ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΤΜΗΜΑ ΜΑΘΗΜΑΤΙΚΩΝ

Σπυρίδων Τζίμας

Προβληματα Κρουσης Μονοπατιων και Κυκλων: Αλγοριθμοι και Πολυπλοκοτητα

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

ΙΩΑΝΝΙΝΑ, 2024

UNIVERSITY OF IOANNINA

DEPARTMENT OF MATHEMATICS

Spyridon Tzimas

Path and Cycle Hitting Problems: Algorithms and Complexity

DOCTORATE THESIS

IOANNINA, 2024

To all the paths
not taken
due to all the cycles
not broken.

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

Εγκρίθηκε την 31/1/2024 από την εξεταστική επιτροπή:

Ονοματεπώνυμο	**Βαθμίδα**
Χάρης Παπαδόπουλος	Αναπληρωτής Καθηγητής
Λουκάς Γεωργιάδης	Αναπληρωτής Καθηγητής
Σταύρος Δ. Νικολόπουλος	Καθηγητής
Δημήτριος Μ. Θηλυκός	Καθηγητής
Μιχαήλ Α. Μπέκος	Επίκουρος Καθηγητής
Χρήστος Νομικός	Αναπληρωτής Καθηγητής
Λεονίδας Παληός	Καθηγητής

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.

Σπυρίδων Τζίμας

Acknowledgements

This thesis is in partial fullfilment of the requirements of the doctoral studies of the author at the Department of Mathematics of the University of Ioannina, Greece, started in 2016. For part of the duration of said studies, the author was financially supported by the General Secretariat for Research and Technology (G.S.R.T.) and the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "1st Call for H.F.R.I. Scholarships to Ph.D. Candidates" (Scholarship Code: 1486).

The author thanks his supervisor Prof. Charis Papadopoulos for his guidance and support throughout what has ended up being too many a year of research and (re)writing for a Ph.D. in Theoretical Computer Science. All this research and (re)writing culminated in this very thesis.

Περιληψη

Τα περισσότερα γραφοθεωρητικά προβλήματα είναι γνωστό πως είναι NP-πλήρη στα γενικά γραφήματα· δεν θεωρείται πίθανο να υπάρχει αλγόριθμος πολυωνυμικού χρόνου για την επίλυσή τους. Με κίνητρο το γεγονός ότι πολλά από αυτά τα προβλήματα έχουν εφαρμογές στον πραγματικό κόσμο, οι ερευνητές έχουν επικεντρωθεί στην σχεδίαση όλο και γρηγορότερων ακριβών αλγορίθμων εκθετικού χρόνου και όλο και καλύτερων προσεγγιστικών αλγορίθμων για την επίλυσής τους. Μία άλλη κατεύθυνση της έρευνας — αυτή που ακολουθούμε στην παρούσα διατριβή —, η οποία έλαβε μεγάλη ώθηση τα τελευταία χρόνια λόγω της ανακάλυψης νέων ισχυρών εργαλείων στο πλαίσιο της Παραμετροποιημένης Πολυπλοκότητας, είναι η μελέτη αυτών των προβλημάτων σε συγκεκριμένα γραφήματα που εμφανίζουν διαφόρων ειδών εσωτερική δομή. Σχεδιάζοντας αλγορίθμους που εκμεταλλεύονται αυτή την εσωτερική δομή, ενδέχεται να παράσχουμε καλύτερες εγγύησεις σε χρόνο εκτέλεσης ή/και ποιότητα λύσης σε αυτά τα συγκεκριμένα γραφήματα. Ενδέχεται επίσης να δείξουμε ότι η δυσκολία ενός προβλήματος διατηρείται σε αυτά τα γραφήματα, παράγοντας με αυτόν τον τρόπο περαιτέρω γνωση πάνω στα είδη δομής που κάνουν το πρόβλημα δύσκολο στην επίλυση.

Στην παρούσα διατριβή, μελετούμε Προβληματα Συνολου Τερματικων Κορυφων: δοθέντων ενός γραφήματος (με βάρη στις κορυφές) και ενός υποσυνόλου κορυφών των γραφήματος, που καλείται το *σύνολο τερματικών κορυφών*, αναζητούμε υποσύνολο κορυφών του γραφήματος ελαχίστου μεγέθους (βάρους) το οποίο τέμνει (*κρούει*) κάθε υποσύνολο κορυφών των γραφήματος που περιέχει τερματική κορυφή και επάγει υπογράφημα που εμφανίζει συγκεκριμένη δομή που εξαρτάται από το πρόβλημα. Αυτή η κλάση προβλημάτων περιέχει εξέχοντα γραφοθεωρητικά προβλήματα τα οποία έχουν εφαρμογές τόσο εντός όσο και πέραν της Επιστήμης Υπολογιστών. Εστιάζουμε την μελέτη μας σε δύο συγκεκριμένα προβλήματα.

Το ένα είναι πρόβλημα κρούσης κύκλων και είναι γνωστό στην σχετική βιβλιογραφία ως το Προβλημα Στοχευμενα Ακυκλου Επαγομενου Υπογραφηματος (SFVS): δοθέντων ενός γραφήματος (με βάρη στις κορυφές) και ενός συνόλου τερματικών κορυφών, αναζητούμε υποσύνολο κορυφών του γραφήματος ελαχίστου μεγέθους (βάρους) το οποίο κρούει κάθε υποσύνολο κορυφών του γραφήματος που περιέχει τερματική κορυφή και επάγει κύκλο. Μελετούμε το SFVS σε υποκλάσεις των AT-ελεύθερων γραφημάτων και σε υποκλάσεις των χορδικών γραφημάτων, τα οποία είναι γραφήματα που είναι γνωστό ότι εμφανίζουν πλούσια εσωτερική δομή. Παρέχουμε αλγορίθμους πολυωνυμικού χρόνου για την επίλυση του έμβαρου SFVS στις ακόλουθες κλάσεις γραφημάτων: γραφήματα διαστημάτων (χρόνου $\mathcal{O}(nm)$), γραφήματα μεταθέσεων (χρόνου $\mathcal{O}(m^3)$), συνδιμερή γραφήματα (χρόνου $\mathcal{O}(n^4)$) και γραφήματα μονοπατιών ριζωμένων δένδρων (χρόνου $\mathcal{O}(n^2m)$)· και δείχνουμε ότι το μη-έμβαρο SFVS είναι NP-πλήρες στα γραφήματα μονοπατιών μη-κατευθυντικών δένδρων. Επιπλέον, για το έμβαρο SFVS, παρέχουμε έναν αλγόριθμο για την επίλυση του στα γραφήματα με φύλλωμα το πολύ k ο οποίος τρέχει σε χρόνο $n^{\mathcal{O}(k)}$ και δείχνουμε είναι W[1]-δύσκολο παραμετροποιημένο από το k. Μελετούμε επίσης το SFVS στα H-ελεύθερα γραφήματα, τα οποία είναι γραφήματα που εμφανίζουν ένα είδος εσωτερικής δομής ως αποτέλεσμα απουσίας ενός άλλου. Για το έμβαρο SFVS, παρέχουμε έναν αλγόριθμο πολυωνυμικού χρόνου για την επίλυση του στα $4K_1$-ελεύθερα γραφήματα και δείχνουμε ότι είναι NP-πλήρες στα $5K_1$-ελεύθερα γραφήματα. Για το μη-έμβαρο SFVS, παρέχουμε έναν αλγόριθμο για την επίλυση του στα $(k+1)K_1$-ελεύθερα γραφήματα ο οποίος τρέχει σε χρόνο $n^{\mathcal{O}(k)}$. Το SFVS αποτελεί γενίκευση του κλασικού Προβληματος Ακυκλου Επαγομενου Υπογραφηματος (FVS): δοθέντων ενός γραφήματος (με βάρη στις κορυφές), αναζητούμε υποσύνολο κορυφών του γραφήματος ελαχίστου μεγέθους (βάρους) το οποίο κρούει κάθε υποσύνολο κορυφών του γραφήματος που επάγει κύκλο. Το έμβαρο FVS είναι γνωστό πως είναι επιλύσιμο σε πολυωνυμικό χρόνο στα AT-ελεύθερα γραφήματα και στα χορδικά γραφήματα. Παρέχουμε έναν αλγόριθμο για την επίλυση του στα $(k+1)K_1$-ελεύθερα γραφήματα ο οποίος τρέχει σε χρόνο $n^{\mathcal{O}(k)}$ και για το μη-έμβαρο FVS, δείχνουμε ότι είναι W[1]-δύσκολο παραμετροποιημένο από το k μέσω μίας αναγωγής η οποία είναι διαφορετική από αυτή που υπάρχει στη βιβλιογραφία και ότι στα $(k+1)K_1$-ελεύθερα γραφήματα δεν μπορεί να επιλυθεί σε χρόνο $f(k) \cdot n^{o(k)}$ υπό την ETH.

Το άλλο πρόβλημα στο οποίο εστιάζουμε είναι ένα πρόβλημα κρούσης μονοπατιών και είναι γνωστό στη σχετική βιβλιογραφία ως το Προβλημα Διαχωρισμου Συνολου Κορυφων (NMC): δοθέντων ενός γραφήματος (με βάρη στις κορυφές) και ένα σύνολο τερματικών κορυφών, αναζητούμε υποσύνολο κορυφών του γραφήματος ελαχίστου μεγέθους (βάρους) το οποίο δεν περιέχει τερματικές κορυφές και κρούει κάθε υποσύνολο κορυφών του γραφήματος που επάγει μονοπάτι μεταξύ δύο τερματικών κορυφών. Για το μη-έμβαρο NMC, παρέχουμε έναν αλγόριθμο πολυωνυμικού χρόνου για την επίλυση του στα $3K_1$-ελεύθερα γραφήματα και δείχνουμε ότι είναι NP-πλήρες στα $4K_1$-ελεύθερα γραφήματα μέσω μίας αναγωγής η οποία στηρίζεται στον περιορισμό ότι η λύση στο NMC δεν περιέχει τερματικές κορυφές. Με κίνητρο αυτό το γεγονός, θεωρούμε επίσης το NMC χωρίς αυτόν τον περιορισμό και το καλούμε Προβλημα Απεριοριστου Διαχωρισμου Συνολου Κορυφων (UNMC). Για το έμβαρο UNMC, παρέχουμε έναν αλγόριθμο πολυωνυμικού χρόνου για την επίλυση του στα $3K_1$-ελεύθερα γραφήματα και δείχνουμε ότι είναι NP-πλήρες στα $4K_1$-ελεύθερα γραφήματα. Για το μη-έμβαρο UNMC, παρέχουμε έναν αλγόριθμο για την επίλυση του στα $(k+1)K_1$-ελεύθερα γραφήματα ο οποίος τρέχει σε χρόνο $n^{\mathcal{O}(k)}$ και δείχνουμε ότι είναι W[1]-δύσκολο παραμετροποιημένο από το k και ότι στα $(k+1)K_1$-ελεύθερα γραφήματα δεν μπορεί να επιλυθεί σε χρόνο $f(k) \cdot n^{o(k)}$ ύπο την ETH.

Η παρούσα διατριβή δομείται ως ακολούθως: Το Κεφάλαιο 1 αποτελεί μια εισαγωγή στο αντικείμενο μελέτης. Το Μέρος I παρέχει το απαραίτητο θεωρητικό υπόβαθρο: Στοιχεία της Θεωρίας Πολυπλοκότητας και της Θεωρίας Γραφημάτων δίνονται στα Κεφάλαια 2 και 3 αντίστοιχα και οι ορισμοί των υπό μελέτη προβλημάτων μαζί με προηγουμένως γνωστά αποτελέσματα σχετικά με την πολυπλοκότητά τους δίνονται στο Κεφάλαιο 4. Το Μέρος II παρέχει τα αποτελέσματα μας σε αλγορίθμους και πολυπλοκότητα: Μετά την παροχή των απαραίτητων προκαταρκτικών στο Κεφάλαιο 5, παρουσιάζουμε τα αποτελέσματά μας σε υποκλάσεις των AT-ελεύθερων γραφημάτων, σε υποκλάσεις των χορδικών γραφημάτων και σε H-ελεύθερα γραφήματα στα Κεφάλαια 6, 7 και 8 αντίστοιχα. Το Κεφάλαιο 9 ολοκληρώνει την διατριβή μας με μερικά σχόλια και ανοικτά προβλήματα που παρέχουν κατευθύνσεις για μελλοντική έρευνα.

Abstract

Most graph-theoretic problems are known to be NP-complete on general graphs; it is considered unlikely that there exist polynomial-time algorithms for solving them. Motivated by the fact that many of these problems have real-world applications, researchers have been primarily focused on designing faster and faster exact exponential-time algorithms and better and better approximation algorithms for solving them. Another direction of research — the one that we follow in this thesis —, which gained a lot of momentum in recent years due to the discovery of new powerful tools within the framework of Parameterized Complexity, is to study these problems on particular graphs that exhibit various kinds of internal structure. By designing algorithms that leverage this internal structure, we may provide better running time and/or solution quality guarantees on these particular graphs. We may also show that a problem's hardness persists on these graphs, thusly yielding further insight into the kinds of structure that make the problem hard to solve.

In this thesis, we study Terminal Set problems: given a (vertex-weighted) graph and a vertex subset of the graph, called the *terminal set*, we search for a minimum-sized (minimum-weighted) vertex subset of the graph which intersects (*hits*) every vertex subset of the graph that contains a terminal and induces a subgraph exhibiting a particular structure dependent on the problem. This class of problems contains prominent graph-theoretic problems which have applications both within and beyond the field of Computer Science. We focus our study on two particular problems.

The one is a cycle hitting problem and is known in the relevant literature as the Subset Feedback Vertex Set (SFVS) problem: given a (vertex-weighted) graph and a terminal set, we search for a minimum-sized (minimum-weighted) vertex subset of the graph which hits every vertex subset of the graph that contains a terminal and induces a cycle. We study SFVS on subclasses of AT-free graphs and on subclasses of chordal graphs, which are graphs that are known to exhibit rich internal structure. We provide polynomial-time algorithms for solving weighted SFVS on the following graph classes: interval graphs ($\mathcal{O}(nm)$-time), permutation graphs ($\mathcal{O}(m^3)$-time), cobipartite graphs ($\mathcal{O}(n^4)$-time) and rooted path graphs ($\mathcal{O}(n^2m)$-time); and we show that unweighted SFVS is NP-complete on undirected path graphs. Moreover, for weighted SFVS, we provide an algorithm for solving it on graphs with leafage at most k which runs in $n^{\mathcal{O}(k)}$ time and we show that it is $\mathsf{W}[1]$-hard parameterized by k. We also study SFVS on H-free graphs, which are graphs that exhibit a kind of internal structure as a result of absence of another. For weighted SFVS, we provide a polynomial-time algorithm for solving it on $4K_1$-free graphs and we show that it is NP-complete on $5K_1$-free graphs. For unweighted SFVS, we provide an algorithm for solving it on $(k+1)K_1$-free graphs which runs in $n^{\mathcal{O}(k)}$ time. SFVS consists a generalization of the classical Feedback Vertex Set (FVS) problem: given a (vertex-weighted) graph, we search for a minimum-sized (minimum-weighted) vertex subset of the graph which hits every vertex subset of the graph that induces a cycle. Weighted FVS is known to be polynomial-time solvable on AT-free graphs and on chordal graphs. We provide an algorithm for solving it on $(k+1)K_1$-free graphs which runs in $n^{\mathcal{O}(k)}$ time and for unweighted FVS, we show that it is $\mathsf{W}[1]$-hard parameterized by k via a reduction which is different than the one existing in the literature and that on $(k+1)K_1$-free graphs it cannot be solved in $f(k) \cdot n^{o(k)}$ time under the ETH.

The other problem that we focus on is a path hitting problem and is known in the relevant literature as the Node Multiway Cut (NMC) problem: given a (vertex-weighted) graph and a terminal set, we search for a minimum-sized (minimum-weighted) *terminal-free* vertex subset of the graph which hits every vertex subset of the graph that induces a path between two terminals. For unweighted NMC, we provide a polynomial-time algorithm for solving it on $3K_1$-free graphs and show that it is NP-complete on $4K_1$-free graphs via a reduction which relies on the constraint that the solution to NMC is terminal-free. Motivated by this fact, we also consider NMC without this constraint

and we call it the Unconstrained Node Multiway Cut (UNMC) problem. For weighted UNMC, we provide a polynomial-time algorithm for solving it on $3K_1$-free graphs and we show that it is NP-complete on $4K_1$-free graphs. For unweighted UNMC, we provide an algorithm for solving it on $(k+1)K_1$-free graphs which runs in $n^{\mathcal{O}(k)}$ time and we show that it is $\mathsf{W}[1]$-hard parameterized by k and that on $(k+1)K_1$-free graphs it cannot be solved in $f(k) \cdot n^{o(k)}$ time under the ETH.

