \mu(G)$.*

Proof. Let (G, k) be an instance of MAXSTC such that $\Delta(G) \leq 4$. Let also $r = k - \mu(G)$.

Recall that a *triangle* is a cycle on three vertices. Slightly abusing notation, we do not distinguish between a triangle and its set of vertices and write $G - T$ instead of $G - V(T)$ for a triangle T and do the same for a union of triangles.

We construct the set of vertices X and the set of edges A as follows. Initially, $X = \emptyset$ and $A = \emptyset$. Then we exhaustively perform the following steps in a greedy way:

1. If there exists a copy of K_4 in $G - X$, we add the vertices of this K_4 to X and the edges between these vertices to A .
2. If there exists a triangle T in $G - X$ such that $\mu(G - X) < 3 + \mu(G - X - T)$, we add the vertices of T to X and the edges of T to A .

Let M be a maximum matching of $G - X$ for the obtained set X . Note that the spanning subgraph H of G with the set of edges $A \cup M$ is a disjoint union of complete graphs with 1, 2, 3 or 4 vertices, that is, H has no induced path on three vertices. Hence, H satisfies the P_3 -closure. Assume that Step 1 was applied p times and we used Step 2 q times. Clearly, $|A| = 6p + 3q$. Notice that the vertices of a copy of K_4 can be incident to at most 4 edges of a matching and the complete graph with 4 vertices has 6 edges. Observe also that by the application of Step 2, we increase the size of A by 3 and $\mu(G - X) - \mu(G - X - T) \leq 2$. This implies that $|E(H)| = |A| + |M| \geq \mu(G) + 2p + q$. Therefore, if $2p + q \geq r$, (G, k) is a yes-instance of MAXSTC. Assume from now that this is not the case. In particular, it means that $|X| \leq 4r$ and $G' = G - X$ is a K_4 -free graph. By the choices made in both steps, notice that every vertex of X has at least two neighbors inside X . Let $Y = V(G) \setminus X = V(G')$.

We need some structural properties of G' and (possible) solutions for the considered instance of MAXSTC.

Claim 6.4.10. *If T is a triangle in G' , then T satisfies the following properties:*

- (i) T contains no edge of M ;
- (ii) every vertex of T is incident to an edge of M .

Proof: If either (i) or (ii) does not hold, the triangle T is such that at most two edges of the matching M are incident to its vertices. This implies that $\mu(G') < 3 + \mu(G' - T)$, which is a contradiction with the fact that Step 2 can no longer be applied. \diamond

We say that a solution H for (G, k) is *regular* if $H[Y]$ is a disjoint union of triangles, edges and isolated vertices. We also say that a solution H is *triangle-maximal* if (i) it contains the maximum number of edges and, subject to (i), (ii) contain the maximum number of pairwise distinct triangles.

Claim 6.4.11. *If (G, k) is a yes-instance of MAXSTC, then every triangle-maximal solution is regular.*

Proof: Let H be a triangle-maximal solution for (G, k) .

We first note that, H has no $K_{1,3}$ as a subgraph. Otherwise it would imply the existence of a K_4 in $G - X$, because for every copy xyz of (not necessarily induced) P_3 in H , $xz \in E(G)$ if H satisfies the P_3 -closure. This implies that H consists of a disjoint union of paths and cycles. Consider an induced path on three vertices $P_3 = v_1v_2v_3$ in H . By the P_3 -closure there is the edge v_1v_3 in G . We prove that the P_3 has a particular form which allows us to make v_1v_3 strong, i.e., the triangle $v_1v_2v_3$ belongs to a solution H . In particular we show that $N_H(v_2) = \{v_1, v_3\}$ and either $N_H(v_1) = \{v_2\}$ and $N_H(v_3) = \{v_2, y\}$ hold or $N_H(v_1) = \{v_2, y\}$ and $N_H(v_3) = \{v_2\}$ hold, where y is a vertex in G .

- First observe that v_2 has no other neighbor in H , because v_2 belongs to a path or a cycle in H . Assume that there is a vertex $x \in X$ that is adjacent to v_2 in H . Then by the P_3 -closure, x is adjacent in G to all three vertices of P_3 which contradicts the fact that $d(x) \leq 4$, because x is adjacent to at least two vertices inside X . Thus v_2 has no other neighbor in H .
- Next assume that there are vertices u_1, u_3 such that $u_1 \in N_H(v_1) \setminus \{v_2\}$ and $u_3 \in N_H(v_3) \setminus \{v_2\}$. If $u = u_1 = u_3$ then u does not belong to Y because there is no K_4 in G' . And if $u \in X$ then by the P_3 -closure, u is adjacent to all three vertices of the P_3 which contradicts the fact that $d(u) \leq 4$. For $u_1 \neq u_3$, notice that v_2 is adjacent to both u_1, u_3 by the P_3 -closure. Then both $u_1v_1v_2$ and $u_3v_3v_2$ form triangles in G , which implies by Claim 6.4.10 that there is an edge v_2v of M with $v \notin \{v_1, v_3, u_1, u_3\}$. This, however, contradicts the fact that $d(v_2) \leq 4$.
- By the previous two arguments, we know that at least one of v_1, v_3 is only adjacent to v_2 in H . Without loss of generality, assume that $N_H(v_1) = \{v_2\}$. If $N_H(v_3) = \{v_2, y, y'\}$ then by the P_3 -closure v_2 is adjacent in G to both y, y' . Applying Claim 6.4.10 shows that there is another edge incident v_2 , contradicting the fact that $d(v_2) \leq 4$. Also note that if $N_H(v_3) = \{v_2\}$ then both v_1, v_3 have no other strong edge incident to them, so that the edge v_1v_3 of G can be made strong which contradicts the maximality of H .

Thus for the given P_3 we know that $N_H(v_1) = \{v_2\}$, $N_H(v_2) = \{v_1, v_3\}$, and $N_H(v_3) = \{v_2, y\}$ or $N_H(v_1) = \{v_2, y\}$, $N_H(v_2) = \{v_1, v_3\}$, and $N_H(v_3) = \{v_2\}$. This means that in both cases we can replace in H the edge v_3y or v_1y by the edge v_1v_3 without violating the P_3 -closure. Iteratively applying such a replacement for every P_3 of H' shows that H is regular. \diamond

In the following we use the notion of distance between two subsets of vertices. For two disjoint subsets of vertices X_1 and X_2 the *distance between X_1 and X_2* is the length of the shortest path among all pairs of vertices v_1 and v_2 with $v_1 \in X_1$ and $v_2 \in X_2$.

Claim 6.4.12. *Let $T = abc$ be a triangle in G' that is at distance one from X . If H is a solution containing T , then H contains no other edge incident to the vertices a , b , and c .*

Proof: Let $T = abc$ be a triangle as described above and let H be a solution containing T . Assume for a contradiction that there exists an edge xa in H that is incident to a vertex of T . Suppose that $x \in X$. This implies that $xb \in E(G)$ and $xc \in E(G)$. Since x has at least two neighbors inside X , we conclude that $d(x) > 4$, a contradiction. If $x \in G - X$, this would imply the existence of K_4 in G' , a contradiction. \diamond

Claim 6.4.13. *Let T be a triangle at distance at least two from X that does not intersect any other triangle. Then T is included in every triangle-maximal regular solution for (G, k) .*

Proof: Let $T = abc$ be a triangle as described above and assume that H is a triangle-maximal regular solution that does not contain T . Since no other triangle intersects T , at most one edge of H is incident to each vertex of T by Claim 6.4.11 and these edges are not included in any other triangle except, possibly, T . If no edge of T is in H , we can replace the edges incident to T by the edges ab , bc and ac and obtain a solution with at least as many edges as H containing T . This solution contains an additional triangle contradicting the condition that H is a triangle-maximal solution. If there exists an edge of T in H , let ab be such an edge. Clearly, ab is the unique edge of the solution in T . Again, since no other triangle intersects T , there is no other edge of the solution that is incident to a or b and at most one edge is incident to c . Then we replace the edge incident to c by the two edges of the triangle abc and obtain a solution with more edges, a contradiction. We conclude that the edges of T are included in H . \diamond

Claim 6.4.14. *If T_1 and T_2 are two intersecting triangles in G' , then the following holds:*

1. T_1 and T_2 have one edge in common;
2. No other triangle intersects T_1 or T_2 .

Proof: Let T_1 and T_2 be two intersecting triangles as described above. Assume for a contradiction that T_1 and T_2 have only a single vertex in common and let a be such a vertex. Recall that M is a maximum matching in $G - X$. By Claim 6.4.10, there exists an edge of the matching incident to a that cannot be contained neither in T_1 nor in T_2 , which implies that $d(a) > 4$, which is a contradiction. We conclude that the triangles must intersect in one edge. Let $T_1 = abc$ and $T_2 = bcd$. Assume for the sake of contradiction that there exists a triangle T that intersects T_1 . Since by the first argument of the claim the triangles T and T_1 cannot intersect in a single vertex, T contains at least one of b or c . Assume $b \in V(T)$. Again, by Claim 6.4.10, there must be an edge of M incident to b that is not contained in any of the triangles, which implies that $d(b) > 4$, a contradiction. This concludes the proof. \diamond

Claim 6.4.15. *If T_1 and T_2 are two intersecting triangles such that T_1 is at distance at least two from X , then either T_1 or T_2 is included in every triangle-maximal regular solution for (G, k) .*

Proof: Let H be a solution. By Claim 6.4.14, T_1 and T_2 have exactly one common edge. Let $T_1 = abc$ and $T_2 = bcd$. Assume that a triangle-maximal regular solution H contains neither T_1 nor T_2 . Note that at most one edge of H is incident to a by Claim 6.4.11. Because H does not contain all the edges of T_2 , the same holds for b and c by Claim 6.4.11. By Claim 6.4.14, these edges are not included in any other triangles except, possibly, T_1 and T_2 . Now we repeat the same arguments as in the proof of Claim 6.4.13. If no edge of T_1 is in H , we can replace the edges incident to T_1 by the edges ab , bc and ac and obtain a solution with at least as many edges as H containing one additional triangle T_1 contradicting the triangle-maximality of H . If there exists an edge of T_1 in H , then at most two edges of H are incident to the vertices of T_1 and we can replace them by the edges of T_1 and increase the number of edges in the solution contradicting the choice of H . \diamond

Given the properties of the triangles in G' and the properties of triangle-maximal regular solutions, we are now ready to solve the problem by finding a regular solution if it exists. Recall that by Claim 6.4.11, a regular solution H to the problem when restricted to $G - X$ is a disjoint union of triangles, edges and isolated vertices. The crucial step is to sort out triangles in G' .