This thesis is structured as follows: Chapter 1 consists an introduction to the subject of study. Part I provides the necessary theoretical background: Elements of Complexity Theory and Graph Theory are given in Chapters 2 and 3 respectively and the definitions of the studied graph-theoretical problems along with previously known results on their complexity are given in Chapter 4. Part II provides our algorithmic and complexity-theoretic results: After providing the necessary preliminaries in Chapter 5, we present our results on subclasses of AT-free graphs, on subclasses of chordal graphs and on H-free graphs in Chapters 6, 7 and 8 respectively. Chapter 9 concludes our thesis with some remarks and open problems providing directions for future research.

Contents

1 Introduction

1.1 Cycle Hitting Problems

In the (weighted) Subset Feedback Vertex Set (SFVS) problem, we are given a (vertex-weighted) graph $G = (V, E)$ and a terminal set $S \subseteq V$ and we are asked to find a set $U \subseteq V$ of minimum size (weight) that intersects every cycle passing through at least one terminal. SFVS was introduced by Even et al. who obtained a constant-factor approximation algorithm for its weighted version [30]. The (weighted) SFVS problem is a generalization of the classical (weighted) Feedback Vertex Set (FVS) problem, in which we are given a (vertex-weighted) graph $G = (V, E)$ and we are asked to find a set $U \subseteq V$ of minimum size (weight) that intersects every cycle; for $S = V$, the SFVS problem is equivalent to the NP-complete FVS problem [52]. Both problems find important applications in several aspects that arise in optimization theory, constraint satisfaction, and bayesian inference [1, 2, 30, 32].

We begin with an brief overview of related work on the complexity of FVS and SFVS restricted to graph classes. FVS is known to be NP-complete on bipartite graphs [75] and on planar graphs [37], whereas it becomes polynomial-time solvable on chordal graphs [24, 73], on interval graphs [60], on permutation graphs [10, 11, 12, 57], on their superclass of cocomparability graphs [58] and even on their superclass of AT-free graphs [54]. Clearly, on graph classes on which the FVS problem is NP-complete, so is the SFVS problem, as the latter is a generalization of the former. Therefore, it is natural to study the complexity of SFVS on graph classes on which FVS is polynomial-time solvable. Prior to our work, very little was known regarding the complexity of SFVS on such graph classes. In fact, it was only known that SFVS is NP-complete on split graphs [35]. However, such a result already implies that there is a difference in the complexity of the two problems, as FVS is polynomial-time solvable on their superclass of chordal graphs. The NP-completeness of SFVS even on split graphs motivated a considerable amount of work by several researchers to obtain fast exponential-time algorithms for solving it even when restricted to chordal graphs or even further to split graphs [22, 33, 35, 41, 71].

We continue with an brief overview of related work on the parameterized complexity of FVS and SFVS. FVS is known to be in FPT parameterized by treewidth [26] and cliquewidth [15], which implies that FVS can be solved in polynomial time on graphs with bounded such parameters. Jansen et al. [51] showed that unweighted FVS is W[1]-hard parameterized by the clique cover number and that it can be solved on graphs with maximum induced matching number at most μ in $n^{\mathcal{O}(\mu)}$ time, so it is in XP parameterized by the maximum induced matching number. Jaffke et al. proposed an algorithm that solves weighted FVS on graphs with *maximum-induced-matching--width (mim-width)* w in $n^{\mathcal{O}(w)}$ time [49]. Independently from the work of [49], Bergougnoux and Kanté showed the same result through the notion of *neighbor equivalence* [4]. Despite their related names, graphs with bounded maximum induced matching number are not related to graphs with bounded mim-width as indicated in [74].

The approach of [49] provides a powerful mechanism, as it unifies polynomial-time algorithms for solving weighted FVS on several graph classes, including interval graphs and permutation graphs. Such a mechanism raises the question of whether the algorithm given in [49] can be extended to the more general setting of weighted SFVS. However the proposed algorithm is based on the crucial fact that the forest formed by deleting the nodes of a solution has bounded number of internal nodes which is not necessarily true for the respective induced subgraph of weighted SFVS. Thus it seems difficult to control the size of the solution whenever $S \subset V$.

Cygan et al. [28] and Kawarabayashi and Kobayashi [53] independently showed that unweighted SFVS is in FPT parameterized by the solution size, while Hols and Kratsch [45] provided a randomized polynomial kernel for the problem. Moreover, by standard algorithmic techniques, weighted SFVS is in FPT parameterized by treewidth. Bergougnoux et al. [5] recently proposed an algorithm that solves weighted SFVS in $n^{\mathcal{O}(w^2)}$ time given a decomposition of the input graph of mim-width w.

Subclasses of AT-free Graphs

A graph is *AT-free* if for every triple of pairwise non-adjacent vertices, the neighborhood of one of them separates the two others. The class of AT-free graphs is well-studied and it properly contains interval graphs, permutation graphs and cocomparability graphs [13, 42]. Let us briefly argue that the approach of [54] to solving the weighted FVS problem on AT-free graphs cannot be directly extended towards solving the weighted SFVS problem on (subclasses of) AT-free graphs. One of the basic tools in [54] relies on growing a small representation of an independent set into a suitable forest. Although such a representation is rather small on AT-free graphs, when considering SFVS it is not necessary that the respective set is an independent set which makes it difficult to control how the partial solution may be extended.

Here we initiate the study of SFVS restricted to graph classes from the positive perspective. We consider its weighted version and give the first positive results on interval graphs and permutation graphs, both being proper subclasses of AT-free graphs. Interval graphs and permutation graphs are unrelated to split graphs and are both characterized by a linear structure implied by certain vertex orderings [13, 42, 73]. For both classes of graphs we design polynomial-time algorithms based on dynamic programming of subproblems implied by their natural linear ordering. One of our key ingredients is that we augment our subproblems with a few additional vertices which are always included in the subsolutions. Although for interval graphs such a strategy leads to a simple algorithm, the case for permutation graphs requires augmenting with more vertices, resulting in more numerous and complex recursive relations.

Moreover, towards the unknown complexity of the problem on the class of AT-free graphs, we consider the class of cobipartite graphs and settle its complexity status. Interestingly, most problems that are NP-hard on AT-free graphs are already NP-hard on cobipartite graphs (see, for example, [62]). Cobipartite graphs are the complements of bipartite graphs and are unrelated to interval graphs and permutation graphs. We show that weighted SFVS admits a simple solution on cobipartite graphs, and, thus, we eliminate the possibility of obtaining NP-hardness on AT-free graphs through NP-hardness on cobipartite graphs. Therefore, we provide the first positive results regarding the complexity of the SFVS problem on subclasses of AT-free graphs. In particular, we show that weighted SFVS can be solved

- on interval graphs in $\mathcal{O}(nm)$ time,

- on permutation graphs in $\mathcal{O}(m^3)$ time and
- on cobipartite graphs in $\mathcal{O}(n^4)$ time.

Subclasses of Chordal Graphs

Several fundamental optimization problems are known to be intractable on chordal graphs, however they admit polynomial-time algorithms when restricted to a proper subclass of chordal graphs such as interval graphs. Typical examples of problems that exhibit this behavior are domination and induced path problems [6, 8, 25, 44, 46, 66]. Although SFVS does not fall under the themes of domination or induced path problems, it is known to be NP-complete on chordal graphs [35], whereas it becomes polynomial-time solvable on interval graphs [69]. Towards a better understanding of why many intractable problems on chordal graphs admit polynomial-time algorithms on interval graphs, we consider the algorithmic usage of a structural parameter called *leafage*. Leafage, introduced by Lin et al. [59], is a graph parameter that captures how close a chordal graph is to being an interval graph. As it concerns chordal graphs, leafage essentially measures the smallest number of leaves in a clique tree, an intersection representation of the given graph [39]. Thus our study of SFVS on subclasses of chordal graphs constists an investigation of the extent to which the structure of the underlying tree representation influences the computational complexity of SFVS.

Habib and Stacho [43] showed that the leafage of a connected chordal graph can be computed in polynomial time. Their described algorithm also constructs a corresponding clique tree with the minimum number of leaves. Regarding other problems that behave well with the leafage, we mention the Minimum Dominating Set problem which Fomin et al. [34] showed to be in FPT parameterized by the leafage. We show that

- weighted SFVS can be solved on chordal graphs with leafage at most ℓ in $\mathcal{O}(n^{2\ell+1})$ time, which is polynomial time on chordal graphs with bounded ℓ.

In particular, we provide an algorithm that given a chordal graph and a tree model of it with ℓ leaves solves the problem in $\mathcal{O}(n^{2\ell+1})$ time. Thus, by combining the algorithm of Habib and Stacho [43] with our algorithm, we deduce that weighted SFVS is in XP parameterized by the leafage. Our algorithm works on an expanded tree model that is obtained from the given tree model and maintains all intersecting information without increasing the number of leaves. Then in a bottom-up dynamic programming fashion, we visit every node of the expanded tree model in order to compute partial solutions. At each intermediate step, we store all necessary information of subsets of vertices that are of size $\mathcal{O}(\ell)$.

One advantage of leafage over mim-width is that we can compute the leafage of a chordal graph in polynomial time, whereas we do not know how to compute the mim-width of a chordal graph in polynomial time. However we note that a graph with bounded leafage implies a graph with bounded mim-width and, further, a decomposition of bounded mim-width can be computed in polynomial time [34]. This can be seen through the notion of H-graphs. For some fixed graph H, a graph is an H-graph if it is the intersection graph of connected subgraphs of some subdivision of H. The intersection model of subtrees of a tree T having ℓ leaves is a T'-graph where T' is obtained from T by contracting nodes of degree two. Thus the size of T' is at most 2ℓ, since T has ℓ leaves. Moreover, given an H-graph and its intersection model, a (linear) decomposition of mim-width at most $2|E(H)|$ can be computed in polynomial time [34]. Therefore, given a graph with leafage at most ℓ, there is a polynomial-time algorithm that computes a decomposition of mim-width $\mathcal{O}(\ell)$. Combined with the algorithm for solving weighted SFVS on graphs of bounded mim-width [5], one can solve weighted SFVS on graphs with leafage at most ℓ in $n^{\mathcal{O}(\ell^2)}$ time. Notably, our $n^{\mathcal{O}(\ell)}$-time algorithm is a non-trivial improvement on the running time obtained from the mim-width approach.

We complement our algorithmic result by showing that

- weighted SFVS is W[1]-hard parameterized by the leafage via a reduction from the Multicolored Clique problem.

Thus we can hardly avoid the dependence of the exponent in the stated running time. Our reduction is inspired by the W[1]-hardness of unweighted FVS parameterized by mim-width given by Jaffke et al. [50]. However we note that our result holds on graphs with arbitrary vertex weights and we are not aware if unweighted SFVS admits the same complexity behaviour.

Leaf power graphs are a subclass of chordal graphs that admit a decomposition of mim-width one [47], thus via the algorithm proposed by Bergougnoux et al. [5] weighted SFVS can be solved in polynomial time on leaf power graphs if an intersection model is given as input. However, to the best of our knowledge, it is not known whether an intersection model of a leaf power graph can be constructed in polynomial time. Rooted path graphs are the intersection graphs of directed paths in a rooted tree. They form a subclass of leaf powers and we observe that they form a class of unbounded leafage. Although rooted path graphs admit a decomposition of mim-width one [47] and such a decomposition can be constructed in polynomial time [29, 40], the running time obtained through the bounded mim-width approach is rather impractical as it requires to store a table of size $\mathcal{O}(n^{13})$ even in this particular case [5]. We show that

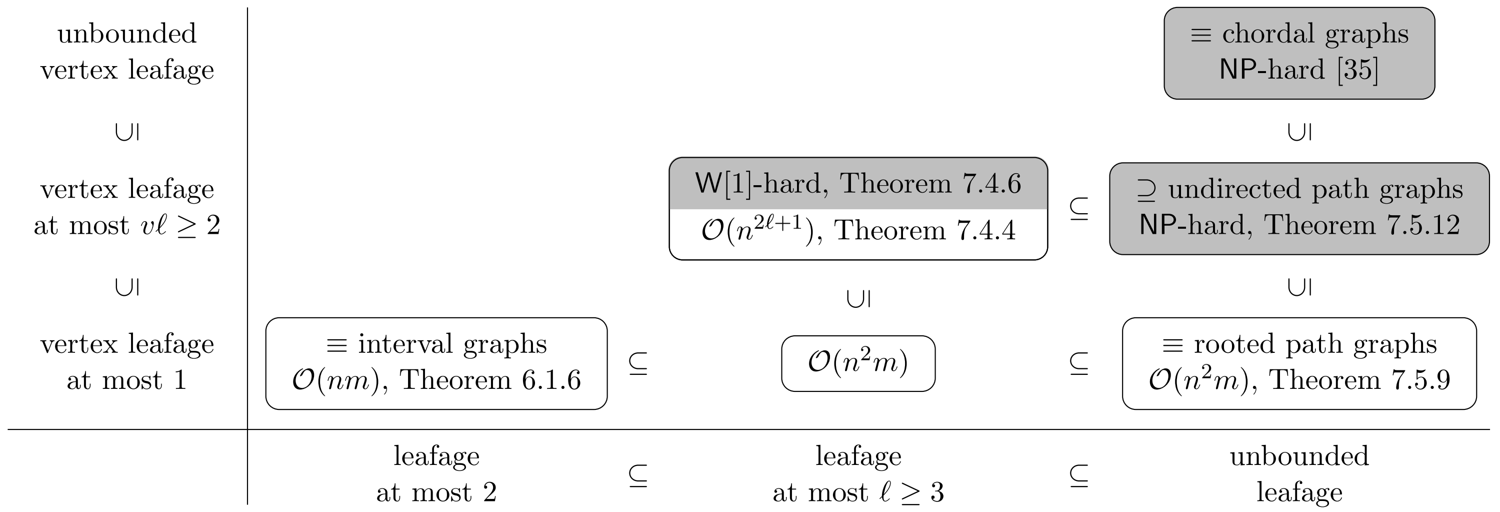
- weighted SFVS can be solved on rooted path graphs in $\mathcal{O}(n^2m)$ time.

As a byproduct of our dynamic programming scheme and the expanded tree model, we show how our approach can be extended in order to handle rooted path graphs. By analyzing further subsets of vertices at each intermediate step, we manage to derive an algorithm for weighted SFVS on rooted path graphs that runs in $\mathcal{O}(n^2m)$ time. Observe that the stated running time is comparable to the $\mathcal{O}(nm)$ time of the previously known algorithm on interval graphs [69]. Interval graphs form a proper subclass of rooted path graphs.

Inspired by the algorithm on graphs with bounded leafage, we also consider the natural relaxation of the leafage that is the *vertex leafage* of a graph. Chaplick and Stacho [19] introduced the vertex leafage of a graph G as the smallest number k such that there exists a tree model for G in which every subtree corresponding to a vertex of G has at most k leaves. As leafage measures the closeness to interval graphs (graphs with leafage at most two), vertex leafage measures the closeness to undirected path graphs which are the intersection graphs of paths in a tree (graphs with vertex leafage at most two). We prove that

- unweighted SFVS is NP-complete on undirected path graphs

and, thus, the problem is para-NP-complete parameterized by the vertex leafage. An interesting trait of our NP-completeness proof is that our reduction comes from the Max Cut problem as opposed to known reductions for SFVS which are usually based on, more natural, covering problems [35, 70]. From our results for rooted path graphs and undirected path graphs, we obtain a complexity dichotomy of the problem with respect to the vertex leafage: if the vertex leafage is at most one (rooted path graphs) then SFVS is polynomial-time solvable; otherwise, if the vertex leafage is at least two, SFVS is NP-complete. Our findings regarding the complexity of SFVS on subclasses of chordal graphs are summarized in Figure 1.1.



unbounded vertex leafage
⊆
vertex leafage at most $v\ell \geq 2$
⊆
vertex leafage at most 1
$\equiv$ chordal graphs
NP-hard [35]
W[1]-hard, Theorem 7.4.6
$\mathcal{O}(n^{2\ell+1})$, Theorem 7.4.4
$\supseteq$ undirected path graphs
NP-hard, Theorem 7.5.12
$\equiv$ interval graphs
$\mathcal{O}(nm)$, Theorem 6.1.6
$\mathcal{O}(n^2m)$
$\equiv$ rooted path graphs
$\mathcal{O}(n^2m)$, Theorem 7.5.9
leafage at most 2
⊆
leafage at most $\ell \geq 3$
⊆
unbounded leafage

Figure 1.1: Computational complexity of the SFVS problem parameterized by leafage and vertex leafage. All NP-hardness results are for the unweighted SFVS problem and all other results are for the weighted SFVS problem.

H-free Graphs

We consider classes of H-free graphs which are characterized by a bounded structural parameter. In particular, we consider $(\alpha + 1)K_1$-free graphs which are exactly the graphs with independent set number at most α. Notice that cobipartite graphs are $3K_1$-free graphs.

We completely characterize the complexity of SFVS with respect to the independent set number α. In particular, we show that

- weighted SFVS can be solved in polynomial time on graphs with $\alpha \leq 3$ by exploiting a structural characterization of the solution which involves the distances from vertices to terminals, whereas it remains NP-complete on graphs with $\alpha \geq 4$ via a reduction from unweighted Vertex Cover on tripartite graphs.

Weighted FVS is polynomial-time solvable on graphs with bounded α [51]. Thus we expand our knowledge on the complexity difference of the two problems with respect to a structural graph parameter. In order to complement our results, we show that

- unweighted SFVS can be solved in $n^{\mathcal{O}(\alpha)}$ time, which is polynomial time on graphs with bounded α.

Thus we provide a complexity difference between the weighted and the unweighted versions of the problem with respect to a natural structural parameter. We also show that

- weighted FVS can be solved in $n^{\mathcal{O}(\alpha)}$ time, which is polynomial time on graphs with bounded α, via an algorithm that is simpler and faster than the one implied by [51] and
- unweighted FVS is W[1]-hard parameterized by α via an reduction from Multicolored Intependent Set which is different than the one given in [51] and linear in α, and thus it cannot be solved in $f(\alpha) \cdot n^{o(\alpha)}$ time under the *Exponential Time Hypothesis* (ETH) by combining our alternative reduction with the conditional lower bound given in [20], which implies that the running times of our algorithms for solving unweighted SFVS and weighted FVS are tight.

Our findings regarding the complexity of FVS and SFVS on graphs with bounded α are summarized in Figure 1.2.

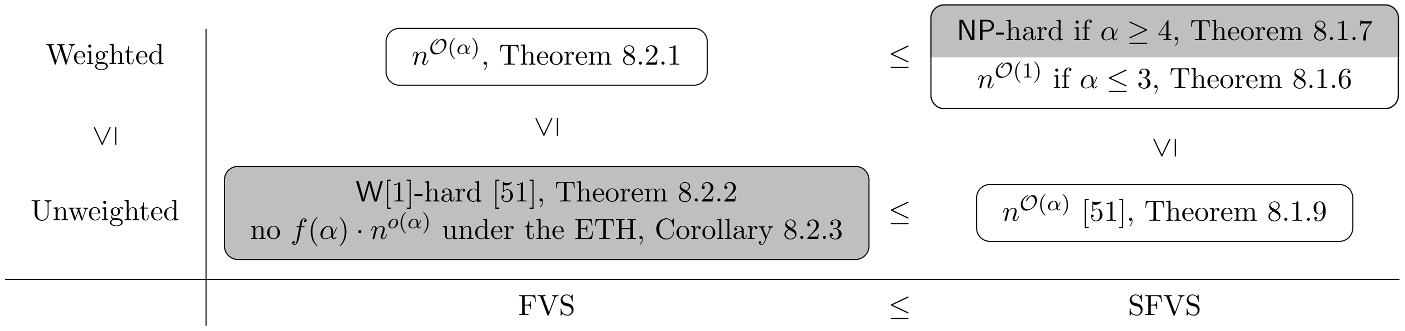

Weighted
$\geq$
Unweighted
$n^{\mathcal{O}(\alpha)}$, Theorem 8.2.1
$\geq$
W[1]-hard [51], Theorem 8.2.2
no $f(\alpha) \cdot n^{o(\alpha)}$ under the ETH, Corollary 8.2.3
$\leq$
$\leq$
NP-hard if $\alpha \geq 4$, Theorem 8.1.7
$n^{\mathcal{O}(1)}$ if $\alpha \leq 3$, Theorem 8.1.6
$\geq$
$n^{\mathcal{O}(\alpha)}$ [51], Theorem 8.1.9
FVS
$\leq$
SFVS

Figure 1.2: Computational complexity of the FVS and SFVS problems on $(\alpha + 1)K_1$-free graphs.

1.2 Path Hitting Problems

In the (weighted) Node Multiway Cut (NMC) problem, we are given a (vertex-weighted) graph $G = (V, E)$ and a terminal set $T \subseteq V$ and we are asked to find a set $U \subseteq V \setminus T$ of minimum size (weight) that intersects every path between two terminals. Interestingly, there is a simple reduction from the NP-complete NMC problem to the weighted SFVS problem where $|S| = 1$ [35]. We also consider a relaxation of NMC in which the solution U is allowed to include terminals and we call it the Unconstrained Node Multiway Cut (UNMC) problem. NMC is known to be in FPT parameterized by the solution size [21, 61] and even above guaranteed value [27]. For further results on variants of NMC we refer to [17, 38, 55].

H-free Graphs

We completely characterize the complexity of NMC with respect to the independent set number α. In particular, we show that

- weighted NMC can be solved in polynomial time on graphs with $\alpha \leq 2$, whereas unweighted NMC remains NP-complete on graphs with $\alpha \geq 3$ by adapting the reduction for weighted SFVS for $\alpha \geq 4$.

We also completely characterize the complexity of UNMC with respect to the independent set number α. In particular, we show that

- weighted UNMC can be solved in polynomial time on graphs with $\alpha \leq 2$ by invoking our algorithm for weighted SFVS on graphs with $\alpha \leq 3$, whereas it remains NP-complete on graphs with $\alpha \geq 3$ via a reduction from unweighted NMC, and
- unweighted UNMC can be solved in $n^{\mathcal{O}(\alpha)}$ time, which is polynomial time on graphs with bounded α, using an idea similar to the polynomial-time algorithm for the unweighted SFVS problem, it is W[1]-hard parameterized by α via a reduction from Multicolored Indepedent Set which is linear on α, and thus it cannot be solved in $f(\alpha) \cdot n^{o(\alpha)}$ time under the ETH, which implies that the running time of our algorithm is tight.

Part I

Theoretical Background and Related Work

2 Complexity Theory

Complexity Theory provides a mathematical framework for comparing the efficiency of algorithms in terms of their requirements in time and space independently of specific implementation or executing machine.

2.1 Computational Complexity

Computational Complexity aims to answer the following question (among others): Given input of size n, how many elementary operations must an algorithm perform *in the worst case* in order to terminate successfully? Thus, we are interested in determining the running time of the algorithm as a function of the input size n. We are particularly interested in determining the behaviour of this function as the input size n grows arbitrarily large, that is, its asymptotic behaviour as $n \to \infty$. For every function $g : \mathbb{N} \to \mathbb{R}_+$, we define the following collections of functions $f : \mathbb{N} \to \mathbb{R}_+$:

$$\mathcal{O}(g(n)) = \left\{ f(n) \;\middle|\; \limsup \frac{f(n)}{g(n)} < \infty \right\} \quad \Theta(g(n)) = \mathcal{O}(g(n)) \cap \Omega(g(n))$$

$$\Omega(g(n)) = \left\{ f(n) \;\middle|\; \liminf \frac{f(n)}{g(n)} > 0 \right\} \quad o(g(n)) = \left\{ f(n) \;\middle|\; \lim \frac{f(n)}{g(n)} = 0 \right\}$$

If the running time is $f(n) \in \mathcal{O}(g(n))$ where g is a polynomial function of n, then we say that the running time is polynomial.

In this thesis, we distinguish two kinds of problems: *decision* problems and *optimization* problems. In a decision problem, the goal is to determine the truth value of a proposition regarding the input. In an optimization problem, the goal is to determine the optimal value of a function of the input.

2.1.1 Computational Complexity Classes

Problems that exhibit similar behaviour with respect to their complexity form complexity classes. A natural starting point is to consider problems which are polynomial-time solvable. These decision problems form the first complexity class.

Polynomial (P)

A decision problem is in the class P if it is solvable in polynomial time.

The desire is for decision problems to be in the class P. However, most problems of interest are not known to be polynomial-time solvable. Of particular interest are the problems for which the problem of verifying a solution to them is polynomial-time solvable. These decision problems form the second complexity class.

Nondeterministic Polynomial (NP)

A decision problem is in the class NP if it is verifiable in polynomial time.

It is not difficult to show that $\mathsf{P} \subseteq \mathsf{NP}$ holds. However, it is not known whether $\mathsf{P} \subset \mathsf{NP}$ or $\mathsf{P} = \mathsf{NP}$ holds. This is known as the P *versus* NP problem and is considered one of the most difficult to resolve open problems in all of Mathematics.

2.1.2 Hardness and Completeness

In tackling the P versus NP problem, we focus on the problems in NP which are the hardest to solve. Consider two decision problems A and B. A *reduction* from A to B is a polynomial-time algorithm that given an instance X of A

constructs an instance Y of B such that X is a `yes`-instance of A if and only if Y is a `yes`-instance of B. If there exists a reduction from A to B, we say that A reduces to B and we write $A \leq B$. Notice that if A reduces to B and B is in P, then A is also in P; a polynomial-time algorithm for solving A is the following: given an instance X of A,

Step 1. we call a reduction from A to B on X to get an instance Y of B,

Step 2. we call a polynomial-time algorithm for solving B on Y, and

Step 3. we return the latter's answer.

Therefore, $A \leq B$ implies that A is not harder to solve than B or, in other words, that B is at least as hard to solve as A. Now consider a complexity class C of decision problems. A problem is said to be C-*hard* if every problem in C reduces to it, and it is said to be C-*complete* if it is both in C and C-hard. The definition implies that the C-complete problems are the problems in C which are the hardest to solve. Observe that $\mathsf{P} = \mathsf{NP}$ if and only if there exists an NP-complete problem in P.

The problem which was first shown to be NP-complete by Cook and independently by Levin is the Boolean Satisfiability problem on formulas in *conjunctive normal form* [23, 56]. A number of additional problems were subsequently shown to be NP-complete by Karp via chained reductions from this problem [52]. Garey and Johnson (1979) [37] contains a long list of NP-complete problems from many fields, including Graph Theory.

Instead of the aforementioned problem, we may first show that the Circuit Satisfiability problem is NP-complete. A *boolean circuit* is a circuit consisting only of input gates, `and`-gates, `or`-gates, `not`-gates and a single output gate.

Circuit Satisfiability

Input:	A boolean circuit C.
Question:	Is there an assignment on the inputs of C for which the output of C is `true`?

2.2 Parameterized Complexity

One of the ways of tackling NP-hard problems is to study their complexity as a function of more than just one variable. All variables beyond the first one, which must be the input size n, are called *parameters*. We typically consider only one parameter which we denote by k.

2.2.1 Parameterized Complexity Classes

The goal of Parameterized Complexity is the design of algorithms with running times that depend on the size n as weakly as possible. In particular, the desire is for problems to be *fixed-parameter tractable* parameterized by the parameter k.

Fixed-parameter Tractable (FPT)

A decision problem having instances of size n with parameter k is in the class FPT if it is solvable in $f(k) \cdot n^{\mathcal{O}(1)}$ time for some computable function f.

"Slicewise" Polynomial (XP)

A decision problem having instances of size n with parameter k is in the class XP if it is solvable in $f(k) \cdot n^{\mathcal{O}(g(k))}$ time for some computable functions f and g.

It is not difficult to show that FPT $\subseteq$ XP. Notice that if a problem is in XP, then it is in P when restricted to instances with parameter $k = c$ for every constant c, hence the name of the class XP.

para-NP

A decision problem having instances of size n with parameter k is in the class para-NP if it is verifiable in $f(k) \cdot n^{\mathcal{O}(1)}$ time for some computable function f.

It is not difficult to show that FPT $\subseteq$ para-NP. A problem is *para*-NP-*hard* if there exists a constant c such that the problem is NP-hard when restricted to instances with parameter $k = c$. It is known that a para-NP-hard problem cannot be in XP, unless $\mathsf{P} = \mathsf{NP}$.

2.2.2 Parameterized Hardness and Completeness

The framework of Parameterized Complexity also provides the tools to show that it is unlikely for a problem to be in FPT under plausible complexity-theoretic assumptions.

Consider two problems A and B. A reduction from A to B is a *parameterized reduction* if given an instance X of size n with parameter k it constructs in $f(k) \cdot n^{\mathcal{O}(1)}$ time an instance Y of size n' with parameter $k' \leq g(k)$ for some computable functions f and g. Notice that if there exists a parameterized reduction from A to B and B is in FPT, then A is in FPT.

We will now define additional parameterized complexity classes with respect to which we will consider hardness and completeness. The *depth* of a boolean circuit is the maximum length of a path from an input gate to the output gate and its *weft* is the maximum number of gates with more than two inputs contained on a path from an input gate to the output gate.

Weighted Circuit Satisfiability (WCS)

Input: A boolean circuit C and a number $k \in \mathbb{N}$.
Question: Is there an assignment on the inputs of C such that exactly k inputs are `true` for which the output of C is `true`?

W-hierarchy

For every $t \in \mathbb{N}^*$, a decision problem is in the class $\mathsf{W}[t]$ if there exists $d \in \mathbb{N}^*$ such that there exists a parameterized reduction from the problem to WCS on boolean circuits of depth at most d and weft at most t.

It is not difficult to show that $\mathsf{W}[t] \subseteq \mathsf{W}[t+1]$ for all $t \in \mathbb{N}^*$. It is known that $\mathsf{FPT} \subseteq W[1]$ and that $\mathsf{FPT} \neq W[1]$ implies that $\mathsf{P} \neq \mathsf{NP}$. To show that a problem is unlikely to be in FPT, we show that it is $\mathsf{W}[1]$-hard via a parameterized reduction from a known $\mathsf{W}[1]$-hard problem.

2.2.3 Parameterized Complexity Lower Bounds

For problems that are shown to be in FPT or in XP, there is a desire for a more fine-grained analysis of the running time with respect to the computable functions f and g. In particular, there is a desire for determining assymptotical lower bounds on f and g under plausible additional complexity-theoretical assumptions. Such an assumption is the following:

Hypothesis (Exponential Time Hypothesis (ETH))**.** *The* 3*-SAT problem on clauses on n variables cannot be solved in* $2^{o(n)}$ *time.*

It is known that the ETH holding implies that $\mathsf{FPT} \neq \mathsf{W}[1]$. Under the assumption that the ETH holds, we can show parameterized lower bounds on the complexity of other problems via appropriate parameterized reductions.

3 Graph Theory

3.1 Graphs

A pair $G = (V, E)$ is called a *graph* if $E \subseteq \{X \subseteq V \mid |X| = 2\}$. The sets V and E are called the *vertex set* and the *edge set* of the graph G respectively and their elements are called *vertices* and *edges* respectively. Unless otherwise stated, we denote the vertex set and the edge set of a graph G by $V(G)$ and $E(G)$ respectively and we use n and m to denote $|V(G)|$ and $|E(G)|$ respectively. A *weighted* graph $G = (V, E)$ is a graph where each vertex $v \in V$ is assigned *weight* that is a positive integer number. We denote by $w(v)$ the weight of each vertex $v \in V$. For a vertex subset $X \subseteq V$, the weight of X is $\sum_{v \in X} w(v)$.

For every set X, we denote by $\mathcal{P}_2(X)$ the collection $\{X' \subseteq X \mid |X'| = 2\}$. Let $G = (V, E)$ be a graph. A graph $G' = (V', E')$ is a *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E \cap \mathcal{P}_2(V')$. For every $X \subseteq V$, the subgraph of G *induced by* X is the graph $(X, E \cap \mathcal{P}_2(X))$ and is denoted by $G[X]$. Moreover, we denote by $G - X$ the graph induced by the vertices of $V \setminus X$. If $X = \{u\}$, we also write $G - u$. The *neighborhood* of a vertex x of G is $N_G(x) = \{v \in V : xv \in E\}$ and the *degree* of x is $|N_G(x)|$. A *leaf* is a vertex whose degree is at most 1. We say that two vertices $u, v \in V$ are *adjacent* if $uv \in E$. We further say that two vertices are true twins (resp. false twins) if they are adjacent (resp. non-adjacent) and they have the same neighborhood. For $X \subseteq V$, $N_G(X) = \bigcup_{v \in X} N_G(v) \setminus X$ and $N_G[X] = N_G(X) \cup X$.

A *clique* is a set of pairwise adjacent vertices, while an *independent set* is a set of pairwise non-adjacent vertices. A *path* is a sequence of vertices $\langle v_1 v_2 \cdots v_k \rangle$ where each pair of consecutive vertices $v_i v_{i+1}$ forms an edge of G. A graph is *connected* if there is a path between any pair of vertices. A *connected component* of G is a maximal connected subgraph of G.

3.1.1 Elementary Graphs

Complete Graphs

A graph is a *complete* graph if its vertex set is a clique. The complete graph on n vertices is denoted by K_n.

Paths and Cycles

A graph is a *path* if there exists an ordering of its vertices such that two vertices are adjacent if and only if they are consecutive elements in the ordering. The path graph on n vertices is denoted by P_n.

A graph is a *cycle* if there exists an ordering of its vertices such that two vertices are adjacent if and only if they are consecutive elements in the ordering or the first and last vertices. The cycle graph on n vertices is denoted by C_n. We call cycles on three and four vertices *triangles* and *squares* respectively.

Forests, Trees and Stars

A graph is a *forest* if it has no cycle as a subgraph. A forest is a *tree* if it is connected. A tree is a *star* if its vertices are all leaves except one. Notice that this implies that the leaves of a star are all adjacent to its remaining vertex. A star with ℓ leaves is called an *ℓ-star*.