We first consider the triangles in G' that are at distance at most one from the set X in G , that is, the triangles that contain at least one vertex that is adjacent to a vertex of X in G . Since $|X| \leq 4r$ and since every vertex of X has at least two neighbors inside X , we have that $|N_G(X)| \leq 8r$. By Claim 6.4.14, at most 2 triangles of G' contain the same vertex. Thus, the number of pairwise distinct triangles in G' that are at distance at most one from the set X in G is at most $16r$. We list all these triangles, and branch on all at most 2^{16r} choices of the triangles that are included in a triangle-maximal regular solution. Then, for each choice of these triangles, we try to extend the partial solution. If we obtain a solution for one of the choices we return it and the algorithm returns NO otherwise.

Assume that we are given a set \mathcal{T}_1 of triangles at distance one from X that should be in a solution. Note that by Claim 6.4.11, the triangles in \mathcal{T}_1 are pairwise disjoint. We apply the following reduction rule.

Rule 6.4.15.1. *Set $G = G - \cup_{T \in \mathcal{T}_1} T$ and set $k = k - 3|\mathcal{T}_1|$.*

By Claim 6.4.12, the original instance has a regular solution if and only if the obtained instance has a regular solution that does not contain triangles in $G - X$ that are

at distance one from X . Our aim now is to find such a solution. For simplicity, we keep the same notation and assume that $G' = G - X$.

Now we deal with triangles that are at distance at least 2 from X . Consider the set \mathcal{T}_2 of triangles in G' that are at distance at least 2 from X and have no common vertices with other triangles in G' . By Claim 6.4.13, all these triangles are in every triangle-maximal regular solution. It immediately gives us the following rule.

Rule 6.4.15.2. Set $G = G - \cup_{T \in \mathcal{T}_2} T$ and set $k = k - 3|\mathcal{T}_2|$.

We again assume that $G' = G - X$. To consider the remaining triangles, recall that by Claim 6.4.14, for every such a triangle T , T is intersecting with a unique triangle T' of G' and T, T' are sharing an edge.

Let \mathcal{T}_3 be the set of triangles in G' that are at distance at least 2 from X in G and have a common edge with a triangle at distance one from X . Recall that we are looking for a regular solution that does not contain triangles in $G - X$ that are at distance one from X . Then by Claim 6.4.15, triangles of \mathcal{T}_3 should be included to a triangle-maximal regular solution, and we get the next rule.

Rule 6.4.15.3. Set $G = G - \cup_{T \in \mathcal{T}_3} T$ and set $k = k - 3|\mathcal{T}_3|$.

As before, let $G' = G - X$. The remaining triangles in G' at distance at least 2 from X in G form pairs $\{T_1, T_2\}$ such that T_1 and T_2 have a common edge and are not intersecting any other triangle. Let \mathcal{P} be the set of all such pairs. By Claim 6.4.15, a triangle-maximal regular solution contains either T_1 or T_2 . We use this to apply the following rule.

Rule 6.4.15.4. For every pair $\{T_1, T_2\} \in \mathcal{P}$, delete the vertices of T_1 and T_2 from G , construct a new vertex u and make it adjacent to the vertices of $N_G((T_1 \setminus T_2) \cup (T_2 \setminus T_1))$. Set $k = k - 3|\mathcal{P}|$.

Denote by (\hat{G}, \hat{k}) the instance of MAXSTC obtained from (G, k) by the application of Rule 6.4.15.4. We show the following claim.

Claim 6.4.16. *If the instance (G, k) has a triangle-maximal regular solution H that has no triangles in $G - X$ at distance one from X , then there is a solution \hat{H} for (\hat{G}, \hat{k}) such that $\hat{H} - X$ is a disjoint union of edges and isolated vertices, and if there is a solution \hat{H} for (\hat{G}, \hat{k}) such that $\hat{H} - X$ is a disjoint union of edges and isolated vertices, then (G, k) has a regular solution H that has no triangles in $G - X$ at distance one from X .*

Proof: Let H be a triangle-maximal regular solution for (G, k) such that H has no triangles in $G - X$ at distance one from X . Notice that if H contains a triangle, then it belongs to one of the pairs of \mathcal{P} . By Claim 6.4.15, we can assume that H contains a triangle from every pair from \mathcal{P} . We construct a solution \hat{H} for (\hat{G}, \hat{k}) by modifying

H as follows. First, we include in \hat{H} the edges of H that are not incident to the vertices of the pairs of triangles of \mathcal{P} . For every pair $\{T_1, T_2\} \in \mathcal{P}$, H contains either T_1 or T_2 . Assume without loss of generality that T_1 is in H . Let v be the vertex of T_2 that is not included in T_1 . By Claims 6.4.11 and 6.4.10, at most one edge of H is incident to v and there is no edge in H that is incident to exactly one vertex of T_1 . Let u be the vertex of \hat{G} constructed by Rule 6.4.15.4 for $\{T_1, T_2\}$. If $vx \in E(H)$ for some $x \in V(G)$, then we include the edge ux' in \hat{H} , where x' is the vertex constructed from x by the rule; note that it can happen that x is a vertex of some other pair of triangles. Since we include in \hat{H} at most one edge incident to a vertex constructed by the rule, \hat{H} does not contain triangles and is a disjoint union of edges and isolated vertices. Moreover, since $|E(H)| \geq k$, we have that $|E(\hat{H})| \geq k - 3|\mathcal{P}| = \hat{k}$.

Suppose now that \hat{H} is a solution for (\hat{G}, \hat{k}) such that $\hat{H} - X$ is a disjoint union of edges and isolated vertices. Now we construct H by modifying \hat{H} . For every edge uv of \hat{H} such that u and v are vertices of the original graph G , we include uv in H . Assume that $uv \in E(\hat{H})$ is such that $v \in V(G)$ and u was obtained from a pair $\{T_1, T_2\} \in \mathcal{P}$. Then v is adjacent in G to a vertex x that belongs to exactly one of the triangles, say T_1 . We include xv and T_2 in H . Suppose that $uv \in E(\hat{H})$ is such that u was obtained from a pair $\{T_1, T_2\} \in \mathcal{P}$ and v was obtained from a pair $\{T'_1, T'_2\} \in \mathcal{P}$. Then G has an edge xy such that x belongs to exactly one of the triangles T_1, T_2 , say T_1 , and y belongs to exactly one of the triangles T'_1, T'_2 , say T'_1 . We include xy , T_2 and T'_2 in H . Finally, if there is a pair $\{T_1, T_2\} \in \mathcal{P}$ such that for the vertex $u \in V(\hat{G})$ constructed from this pair, \hat{H} has no edge incident to u , we include T_1 in H . With this way we obtain H such that $H - X$ is a disjoint union of triangles, edges and isolated vertices. It remains to note that because $|E(\hat{H})| \geq \hat{k}$, we have that $|E(H)| \geq k$, that is, H is a regular solution. \diamond

By Claim 6.4.16, we have to find a solution for the instance (\hat{G}, \hat{k}) such that $\hat{H} - X$ is a disjoint union of edges and isolated vertices. We do it by branching on all possible choices of edges in a solution that are incident to the vertices of X . Since $|X| \leq 4r$ and $\Delta(G) \leq 4$, there are at most $16r$ edges that are incident to the vertices of X and, therefore, we branch on at most 2^{16r} choices of a set of edges S . Then for each choice of S , we are trying to extend it to a solution. If we can do it for one of the choices, we return the corresponding solution, and the algorithm returns NO otherwise.

Assume that S is given. First, we verify whether the spanning subgraph of G with the set of edges S satisfies the P_3 -closure. If it is not so, we discard the current choice of S since, trivially, S cannot be extended to a solution. Assume that this is not the case. Let $R = \hat{G} - X$. We modify R by the exhaustive application of the following rule.

Rule 6.4.16.1. *If there are vertices x, y, z such that $xy \in E(R)$, $z \in X$, $xz \in S$, and $yz \notin E(\hat{G})$, then delete xy from R .*

Let R' be the graph obtained from R by the rule. Observe that the edges deleted by

Rule 6.4.16.1 cannot belong to a solution. Hence, to extend S , we have to complement it by some edges of R' that form a matching. Moreover, every matching of R' could be used to complement S . To see this, observe that every matching of R' and the edges of S satisfy the P_3 -closure. By Rule 6.4.16.1, we ensure that the edges of $S \cup M$ in \hat{G} satisfy the P_3 -closure. Respectively, we find a maximum matching M in R' in polynomial time [105]. We obtain that the spanning subgraph \hat{H} of \hat{G} with $E(\hat{H}) = S \cup M$ satisfies the P_3 -closure. We verify whether $|S| + |M| \geq \hat{k}$. If it holds, we return \hat{H} . Otherwise, we discard the current choice of S .

The correctness of the algorithm follows from the properties of Rules 6.4.15.1–6.4.16.1 and Claim 6.4.16. To evaluate the running time, observe that Steps 1 and 2 that are used to construct X and A can be done in polynomial time. Then we branch on at most 2^{16r} choices of \mathcal{T}_1 . For each choice, we apply Rules 6.4.15.1–6.4.15.4 in polynomial time. Then we consider at most 2^{16r} choices of a set of edges S . For each choice, we apply Rule 6.4.16.1 in polynomial time and then compute a maximum matching in R' [105]. Summarizing, we obtain the running time $2^{O(r)} \cdot n^{O(1)}$. \square

6.5 Further Results

To complement our results so far, we give here the parameterized complexity results when our problem is parameterized by the number of weak edges. The following result is not difficult to deduce using similar ideas to those used in proving that F -FREE EDGE DELETION is FPT by the number of deleted edges [16].

Theorem 6.5.1. *For every fixed graph F , STRONG F -CLOSURE can be solved in time $2^{O(\ell)} \cdot n^{O(1)}$, where $\ell = |E(G)| - k$.*

Proof. We basically use the main idea given in [16]. Since F is of fixed size, we can list all induced subgraphs of G isomorphic to F in polynomial time. For each induced subgraph F' we check whether $G[F'] \simeq F$. If $G[F'] \simeq F$, then we must remove at least one of the edges of F' . We branch at all such possible $|E(F)|$ edges and on each resulting graph we apply the same procedure for at most ℓ steps. If at some intermediate graph we have $G[F'] \not\simeq F$ for all of its induced subgraphs then we have found the desired subgraph within at most ℓ edge deletions. Otherwise, we can safely output that there is no such subgraph with at most ℓ edge removals. As the depth of the search tree is bounded by ℓ , the overall running time is $2^{O(\ell)} \cdot n^{O(1)}$. \square

Next we show that STRONG F -CLOSURE has a generalized polynomial kernel with this parameterization whenever F is a fixed graph. We obtain this result by constructing generalized kernelization that reduces STRONG F -CLOSURE to the d -HITTING SET

problem that is the variant of HITTING SET with all the sets in \mathcal{C} having d elements. Notice that this result comes in contrast to the F -FREE EDGE DELETION problem, as it is known that there are fixed graphs F for which there is no polynomial compression [17] unless $\text{NP} \subseteq \text{coNP}/\text{poly}$.