Multipartite Graphs and Their Complements

For every $k \in \mathbb{N}$, a graph is a *k-partite* graph if its vertex set can be partitioned into k independent sets. For the cases of $k = 2$ and $k = 3$, k-partite graphs are also called *bipartite* graphs and *tripartite* graphs respectively.

For every $k \in \mathbb{N}$, a graph is a *co-k-partite* graph if its vertex set can be partitioned into k cliques. For the cases of $k = 2$ and $k = 3$, co-k-partite graphs are also called *cobipartite* graphs and *cotripartite* graphs respectively.

3.2 Graph Classes

Graph classes are collections of graphs that exhibit common structure. A property that holds for and only for the graphs of a class is called a *characterization* of the class. In particular, the definition of a class is a characterization of the class; and further characterizations of a class can be viewed as alternative equivalent definitions of the class. In this thesis, we are particularly interested in the following two types of characterizations.

$\mathcal{H}$-free Graphs

Let $\mathcal{H}$ be a (possibly infinite) collection of graphs. A graph G is an *$\mathcal{H}$-free* graph if no graph in $\mathcal{H}$ is an induced subgraph of G. For the case of $\mathcal{H} = \{H\}$, $\mathcal{H}$-free graphs are also called *H-free* graphs.

In this thesis, we are particularly interested in the classes of H-free graphs where $H = (k+1)K_1$, $k \in \mathbb{N}$. Observe that for every $k \in \mathbb{N}$, co-k-partite graphs are a subclass of $(k+1)K_1$-free graphs.

Intersection Graphs

Let X be a set and let $\mathcal{C}$ be a collection of subsets of X. Set $\mathcal{M} = (X, \mathcal{C})$. We define $G(\mathcal{M})$ to be the graph $(\mathcal{C}, \{\{U_1, U_2\} \in \mathcal{P}_2(\mathcal{C}) \mid U_1 \cap U_2 \neq \varnothing\})$. We say that $\mathcal{M}$ is an *intersection model* of $G(\mathcal{M})$ and $G(\mathcal{M})$ is an *intersection graph* of $\mathcal{M}$.

Let H be a graph and let $\mathcal{C}$ be a collection of subgraphs of H. Set $\mathcal{M} = (H, \mathcal{C})$ and set $\mathcal{M}' = \{V(H), \{V(H') \mid H' \in \mathcal{C}\}\}$. We define $G(\mathcal{M})$ to be the graph $G(\mathcal{M}')$. We say that $\mathcal{M}$ is an *intersection model* of $G(\mathcal{M})$ and $G(\mathcal{M})$ is an *intersection graph* of $\mathcal{M}$.

3.2.1 AT-free Graphs

A set of three vertices of a graph is called an *asteroidal triple* if for every two vertices of the set there exists a path containing them that does not intersect the neighbourhood of the third one. A graph is an *AT-free* graph if it has no asteroidal triple. We note that co-k-partite graphs are a subclass of AT-free graphs if and only if $k \leq 2$.

Interval Graphs

A graph is an *interval* graph if it is the intersection graph of $(\mathbb{R}, \mathcal{I})$ where $\mathcal{I}$ is a collection of closed intervals of $\mathbb{R}$. We will subsequently call such an intersection model an *interval model.* Whether a given graph is an interval graph can be decided in linear time and if so, an interval model can be constructed in linear time [36]. It is known that every induced cycle of an interval graph is a triangle [60, 73].

Permutation Graphs

Given a permutation $\pi = (\pi(1), \ldots, \pi(n))$ over $\{1, \ldots, n\}$, the *inversion graph* of π, denoted by $G(\pi)$, has vertex set $\{1, \ldots, n\}$ and two vertices i, j are adjacent if and only if $(i-j)(\pi(i)-\pi(j)) < 0$. A graph is a *permutation* graph if it is isomorphic to the inversion graph of a permutation [13, 42]. Permutation graphs can also be characterized as a particular class of intersection graphs. A graph is a permutation graph if it is the intersection graph of a collection of line segments with one endpoint lying on L_0 and the other on L_1 where L_0, L_1 are two parallel lines of the plane. We refer to the two parallel lines as the *top* and *bottom* lines. We will subsequently call such an intersection model a *permutation model.* Whether a given graph is a permutation graph can be decided in linear time and if so, a permutation model can be constructed in linear time [63]. It is known that every induced cycle of a permutation graph is either an triangle or a square [10, 11, 12, 57, 73].

A graph is a k-permutation graph if it is the intersection graph of a collection of polygonal chains on $k+1$ points with the i-th point lying on L_{i-1} for every $i \in [k+1]$ where $L_0, L_1, \ldots, L_k$ are $k+1$ parallel lines in ascending order of their distance to the first one and descending order of the distance to the last one among them. It is not difficult to show that permutation graphs are 1-permutation graphs and k-permutation graphs are $(k+1)$-permutation graphs for every $k \in \mathbb{N}^*$.

Trapezoid Graphs

A graph is a *trapezoid* graph if it is the intersection graph of a collection of trapezoids with one base lying on L_0 and the other on L_1 where L_0, L_1 are two parallel lines of the plane. It is not difficult to show that interval graphs and permutation graphs are trapezoid graphs.

We call the union of k trapezoids a *k-trapezoid* if there exists an ordering of the k trapezoids and for every trapezoid, there exists an ordering of its bases, such that the second base of the i-th trapezoid is also the first base of the $(i+1)$-th trapezoid for every $i \in [k-1]$. We conclude that a k-trapezoid can be specified by a sequence of $k+1$ bases. A graph is a k-trapezoid graph if it is the intersection graph of a collection of k-trapezoids with the i-th base lying on L_{i-1} for every $i \in [k+1]$ where $L_0, L_1, \ldots, L_k$ are $k+1$ parallel lines of the plane in ascending order of their distance to the first one and descending order of the distance to the last one among them. It is not difficult to show that trapezoid graphs are 1-trapezoid graphs and k-trapezoid graphs are $(k+1)$-trapezoid graphs for every $k \in \mathbb{N}^*$.

Cocomparability Graphs

A graph is a *cocomparability* graph if it is the intersection graph of a collection of curves with one endpoint lying on L_0 and the other on L_1 where L_0, L_1 are two parallel lines of the plane. It is known that k-trapezoid graphs are cocomparability graphs.

3.2.2 Chordal Graphs

A graph is a *chordal* graph if every cycle of the graph that is not a triangle has a chord or, equivalently, if every induced cycle of the graph is a triangle. Chordal graphs can also be characterized as a particular class of intersection graphs: A graph is a chordal graph if it is the intersection graph of $(T, \mathcal{T})$ where T is a tree and $\mathcal{T}$ is a collection of subtrees of T. We will subsequently call such an intersection model a *tree model.*

Split Graphs

A graph is a *split* graph if its vertex set can be partitioned into a clique and an independent set. Observe that the host tree of a tree model of a split graph can be an ℓ-star where ℓ is the size of the independent set in the bipartition of the split graph's vertex set.

Strongly Chordal Graphs

A chord of an even cycle is odd if the distance of its endpoints in the cycle odd. A graph is a *strongly chordal* graph if it is a chordal graph and every even cycle of the graph that is not a square has an odd chord. Thus strongly chordal graphs are chordal graphs by definition.

Leaf Powers

Let G be a graph. We denote by $\mathcal{T}(G)$ the collection of all trees T such that $L(T) = V(G)$. The graph G is a *leaf power* graph if there exist $k \in \mathbb{N}^*$ and $T \in \mathcal{T}(G)$ such that two vertices of G are adjacent if and only if their distance in T is at most k. It is known that leaf power graphs are strongly chordal graphs.

Undirected/Directed/Rooted Path Graphs and Interval Graphs

A number of subclasses of chordal graphs are defined as the classes of intersection graphs of tree models restricted to various kinds of trees and subtrees. These are

- *undirected path* graphs if all subtrees are paths,
- *directed path* graphs if the host tree is a directed tree and all subtrees are directed paths,
- *rooted path* graphs if the host tree is a rooted tree and all subtrees are directed paths, and
- *interval* graphs if the host tree is a path.

Structural properties and recognition algorithms are known for all these graph classes [18, 65, 68]. It is not difficult to see that interval graphs are rooted path graphs, which are directed path graphs, which are undirected path graphs.

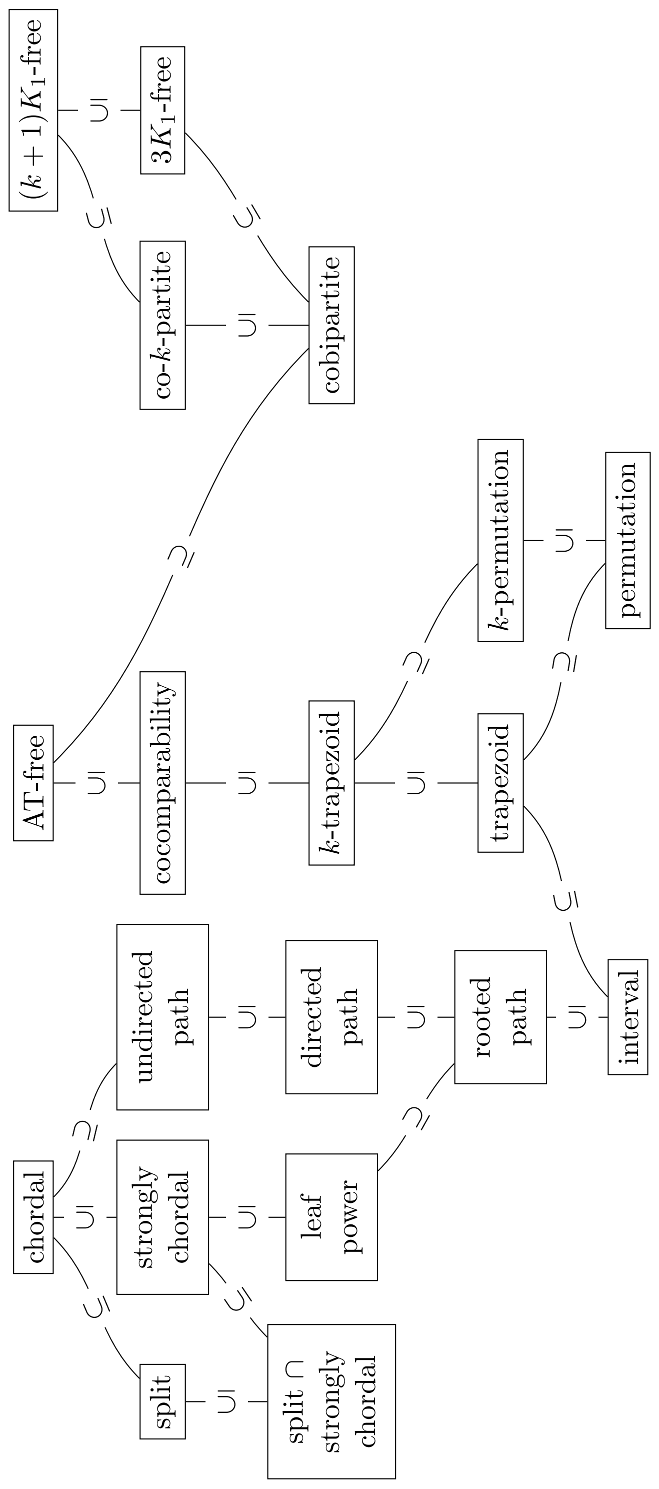


Figure 3.1: Relations between the graph classes.

3.3 Graph Parameters

3.3.1 Graph Parameters of General Graphs

Parameters of graphs that are being studied in the literature generally fall into one of two categories. The *numbers* are sizes of optimal substructures of graphs and the *widths* are measures indicative of the complexity of the global structure of graphs.

Clique Cover Number

A collection $\mathcal{C}$ of subsets of a set S is called a *cover* of S if $\cup\mathcal{C} = S$. A cover $\mathcal{C}$ of a set S is called a *partition* of S if all elements of $\mathcal{C}$ are non-empty and pairwise disjoint. Given a cover $\mathcal{C}$ of a set S and a total order $\leq$ on $\mathcal{C}$, it is not difficult to show that the collection $\mathcal{C}' = \{X \setminus \cup\{X' \in \mathcal{C} \mid X' < X\} \mid X \in \mathcal{C}\} \setminus \{\varnothing\}$ is a partition of S such that $|\mathcal{C}'| \leq |\mathcal{C}|$.

A collection $\mathcal{C}$ of vertex subsets of a graph G is called a *clique cover* of G if $\mathcal{C}$ is a cover of $V(G)$ and all elements of $\mathcal{C}$ are cliques of G. The *clique cover number* of a graph G, denoted by $\kappa(G)$, is the minimum cardinality of a clique cover of G.

Independent Set Number

A set I of vertices of a graph G is called an *independent set* of G if there are no edges in $G[I]$. The *independent set number* of a graph G, denoted by $\alpha(G)$, is the maximum cardinality of an independent set of G. Observe that $\alpha(G) \leq \kappa(G)$ for every graph G.

Maximum Induced Matching Number

A set P of pairwise-disjoint edges of a graph G is called an *induced matching* of G if there are no edges in $G[\cup P]$ beside the ones in P. The *maximum induced matching number* of a graph G, denoted by $\mu(G)$, is the maximum cardinality of an induced matching of G. Observe that $\mu(G) \leq \alpha(G)$ for every graph G.

Maximum-Induced-Matching--width (Mim-width)

For every graph G and for every bipartition $\{V_1, V_2\}$ of $V(G)$, we define $G[V_1, V_2]$ to be the graph $G[V_1 \cup V_2]$ minus all edges in $G[V_1]$ and all edges in $G[V_2]$. Thus, $G[V_1, V_2]$ is a bipartite graph and $\{V_1, V_2\}$ is a bipartition of its vertex set into independent sets.

Let G be a graph. We denote by $\mathcal{T}_{\leq 3}(G)$ the collection of all trees T with maximum degree at most 3 such that $L(T) = V(G)$. For every tree $T \in \mathcal{T}_{\leq 3}(G)$ and for every edge e of T, we denote the two connected components of $T - e$ by T_1^e, T_2^e. Then $\{L_1^e(T) = L(T) \cap V(T_1^e), L_2^e(T) = L(T) \cap V(T_2^e)\}$ is a bipartition of $L(T) = V(G)$. The *mim-width* of G is the number:

$$\min\{\max\{\mu(G[L_1^e(T), L_2^e(T)]) \mid e \in E(T)\} \mid T \in \mathcal{T}_{\leq 3}(G)\}$$

3.3.2 Graph Parameters of Chordal Graphs

For the class of chordal graphs in particular, a couple of parameters have been introduced that measure the structural complexity of a chordal graph via measuring the size of an optimal tree model of the graph.

For every chordal graph G, we denote the collection of all tree models of G by $\mathcal{M}_{tree}(G)$.

Leafage

The *leafage* of a chordal graph G is the minimum number of leaves that the host tree of a tree model of G can have.

$$\ell(G) = \min\{L(T) \mid (T, \mathcal{T}) \in \mathcal{M}_{tree}(G)\}$$

Vertex Leafage

The *vertex leafage* of a chordal graph G is the minimum maximum number of leaves that a subtree of a tree model of G can have.

$$v\ell(G) = \min\{\max\{L(T') \mid T' \in \mathcal{T}\} \mid (T, \mathcal{T}) \in \mathcal{M}_{tree}(G)\}$$

Notice that by definition $v\ell(G) \leq \ell(G)$ for every chordal graph G.

4 Subgraph Hitting Problems

As mentioned in the previous chapter, graphs lend themselves very well to modelling real world structures, be they physical or virtual. Thus, it comes at no surprise that solving problems defined on graphs tend to have various diverse real world applications. In this thesis, we consider a number of hitting problems on graphs and we consider them both as decision problems and as optimization problems.

In this chapter, we provide these problems' definitions. First, we introduce some relevant definitions and notation. We say that a set A *hits* a set B if $A \cap B \neq \varnothing$. Given a collection $\mathcal{C}$ of subsets of a set X and a set $A \subseteq X$, we say that A is a *hitting set* of $\mathcal{C}$ if A hits all elements of $\mathcal{C}$. For any collection $\mathcal{C}$ of graphs, we denote by $\mathcal{V}(\mathcal{C})$ the collection of vertex sets of all elements of $\mathcal{C}$. For every collection $\mathcal{C} \subseteq \mathcal{P}(V(G))$, we use $\max\limits_{\text{weight}} \mathcal{C}$ to denote the collection $\arg\max\{w(U) \mid U \in \mathcal{C}\}$.

4.1 Path Hitting Problems

For any graph G and $k \in \mathbb{N}$, we denote by $\mathcal{P}_k(G)$ the collection of all P_k subgraphs of G. Furthermore, for any graph G, we denote by $\mathcal{P}(G)$ the collection $\bigcup\{\mathcal{P}_k(G) \mid k \in \mathbb{N}\}$, and for any $T \subseteq V(G)$, we denote by $\mathcal{P}(G, T)$ the collection of all paths of G with endpoints in T.

4.1.1 Vertex Cover (VC)

Decision Problem

VERTEX COVER (VC)

Input: A (vertex-weighted) graph $G = (V, E)$ and a number $k \in \mathbb{N}$.
Question: Is there a set $U \subseteq V$ which is a hitting set of E of cardinality (weight) at most k?

Optimization Problem

VERTEX COVER (VC)

Input: A (vertex-weighted) graph $G = (V, E)$.
Output: A set $U \subseteq V$ which is a hitting set of E of minimum cardinality (weight).

Notice that the edge set E of a graph G is exactly the collection $\mathcal{V}(\mathcal{P}_2(G))$.

4.1.2 Node Multiway Cut (NMC)

Decision Problem

NODE MULTIWAY CUT (NMC)

Input: A (vertex-weighted) graph $G = (V, E)$, a set $T \subseteq V$ and a number $k \in \mathbb{N}$.
Question: Is there a set $U \subseteq V \setminus T$ which is a hitting set of $\mathcal{V}(\mathcal{P}(G, T))$ of cardinality (weight) at most k?

Optimization Problem

NODE MULTIWAY CUT (NMC)

Input: A (vertex-weighted) graph $G = (V, E)$ and a set $T \subseteq V$.
Output: A set $U \subseteq V \setminus T$ which is a hitting set of $\mathcal{V}(\mathcal{P}(G, T))$ of minimum cardinality (weight).

4.1.3 Unconstrained Node Multiway Cut (UNMC)

Decision Problem

Unconstrained Node Multiway Cut (UNMC)

Input: A (vertex-weighted) graph $G = (V, E)$, a set $T \subseteq V$ and a number $k \in \mathbb{N}$.

Question: Is there a set $U \subseteq V$ which is a hitting set of $\mathcal{V}(\mathcal{P}(G, T))$ of cardinality (weight) at most k?

Optimization Problem

Unconstrained Node Multiway Cut (UNMC)

Input: A (vertex-weighted) graph $G = (V, E)$ and a set $T \subseteq V$.

Output: A set $U \subseteq V$ which is a hitting set of $\mathcal{V}(\mathcal{P}(G, T))$ of minimum cardinality (weight).

4.2 Cycle Hitting Problems

For any graph G and $k \in \mathbb{N}$ such that $k \geq 3$, we denote by $\mathcal{C}_k(G)$ the collection of all C_k subgraphs of G. Furthermore, for any graph G, we denote by $\mathcal{C}(G)$ the collection $\bigcup\{\mathcal{C}_k(G) \mid k \in \mathbb{N},\ k \geq 3\}$, and for any $T \subseteq V(G)$, we denote by $\mathcal{C}(G, T)$ the collection of all cycles of G that are hit by T.

4.2.1 Feedback Vertex Set (FVS)

Decision Problem

Feedback Vertex Set (FVS)

Input: A (vertex-weighted) graph $G = (V, E)$ and a number $k \in \mathbb{N}$.

Question: Is there a set $U \subseteq V$ which is a hitting set of $\mathcal{V}(\mathcal{C}(G))$ of cardinality (weight) at most k?

Optimization Problem

FEEDBACK VERTEX SET (FVS)

Input: A (vertex-weighted) graph $G = (V, E)$.
Output: A set $U \subseteq V$ which is a hitting set of $\mathcal{V}(\mathcal{C}(G))$ of minimum cardinality (weight).

4.2.2 Subset Feedback Vertex Set (SFVS)

Decision Problem

SUBSET FEEDBACK VERTEX SET (SFVS)

Input: A (vertex-weighted) graph $G = (V, E)$, a set $T \subseteq V$ and a number $k \in \mathbb{N}$.
Question: Is there a set $U \subseteq V$ which is a hitting set of $\mathcal{V}(\mathcal{C}(G, T))$ of cardinality (weight) at most k?

Optimization Problem

SUBSET FEEDBACK VERTEX SET (SFVS)

Input: A (vertex-weighted) graph $G = (V, E)$ and a set $T \subseteq V$.
Output: A set $U \subseteq V$ which is a hitting set of $\mathcal{V}(\mathcal{C}(G, T))$ of minimum cardinality (weight).

Part II

Our Results and Open Problems

5 Preliminaries

We begin this chapter with some general mathematical definitions and notation. Then we give our definitions and notation regarding the SFVS problem. Finally, we present our dynamic programming scheme for solving SFVS on classes of graphs that exhibit a suitable structure.

For every $p \in \mathbb{N}$, we use $[p]$ and $-[p]$ to denote the sets $\{1, 2, \dots, p\}$ and $\{-1, -2, \dots, -p\}$ respectively. A binary relation over a set is called a *partial order* if it is reflexive, transitive and antisymmetric. Let X be a set and let $\leq$ be a partial order on X. For all $X' \subseteq X$, we use $\min_{\leq} X'$ and $\max_{\leq} X'$ to denote the sets of all minimal and maximal elements of X' with respect to $\leq$ respectively. Any two elements u and v of X are called *comparable* with respect to $\leq$ if $u \leq v$ or $v \leq u$; otherwise, u and v are called *incomparable* with respect to $\leq$. If $u \leq v$ and $u \neq v$, then we simply write $u < v$. A partial order on a set is called a *total order* if any two elements of the set are comparable with respect to the order. In the particular case that $\leq$ is a total order on X, we use $\min_{\leq} X'$ and $\max_{\leq} X'$ to denote the (unique) minimum and maximum element of X' with respect to $\leq$ respectively. We omit explicit textual or symbolic reference to the particular relation if it can be safely inferred from context.

5.1 Subset Forests and Subset Feedback Vertex Sets

Let G be a graph and let $S \subseteq V(G)$ be a set of terminals to be given as input to the SFVS problem. We say that

- a cycle (resp. triangle, square) is an *S-cycle* (resp. *S-triangle*, *S-square*) if it contains a vertex in S;
- a subgraph F of G is an *S-forest* of G if F does not contain an S-cycle; and
- a set $U \subseteq V(G)$ is an *S-fvs* of G if $G - U$ is an S-forest.

We use $\mathcal{F}_S$ to denote the collection of all S-forests of G. In such terms, SFVS can be restated as follows:

Subset Feedback Vertex Set (SFVS)

Input: A graph G and a set $S \subseteq V(G)$.
Output: An S-fvs of G of minimum weight.

5.2 Solving SFVS via Dynamic Programming

For solving SFVS we consider the following more convenient problem:

Induced Subset Forest (ISF)

Input: A graph G and a set $S \subseteq V(G)$.
Output: A set $U \subseteq V(G)$ of maximum weight such that $G[U] \in \mathcal{F}_S$.

Notice that a set U is a solution to ISF on (G, S) if and only if $V(G) \setminus U$ is a solution to SFVS on (G, S). On classes of graphs that exhibit a suitable structure, we solve SFVS via a dynamic programming scheme. We consider the following subproblems of ISF:

Partial Induced Subset Forest (PISF)

Input: A graph G and sets $S, X, Y \subseteq V(G)$ such that $X \cap Y \neq \varnothing$.
Output: A set $U \subseteq X$ of maximum weight such that $G[U \cup Y] \in \mathcal{F}_S$.

Observe that a set is a solution to ISF on (G, S) if and only if it is a solution to PISF on $(G, S, V(G), \varnothing)$. We use A_X^Y to denote an arbitrary solution to PISF on (G, S, X, Y). For every two expressions e_1 and e_2 involving such arbitrary solutions, we write $e_1 \leftrightarrow e_2$ to denote that the collection of all evaluations of e_1 equals the collection of all evaluations of e_2. Clearly $A_\varnothing^Y \leftrightarrow \varnothing$ for every $Y \subseteq V(G)$. For each considered graph class, we derive recursive formulas for computing A_X^Y for particular sets X and Y necessary for that class and we show that it is sufficient to compute A_X^Y only for a polynomial number of sets X and Y. Within the scope of our algorithms, we use A_X^Y to denote the particular solution to PISF on (G, S, X, Y) that we compute instead of an arbitrary one and we use $\max_{\text{weight}} \mathcal{C}$ to denote an arbitrary element of the collection $\arg\max\{w(U) \mid U \in \mathcal{C}\}$ instead of the collection itself.

In the proofs of statements appearing in subsequent chapters, we will make implicit use of the following Observation:

Observation 5.2.1. *Let $X, Y \subseteq V(G)$ such that $X \neq Y$.*

(1) For every $x \in X$, if no subset of $X \cup Y$ containing x induces an S-cycle of G, then $A_X^Y \leftrightarrow A_{X \setminus \{x\}}^Y \cup \{x\}$.

(2) For every $y \in Y$, if no subset of $X \cup Y$ containing y induces an S-cycle of G, then $A_X^Y \leftrightarrow A_X^{Y \setminus \{y\}}$.

Proof. We show the first statement. Showing the second statement is completely analogous. Let $x \in X$ such that no subset of $X \cup Y$ containing x induces an S-cycle of G. Clearly, for every $U \subseteq X$, if $G[U \cup \{x\} \cup Y] \in \mathcal{F}_S$, then $G[U \cup Y] \in \mathcal{F}_S$. Now let $U \subseteq X$ such that $G[U \cup Y] \in \mathcal{F}_S$. We show that $G[U \cup \{x\} \cup Y] \in \mathcal{F}_S$ as well. Assume for contradiction that a subset C of $U \cup \{x\} \cup Y$ induces an S-cycle of G. If $x \notin C$, then we obtain a contradiction to $G[U \cup Y] \in \mathcal{F}_S$. If $x \in C$, then we obtain a contradiction to the property of x. We conclude that no subset of $U \cup \{x\} \cup Y$ induces an S-cycle of G, completing our proof. □

6 SFVS on Subclasses of AT-free Graphs

In this chapter, we provide polynomial-time algorithms for solving SFVS on interval graphs, permutation graphs and cobipartite graphs. These are the first polynomial-time algorithms for solving SFVS on particular classes of graphs appearing in the literature. The results presented in this chapter were published in the following works:

- Charis Papadopoulos and Spyridon Tzimas. Polynomial-Time Algorithms for the Subset Feedback Vertex Set Problem on Interval Graphs and Permutation Graphs. 21st International Symposium on Fundamentals of Computation Theory (FCT 2017). *Lecture Notes in Computer Science (LNCS)*, 10472:381–394 (2017).

- Charis Papadopoulos and Spyridon Tzimas. Polynomial-time algorithms for the subset feedback vertex set problem on interval graphs and permutation graphs. *Discrete Applied Mathematics*, 258:204–221 (2019).

6.1 Interval Graphs

We begin with the case of interval graphs. An example of an interval graph is shown in Figure 6.1. Let $G = (V, E)$ be a vertex-weighted interval graph. We compute an interval model $(\mathbb{R}, \mathcal{I})$ of G. Without loss of generality, we assume that the endpoints of the n intervals in $\mathcal{I}$ are $2n$ distinct points of $\mathbb{R}$. We sort the vertices of G in ascending order of the right endpoints of their corresponding intervals in $\mathcal{I}$. To simplify our presentation, we identify the sorted vertices of G with the integers of $[n]$. For every $i \in [n]$, we denote the left and right endpoints of its corresponding interval in $\mathcal{I}$ by $\ell(i)$ and $r(i)$ respectively.

We consider the two relations on $[n]$ that are defined by the endpoints of the intervals as follows: $i \leq_\ell j \Leftrightarrow \ell(i) \leq \ell(j)$ and $i \leq_r j \Leftrightarrow r(i) \leq r(j)$. Since $\leq$ is a total order on $\mathbb{R}$, we get that $\leq_\ell$ and $\leq_r$ are total orders on $[n]$. For every $i \in [n]$, notice that $[i] = \{h \in [n] : h \leq_r i\}$ and $[n] \setminus [i] = \{x \in [n] : i <_r x\}$.

We define two different types of predecessors of the interval i with respect to $\leq_r$, which correspond to the subproblems that our dynamic programming algorithm wants to solve. These are $i - 1$ and $\lll\!\!\lhd i := \max(\{0\} \cup [i] \setminus N[i])$. Intuitively, if we consider time increasing from left to right, then $i - 1$ is the last interval that ends before i ends and $\lll\!\!\lhd i$ is the last interval that ends before i begins. For both predecessors, our definition returns 0 if and only if such an interval does not exist. For the example of Figure 6.1, denoting the red, green, blue, cyan, magenta and yellow vertices by r, g, b, c, m and y respectively, the following hold:

$$0 < c \equiv 1 < m \equiv 2 < b \equiv 3 < r \equiv 4 < y \equiv 5 < g \equiv 6$$

$$\begin{array}{lll}
[r] = \{r, b, c, m\} & r - 1 = b & \lll\!\!\lhd r = c \\
[g] = \{r, g, b, c, m, y\} & g - 1 = y & \lll\!\!\lhd g = 0 \\
[b] = \{b, c, m\} & b - 1 = m & \lll\!\!\lhd b = 0 \\
[c] = \{c\} & c - 1 = 0 & \lll\!\!\lhd c = 0 \\
[m] = \{c, m\} & m - 1 = c & \lll\!\!\lhd m = c \\
[y] = \{r, b, c, m, y\} & y - 1 = r & \lll\!\!\lhd y = b
\end{array}$$

Observation 6.1.1. *Let $i \in [n]$ and let $j \in [n] \setminus [i]$ such that $ij \in E$. Then,*

(1) $[i] = [i - 1] \cup \{i\}$ *and*

(2) $[i - 1] = [\lll\!\!\lhd j] \cup ([i - 1] \cap N(j))$.

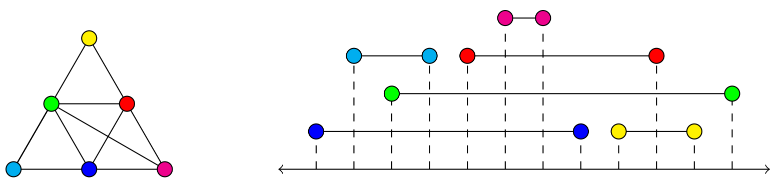


Figure 6.1: Illustration of an interval graph G and an interval model of G.

Proof. Notice that $[0] = \varnothing$. The first statement trivially holds. For the second statement, observe that $[i-1]$ can be partitioned into the set of non-neighbors and the set of neighbors of j contained in $[i-1]$. The definition of $\lll j$ implies that $[\lll j] = [j] \setminus N[j]$. Since $i < j$ and $ij \in E$, we get that $\lll j \leq i-1 < j$, so $[\lll j] \subseteq [i-1] \subset [j]$. We conclude that $[\lll j] = [i-1] \setminus N(j)$. □

Lemma 6.1.2. *Let $i \in [n]$. Then $A^{\varnothing}_{[i]} \leftrightarrow \max\limits_{\text{weight}} \left\{ A^{\varnothing}_{[i-1]}, A^{\{i\}}_{[i-1]} \cup \{i\} \right\}$.*

Proof. By Observation 6.1.1 (1), $[i] = [i-1] \cup \{i\}$. If $i \notin A^{\varnothing}_{[i]}$, then we get that $A^{\varnothing}_{[i]} \leftrightarrow A^{\varnothing}_{[i-1]}$. Otherwise, we get that $A^{\varnothing}_{[i]} \leftrightarrow A^{\{i\}}_{[i-1]} \cup \{i\}$. □

To simplify the proofs in the forthcoming lemmas, we use the following observation.

Observation 6.1.3. *Let $i \in [n]$ and let $x, y \in [n] \setminus [i]$ such that $x <_\ell y$ and $iy, xy \in E$. Then $\langle i, x, y \rangle$ is a triangle of G.*

Proof. Assuming $r(i) < \ell(y)$ results in vertices i and y being non-adjacent, a contradiction to $iy \in E$, so we have $\ell(y) < r(i)$. This inequality along with $r(i) < r(x)$ and $\ell(x) < \ell(y)$ yields $\ell(x) < r(i) < r(x)$, which implies that $ix \in E$. Therefore, $\langle i, x, y \rangle$ is a triangle of G. □

Lemma 6.1.4. *Let $i \in [n]$ and let $x \in [n] \setminus [i]$. Moreover, let $\{x', y'\} = \{i, x\}$ such that $x' <_\ell y'$.*

(1) If $ix \notin E$, then $A^{\{x\}}_{[i]} \leftrightarrow A^{\varnothing}_{[i]}$.

(2) If $ix \in E$, then
$$A^{\{x\}}_{[i]} \leftrightarrow \begin{cases} \max\limits_{\text{weight}} \left\{ A^{\{x\}}_{[i-1]}, A^{\{x'\}}_{[\lll y']} \cup \{i\} \right\}, & \text{if } i \in S \text{ or } x \in S \\ \max\limits_{\text{weight}} \left\{ A^{\{x\}}_{[i-1]}, A^{\{i,x\}}_{[i-1]} \cup \{i\} \right\}, & \text{if } i, x \notin S. \end{cases}$$

Proof. Assume first that $ix \notin E$. Then $r(i) < \ell(x)$, because we already have $r(i) < r(x)$, so x has no neighbor in $G[[i] \cup \{x\}]$. Thus no subset of $[i] \cup \{x\}$ containing x induces an S-cycle of G, implying that $A_{[i]}^{\{x\}} \leftrightarrow A_{[i]}^{\varnothing}$.