Theorem 6.5.2. *For every fixed graph F , STRONG F -CLOSURE has a generalized polynomial kernel, when parameterized by $\ell = |E(G)| - k$.*

Proof. Let d be the number of edges of F . We enumerate all the induced subgraphs of G isomorphic to F in polynomial time. Let $\mathcal{F}_G = \{F_1, \dots, F_q\}$ be the produced subgraphs isomorphic to F such that $V(F_i) \neq V(F_j)$. For each $F_i \in \mathcal{F}_G$, we construct the set $E_i = E(F_i)$. Notice that $|E_1| = \dots = |E_q| = d$. Now our task is to select at most ℓ edges E' from G such that $E' \cap E_i \neq \emptyset$ for every E_i . We claim that such a subset of edges is enough to produce a solution for the STRONG F -CLOSURE. To see this, consider an F -graph F_i of G and denote by G' the graph obtained from G by removing an edge $e = xy$ of F_i . Assume for contradiction that at least one new F -graph F' is created in G' so that $F' \notin \mathcal{F}_G$ and $F' \in \mathcal{F}_{G'}$. Then both x and y must belong to F' which implies that x and y are non-adjacent in $G'[F']$. This, however, contradicts the fact that $G[F']$ induces a graph isomorphic to F , because x and y are adjacent in G . Thus $\mathcal{F}_{G'} \subset \mathcal{F}_G$ which implies that the described set of edges E' constitutes a solution. This actually corresponds to the d -HITTING SET problem: given a collection of sets $C_i = E_i$ each of size d from a universe $U = E(G)$, select at most ℓ elements from U such that every set C_i contains a selected element. Then we use the result of Abu-Khzam [1] (see also [30]) that d -HITTING SET admits a polynomial kernel with the universe size $O(\ell^d)$ and with $O(\ell^d)$ sets. \square

Observe that whenever STRONG F -CLOSURE is polynomially solvable or NP-complete for a given F , Theorem 6.5.2 implies that STRONG F -CLOSURE admits a polynomial kernel. If the problem can be solved in polynomial time, then it has a trivial kernel. If STRONG F -CLOSURE is NP-complete, then there is a polynomial reduction of d -HITTING SET to STRONG F -CLOSURE. Combining the generalized kernelization and this reduction, we obtain a polynomial kernel.

We would like to underline that Theorems 6.5.1 and 6.5.2 are fulfilled for the case when F is a fixed graph of constant size, as the degree of the polynomial in the running time of our algorithm depends on the size of F and, similarly, the size of F is in the exponent of the function defining the size of our generalized kernel. We can hardly avoid this dependence as it can be observed that for $\ell = 0$, STRONG F -CLOSURE is equivalent to asking whether the input graph G is F -free, that is, we have to solve the INDUCED SUBGRAPH ISOMORPHISM problem. It is well known that INDUCED SUBGRAPH ISOMORPHISM parameterized by the size of F is $\text{W}[1]$ -hard when F is a complete

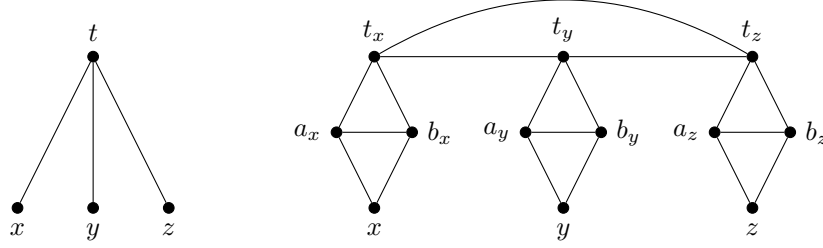


Figure 6.3: The planar configuration used in the proof of Theorem 6.5.3.

graph or graph without edges [39], and the problem is $W[1]$ -hard when F belongs to other restricted families of graphs [81].

We conclude with a few open problems. An interesting question is whether MAXSTC is FPT when parameterized by $r = k - \mu(G)$. We proved that this holds on graphs of maximum degree at most 4, and we believe that this question is interesting not only on general graph but also on various other graph classes. In particular, what can be said about planar graphs? To set the background, we show that MAXSTC is NP-hard on planar graphs and $(3K_1, 2K_2)$ -free graphs. The Lemma 3.2.4 is needed for the proofs of Theorems 6.5.3 and 6.5.5.

Lemma 3.2.4: Let x and y be true twins in G . Then, there is an optimal solution H for MAXSTC such that $xy \in E(H)$ and for every vertex $u \in N(x)$, $xu \in E(H)$ if and only if $yu \in E(H)$.

Theorem 6.5.3. MAXSTC is NP-hard on planar graphs.

Proof. We show the theorem by a reduction from $\text{PLANARX}_3\text{C}$. In X_3C we are given a set X with $|X| = 3q$ elements and a collection C of triplets of X and the problem asks for a subcollection $C' \subseteq C$ such that every element of X occurs in exactly one member of C' . If such a subcollection C' exists, then it is called an *exact cover* of X . For the $\text{PLANARX}_3\text{C}$ we associate a bipartite graph G with this instance as follows: we have a vertex for every element of X and a vertex for every triplet of C and there is an edge between an element and a triplet if and only if the element belongs to the triplet. The problem is known to be NP-complete even restricted to instances whose associated graph is planar [40]. Let $G = (X \cup C, E)$ be an instance of $\text{PLANARX}_3\text{C}$ with $|C| = m \geq q$. We construct another graph G' by replacing the three edges incident to each triplet with the configuration shown in Figure 6.3. More precisely, we replace each triplet vertex t by a triangle $\{t_x, t_y, t_z\}$ (*middle triangle*) and for each original edge tx we introduce two triangles $\{t_x, a_x, b_x\}$ (*inner triangle*) and $\{a_x, b_x, x\}$ (*outer triangle*). Thus for every triplet we associate seven triangles in which four of them are

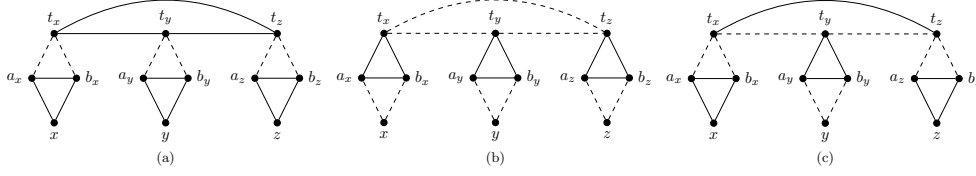


Figure 6.4: A solid edge corresponds to a strong edge, whereas a dashed edge corresponds to a weak edge. Form (a) has 12 strong edges and corresponds to a triplet that is a member of an exact cover. Form (b) has 9 strong edges and corresponds to a triplet that does not belong to an exact cover. Form (c) contains all other cases; we depict only one of them.

vertex-disjoint (the middle and the outer triangles) and the other three triangles (inner triangles) share all their vertices with two vertex-disjoint triangles. Such a subgraph corresponding to the triplet $(x, y, z) \in C$ is simply called *triplet subgraph*. Observe that any two triplet subgraphs have in common only a subset of the vertices x, y, z of their outer triangles. Notice also that G' remains a planar graph. We prove that $\text{PLANARX}_3\text{C}$ has an exact cover if and only if G' has a spanning subgraph with at least $9m + 3q$ strong edges that satisfies the P_3 -closure.

Assume C' is an exact cover for $\text{PLANARX}_3\text{C}$ with $|C'| = q$. If a triplet belongs to C' then we make the edges of all four vertex-disjoint triangles strong (see Figure 6.4 (a)). If a triplet does not belong to C' then we make the edges of all inner triangles strong (see Figure 6.4 (b)). This labeling satisfies the P_3 -closure as there is no P_3 spanned by strong edges and the total number of strong edges is $12q + 9(m - q)$ which gives the claimed bound.

For the opposite direction, assume that G' has a spanning subgraph H with at least $9m + 3q$ strong edges. Consider the graph induced by the vertices $\{t_x, a_x, b_x, x\}$ that corresponds to an original edge between an element x and a triplet t . Since a_x, b_x are true twins in G , by Lemma 3.2.4, $E(H)$ contains the edge $a_x b_x$ and a_x, b_x are also true twins in H . The latter implies that either one of the two triangles $\{x, a_x, b_x\}, \{t_x, a_x, b_x\}$ belongs to H , or no such triangle belongs to H . The same observation carries along the vertices a_y, b_y and a_z, b_z . Thus for every triplet subgraph, $E(H)$ contains all its outer triangles, or all its inner triangles, or a combination of some inner and outer triangles. These cases correspond to the three forms given in Figure 6.4. We show that there exists an optimal solution H only with the first two forms of Figure 6.4, which particularly means that every triplet subgraph of H contains either all its outer triangles or all its inner triangles.

To prove this, we first show that every middle triangle in a triplet subgraph has either

all its edges strong or none of its edges is strong. We refer to the former case as strong middle triangle and the later as weak middle triangle. Assume that the middle triangle contains at least one strong edge $t_x t_z$. Then there is no other strong edge incident to t_x or t_z . If the inner triangle of t_y is not strong, then we can safely make the edges $t_y t_x, t_y t_z$ strong. Otherwise, the inner triangle of t_y is strong and we remove both edges $t_y a_y$ and $t_y b_y$ from H and add the edges $t_y t_x, t_y t_z$. Thus if there is a strong edge in the middle triangle then there is a solution with a strong middle triangle.

Next we consider a (strong or weak) middle triangle. If a middle triangle is weak then $E(H)$ contains at most 9 edges from its triplet subgraph. In such a case we replace all its edges from $E(H)$ by the 9 edges of its inner triangles by keeping the same size for $E(H)$. The replacement is safe with respect to the P_3 -closure because the inner triangles of each triplet subgraph are vertex-disjoint with any other triplet subgraph. For every strong middle triangle notice that all the edges of its inner triangles are weak. If there is at most one outer triangle that is strong then we make the middle triangle weak and we replace its edges of $E(H)$ by the edges of its inner triangles. Thus for every strong middle triangle we know that either two or three outer triangles are strong. Also recall that for every weak middle triangle, all its outer triangles are weak.

For $i \in \{0, 2, 3\}$, let ℓ_i be the number of triplet subgraphs in which there are i outer triangles strong. We will show that, since H contains at least $9m + 3q$ edges, there are no triplet subgraphs with exactly two outer triangles strong, i.e., $\ell_2 = 0$. Observe that $\ell_0 + \ell_2 + \ell_3 = m$. Also notice that each of the subgraphs corresponding to ℓ_0 contains 9 strong edges, ℓ_2 contains 10 strong edges, and ℓ_3 contains 12 strong edges. Therefore the total number of strong edges is $9\ell_0 + 10\ell_2 + 12\ell_3$. As H contains at least $9m + 3q$ edges, we get $\ell_2 + 3\ell_3 \geq 3q$. Now notice that every vertex of X is incident to at most one strong triangle. Thus for each of the ℓ_2 subgraphs there are 2 vertices in X that are incident to strong edges, whereas for each of the ℓ_3 subgraphs there are 3 such vertices in X . This implies that $2\ell_2 + 3\ell_3 \leq |X| = 3q$. Therefore $|E(H)| \geq 9m + 3q$ holds only if $\ell_2 = 0$ and $\ell_3 = q$, so that all triplet subgraphs with strong middle triangles correspond to an exact cover for the elements of X . \square

Moreover, in proof of Theorem 6.5.3 every strong connected component forms a clique and, thus, we have the following corollary.

Corollary 6.5.4. *CLUSTER DELETION is NP-hard on planar graphs.*

We next proceed with the $(3K_1, 2K_2)$ -free graphs. The reduction comes from the CLIQUE problem which is known to be NP-complete on such graphs [58].

Theorem 6.5.5. *MAXSTC restricted on $(3K_1, 2K_2)$ -free graphs remains NP-hard.*

Proof. Let (G, k) be an instance of CLIQUE with G being a $(3K_1, 2K_2)$ -free graph. From G we construct G' by adding a clique X of size $x = nk$ such that every vertex of X is adjacent to every vertex of G . Clearly G' remains $(3K_1, 2K_2)$ -free graph. We show that G has a solution for CLIQUE of size at least k if and only if G' has a spanning subgraph that satisfies the P_3 -closure with at least $q = \frac{x(x-1)}{2} + \frac{k(k-1)}{2} + kx$ strong edges.

Assume that $C \subseteq V(G)$ is a solution for CLIQUE on G of size at least k . Then $C \cup X$ is a clique in G' . Maintaining only the edges of $C \cup X$ in a spanning subgraph of G' does not create any P_3 . Thus there is a spanning subgraph of G' that satisfies the P_3 -closure and the number of edges in $G'[C \cup X]$ gives the desired bound.

For the opposite direction, assume that H is such a solution for STRONG P_3 -CLOSURE on G' . Observe that the vertices of X have the same closed neighborhood in G' , so they are true twins. By Lemma 3.2.4 we know that all vertices of X have the same neighborhood in H and all edges inside X are strong. If there is a vertex of X with k strong neighbors in $G'[V]$ then there is a k -clique in G . Moreover if there is a vertex of $G'[V]$ with $k - 1$ strong neighbors then those vertices induce a clique of size k . We show that at least one of the two conditions holds in H . Assume for contradiction, that there is no such vertex: all the vertices of X have the same $k - 1$ strong neighbors in $G'[V]$, and every vertex of $G'[V]$ has at most $k - 2$ strong neighbors. This means that the claimed solution has at most p strong edges where $p = \frac{x(x-1)}{2} + x(k-1) + n(k-2)$. Since $p < q$, we get a contradiction to the number of strong edges in H . Thus there is at least one vertex v of the following type: either $v \in X$ with at least k strong neighbors in G or $v \in V(G)$ with at least $k - 1$ strong neighbors in G . Therefore in both cases we get a k -clique in G . \square

The proof of Theorem 6.5.5 can be generalized to any graph class Π for which the following two conditions hold: (i) CLIQUE is NP-hard on Π and (ii) Π is closed under addition of a universal vertex.

Regarding the parameterization by $r = k - \mu(G)$, it is still interesting to extend STRONG F -CLOSURE when $F \neq P_3$ has a connected component with at least three vertices. As a first step, we give an FPT result when F is a star.

Theorem 6.5.6. *For every $t \geq 3$, STRONG $K_{1,t}$ -CLOSURE can be solved in time $2^{O(r^2)} \cdot n^{O(1)}$, where $r = k - \mu(G)$.*

Proof. We prove the theorem by constructing a kernel for the problem.

Let G be a graph and M be a maximum matching of G . We assume without loss of generality that G has no isolated vertices. Otherwise, we just delete such vertices and, trivially, obtain an equivalent instance of the problem. Let V_M be the set of vertices of G that are covered by M . Let X be a subset of vertices of $V(G)$ and A be a subset of edges

of $G[X]$, both initially set to be empty. We add elements to X and A by performing the following steps in a greedy way:

1. If there is $v \in V(G) \setminus V_M$ and $xy \in M$ such that $vx \in E(G)$ or $vy \in E(G)$, then we add v, x and y to X and add all the edges between $\{v, x, y\}$ to A .
2. If there is $xy, wz \in M$ such that $G[\{x, y, w, z\}] \not\cong 2K_2$, then we add x, y, w and z to X and add all the edges between $\{x, y, w, z\}$ to A .

Note that since the set $\{v, x, y\}$ does not induce a $K_{1,t}$, and since $xy, wz \in E(G)$, the set $\{x, y, w, z\}$ does not induce a $K_{1,t}$ either, the edges added to A in each step can be part of a solution. Moreover, after each application of step 1 or step 2, the size of the set $A \cup M$ is increased by at least one. As a consequence, if the steps can be applied at least r times, then $|A \cup M| \geq |M| + r$ and therefore we have a yes instance. Assume that this is not the case. This implies that $|X| < 4r$.

After the exhaustive application of steps 1 and 2 in a greedy way, we consider the matching obtained from M by the deletion of the edges included in A . For simplicity, we call this matching M again and use V_M to denote the set of vertices of $V(G) \setminus X$ that are covered by M . Observe that M is a maximum matching of $G - X$. Since step 1 cannot be applied, we have that the vertices of the set $W = V(G) \setminus (X \cup V_M)$ are not adjacent to the vertices of V_M . By the maximality of M , the vertices of W are pairwise nonadjacent. Because step 2 can no longer be applied, M is actually an induced matching of G . That is, $G - X$ is the disjoint union of the edges of M and the isolated vertices of W .

In what follows, we show that the sizes of W and M can be reduced to bound them by a function of r .

Recall that G has no isolated vertices. Hence, each vertex of W is adjacent to a vertex of X . We partition the vertices of W according to their neighborhood in X , that is, two vertices $x, y \in W$ are in the same class if and only if $N_G(x) = N_G(y)$. Clearly, we obtain at most $2^{|X|} \leq 2^{4r}$ classes. We exhaustively apply the following rule.

Rule 6.5.6.1. *If there exists a class of vertices of W that has size at least $(t - 1) \cdot 4r + 1$, then delete one vertex of the class from the graph.*

To see that Rule 6.5.6.1 is safe, assume that one given class contains at least $(t - 1) \cdot 4r + 1$ vertices and we applied the rule for this class. Denote by G' the obtained graph. Observe that every vertex of X can be adjacent to at most $t - 1$ vertices of $G - X$ in a solution, otherwise the solution would contain a set of strong edges inducing a $K_{1,t}$ in G . This, together with the fact that $|X| < 4r$, gives us that at most $(t - 1) \cdot 4r$ vertices of W are adjacent to vertices of X in a solution. Since the class contains at least $(t - 1) \cdot 4r + 1$ vertices, at least one vertex of the class has no incident edges in the

solution. Notice that $\mu(G') = \mu(G)$. Therefore, if (G, k) is a yes instance, then (G', k) is a yes instance as well. For the opposite direction, it is sufficient to observe that every solution to (G', k) is a solution for (G, k) .

We use similar approach to reduce the size of M . We partition M to classes according to their neighborhood in X . More precisely, two edges $x_1y_1, x_2y_2 \in M$ are in the same class if and only if either $N_G(x_1) \cap X = N_G(x_2) \cap X$ and $N_G(y_1) \cap X = N_G(y_2) \cap X$ or, symmetrically, $N_G(x_1) \cap X = N_G(y_2) \cap X$ and $N_G(y_1) \cap X = N_G(x_2) \cap X$. There are at most 2^{4r} possible subsets of X that can be the neighborhood of a given vertex of $G - X$. Then, we can partition the edges of M into at most 2^{8r} classes, according to the neighborhoods of the two endpoints of the edge. We exhaustively apply the following rule.

Rule 6.5.6.2. *If there exists a class of edges of M that has size at least $(t-1) \cdot 4r + 1$, then delete the end-vertices of one edge of the class from the graph and reduce the parameter k by one.*

To show safeness, suppose that one given class contains at least $(t-1) \cdot 4r + 1$ edges. Assume that Rule 6.5.6.2 is applied for this class and denote by G' the graph obtained by the application of the rule. Since M is an induced matching in G , every vertex of X can be adjacent to end-vertices of at most $t-1$ edges of M in a solution, otherwise the solution would contain a set of strong edges inducing a $K_{1,t}$ in G . Together with the fact that $|X| < 4r$, we obtain that at most $(t-1) \cdot 4r$ edges of M have end-vertices that are adjacent to vertices of X in a solution. Since the class contains at least $(t-1) \cdot 4r + 1$ edges, at least one edge of the class is such that both of its end-vertices are not adjacent to any vertex of X in the solution. This edge can therefore be part of every solution H . Note that $\mu(G') = \mu(G) - 1$. This implies that if (G, k) is a yes instance, then $(G', k-1)$ is a yes instance. For the opposite direction, consider any solution for $(G', k-1)$. Clearly, we can construct a solution for (G, k) by adding the edge that was deleted by the rule. Hence, if $(G', k-1)$ is a yes instance, then (G, k) is a yes instance.