Next assume that $ix \in E$. If $i \notin A_{[i]}^{\{x\}}$, then, by Observation 6.1.1 (1), we get that $A_{[i]}^{\{x\}} \leftrightarrow A_{[i-1]}^{\{x\}}$. In what follows, let us assume that $i \in A_{[i]}^{\{x\}}$. We distinguish two cases according to whether i and x belong to S.

- Let $i \in S$ or $x \in S$. By Observation 6.1.1 (1), we have $A_{[i]}^{\{x\}} \setminus \{i\} \subseteq [i-1]$. If there exists a vertex $h \in A_{[i]}^{\{x\}} \setminus \{i\}$ such that $hy' \in E$, then, by Observation 6.1.3, $\langle h, x', y' \rangle$ is an S-triangle of G. Thus, we have $hy' \notin E$ for every vertex $h \in A_{[i]}^{\{x\}} \setminus \{i\}$. By Observation 6.1.1 (2), we get that $A_{[i]}^{\{x\}} \setminus \{i\} \subseteq [\lll y']$. Observe that the neighborhood of y' in $G[[\lll y'] \cup \{x', y'\}]$ is $\{x'\}$. Thus, no subset of $[\lll y'] \cup \{x', y'\}$ containing y' induces an S-cycle of G. We conclude that $A_{[i]}^{\{x\}} \leftrightarrow A_{[\lll y']}^{\{x'\}} \cup \{i\}$.

- Let $i, x \notin S$. By Observation 6.1.1 (1), we get $A_{[i]}^{\{x\}} \leftrightarrow A_{[i-1]}^{\{i,x\}} \cup \{i\}$.

Therefore, in every case, we obtain the desired formula. $\square$

Lemma 6.1.5. *Let $i \in [n]$ and let $x, y \in ([n] \setminus [i]) \setminus S$ such that $x <_\ell y$ and $xy \in E$. Moreover, let $\{x', y', z'\} = \{i, x, y\}$ such that $x' <_\ell y' <_\ell z'$.*

(1) If $iy \notin E$, then $A_{[i]}^{\{x,y\}} \leftrightarrow A_{[i]}^{\{x\}}$.

(2) If $iy \in E$, then
$$A_{[i]}^{\{x,y\}} \leftrightarrow \begin{cases} A_{[i-1]}^{\{x,y\}} & , \text{ if } i \in S \\ \max\limits_{\text{weight}} \left\{ A_{[i-1]}^{\{x,y\}}, A_{[i-1]}^{\{x',y'\}} \cup \{i\} \right\}, & \text{if } i \notin S. \end{cases}$$

Proof. First assume that $iy \notin E$. Then $r(i) < \ell(y)$, because we already have $r(i) < r(y)$, so the neighborhood of y in $G[[i] \cup \{x, y\}]$ is $\{x\}$. Thus, no subset of $[i] \cup \{x, y\}$ containing y induces an S-cycle of G. It follows that $A_{[i]}^{\{x,y\}} \leftrightarrow A_{[i]}^{\{x\}}$.

Next assume that $iy \in E$. By Observation 6.1.3, $\langle i, x, y \rangle$ is a triangle of G. If $i \notin A_{[i]}^{\{x,y\}}$, then, by Observation 6.1.1 (1), we have $A_{[i]}^{\{x,y\}} \leftrightarrow A_{[i-1]}^{\{x,y\}}$. If $i \in S$, then $\langle i, x, y \rangle$ is an S-triangle of G, so $i \notin A_{[i]}^{\{x,y\}}$. In what follows, we will assume that $i \notin S$ and $i \in A_{[i]}^{\{x,y\}}$ and we will show that $A_{[i]}^{\{x,y\}} \leftrightarrow A_{[i-1]}^{\{x',y'\}} \cup \{i\}$.

As $i \in A_{[i]}^{\{x,y\}}$, by Observation 6.1.1 (1), we get that $A_{[i]}^{\{x,y\}} \leftrightarrow A_{[i-1]}^{\{i,x,y\}} \cup \{i\}$. By definition, $A_{[i-1]}^{\{i,x,y\}}$ and $A_{[i-1]}^{\{x',y'\}}$ are subsets of $[i-1]$ of maximum weight such that their unions with $\{i, x, y\} = \{x', y', z'\}$ and $\{x', y'\}$ respectively induce an S-forest of G. Observe that no vertex of $\{x', y', z'\}$ belongs to S by the hypothesis on x, y and the assumption on i. Given a subset U of $[i-1]$ such that $U \cup \{x', y', z'\}$ induces an S-forest of G, it is clear that $U \cup \{x', y'\}$ induces an S-forest of G as well. Let U be a subset of $[i-1]$ such that $U \cup \{x', y'\}$ induces an S-forest of G. We show that $U \cup \{x', y', z'\}$ induces an S-forest of G. Assume for contradiction that a subset of $U \cup \{x', y', z'\}$ induces an S-triangle $\langle v_1, v_2, z' \rangle$ of G. Since $z' \notin S$, without loss of generality, assume that $v_1 \in S$. This particularly means that $v_1 \in U$, because $x', y' \notin S$ as well. By the fact that $v_1 \in [i-1]$, we have $v_1 <_r x', y', z'$. Recall that $x' <_\ell y' <_\ell z'$ and that $\langle x', y', z' \rangle$ is a triangle of G. By Observation 6.1.3, we get that $\langle v_1, x', y' \rangle$ is an S-triangle of G, a contradiction to $U \cup \{x', y'\}$ inducing an S-forest of G. We conclude that $A_{[i-1]}^{\{i,x,y\}} \leftrightarrow A_{[i-1]}^{\{x',y'\}}$. Therefore, $A_{[i]}^{\{x,y\}} \leftrightarrow A_{[i-1]}^{\{x',y'\}} \cup \{i\}$ as desired. □

Now we are equipped with the necessary tools to obtain the main result of this section, namely a polynomial-time algorithm for solving SFVS on interval graphs.

Theorem 6.1.6. *The weighted SFVS optimization problem can be solved on interval graphs in $\mathcal{O}(nm)$ time.*

Proof. We briefly describe such an algorithm based on Lemmas 6.1.2, 6.1.4, and 6.1.5. In a preprocessing step, we compute $\lll i$ for all intervals $i \in [n]$. We visit all intervals from 1 to n in ascending order with respect to $<_r$. For every interval i that we visit, we first compute $A_{[i]}^{\varnothing}$ according to Lemma 6.1.2 and then compute $A_{[i]}^{\{x\}}$ and $A_{[i]}^{\{x,y\}}$ for every $x, y \in [n] \setminus [i]$ such that $x <_\ell y$ and $xy \in E$ according to Lemmas 6.1.4 and 6.1.5 respectively. We output $[n] \setminus A_{[n]}^{\varnothing}$, as already explained. The correctness of the algorithm follows from the aforementioned Lemmas.

Regarding its running time, recall that $n \leq m$. Computing $\lll i$ for a single interval $i \in [n]$ can be done in $\mathcal{O}(m)$ time, because the intervals are sorted with respect to $<_r$. The computation of a single set A_X^Y takes constant time. Moreover, for each $i \in [n]$, the number of sets $A_{[i]}^{\{x\}}$ and $A_{[i]}^{\{x,y\}}$ being computed are at most $n + m$. Therefore, the overall running time of the algorithm is $\mathcal{O}(nm)$. □

6.2 Permutation Graphs

We continue with the case of permutation graphs. An example of a permutation graph is shown in Figure 6.2. Let $G = (V, E)$ be a vertex-weighted permutation graph such that $G = G(\pi)$ for some permutation π. Then $V = [n]$ and $E = \{ij \in \mathcal{P}_2([n]) \mid (i-j)(\pi^{-1}(i) - \pi^{-1}(j)) < 0\}$ by definition. We construct an equivalent permutation model as follows: For every $i \in [n]$, the line segment corresponding to the vertex i is the line segment with one endpoint being the i-th point of the top line and the other being the $(\pi^{-1}(i))$-th point of the bottom line.

To simplify our presentation, we also consider the line segment with endpoints being the 0-th points of both lines and extent π to include $0 \mapsto \pi(0) = 0$. The orderings of the considered line segments' endpoints on the top and bottom lines of the permutation model induce two total orders on $\{0\} \cup [n]$ which we denote by $\leq_t$ and $\leq_b$ respectively. In terms of the permutation π, these orders are defined as follows: $i \leq_t j \Leftrightarrow i \leq j$ and $i \leq_b j \Leftrightarrow \pi^{-1}(i) \leq \pi^{-1}(j)$ for all $i, j \in \{0\} \cup [n]$.

We define $\mathcal{X} := \{ii := \{i\} \mid i \in \{0\} \cup [n]\} \cup E$. We define two partial orders $\leq_\ell$ and $\leq_r$ on $\mathcal{X}$ as follows: $gh \leq_\ell ij \Leftrightarrow (g \leq_t i \text{ and } h \leq_b j)$ and $gh \leq_r ij \Leftrightarrow (g \leq_b i \text{ and } h \leq_t j)$ for all $gh, ij \in \mathcal{X}$ such that $g \leq h$ and $i \leq j$. Intuitively, every element of $\mathcal{X}$ corresponds to the union of one or two intersecting line segments and $\leq_\ell$ and $\leq_r$ correspond to partial orders with respect to their leftmost and rightmost endpoints on both lines respectively. We particularly write $<_\ell$ and $<_r$ to denote that the inequalities on *both* lines are strict. For every set $X \subseteq \{0\} \cup [n]$, we define $\mathcal{X}[X] := \{ij \in \mathcal{X} \mid i, j \in X\}$. If $X \neq \varnothing$, then it is not difficult to see that the minimum element of $\mathcal{X}[X]$ with respect to $\leq_\ell$ and its maximum element with respect to $\leq_r$ are $\min_{\leq_b} X \min_{\leq_t} X$ and $\max_{\leq_b} X \max_{\leq_t} X$ respectively and, consequently, are the unique elements of $\min_{\leq_\ell} \mathcal{X}[X]$ and $\max_{\leq_r} \mathcal{X}[X]$ respectively.

Let $ij \in \mathcal{X}$ such that $i \leq_t j$. We define $V_{ij} := \{h \in [n] \mid hh \leq_r ij\}$. We next define the predecessors $\overline{\lhd} ij$, $\unlhd ij$, $\lhd ij$ and $\lll ij$ of ij with respect to $\leq_r$ to be the (unique) elements of $\max_{\leq_r} \mathcal{X}[\{0\} \cup V_{ij} \setminus \{i\}]$, $\max_{\leq_r} \mathcal{X}[\{0\} \cup V_{ij} \setminus \{j\}]$, $\max_{\leq_r} \mathcal{X}[\{0\} \cup V_{ij} \setminus \{i,j\}]$ and $\max_{\leq_r} \mathcal{X}[\{0\} \cup V_{ij} \setminus N[\{i,j\}]]$ respectively. These are precisely the greatest predecessors of ij with respect to $\leq_r$ such that they have no element in $\{i\}$, $\{j\}$, $\{i,j\}$ and $N[\{i,j\}]$ respectively. Given $gh \in \mathcal{X}$, we also define $gh \bowtie ij$ to be the (unique) element of $\max_{\leq_r} \mathcal{X}[\{0\} \cup (V_{gh} \cap V_{ij})]$, which is the greatest common predecessor of gh and ij with respect to $\leq_r$. For the example of Figure 6.2, denoting the red, green, blue, cyan, magenta

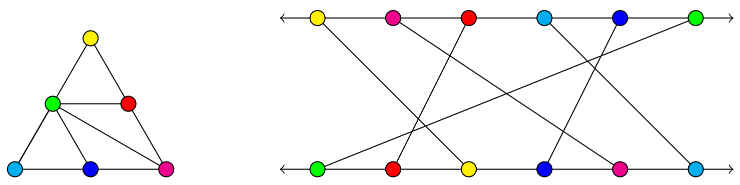


Figure 6.2: An example of a permutation graph G and a permutation model of G.

and yellow vertices by r, g, b, c, m and y respectively, the following hold:

$$0 < y \equiv 1 < m \equiv 2 < r \equiv 3 < c \equiv 4 < b \equiv 5 < g \equiv 6$$

$$\begin{array}{lll}
V_{rr} = \{r\} & V_{cg} = \{r, g, b, c, m, y\} & V_{mb} = \{r, b, m, y\} \\
V_{rg} = \{r, g\} & V_{cb} = \{r, b, c, m, y\} & V_{mm} = \{m, y\} \\
V_{gg} = \{g\} & V_{cc} = \{r, c, m, y\} & V_{yr} = \{r, y\} \\
V_{bg} = \{r, g, b, y\} & V_{mr} = \{r, m, y\} & V_{yg} = \{r, g, y\} \\
V_{bb} = \{r, b, y\} & V_{mg} = \{r, g, b, m, y\} & V_{yy} = \{y\}
\end{array}$$

$$\begin{array}{l}
\overline{\lhd}rr = \unlhd rr = \lhd rr = \lll rr = 00 \\
\overline{\lhd}gg = \unlhd gg = \lhd gg = \lll gg = 00 \\
\overline{\lhd}bb = \unlhd bb = \lhd bb = \lll bb = yr \\
\overline{\lhd}cc = \unlhd cc = \lhd cc = \lll cc = mr \\
\overline{\lhd}mm = \unlhd mm = \lhd mm = \lll mm = yy \\
\overline{\lhd}yy = \unlhd yy = \lhd yy = \lll yy = 00
\end{array}$$

$$\begin{array}{llll}
\overline{\lhd}rg = gg & \unlhd rg = rr & \lhd rg = 00 & \lll rg = 00 \\
\overline{\lhd}bg = yg & \unlhd bg = bb & \lhd bg = yr & \lll bg = 00 \\
\overline{\lhd}cg = mg & \unlhd cg = cb & \lhd cg = mb & \lll cg = 00 \\
\overline{\lhd}cb = mb & \unlhd cb = cc & \lhd cb = mr & \lll cb = yr \\
\overline{\lhd}mr = yr & \unlhd mr = mm & \lhd mr = yy & \lll mr = 00 \\
\overline{\lhd}mg = bg & \unlhd mg = mb & \lhd mg = bb & \lll mg = 00 \\
\overline{\lhd}mb = bb & \unlhd mb = mr & \lhd mb = yr & \lll mb = yy \\
\overline{\lhd}yr = rr & \unlhd yr = yy & \lhd yr = 00 & \lll yr = 00 \\
\overline{\lhd}yg = rg & \unlhd yg = yr & \lhd yg = rr & \lll yg = 00
\end{array}$$

Observation 6.2.1. *Let $gh, ij \in \mathcal{X}$ such that $i \leq_t j$. Then (1) $V_{\overline{\lhd}ij} = V_{ij} \setminus \{i\}$, (2) $V_{\unlhd ij} = V_{ij} \setminus \{j\}$, (3) $V_{\lhd ij} = V_{ij} \setminus \{i, j\}$, (4) $V_{\lll ij} = V_{ij} \setminus N[\{i, j\}]$ and (5) $V_{gh \bowtie ij} = V_{gh} \cap V_{ij}$.*

Proof. We show the first statement. The remaining statements are shown in similar fashion. By the definition of $\overline{\lhd}ij$, we have $hh \leq_r \overline{\lhd}ij$ for every vertex $h \in V_{ij} \setminus \{i\}$, so $V_{ij} \setminus \{i\} \subseteq V_{\overline{\lhd}ij}$. We show that $V_{\overline{\lhd}ij} \subseteq V_{ij} \setminus \{i\}$ as well. Let $\overline{\lhd}ij = i'j'$ such that $i' \leq_t j'$. Then $i', j' \in V_{ij} \setminus \{i\}$, which implies that $j' \leq_t j$ and $i' <_b i$. Consider a vertex $h \in V_{\overline{\lhd}ij}$. Then $h \leq_t j' \leq_t j$ and $h \leq_b i' <_b i$, which implies that $h \in V_{ij} \setminus \{i\}$. Thus $V_{\overline{\lhd}ij} \subseteq V_{ij} \setminus \{i\}$ as well. We conclude that $V_{\overline{\lhd}ij} = V_{ij} \setminus \{i\}$ as desired. $\square$

Observation 6.2.2. *Let $ij \in \mathcal{X}$ such that $i \leq_t j$ and let $x \in [n] \setminus V_{ij}$. Then*

(1) $V_{ij} = V_{\overline{\lhd}ij} \cup \{i\} = V_{\unlhd ij} \cup \{j\} = V_{\lhd ij} \cup \{i, j\}$,

(2) $V_{\lhd ij} = V_{\lll ii} \cup (V_{\lhd ij} \cap N(i)) = V_{\lll jj} \cup (V_{\lhd ij} \cap N(j))$,

(3) $V_{\lll ii} = V_{\lll ij} \cup (V_{\lll ii} \cap N(j))$, $V_{\lll jj} = V_{\lll ij} \cup (V_{\lll jj} \cap N(i))$, *and*

(4) $V_{\lhd ij} = V_{(\lhd ij) \bowtie (\lll xx)} \cup (V_{\lhd ij} \cap N(x))$.