Once Rules 6.5.6.1 and 6.5.6.2 have been exhaustively applied, the number of vertices of the graph is bounded by $4r + 2^{4r} \cdot (t-1) \cdot 4r + 2^{8r} \cdot (t-1) \cdot 4r \cdot 2 = g(r)$. It is now possible to use brute force to solve the problem in the following way. First, we guess which edges inside X go into the solution. Since $|X| < 4r$, this guessing takes $2^{O(r^2)}$ time. Since every vertex of X can have at most $2t-2$ neighbors in $G - X$ in a solution, we can again guess which edges from X to $G - X$ go into the solution. This takes $2^{O(r^2)}$ time. Finally, for each of these guesses made for the edges in $E(G) \setminus M$, we test which edges of M can be added into the solution without forming an induced $K_{1,t}$ in H that also induce a $K_{1,t}$ in G . This takes time $2^{O(r)}$. The total running time of the brute force algorithm is therefore $2^{O(r^2)} \cdot n^{O(1)}$. \square

CONCLUSION

7.1 Summary

In this thesis, we mainly considered MAXSTC, an optimization problem that arises in social network analysis, and CLUSTER DELETION, a classical edge modification problem, and we studied their computational complexity on several graph classes. Moreover, we introduced the STRONG F -CLOSURE problem, which serves as a generalization of MAXSTC, as well as a relaxation of F -FREE EDGE DELETION. We studied STRONG F -CLOSURE from a parameterized perspective with various natural parameterizations.

In particular, we studied the NP-complete problem MAXSTC in important classes of graphs, since, to the best of our knowledge, no previous results were known prior to our work when restricting the input graph for the MAXSTC problem. We first considered subclasses of chordal graphs. We showed that MAXSTC remains NP-hard on split graphs (Theorem 3.3.4), and consequently also on chordal graphs. However, we solved MAXSTC in polynomial time on proper interval graphs (Theorem 3.4.17) and trivially perfect graphs (Theorem 3.2.3). On proper interval graphs, we characterized a specific solution by taking advantage of the ordering of the vertices and we designed a dynamic programming algorithm to construct such a solution. On trivially perfect graphs we characterized their line-incompatibility graph and we solved the INDEPENDENT SET on their line-incompatibility graph which consists an optimal solution for MAXSTC. Moreover, we expressed MAXSTC and CLUSTER DELETION in monadic second order logic of second type (MSO_2) and, thus, both problems can be solved in linear time on graphs of bounded treewidth (Subsection 3.2.1).

Furthermore, we presented a 2-dimension scheme which helped us to solve MAXSTC in polynomial time on cographs (Theorem 4.3.6). The optimal solution of the problem on cograph can be greedily taken by a sequence of maximum cliques. As a byproduct of this procedure, we show that INDEPENDENT SET of the cartesian product of two cographs is polynomial solvable (Theorem 4.3.8). Moreover, we studied the influence of low maximum degree for the MAXSTC problem. We show an interesting complexity

Graphs	CLUSTER DELETION	MAXSTC	Matched
General	NP-hard [116]	NP-hard [118]	<
Chordal	NP-hard [11]	NP-hard	<
Interval	Poly-time [Th. 5.3.14]	?	<
Proper interval	Poly-time [11]	Poly-time [Th. 3.4.17]	<
Split	Poly-time [11]	NP-hard [Th. 3.3.4]	<
Starlike	NP-hard [[11], Th. 5.4.3]	NP-hard	<
Cograph	Poly-time [50]	Poly-time [Th. 4.3.6]	✓
Trivially-perfect	Poly-time [50]	Poly-time [Th. 3.2.3]	✓
Triangle-free	Poly-time	Poly-time	✓
Bounded treewidth	Poly-time	Poly-time	
Planar	NP-hard [Cor. 6.5.4]	NP-hard [Th. 6.5.3]	
$(3K_1, 2K_2)$ -free	NP-hard [50]	NP-hard [Th. 6.5.5]	
$\Delta = 3$	Poly-time [85]	Poly-time [Th. 4.4.3]	✓
$\Delta \geq 4$	NP-hard [85]	NP-hard [Th. 4.4.1]	

Table 7.1: Complexity of CLUSTER DELETION and MAXSTC restricted on particular graph classes. Our results obtained within this thesis are presented in **bold**. In column Matched, the symbols “✓” or “<” indicate whether the optimal value of MAXSTC matches or is greater than, respectively, to the optimal value of CLUSTER DELETION.

dichotomy result: for graphs of maximum degree four MAXSTC remains NP-complete (Theorem 4.4.1), whereas for graphs of maximum degree three the problem is solved in polynomial time (Theorem 4.4.3). Our reduction for the NP-completeness on graphs of maximum degree four implies that, under the Exponential-Time Hypothesis, there is no subexponential time algorithm for MAXSTC.

Next, we considered the close related CLUSTER DELETION problem. It is a well-studied NP-complete problem and there are several computational results when the input graph is restricted on graph classes. However, the complexity of CLUSTER DELETION on interval graphs was left as an open problem by several researchers for more than a decade. Here, we provided a polynomial time algorithm for computing an optimal solution for CLUSTER DELETION when the input graph is an interval graph (Theorem 5.3.14) and, thus, we settled the open problem in the affirmative. Our algorithm relied on the linear structure obtained from the clique path of the interval graph. We considered a dynamic programming approach defined by two vertex ordering to solve the problem. Moreover, we studied the class of starlike graphs, a slight generalization of split graphs. We characterized starlike graphs by a list of forbidden subgraphs and we showed that CLUSTER DELETION is NP-hard ([11], Theorem 5.4.3). We also studied stable-like, threshold-like and laminar-like graphs that consist three subclasses of starlike graphs. In all cases we proved that CLUSTER DELETION can be solved in

polynomial time (Subsection 5.4.1). In Table 7.1 we summarize our results together with previously-known results concerning the complexity of CLUSTER DELETION and MAXSTC on graph classes.

Apart from our results on classical computational complexity of MAXSTC and CLUSTER DELETION on particular graph classes, we proposed sufficient conditions so that the optimal solutions of MAXSTC and CLUSTER DELETION coincide. We deduced that when the input graph is restricted either on K_3 -free graphs, on P_4 -free graphs or on graphs with maximum degree 3, a cluster graph corresponds to an optimal solution for both problems. However, we found instances on split graphs ($(2K_2, C_4, C_5)$ -free graphs) and proper interval graphs (claw-free interval graphs) for which an optimal solution of MAXSTC is slightly larger than an optimal solution of CLUSTER DELETION (see for e.g. Figure 4.1). Furthermore, it should be noticed that among the considered graph classes, split graphs consists the only graph family for which the complexity of MAXSTC differs from the complexity of CLUSTER DELETION. In Table 7.1 we highlight this relationship between the two problems regarding their optimal solutions.

Motivated by the role of triadic closures in social networks, and the importance of finding a maximum subgraph that avoids a fixed pattern, we introduced and initiated the parameterized study of the STRONG F -CLOSURE problem, where F is a fixed graph. This study has been done with three different natural parameters: the number of strong edges k , the number of strong edges above guarantee (maximum matching size) $k - \mu(G)$, and the number of weak edges ℓ . We showed that the STRONG F -CLOSURE is FPT when parameterized by k , for a fixed F (Theorem 6.3.8). Also, we proved that the problem is FPT even when we allow the size of F to be a parameter, that is, if we parameterize the problem by $k + |V(F)|$ (Corollary 6.3.9), unless F has at most one edge. In the latter case STRONG F -CLOSURE is co- $W[1]$ -hard when parameterized by $|V(F)|$ even if $k \leq 1$ (Propositions 6.3.2, 6.3.3). With this parameterization we observed that STRONG F -CLOSURE admits a polynomial kernel when the input graph is planar graph or a d -degenerate graph (Proposition 6.3.10). Next, we studied the special case of $F = P_3$ which coincides with the MAXSTC problem. We complemented our FPT results by proving that MAXSTC does not admit polynomial kernel even on split graphs (Theorem 6.4.1). Moreover, on graphs of maximum degree at most 4, we were able to show that MAXSTC is FPT with the above guarantee parameterization $k - \mu(G)$ (Theorem 6.4.9). Under the same parameterization, we showed that for every $t \geq 3$, STRONG $K_{1,t}$ -CLOSURE is FPT (Theorem 6.5.6). Furthermore, we proved that STRONG F -CLOSURE is FPT and admits a polynomial kernel when parameterized by the number of weak edges and F is a fixed graph (Theorems 6.5.1, 6.5.2). We concluded with some results related to classical computational complexity. We showed that MAXSTC remains NP-hard on $(3K_1, 2K_2)$ -free graphs (Theorem 6.5.5) and on planar graphs (Theorem 6.5.3). Our reduction for planar graphs also works for the

CLUSTER DELETION problem and, thus, CLUSTER DELETION remains NP-hard on planar graphs (Corollary 6.5.4).

7.2 Open Problems

Here, we discuss possible future directions for further research and we highlight few open problems that arise from the results obtained within this thesis.

Given the first study with positive and negative results for the MAXSTC problem on restricted input, there are some interesting open problems. As we pointed out, MAXSTC is more difficult than CLUSTER DELETION in the following sense: a solution for CLUSTER DELETION forms a solution for MAXSTC but the converse is not necessarily true. We have given examples showing that such an observation carries out for split graphs as well as for proper interval graphs. Despite the structural difference of both problems, our result on split graphs points out an important and interesting complexity difference between the two problems: on split graphs CLUSTER DELETION has already been shown to be polynomially solvable [11], whereas we proved that MAXSTC remains NP-complete. It is interesting to explore other graph classes that exhibit the same behavior. A natural graph class towards such a direction, can be considered as the class of interval graphs. The structural properties that we proved for the solution of MAXSTC on proper interval graphs, as well as, the extensive analysis that we did for computing an optimal solution for CLUSTER DELETION on interval graphs seem to be helpful in order to attack MAXSTC on interval graphs.

Moreover, our algorithm for CLUSTER DELETION on interval graphs, which heavily relies on the linear structure obtained from their clique paths, leads us to consider few open questions regarding two main directions. On the one hand, it seems tempting to adjust our algorithm for other vertex partitioning problems on interval graphs within a more general framework, as already have been studied for particular graph properties [13, 53, 70, 79, 120]. On the other hand, it is reasonable to ask whether our approach works for CLUSTER DELETION on graphs admitting similar linear structure such as permutation graphs [43, 58, 104]. In addition, permutation graphs consist a superclasses of cographs, where the optimal solutions for MAXSTC and CLUSTER DELETION coincide. On permutation graphs both problems do not coincide (the lowest example of Figure 4.1 is a permutation graph) and both problems restricted to such graphs have unresolved complexity status.