Proof. The first statement consists an alternative formulation of the first three statements of Observation 6.2.1.

For showing the first equality of the second statement, it suffices to show that $V_{\lll ii} = V_{\lhd ij} \setminus N(i)$. Notice that $ii \leq_r ij$ implies that $V_{ii} \subseteq V_{ij} \Rightarrow$

$$\begin{aligned}
&\Rightarrow V_{ii} \setminus (\{i, j\} \cup N(i)) \subseteq V_{ij} \setminus (\{i, j\} \cup N(i)) & \\
&\Rightarrow V_{ii} \setminus N[i] \subseteq (V_{ij} \setminus \{i, j\}) \setminus N(i) & \text{(if } i <_t j, \text{ then } j \notin V_{ii}) \\
&\Rightarrow V_{\lll ii} \subseteq V_{\lhd ij} \setminus N(i). & \text{(Observation 6.2.1 (3)–(4))}
\end{aligned}$$

We show that $V_{\lhd ij} \setminus N(i) \subseteq V_{\lll ii}$ as well. Let $\lhd ij = i'j'$ such that $i' \leq_t j'$. Then $i', j' \in V_{ij} \setminus \{i, j\}$, which implies that $j' <_t j$ and $i' <_b i$. Consider a vertex $h \in V_{\lhd ij}$. Then $h \leq_t j' <_t j$ and $h \leq_b i' <_b i$. Assume that $hi \notin E$. Since we already have $h <_b i$, we get that $h <_t i$ and $h <_b i$, which implies that $h \in V_{ii} \setminus N[i] = V_{\lll ii}$. Thus $V_{\lhd ij} \setminus N(i) \subseteq V_{\lll ii}$ as well. We conclude that $V_{\lll ii} = V_{\lhd ij} \setminus N(i)$ as desired. Completely symmetrical arguments show the second equality.

For showing the first equality of the third statement, it suffices to show that $V_{\lll ij} = V_{\lll ii} \setminus N(j)$. Notice that $ii \leq_r ij$ implies that $V_{ii} \subseteq V_{ij} \Rightarrow$

$$\begin{aligned}
&\Rightarrow V_{ii} \setminus N[\{i, j\}] \subseteq V_{ij} \setminus N[\{i, j\}] & \\
&\Rightarrow (V_{ii} \setminus N[i]) \setminus N(j) \subseteq V_{ij} \setminus N[\{i, j\}] & \text{(if } i <_t j, \text{ then } j \notin V_{ii}) \\
&\Rightarrow V_{\lll ii} \setminus N(j) \subseteq V_{\lll ij}. & \text{(Observation 6.2.1 (4))}
\end{aligned}$$

We show that $V_{\lll ij} \subseteq V_{\lll ii} \setminus N(j)$ as well. Let $\lll ij = i'j'$ such that $i' \leq_t j'$. Then $i', j' \in V_{ij} \setminus N[\{i, j\}]$, which implies that $j' <_t i \leq_t j$ and $i' <_b j \leq_b i$. Consider a vertex $h \in V_{\lll ij}$. Then $h \leq_t j' <_t i \leq_t j$ and $h \leq_b i' <_b j \leq_b i$, which implies that $h \in (V_{ii} \setminus N[i]) \setminus N(j) = V_{\lll ii} \setminus N(j)$. Thus $V_{\lll ij} \subseteq V_{\lll ii} \setminus N(j)$ as well. We conclude that $V_{\lll ij} = V_{\lll ii} \setminus N(j)$ as desired. Completely symmetrical arguments show the second equality.

For showing the last statement, it suffices to show that $V_{(\lhd ij) \bowtie (\lll xx)} = V_{\lhd ij} \setminus N(x)$. Notice that $V_{\lhd ij} \cap V_{\lhd xx} \subseteq V_{\lhd ij} \Rightarrow (V_{\lhd ij} \cap V_{\lhd xx}) \setminus N(x) \subseteq V_{\lhd ij} \setminus N(x) \Rightarrow$

$$\begin{aligned}
&\Rightarrow V_{\lhd ij} \cap (V_{xx} \setminus N[x]) \subseteq V_{\lhd ij} \setminus N(x) && \text{(Observation 6.2.1 (3))}\\
&\Rightarrow V_{\lhd ij} \cap V_{\lll xx} \subseteq V_{\lhd ij} \setminus N(x) && \text{(Observation 6.2.1 (4))}\\
&\Rightarrow V_{(\lhd ij) \bowtie (\lll xx)} \subseteq V_{\lhd ij} \setminus N(x). && \text{(Observation 6.2.1 (5))}
\end{aligned}$$

We show that $V_{\lhd ij} \setminus N(x) \subseteq V_{(\lhd ij) \bowtie (\lll xx)}$ as well. Let $\lhd ij = i'j'$ such that $i' \leq_t j'$. Then $i', j' \in V_{ij} \setminus \{i, j\}$, which implies that $j' <_t j$ and $i' <_b i$. Since $x \notin V_{ij}$, we have $j <_t x$ or $i <_b x$. Without loss of generality, assume that $j <_t x$. Consider a vertex $h \in V_{\lhd ij}$. Then $h \leq_t j' <_t j$ and $h \leq_b i' <_b i$. Assume that $hx \notin E$. Since we already have $h <_t x$, we get that $h <_t x$ and $h <_b x$, which implies that $h \in V_{\lhd ij} \cap (V_{xx} \setminus N[x]) = V_{\lhd ij} \cap V_{\lll xx} = V_{(\lhd ij) \bowtie (\lll xx)}$. Thus $V_{\lhd ij} \setminus N(x) \subseteq V_{(\lhd ij) \bowtie (\lll xx)}$ as well. We conclude that $V_{(\lhd ij) \bowtie (\lll xx)} = V_{\lhd ij} \setminus N(x)$ as desired. □

Our dynamic programming algorithm for this section solves SFVS on permutation graphs by recursively solving appropriate subproblems as previously. However, it differs from our dynamic programming algorithm for the previous section in the following manner: for solving SFVS on interval graphs we consider subproblems that correspond to elements of $\{0\} \cup [n]$, whereas for solving SFVS on permutation graphs we consider subproblems that correspond to elements of $\mathcal{X}$. Recall that for solving SFVS on interval graphs it suffices to compute solutions A_X^Y to subproblems for cases of Y consisting of at most two elements of $\{0\} \cup [n]$. In the following lemma, we show that for solving SFVS on permutation graphs it suffices to compute solutions A_X^Y to subproblems for cases of Y being the union of at most two elements of $\mathcal{X}$.

Lemma 6.2.3. *Let $gh \in \mathcal{X}$, $g \leq_t h$ and let $a, b, c, d, e, f \in ([n] \setminus V_{gh}) \setminus S$ such that $ab \in E$, $a <_t b$ and $gh <_r ab$, $cd \in E$, $c <_t d$ and $gh <_r cd$, $ef \in \mathcal{X}$, $e \leq_t f$ and $\begin{cases} gh <_r ef & \text{, if } ef \in E \\ h <_t f \text{ or } g <_b e, & \text{if } e = f \end{cases}$ and $ab <_\ell cd <_\ell ef$. Then, for every $U \subseteq V_{gh}$, the following statements are equivalent:*

(i) $G[U \cup \{a,b,c,d,e,f\}] \in \mathcal{F}_S$ *(iii)* $G[U \cup \{a,b,c,d,f\}] \in \mathcal{F}_S$
(ii) $G[U \cup \{a,b,c,d,e\}] \in \mathcal{F}_S$ *(iv)* $G[U \cup \{a,b,c,d\}] \in \mathcal{F}_S$

Proof. In the context of this proof, we consider statement (i) only if $ef \in E$, statement (ii) only if $ef \in E$ or $e = f$ such that $g <_b e$, and statement (iii) only if $ef \in E$ or $e = f$ such that $h <_b f$. Notice that this is sufficient, because statements (i), (ii), and (iii) are equivalent if $e = f$. We next show all directions among the four statements.

(i) $\Rightarrow$ (ii) $\Rightarrow$ (iv) and (i) $\Rightarrow$ (iii) $\Rightarrow$ (iv). These facts are trivial, because an induced subgraph of an S-forest of G is also an S-forest of G.

(iv) $\Rightarrow$ (ii). Assume for contradiction that a subset of $U \cup \{a,b,c,d,e\}$ containing e induces an S-cycle C of G. Since every induced cycle of a permutation graph is either a triangle or a square, C is either an S-triangle or an S-square.

- Let C be the S-triangle $\langle v_1, v_2, e\rangle$. Since $e \notin S$, without loss of generality, assume that $v_1 \in S$. Then $v_1 \in U$, because $a,b,c,d \notin S$ as well. Since $gh <_r ab, cd$ and $g <_b e$, we have $v_1 \leq_t h <_t b, d$ and $v_1 \leq_b g <_b a, c, e$. For $v_1 e \in E$ to hold, we get that $e <_t v_1$. By $ab <_\ell cd <_\ell ef$, we also have $a <_t c <_t e$ and $b <_b d <_b c$. Putting it all together, we get that $a <_t c <_t v_1 <_t b$ and $v_1, b <_b a, c$, and in particular that $v_1 a, v_1 c, bc \in E$. If additionally $v_1 b \in E$ or $ac \in E$, then $\langle v_1, a, b\rangle$ or $\langle v_1, a, c\rangle$ is an S-triangle of G; otherwise, $\langle v_1, a, b, c\rangle$ is an S-square of G.
- Let C be the S-square $\langle v_1, v_2, v_3, e\rangle$. If $v_1 \in S$ or $v_3 \in S$, then we obtain that there exists a subset of $U \cup \{a,b,c,d\}$ that induces an S-cycle of G by following the exact previous argumentation. Let us assume that $v_2 \in S$. Then $v_2 \in U$. Since $gh <_r ab$ and $ab <_\ell cd <_\ell ef$, we have $v_2 \leq_t h <_t b$ and $v_2 \leq_b g <_b a$ as well as $a <_t e \leq_t f$ and $b <_b f \leq_b e$. Moreover, for $v_1 v_2, v_2 v_3, v_3 e, e v_1 \in E$ to hold, either $v_1, v_3 <_t v_2, e$ and $v_2, e <_b v_1, v_3$ hold, or $v_2, e <_t v_1, v_3$ and $v_1, v_3 <_b v_2, e$ hold. Assume the former. Then $v_1, v_3 <_t b$ and $b <_b v_1, v_3$, so $v_1 b, v_3 b \in E$. If additionally $v_2 b \in E$, then $\langle v_1, v_2, b\rangle$ is an S-triangle of G; otherwise, $\langle v_1, v_2, v_3, b\rangle$ is an S-square of G. Now assume the latter. Then $a <_t v_1, v_3$ and $v_1, v_3 <_b a$, so $v_1 a, v_3 a \in E$. If additionally $v_2 a \in E$, then $\langle v_1, v_2, a\rangle$ is an S-triangle of G; otherwise, $\langle v_1, v_2, v_3, a\rangle$ is an S-square of G.

Thus, there exists a subset of $U \cup \{a,b,c,d\}$ that induces an S-cycle of G in all cases, which is a contradiction to statement (iv). Therefore, no subset of $U \cup \{a,b,c,d,e\}$ containing e induces an S-cycle of G. This fact and statement (iv) imply statement (ii).

(iv) $\Rightarrow$ (iii). Symmetrical arguments to the previous case show this direction.

(ii) $\Rightarrow$ (i). Assume for contradiction that a subset of $U \cup \{a, b, c, d, e, f\}$ containing f induces an S-cycle C of G.

- Let C be the S-triangle $\langle v_1, v_2, f\rangle$. Since $f \notin S$, without loss of generality, assume that $v_1 \in S$. Then $v_1 \in U$, because $a, b, c, d, e \notin S$ as well. Since $gh <_r ab, cd, ef$, we have $v_1 \leq_t h <_t b, d, f$ and $v_1 \leq_b g <_b a, c, e$. For $v_1 f \in E$ to hold, we get that $f <_b v_1$. By $ab <_\ell cd <_\ell ef$, we also have $a <_t c <_t d$ and $b <_b d <_b f$. Putting it all together, we get that $v_1, a <_t b, d$ and $b <_b d <_b v_1 <_b a$, and in particular that $v_1 b, v_1 d, ad \in E$. If additionally $v_1 a \in E$ or $bd \in E$, then $\langle v_1, a, b\rangle$ or $\langle v_1, b, d\rangle$ is an S-triangle of G; otherwise, $\langle v_1, b, a, d\rangle$ is an S-square of G.

- Let C be the S-square $\langle v_1, v_2, v_3, f\rangle$. If $v_1 \in S$ or $v_3 \in S$, then we obtain that there exists a subset of $U \cup \{a, b, c, d, e\}$ that induces an S-cycle of G by following the exact previous argumentation. Let us assume that $v_2 \in S$. Then $v_2 \in U$. Since $gh <_r ab$ and $ab <_\ell cd <_\ell ef$, we have $v_2 \leq_t h <_t b$ and $v_2 \leq_b g <_b a$ as well as $a <_t e <_t f$ and $b <_b f <_b e$. Moreover, for $v_1 v_2, v_2 v_3, v_3 e, e v_1 \in E$ to hold, either $v_1, v_3 <_t v_2, f$ and $v_2, f <_b v_1, v_3$ hold, or $v_2, f <_t v_1, v_3$ and $v_1, v_3 <_b v_2, f$ hold. Assume the former. Then $v_1, v_3 <_t b$ and $b <_b v_1, v_3$, so $v_1 b, v_3 b \in E$. If additionally $v_2 b \in E$, then $\langle v_1, v_2, b\rangle$; otherwise, $\langle v_1, v_2, v_3, b\rangle$ is an S-square of G. Now assume the latter. Then $a <_t v_1, v_3$ and $v_1, v_3 <_b a$, so $v_1 a, v_3 a \in E$. If additionally $v_2 a \in E$, then $\langle v_1, v_2, a\rangle$ is an S-triangle of G; otherwise, $\langle v_1, v_2, v_3, a\rangle$ is an S-square of G.

Thus, there exists a subset of $U \cup \{a, b, c, d, e\}$ that induces an S-cycle of G in all cases, which is a contradiction to statement (ii). Therefore, no subset of $U \cup \{a, b, c, d, e, f\}$ containing f induces an S-cycle of G. This fact and statement (ii) imply statement (i).

(iii) $\Rightarrow$ (i). Symmetrical arguments to the previous case show this direction.

Notice that all other directions ((ii) $\Rightarrow$ (iii), (iii) $\Rightarrow$ (ii), and (iv) $\Rightarrow$ (i)), follow from the previous cases. Therefore, all four statements are equivalent, as desired. $\square$

The following lemmas provide recursive formulas to be used for the computation of A_X^Y in the cases that are considered by our algorithm. We first address the cases of $X = V_{ii}$, $i \in [n]$.

Lemma 6.2.4. *Let $i \in [n]$. Then $A^{\varnothing}_{V_{ii}} \leftrightarrow A^{\varnothing}_{V_{\lhd ii}} \cup \{i\}$.*

Proof. Notice that we have $h <_t i$ and $h <_b i$ for every $h \in V_{ii} \setminus \{i\}$ by definition, so i has no neighbor in $G[V_{ii}]$. This implies that no subset of V_{ii} containing i induces an S-cycle of G. By Observation 6.2.2 (1), it follows that $A^{\varnothing}_{V_{ii}} \leftrightarrow A^{\varnothing}_{V_{\lhd ii}} \cup \{i\}$. □

Lemma 6.2.5. *Let $i \in [n]$ and let $x \in [n] \setminus V_{ii}$.*

(1) If $ix \notin E$, then $A^{xx}_{V_{ii}} \leftrightarrow A^{\varnothing}_{V_{ii}}$.

(2) If $ix \in E$, then $A^{xx}_{V_{ii}} \leftrightarrow A^{xx}_{V_{\lhd ii}} \cup \{i\}$.

Proof. First assume that $ix \notin E$. Then either $i <_t x$ and $i <_b x$ hold or $x <_t i$ and $x <_b i$ hold. Since $x \notin V_{ii}$, we get that $i <_t x$ and $i <_b x$ must hold. We also have $h \leq_t i$ and $h \leq_b i$ for every $h \in V_{ii}$ by definition. We conclude that $h <_t x$ and $h <_b x$ for every $h \in V_{ii}$, so x has no neighbor in $G[V_{ii} \cup \{x\}]$. This implies that no subset of $V_{ii} \cup \{x\}$ containing x induces an S-cycle of G. Hence, $A^{xx}_{V_{ii}} \leftrightarrow A^{\varnothing}_{V_{ii}}$ follows.