Furthermore, line-incompatibility graph is a useful tool to study MAXSTC because of the equivalency between INDEPENDENT SET and MAXSTC by Proposition 3.2.2. Observing that INDEPENDENT SET is polynomial solvable on perfect graphs [61], a potential characterization of whether the line-incompatibility of a graph is perfect, will give

us immediately a polynomial time solution for MAXSTC. Besides, line-incompatibility graph has already been considered under the term of Gallai graph. Gallai graph has attracted many researchers [78, 96, 119], where they tried either to characterize the Gallai graph of a graph or to provide interesting structural properties.

Given the fact that the solutions of both problems coincide on P_4 -free and K_3 -free but not on $(2K_2, C_4, C_5)$ -free graphs and claw-free interval graphs, an interesting topic is to completely characterize graphs by forbidden subgraphs for which MAXSTC and CLUSTER DELETION coincide. In this direction, regarding H -free graphs, Grüttemeier et al. [62], showed a complexity dichotomy result for any graph H consisting of at most four vertices. In particular, for any graph H on four vertices with $H \notin \{P_4, \text{paw}\}$, CLUSTER DELETION is NP-hard on H -free graphs, whereas it can be solved in polynomial time on P_4 - or paw-free graphs. It is interesting to exhibit similar characterization for a family \mathcal{H} of graphs that handles \mathcal{H} -free graphs.

Another area of interest is graphs with small structural parameters, since CLUSTER DELETION and MAXSTC can be solved in linear time on graphs of bounded treewidth by using Courcelle's machinery [28]. Although, for other structural parameters it seems rather difficult to obtain a similar result, it is still interesting to settle the complexity for both problems on distance hereditary graphs as they admit constant clique-width [59]. In fact, we would like to settle the case in which from a given cograph (P_4 -free graph) we can append degree-one vertices. This comes in conjunction with the starlike graphs, as they can be seen as a degree-one extension of a clique.

Moreover, other structural parameters that seem promising on parameterized complexity are *neighborhood diversity* and *leafage* [49, 84, 47, 65, 100]. *Neighborhood diversity* has been introduced by Lampis in [93], as a new graph parameter which generalizes vertex cover to dense graphs. A graph $G = (V, E)$ has neighborhood diversity at most w , if there exists a partition of V into at most w sets, such that all the vertices in each set have the same *type*: two vertices v, v' of G have the same type iff $N(v) \setminus \{v'\} = N(v') \setminus \{v\}$. It is a prominent parameter, since it can be seen as a generalization of false twins and we showed that twin vertices play important role in MAXSTC as well as in CLUSTER DELETION.

According to Gavril [52], a graph G is chordal if and only if G can be represented as the intersection graph of a collection of subtrees of a host tree, the so-called tree model of G . The *leafage* $\ell(G)$ of a connected chordal graph G is defined as the minimum number of leaves of the host tree of a tree model of G . This concept was first defined by Lin et al. in [100]. It is known that leafage can be computed in polynomial time on chordal graphs [65] and when the graph G is interval it holds $\ell(G) = 2$. Considering that CLUSTER DELETION is polynomial solvable on interval graphs and both CLUSTER DELETION and MAXSTC are NP-hard on chordal graphs, it is interesting if this parameter can lead to an FPT algorithm on chordal graphs. For example, an

FPT algorithm for DOMINATION SET on connected chordal graphs parameterized by leafage was given in [47].

As we already mentioned, the above structural parameters may be proved to be helpful in order to show that MAXSTC and CLUSTER DELETION admit FPT algorithms with respect to such parameters. Additionally, the parameterized complexity of STRONG F -CLOSURE can be further explored. Since maximum matching of a graph satisfy F -Closure when F has a component with three vertices and MAXSTC admits an FPT algorithm on graphs of maximum degree at most 4 with the above guarantee parameterization $k - \mu(G)$, we believe that this parameterization is interesting not only on general graph but also on various other graph classes. In particular, is MAXSTC fixed-parameter tractable when parameterized by $k - \mu(G)$? What can be said about planar graphs, since MAXSTC is already NP-hard on planar graphs?

A more general and realistic scenario for CLUSTER DELETION and MAXSTC is to restrict the choice of the considered edges. Assume that a subset F of edges is required to be included in the same clusters for CLUSTER DELETION or, analogously, the described edges are required to be strong for MAXSTC. Then, it is natural to ask for a suitable set of edges $E' \subseteq E \setminus F$ with $|E'|$ as large as possible such that the edges of $E' \cup F$ span a cluster graph or satisfy the strong triadic closure, respectively. Clarifying the complexity of such generalized problems is interesting on graphs for which CLUSTER DELETION or MAXSTC are solved in polynomial time. Another generalization of MAXSTC, that is introduced by Sintos and Tsaparas [118], is having multiple type of strong edges, since in real world scenarios every person has different type of relation with other people. This problem is referred to as MULTI-STC. In MULTI-STC an induced P_3 may receive two strong labels as long as they are different. Towards this direction, there are some results by Bulteau et al. [14] who studied the classical and parameterized complexity of MULTI-STC and its variations such as VL-MULTI-STC and EL-MULTI-STC that concern vertex or edge restrictions. Also Grüttemeier et al. [64] studied the parameterized complexity of MULTI-STC and EL-MULTI-STC. It is interesting to study graph classes for which VL-MULTI-STC or EL-MULTI-STC admit polynomial solutions.

BIBLIOGRAPHY

- [1] ABU-KHZAM, F. N. A kernelization algorithm for d-hitting set. *J. Comput. Syst. Sci.* 76, 7 (2010), 524–531.
- [2] ADCOCK, A. B., SULLIVAN, B. D., AND MAHONEY, M. W. Tree decompositions and social graphs. *Internet Mathematics* 12 (2016), 315–361.
- [3] ADRIAENS, F., BIE, T. D., GIONIS, A., LIJFFIJT, J., MATAKOS, A., AND ROZENSHTEIN, P. Relaxing the strong triadic closure problem for edge strength inference. *Data Min. Knowl. Discov.* 34, 3 (2020), 611–651.
- [4] ALON, N., GUTIN, G., KIM, E. J., SZEIDER, S., AND YEO, A. Solving max- r -sat above a tight lower bound. *Algorithmica* 61, 3 (2011), 638–655.
- [5] BACKSTROM, L., AND KLEINBERG, J. Romantic partnerships and the dispersion of social ties: a network analysis of relationship status on facebook. In *Proceedings of CSCW 2014* (2014), pp. 831–841.
- [6] BANSAL, N., BLUM, A., AND CHAWLA, S. Correlation clustering. *Machine Learning* 56 (2004), 89–113.
- [7] BLIZNETS, I., CYGAN, M., KOMOSA, P., AND PILIPCZUK, M. Hardness of approximation for H -free edge modification problems. *TOCT* 10, 2 (2018), 9:1–9:32.
- [8] BODLAENDER, H. L., DOWNEY, R. G., FELLOWS, M. R., AND HERMELIN, D. On problems without polynomial kernels. *J. Comput. Syst. Sci.* 75, 8 (2009), 423–434.
- [9] BODLAENDER, H. L., JANSEN, B. M. P., AND KRATSCH, S. Kernelization lower bounds by cross-composition. *SIAM J. Discrete Math.* 28, 1 (2014), 277–305.

- [10] BONOMO, F., DURÁN, G., NAPOLI, A., AND VALENCIA-PABON, M. A one-to-one correspondence between potential solutions of the cluster deletion problem and the minimum sum coloring problem, and its application to P_4 -sparse graphs. *Inf. Proc. Lett.* 115 (2015), 600–603.
- [11] BONOMO, F., DURÁN, G., AND VALENCIA-PABON, M. Complexity of the cluster deletion problem on subclasses of chordal graphs. *Theor. Comp. Science* 600 (2015), 59–69.
- [12] BRANDSTÄDT, A., LE, V. B., AND SPINRAD, J. *Graph Classes: A Survey*. Society for Industrial and Applied Mathematics, 1999.
- [13] BUI-XUAN, B., TELLE, J. A., AND VATSHELLE, M. Fast dynamic programming for locally checkable vertex subset and vertex partitioning problems. *Theor. Comput. Sci.* 511 (2013), 66–76.
- [14] BULTEAU, L., GRÜTTEMEIER, N., KOMUSIEWICZ, C., AND SORGE, M. Your rugby mates don't need to know your colleagues: Triadic closure with edge colors. In *Algorithms and Complexity* (2019), Springer International Publishing, pp. 99–111.
- [15] BURZYN, P., BONOMO, F., AND DURÁN, G. Np-completeness results for edge modification problems. *Discret. Appl. Math.* 154, 13 (2006), 1824–1844.
- [16] CAI, L. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters* 58 (1996), 171–176.
- [17] CAI, L., AND CAI, Y. Incompressibility of H -free edge modification problems. *Algorithmica* 71 (2015), 731–757.
- [18] CAI, L., CHAN, S., AND CHAN, S. Random separation: a new method for solving fixed-cardinality optimization problems. In *IWPEC 2006* (2006), pp. 239–250.
- [19] CERIOLI, M. R., AND SZWARCFITER, J. L. Characterizing intersection graphs of substars of a star. *Ars Comb.* 79 (2006).
- [20] CHARIKAR, M., GURUSWAMI, V., AND WIRTH, A. Clustering with qualitative information. In *Proceedings of FOCS 2003* (2003), pp. 524–533.
- [21] CHEN, J., KANJ, I. A., AND XIA, G. Improved parameterized upper bounds for vertex cover. In *Mathematical Foundations of Computer Science 2006, 31st International Symposium, MFCS 2006, Stará Lesná, Slovakia, August 28–September 1, 2006, Proceedings* (2006), R. Kralovic and P. Urzyczyn, Eds., vol. 4162 of *Lecture Notes in Computer Science*, Springer, pp. 238–249.

- [22] CHITNIS, R., CYGAN, M., HAJIAGHAYI, M., PILIPCZUK, M., AND PILIPCZUK, M. Designing FPT algorithms for cut problems using randomized contractions. *SIAM J. Comput.* 45, 4 (2016), 1171–1229.
- [23] CHUDNOVSKY, M., ROBERTSON, N., SEYMOUR, P., AND THOMAS, R. The strong perfect graph theorem. *Annals of Mathematics* 164 (2006), 51–229.
- [24] COCHEFERT, M., COUTURIER, J.-F., GOLOVACH, P. A., KRATTSCH, D., AND PAULUSMA, D. Parameterized algorithms for finding square roots. *Algorithmica* 74 (2016), 602–629.
- [25] CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., AND STEIN, C. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009.
- [26] CORNEIL, D., LERCHS, H., AND STEWART, L. Complement reducible graphs. *Discrete Applied Mathematics* 3 (1981), 163–174.
- [27] CORNEIL, D., PERL, Y., AND STEWART, L. A linear recognition algorithm for cographs. *SIAM Journal on Computing* 14 (1985), 926–934.
- [28] COURCELLE, B. The monadic second-order logic of graphs i: Recognizable sets of finite graphs. *Information and Computation* 85 (1990), 12–75.
- [29] CRESPELLE, C., DRANGE, P. G., FOMIN, F. V., AND GOLOVACH, P. A. A survey of parameterized algorithms and the complexity of edge modification. *CoRR abs/2001.06867* (2020).
- [30] CYGAN, M., FOMIN, F. V., KOWALIK, L., LOKSHTANOV, D., MARX, D., PILIPCZUK, M., PILIPCZUK, M., AND SAURABH, S. *Parameterized Algorithms*. Springer, 2015.
- [31] DE RIDDER, H. N. Information system on graph classes and their inclusions. graphclasses.org, accessed December 2020.
- [32] DENG, X., HELL, P., AND HUANG, J. Linear-time representation algorithms for proper circular-arc graphs and proper interval graphs. *SIAM J. Comput.* 25 (1996), 390–403.
- [33] DESSMARK, A., JANSSON, J., LINGAS, A., LUNDELL, E., AND PERSSON, M. On the approximability of maximum and minimum edge clique partition problems. *Int. J. Found. Comput. Sci.* 18 (2007), 217–226.
- [34] DIESTEL, R. *Graph Theory, 4th Edition*, vol. 173 of *Graduate Texts in Mathematics*. Springer, 2012.

- [35] DOM, M., LOKSHTANOV, D., AND SAURABH, S. Kernelization lower bounds through colors and ids. *ACM Trans. Algorithms* 11, 2 (2014), 13:1–13:20.
- [36] DOWNEY, R. G., AND FELLOWS, M. R. Fixed parameter tractability and completeness. In *Complexity Theory: Current Research, Dagstuhl Workshop, February 2-8, 1992* (1992), K. Ambos-Spies, S. Homer, and U. Schöning, Eds., Cambridge University Press, pp. 191–225.
- [37] DOWNEY, R. G., AND FELLOWS, M. R. Fixed-parameter tractability and completeness II: on completeness for $W[1]$. *Theor. Comput. Sci.* 141, 1&2 (1995), 109–131.
- [38] DOWNEY, R. G., AND FELLOWS, M. R. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999.
- [39] DOWNEY, R. G., AND FELLOWS, M. R. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.
- [40] DYER, M. E., AND FRIEZE, A. M. Planar 3DM is NP-complete. *Journal of Algorithms* 7 (1986), 174–184.
- [41] EASLEY, D., AND KLEINBERG, J. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, 2010.
- [42] EDMONDS, J. Paths, trees and flowers. *Canad. J. Math.* 17 (1965), 449–467.
- [43] EVEN, S., PNUELI, A., AND LEMPEL, A. Permutation graphs and transitive graphs. *J. Assoc. Comput. Mach.* 19 (1972), 400–410.
- [44] FLUM, J., AND GROHE, M. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.
- [45] FÖLDES, S., AND HAMMER, P. L. Split graphs. *Congressus Numerantium* 19 (1977), 311–315.
- [46] FOMIN, F., AND KRATSCH, D. *Exact Exponential Algorithms*. Springer, 2010.
- [47] FOMIN, F. V., GOLOVACH, P. A., AND RAYMOND, J. On the tractability of optimization problems on h-graphs. *Algorithmica* 82, 9 (2020), 2432–2473.
- [48] FULKERSON, D. R., AND GROSS, O. A. Incidence matrices and interval graphs. *Pacific Journal of Mathematics* 15 (1965), 835–855.
- [49] GANIAN, R. Using neighborhood diversity to solve hard problems. *CoRR abs/1201.3091* (2012).

- [50] GAO, Y., HARE, D. R., AND NASTOS, J. The cluster deletion problem for cographs. *Discrete Mathematics* 313 (2013), 2763–2771.
- [51] GAREY, M. R., AND JOHNSON, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., USA, 1979.
- [52] GAVRIL, F. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B* 16, 1 (1974), 47 – 56.
- [53] GERBER, M. U., AND KOBLER, D. Algorithms for vertex-partitioning problems on graphs with fixed clique-width. *Theor. Comp. Science* 299 (2003), 719 – 734.
- [54] GOLDBERG, P. W., GOLUMBIC, M. C., KAPLAN, H., AND SHAMIR, R. Four strikes against physical mapping of dna. *Journal of Computational Biology* 2, 1 (1995), 139–152.
- [55] GOLOVACH, P. A., HEGGERNES, P., KONSTANTINIDIS, A. L., LIMA, P. T., AND PAPADOPOULOS, C. Parameterized aspects of strong subgraph closure. In *Proceedings of SWAT 2018* (2018), pp. 23:1–23:13.
- [56] GOLOVACH, P. A., HEGGERNES, P., KONSTANTINIDIS, A. L., LIMA, P. T., AND PAPADOPOULOS, C. Parameterized aspects of strong subgraph closure. *Algorithmica* 82, 7 (2020), 2006–2038.
- [57] GOLUMBIC, M. Trivially perfect graphs. *Discrete Mathematics* 24 (1978), 105–107.
- [58] GOLUMBIC, M. C. *Algorithmic Graph Theory and Perfect Graphs*. Annals of Discrete Mathematics, Elsevier, 2004.
- [59] GOLUMBIC, M. C., AND ROTICS, U. On the clique-width of some perfect graph classes. *Int. J. Found. Comput. Sci.* 11 (2000), 423–443.
- [60] GRANOVETTER, M. The strength of weak ties. *American Journal of Sociology* 78 (1973), 1360–1380.
- [61] GRÖTSCHEL, M., LOVÁSZ, L., AND SCHRIJVER, A. Polynomial algorithms for perfect graphs. In *Topics on Perfect Graphs*, C. Berge and V. Chvátal, Eds., vol. 88 of *North-Holland Mathematics Studies*. North-Holland, 1984, pp. 325 – 356.
- [62] GRÜTTEMEIER, N., AND KOMUSIEWICZ, C. On the relation of strong triadic closure and cluster deletion. In *WG 2018* (2018), vol. 11159 of *Lecture Notes in Computer Science*, Springer, pp. 239–251.

- [63] GRÜTTEMEIER, N., AND KOMUSIEWICZ, C. On the relation of strong triadic closure and cluster deletion. *Algorithmica* 82 (2020), 853–880.
- [64] GRÜTTEMEIER, N., KOMUSIEWICZ, C., AND MORAWIETZ, N. Maximum Edge-Colorable Subgraph and Strong Triadic Closure Parameterized by Distance to Low-Degree Graphs. In *17th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2020)* (2020), vol. 162 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 26:1–26:17.
- [65] HABIB, M., AND STACHO, J. Polynomial-time algorithm for the leafage of chordal graphs. In *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings* (2009), A. Fiat and P. Sanders, Eds., vol. 5757 of *Lecture Notes in Computer Science*, Springer, pp. 290–300.
- [66] HAGEN, L., AND KAHNG, A. B. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 11, 9 (1992), 1074–1085.
- [67] HAMMER, P. L., AND SIMEONE, B. The splittance of a graph. *Combinatorica* 1 (1981), 275–284.
- [68] HANSEN, P., AND JAUMARD, B. Cluster analysis and mathematical programming. *Math. Programming* 79 (1997), 191–215.
- [69] HARTIGAN, J. *Clustering Algorithms*. Wiley, New York, 1975.
- [70] HEGGERNES, P., LOKSHTANOV, D., NEDERLOF, J., PAUL, C., AND TELLE, J. A. Generalized graph clustering: recognizing (p, q) -cluster graphs. In *Proceedings of WG 2010* (2010), pp. 171–183.
- [71] HON, W., KLOKS, T., LIU, H., POON, S., AND WANG, Y. On independence domination. In *Proceedings of FCT 2013* (2013), pp. 183–194.
- [72] HOPCROFT, J. E., AND KARP, R. M. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.* 2 (1973), 225–231.
- [73] IBARRA, L. The clique-separator graph for chordal graphs. *Discrete Applied Mathematics* 157 (2009), 1737–1749.
- [74] IMPAGLIAZZO, R., PATURI, R., AND ZANE, F. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences* 63 (2001), 512–530.

- [75] IMRICH, W., KLAVZAR, S., AND RALL, D. *Topics in Graph Theory: Graphs and Their Cartesian Product*. AK Peters Ltd, 2008.
- [76] JACKSON, M. O. *Social and economic networks*. Princeton University press, vol. 3, 2008.
- [77] JHA, P., AND SLUTZKI, G. Independence numbers of product graphs. *Applied Mathematics Letters* 7 (1994), 91–94.
- [78] JOOS, F., LE, V. B., AND RAUTENBACH, D. Forests and trees among gallai graphs. *Discret. Math.* 338, 2 (2015), 190–195.
- [79] KANJ, I. A., KOMUSIEWICZ, C., SORGE, M., AND VAN LEEUWEN, E. J. Solving partition problems almost always requires pushing many vertices around. In *Proceedings of ESA 2018* (2018), pp. 51:1–51:14.
- [80] KARP, R. M. Reducibility among combinatorial problems. *Complexity of Computer Computations* (1972), 85–103.
- [81] KHOT, S., AND RAMAN, V. Parameterized complexity of finding subgraphs with hereditary properties. *Theor. Comput. Sci.* 289, 2 (2002), 997–1008.
- [82] KLEINBERG, J. M., AND TARDOS, É. *Algorithm design*. Addison-Wesley, 2006.
- [83] KLEITMAN, D. J., AND VOHRA, R. V. Computing the bandwidth of interval graphs. *SIAM J. Disc. Math.* 3 (1990), 373–375.
- [84] KNOP, D. Partitioning graphs into induced subgraphs. *Discret. Appl. Math.* 272 (2020), 31–42.
- [85] KOMUSIEWICZ, C., AND UHLMANN, J. Cluster editing with locally bounded modifications. *Discrete Applied Mathematics* 160 (2012), 2259–2270.
- [86] KONSTANTINIDIS, A. L., NIKOLOPOULOS, S. D., AND PAPADOPOULOS, C. Strong triadic closure in cographs and graphs of low maximum degree. In *COCOON 2017* (2017), pp. 346–358.
- [87] KONSTANTINIDIS, A. L., NIKOLOPOULOS, S. D., AND PAPADOPOULOS, C. Strong triadic closure in cographs and graphs of low maximum degree. *Theor. Comput. Sci.* 740 (2018), 76–84.
- [88] KONSTANTINIDIS, A. L., AND PAPADOPOULOS, C. Maximizing the strong triadic closure in split graphs and proper interval graphs. In *ISAAC 2017* (2017), pp. 53:1–53:12.

- [89] KONSTANTINIDIS, A. L., AND PAPADOPOULOS, C. Cluster deletion on interval graphs and split related graphs. In *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany* (2019), P. Rossmanith, P. Heggernes, and J. Katoen, Eds., vol. 138 of *LIPICs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 12:1–12:14.
- [90] KONSTANTINIDIS, A. L., AND PAPADOPOULOS, C. Maximizing the strong triadic closure in split graphs and proper interval graphs. *Discret. Appl. Math.* 285 (2020), 79–95.
- [91] KONSTANTINIDIS, A. L., AND PAPADOPOULOS, C. Cluster deletion on interval graphs and split related graphs. *Algorithmica* (2021).
- [92] KRATSCH, S., AND WAHLSTROM, M. Two edge modification problems without polynomial kernels. *Discrete Optimization* 10 (2013), 193–199.
- [93] LAMPIS, M. Algorithmic meta-theorems for restrictions of treewidth. *Algorithmica* 64, 1 (2012), 19–37.
- [94] LAU, L. C. Bipartite roots of graphs. *ACM Transactions on Algorithms* 2 (2006), 178–208.
- [95] LAU, L. C., AND CORNEIL, D. G. Recognizing powers of proper interval, split, and chordal graphs. *SIAM J. Disc. Math.* 18 (2004), 83–102.
- [96] LE, V. B. Gallai graphs and anti-gallai graphs. *Discrete Mathematics* 159 (1996), 179–189.
- [97] LE, V. B., OVERSBERG, A., AND SCHAUDT, O. Polynomial time recognition of squares of ptolemaic graphs and 3-sun-free split graphs. *Theor. Comp. Science* 602 (2015), 39–49.
- [98] LEKKERKERKER, C. G., AND BOLAND, J. C. Representation of a finite graph by a set of intervals on the real line. *Fundam. Math.* 51 (1962), 45–64.
- [99] LEWIS, J. M., AND YANNAKAKIS, M. The node-deletion problem for hereditary properties is np-complete. *Journal of Computer and System Sciences* 20, 2 (1980), 219 – 230.
- [100] LIN, I., MCKEE, T. A., AND WEST, D. B. The leafage of a chordal graph. *Discuss. Math. Graph Theory* 18, 1 (1998), 23–48.
- [101] LOKSHTANOV, D., MARX, D., AND SAURABH, S. Lower bounds based on the exponential time hypothesis. *Bulletin of the European Association for Theoretical Computer Science* 105 (2011), 41–72.

- [102] LOOGES, P. J., AND OLARIU, S. Optimal greedy algorithms for indifference graphs. *Computers & Mathematics with Applications* 25 (1993), 15–25.
- [103] MAHADEV, N. V. R., AND PELED, U. N. *Threshold Graphs and Related Topics*, vol. 56. North Holland, 1995.
- [104] MCCONNELL, R. M., AND SPINRAD, J. P. Modular decomposition and transitive orientation. *Discrete Mathematics* 201, 1 (1999), 189 – 241.
- [105] MICALI, S., AND VAZIRANI, V. V. An $O(\sqrt{|V||E|})$ algorithm for finding maximum matching in general graphs. In *FOCS 1980* (1980), pp. 17–27.
- [106] MILANIĆ, M., AND SCHAUDT, O. Computing square roots of trivially perfect and threshold graphs. *Discrete Applied Mathematics* 161 (2013), 1538–1545.
- [107] NATANZON, A., SHAMIR, R., AND SHARAN, R. Complexity classification of some edge modification problems. *Discret. Appl. Math.* 113, 1 (2001), 109–128.
- [108] PFALTZ, J. L. Chordless cycles in networks. In *Proceedings of ICDE Workshops 2013* (2013), pp. 223–228.
- [109] PROTTI, F., DA SILVA, M. D., AND SZWARCFITER, J. L. Applying modular decomposition to parameterized cluster editing problems. *Theory Comput. Syst.* 44 (2009), 91–104.
- [110] ROBERTS, F. S. Indifference graphs. In *Proof Techniques in Graph Theory*, Academic Press, New York (1969), pp. 139–146.
- [111] ROSE, D. J. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. In *Graph Theory and Computing*, R. C. READ, Ed. Academic Press, 1972, pp. 183 – 217.
- [112] ROTABI, R., KAMATH, K., KLEINBERG, J., AND SHARMA, A. Detecting strong ties using network motifs. In *Proceedings of WWW 2017* (2017), pp. 983–992.
- [113] ROZENSHTAIN, P., TATTI, N., AND GIONIS, A. Inferring the strength of social ties: a community-driven approach. In *Proceedings of KDD 2017* (2017), pp. 1017–1025.
- [114] SCHAEFFER, S. E. Graph clustering. *Computer Science Review* 1, 1 (2007), 27–64.
- [115] SHAMIR, R., AND SHARAN, R. A fully dynamic algorithm for modular decomposition and recognition of cographs. *Discrete Applied Mathematics* 136 (2004), 329–340.

- [116] SHAMIR, R., SHARAN, R., AND TSUR, D. Cluster graph modification problems. *Discrete Applied Mathematics* 144 (2004), 173–182.
- [117] SHARAN, R., AND SHAMIR, R. Click: a clustering algorithm with applications to gene expression analysis. *Proceedings. International Conference on Intelligent Systems for Molecular Biology* 8 (2000), 307–16.
- [118] SINTOS, S., AND TSAPARAS, P. Using strong triadic closure to characterize ties in social networks. In *Proceedings of KDD 2014* (2014), pp. 1466–1475.
- [119] SUN, L. Two classes of perfect graphs. *Journal of Combinatorial Theory, Series B* 53, 2 (1991), 273 – 292.
- [120] TELLE, J. A., AND PROSKUROWSKI, A. Algorithms for vertex partitioning problems on partial k -trees. *SIAM J. Discrete Math.* 10 (1997), 529–550.
- [121] UGANDER, J., BACKSTROM, L., AND KLEINBERG, J. Subgraph frequencies: Mapping the empirical and extremal geography of large graph collections. In *Proceedings of WWW 2013* (2013), pp. 1307–1318.
- [122] VAN BEVERN, R., TSIDULKO, O. Y., AND ZSCHOCHE, P. Fixed-parameter algorithms for maximum-profit facility location under matroid constraints. In *Algorithms and Complexity - 11th International Conference, CIAC 2019, Rome, Italy, May 27-29, 2019, Proceedings* (2019), P. Heggernes, Ed., vol. 11485 of *Lecture Notes in Computer Science*, Springer, pp. 62–74.
- [123] WOEGINGER, G. *Exact algorithms for NP-hard problems: a survey*, vol. 2570. *Lecture Notes in Computer Science*, Springer, 2003.
- [124] WU, Z., AND LEAHY, R. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15, 11 (1993), 1101–1113.
- [125] YANNAKAKIS, M. Edge-deletion problems. *SIAM Journal on Computing* 10, 2 (1981), 297–309.

SHORT CV

Athanasios L. Konstantinidis is a PhD candidate at the Department of Mathematics of the University of Ioannina and supervised by Associate Professor Charis Papadopoulos. His PhD studies are supported by a grant from the National Grant Foundation of Greece (IKY). He received his B.Sc and M.Sc degrees from the same department at 2014 and 2016 respectively. Throughout his studies he was labs assistant in the following courses: Introduction to Computers, Introduction to Programming and Data Structure. His research focuses on theoretical computer science and in particular on design and analysis of algorithms and algorithmic graph theory. He has published research papers in journals and refereed conferences. He has given research talks at ACAC 2017 (Greece), Algorithms Seminars 2017 (Dept. of Informatics, University of Bergen, Norway), ISAAC 2017 (Thailand), 2nd Conference of Young Researchers in the Branches of Mathematical Science 2018 (Dept. of Mathematics, University of Ioannina, Greece), SWAT 2018 (Sweden), MFCS 2019 (Germany).

LIST OF PUBLICATIONS

The results of this thesis have already been published [55, 56, 86, 87, 88, 89, 90, 91]. For completeness, here we list the related publications in chronological order:

- **Maximizing the strong triadic closure in split graphs and proper interval graphs.** Athanasios L. Konstantinidis, and Charis Papadopoulos. *In 28th International Symposium on Algorithms and Computation (ISAAC 2017), Leibniz International Proceedings in Informatics (LIPIcs), pages 53:1–53:12, 2017.*
- **Strong triadic closure in cographs and graphs of low maximum degree.** Athanasios L. Konstantinidis, Stavros D. Nikolopoulos, and Charis Papadopoulos. *23rd Annual International Computing and Combinatorics Conference, (COCOON 2017), Hong Kong, China, 2017. Springer Verlag, LNCS 10392: 346–358.*
- **Strong triadic closure in cographs and graphs of low maximum degree.** Athanasios L. Konstantinidis, Stavros D. Nikolopoulos, and Charis Papadopoulos. *Theoretical Computer Science 740: 76–84, 2018.*
- **Parameterized aspects of strong subgraph closure.** Petr A. Golovach, Pinar Heggernes, Athanasios L. Konstantinidis, Paloma T. Lima and Charis Papadopoulos. *16th Scandinavian Symposium and Workshops on Algorithm Theory, (SWAT 2018), Malmo, Sweden, 2018. Leibniz-Zentrum für Informatik, LIPIcs 101: 23(1)–23(13), 2018.*
- **Cluster deletion on interval graphs and split related graphs.** Athanasios L. Konstantinidis and Charis Papadopoulos. *44th International Symposium on Mathematical Foundations of Computer Science, (MFCS 2019), Aachen, Germany, 2019. Leibniz-Zentrum für Informatik, LIPIcs 138: 12(1)–12(14), 2019.*
- **Parameterized aspects of strong subgraph closure.** Petr A. Golovach, Pinar Heggernes, Athanasios L. Konstantinidis, Paloma T. Lima, and Charis Papadopoulos. *Algorithmica 82: 2006–2038, 2020.*

- **Maximizing the strong triadic closure in split graphs and proper interval graphs.** Athanasios L. Konstantinidis and Charis Papadopoulos. *Discrete Applied Mathematics* 285: 79-95, 2020.
- **Cluster deletion on interval graphs and split related graphs.** Athanasios L. Konstantinidis and Charis Papadopoulos. *Algorithmica*, 2021.