Next assume that $ix \in E$. Notice that i has no neighbor in $G[V_{ii}]$, so the neighborhood of i in $G[V_{ii} \cup \{x\}]$ is $\{x\}$. This implies that no subset of $V_{ii} \cup \{x\}$ containing i induces an S-cycle of G. By Observation 6.2.2 (1), it follows that $A^{xx}_{V_{ii}} \leftrightarrow A^{xx}_{V_{\lhd ii}} \cup \{i\}$. □

Lemma 6.2.6. *Let $i \in [n]$ and let $x, y \in ([n] \setminus V_{ii}) \setminus S$ such that $xy \in E$, $x <_t y$ and $ii <_r xy$.*

(1) If $iy \notin E$, then $A^{xy}_{V_{ii}} \leftrightarrow A^{xx}_{V_{ii}}$.

(2) If $ix \notin E$, then $A^{xy}_{V_{ii}} \leftrightarrow A^{yy}_{V_{ii}}$.

(3) If $ix, iy \in E$, then $$A^{xy}_{V_{ii}} \leftrightarrow \begin{cases} A^{xy}_{V_{\lhd ii}} & \text{, if } i \in S \\ A^{xy}_{V_{\lhd ii}} \cup \{i\}, & \text{if } i \notin S. \end{cases}$$

Proof. First assume that $iy \notin E$. Then either $i <_t y$ and $i <_b y$ hold or $y <_t i$ and $y <_b i$ hold. Since $xy \in E$, $x <_t y$ and $ii <_r xy$, we have $i <_t y$ and $i <_b x$. Putting it all together, we get that $i <_t y$ and $i <_b y$ must hold. We also have $h \leq_t i$ and $h \leq_b i$ for every $h \in V_{ii}$ by definition. We conclude that $h <_t y$ and $h <_b y$ for every $h \in V_{ii}$, so the neighborhood of y in $G[V_{ii} \cup \{x, y\}]$ is $\{x\}$. This implies that no subset of $V_{ii} \cup \{x, y\}$ containing y induces an S-cycle of G. Hence, $A^{xy}_{V_{ii}} \leftrightarrow A^{xx}_{V_{ii}}$ follows. If $ix \notin E$, then completely symmetrical arguments apply in showing that $A^{xy}_{V_{ii}} \leftrightarrow A^{yy}_{V_{ii}}$.

Next assume that $ix, iy \in E$. Notice that i has no neighbor in $G[V_{ii}]$, so the neighborhood of i in $G[V_{ii} \cup \{x, y\}]$ is $\{x, y\}$. We distinguish two cases according to whether i belongs to S. Suppose that $i \in S$. Then $\langle i, x, y\rangle$ is an S-triangle of G, so $i \notin A^{xy}_{V_{ii}}$. By this fact and Observation 6.2.2 (1), it follows that $A^{xy}_{V_{ii}} \leftrightarrow A^{xy}_{V_{\lhd ii}}$.

Next suppose that $i \notin S$. We will show that no subset of $V_{ii} \cup \{x, y\}$ containing i induces an S-cycle of G. Assume for contradiction that a subset of $V_{ii} \cup \{x, y\}$ containing i induces an S-cycle C of G. Recall that every induced cycle of a permutation graph is either a triangle or a square. If C is the S-triangle $\langle v_1, v_2, i\rangle$ of G, then $\{v_1, v_2\} = \{x, y\}$, as we recall that the neighborhood of i in $G[V_{ii} \cup \{x, y\}]$ is $\{x, y\}$. However, this implies a contradiction to $\{v_1, v_2, i\}$ containing a vertex that belongs to S, because $i, x, y \notin S$. If C is the S-square $\langle v_1, v_2, v_3, i\rangle$ of G, then $\{v_1, v_3\} = \{x, y\}$, which implies a contradiction to $v_1v_3 \notin E$, because $xy \in E$. Therefore, no subset of $V_{ii} \cup \{x, y\}$ containing i induces an S-cycle of G. By this fact and Observation 6.2.2 (1), it follows that $A^{xy}_{V_{ii}} \leftrightarrow A^{xy}_{V_{\lhd ii}} \cup \{i\}$. □

Lemma 6.2.7. *Let $i \in [n]$ and let $x, y, z \in ([n] \setminus V_{ii}) \setminus S$ such that $xy \in E$, $x <_t y$ and $ii <_r xy$ and $xy <_\ell zz$.*

(1) If $iz \notin E$, then $A^{xy \cup zz}_{V_{ii}} \leftrightarrow A^{xy}_{V_{ii}}$.

(2) If $iz \in E$, then $A^{xy \cup zz}_{V_{ii}} \leftrightarrow \begin{cases} A^{xy \cup zz}_{V_{\lhd ii}} & \text{, if } i \in S \\ A^{xy \cup zz}_{V_{\lhd ii}} \cup \{i\}, & \text{if } i \notin S. \end{cases}$

Proof. First assume that $iz \notin E$. Then either $i <_t z$ and $i <_b z$ hold or $z <_t i$ and $z <_b i$ hold. Since $z \notin V_{ii}$, we get that $i <_t z$ and $i <_b z$ must hold. We also have $h \leq_t i$ and $h \leq_b i$ for every $h \in V_{ii}$ by definition. We conclude that $h <_t z$ and $h <_b z$ for every $h \in V_{ii}$, so the neighborhood of z in $G[V_{ii} \cup \{x, y, z\}]$ is a subset of $\{x, y\}$. Now completely analogous arguments as in the proof of Lemma 6.2.6 apply in showing that no subset of $V_{ii} \cup \{x, y, z\}$ containing z induces an S-cycle of G. Hence, $A^{xy \cup zz}_{V_{ii}} \leftrightarrow A^{xy}_{V_{ii}}$ follows.

Next assume that $iz \in E$. Then either $i <_t z$ and $z <_b i$ hold or $z <_t i$ and $i <_b z$ hold. Since $xy \in E$, $x <_t y$ and $ii <_r xy$ and $xy <_l zz$, we have $i, x <_t y$ and $i, y <_b x$ as well as $x <_t y, z$ and $y <_b x, z$. Putting it all together, we get that either $i, x <_t y, z$ and $y <_b z <_b i <_b x$ hold or $x <_t z <_t i <_t y$ and

$i, y <_b x, z$ hold. In particular, the former implies that $iy, xz \in E$ and the latter implies that $ix, yz \in E$. Notice that i has no neighbor in $G[V_{ii}]$, so the neighborhood of i in $G[V_{ii} \cup \{x, y, z\}]$ is a subset N of $\{x, y, z\}$. We distinguish two cases depending on whether i belongs in S.

- Let $i \in S$. We will show that $i \notin A^{xy \cup zz}_{V_{ii}}$. For the sake of contradiction, let $i \in A^{xy \cup zz}_{V_{ii}}$. If $ix, iy \in E$, then $\langle i, x, y\rangle$ is an S-triangle of G. If $ix \notin E$, then $iy, xz \in E$ must hold. If additionally $yz \in E$, then $\langle i, y, z\rangle$ is an S-triangle of G; otherwise, $\langle i, y, x, z\rangle$ is an S-square of G. The case for $iy \notin E$ is completely symmetric. We conclude that there always exists a subset of $\{i, x, y, z\}$ containing i that induces an S-cycle of G, which is a contradiction, so $i \notin A^{xy \cup zz}_{V_{ii}}$. By this fact and Observation 6.2.2 (1), it follows that $A^{xy \cup zz}_{V_{ii}} \leftrightarrow A^{xy \cup zz}_{V_{\lhd ii}}$.
- Let $i \notin S$. Given a subset U of $V_{ii} \setminus \{i\}$ such that $U \cup \{i, x, y, z\}$ induces an S-forest of G, it is clear that $U \cup \{x, y, z\}$ induces an S-forest of G as well. Let U be a subset of $V_{ii} \setminus \{i\}$ such that $U \cup \{x, y, z\}$ induces an S-forest of G. We show that $U \cup \{i, x, y, z\}$ induces an S-forest of G. Assume for contradiction that a subset of $U \cup \{i, x, y, z\}$ induces an S-cycle C of G. If $C = \langle v_1, v_2, i\rangle$, then $\{v_1, v_2\} \subseteq N$, which implies a contradiction to $\{v_1, v_2, i\}$ containing a vertex that belongs to S, because $i, x, y, z \notin S$. If $C = \langle v_1, v_2, v_3, i\rangle$, then $\{v_1, v_3\} \subseteq N$, which implies that v_2 must belong to S and, consequently, $v_2 \in U$. The case for $\{v_1, v_3\} = \{x, y\}$ implies a contradiction to $v_1v_3 \notin E$, because $xy \in E$. Assume that $\{v_1, v_3\} = \{y, z\}$. Then $yz \notin E$ and, consequently, $iy, xz \in E$ must hold. If additionally $v_2x \in E$, then $\langle y, v_2, x\rangle$ is an S-triangle of G, which implies a contradiction to $v_2 \in U$; otherwise, $\langle y, v_2, z, x\rangle$ is an S-square of G, which implies a contradiction to $v_2 \in U$ as well. The case for $\{v_1, v_3\} = \{x, z\}$ is completely symmetric. Therefore, we obtain a contradiction in all cases. By Observation 6.2.2 (1), it follows that $A^{xy \cup zz}_{V_{ii}} \leftrightarrow A^{xy \cup zz}_{V_{\lhd ii}} \cup \{i\}$.

Thus, in every case, we show that the stated formula holds, which completes our proof. □

Lemma 6.2.8. *Let $i \in [n]$ and let $x, y, z, w \in ([n] \setminus V_{ii}) \setminus S$ such that $xy \in E$, $x <_t y$, $zw \in E$, $z <_t w$ and $ii <_r xy, zw$ and $xy <_\ell zw$.*

(1) If $iw \notin E$, then $A^{xy \cup zw}_{V_{ii}} \leftrightarrow A^{xy \cup zz}_{V_{ii}}$.

(2) If $iz \notin E$, then $A^{xy \cup zw}_{V_{ii}} \leftrightarrow A^{xy \cup ww}_{V_{ii}}$.

(3) If $iz, iw \in E$, then $A_{V_{ii}}^{xy \cup zw} \leftrightarrow \begin{cases} A_{V_{\lhd ii}}^{xy \cup zw} & , \textit{ if } i \in S \\ A_{V_{\lhd ii}}^{xy \cup zw} \cup \{i\}, & \textit{if } i \notin S. \end{cases}$

Proof. First assume that $iw \notin E$. Then either $i <_t w$ and $i <_b w$ hold or $w <_t i$ and $w <_b i$ hold. Since $zw \in E$, $z <_t w$ and $ii <_r zw$, we have $i <_t z$ and $i <_b w$. Putting it all together, we get that $i <_t w$ and $i <_b w$ must hold. We also have $h \leq_t i$ and $h \leq_b i$ for every $h \in V_{ii}$ by definition. We conclude that $h <_t w$ and $h <_b w$ for every $h \in V_{ii}$, so the neighborhood of w in $G[V_{ii} \cup \{x, y, z, w\}]$ is a subset N of $\{x, y, z\}$. Given a subset U of V_{ii} such that $U \cup \{x, y, z, w\}$ induces an S-forest of G, it is clear that $U \cup \{x, y, z\}$ induces an S-forest of G as well. Let U be a subset of V_{ii} such that $U \cup \{x, y, z\}$ induces an S-forest of G. We show that $U \cup \{x, y, z, w\}$ induces an S-forest of G. Assume for contradiction that a subset of $U \cup \{x, y, z, w\}$ containing w induces an S-cycle C of G. If $C = \langle v_1, v_2, w \rangle$, then $\{v_1, v_2\} \subseteq N$, which implies a contradiction to $\{v_1, v_2, w\}$ containing a vertex that belongs to S, because $x, y, z, w \notin S$. If $C = \langle v_1, v_2, v_3, w \rangle$, then $\{v_1, v_3\} \subseteq N$, which implies that v_2 must belong to S and, consequently, $v_2 \in U$.

- The case for $\{v_1, v_3\} = \{x, y\}$ implies a contradiction to $v_1 v_3 \notin E$, because $xy \in E$.

- Assume that $\{v_1, v_3\} = \{y, z\}$. Then $yz \notin E$ and $v_2 z \in E$ must hold. Since $xy \in E$, $x <_t y$, $zw \in E$, $z <_t w$ and $ii <_r xy, zw$ and $xy <_\ell zw$, we have $i <_t y, w$ and $i <_b x, z$ as well as $x <_t z <_t w$ and $y <_b w <_b z$. For $yz \notin E$ to hold, we get that $v_2 \leq_t i <_t y <_t z$ and $v_2 \leq_b i <_b w <_b z$, and in particular that $v_2 z \notin E$, which is a contradiction.

- Assume that $\{v_1, v_3\} = \{x, z\}$. Then $xz \notin E$ and $v_2 z \in E$ must hold. Since $xy \in E$, $x <_t y$, $zw \in E$, $z <_t w$ and $ii <_r xy, zw$ and $xy <_\ell zw$, we have $i <_t y, w$ and $i <_b x, z$ as well as $x <_t z <_t w$ and $y <_b w <_b z$. For $xz \notin E$ and $v_2 z \in E$ to hold, we get that $x <_t z <_t v_2 \leq_t i <_t y$ and $v_2 \leq_b i <_b x <_b z$ and $y <_b x <_b z$, and in particular that $yz \in E$. If additionaly $v_2 y \in E$, then $\langle x, v_2, y \rangle$ is an S-triangle of G, which implies a contradiction to $v_2 \in U$; otherwise, $\langle x, v_2, z, y \rangle$ is an S-square of G, which implies a contradiction to $v_2 \in U$ as well.

Therefore, we obtain a contradiction in all cases. We conclude that $A_{V_{ii}}^{xy \cup zw} \leftrightarrow A_{V_{ii}}^{xy \cup zz}$. If $iz \notin E$, then completely symmetrical arguments apply in showing that $A_{V_{ii}}^{xy \cup zw} \leftrightarrow A_{V_{ii}}^{xy \cup ww}$.

Now assume that $iz, iw \in E$. Putting together the inequalities implied by this fact along with $xy \in E$, $x <_t y$, $zw \in E$, $z <_t w$ and $ii <_r xy, zw$ and $xy <_\ell zw$, we get that $x <_t z <_t i <_t y, w$ and $y <_b w <_b i <_b x, z$, and in particular that $ix, iy, xw, yz \in E$. Notice that i has no neighbor in $G[V_{ii}]$, so the neighborhood of i in $G[V_{ii} \cup \{x, y, z, w\}]$ is $\{x, y, z, w\}$. Assume that $i \in S$. Then $\langle i, x, y\rangle$ is an S-triangle of G, which implies that $i \notin A^{xy \cup zw}_{V_{ii}}$. By this fact and Observation 6.2.2 (1), it follows that $A^{xy\cup zw}_{V_{ii}} \leftrightarrow A^{xy\cup zw}_{V_{\lhd ii}}$. Now let us assume that $i \notin S$. Given a subset U of $V_{ii} \setminus \{i\}$ such that $U \cup \{i, x, y, z, w\}$ induces an S-forest of G, it is clear that $U \cup \{x, y, z, w\}$ induces an S-forest of G as well. Let U be a subset of $V_{ii} \setminus \{i\}$ such that $U \cup \{x, y, z, w\}$ induces an S-forest of G. We show that $U \cup \{i, x, y, z, w\}$ induces an S-forest of G. Assume for contradiction that a subset of $U \cup \{i, x, y, z, w\}$ containing i induces an S-cycle C of G.

- If $C = \langle v_1, v_2, i\rangle$, then $\{v_1, v_2\} \subset \{x, y, z, w\}$, which implies a contradiction to $\{v_1, v_2, i\}$ containing a vertex that belongs to S, because $i, x, y, z, w \notin S$.

- If $C = \langle v_1, v_2, v_3, i\rangle$, then $\{v_1, v_3\} \subset \{x, y, z, w\}$, which implies that v_2 must belong to S and, consequently, $v_2 \in U$. Moreover, $v_1v_3 \notin E$ must hold, so either $\{v_1, v_3\} = \{x, z\}$ or $\{v_1, v_3\} = \{y, w\}$. Assume that $\{v_1, v_3\} = \{x, z\}$. If additionally $v_2y \in E$, then $\langle x, v_2, y\rangle$ is an S-triangle of G, which implies a contradiction to $v_2 \in U$; otherwise, $\langle x, v_2, z, y\rangle$ is an S-square of G, which implies a contradiction to $v_2 \in U$ as well. The case for $\{v_1, v_3\} = \{y, w\}$ is completely symmetric.

Therefore, we obtain a contradiction in all cases. By Observation 6.2.2 (1), it follows that $A^{xy\cup zw}_{V_{ii}} \leftrightarrow A^{xy\cup zw}_{V_{\lhd ii}} \cup \{i\}$. □

Based on Lemmas 6.2.4–6.2.8, we can compute A^Y_X in the cases of $X = V_{ii}$, $i \in [n]$ that are considered by our algorithm. In the remaining lemmas, we address the cases of $X = V_{ij}$, $ij \in E$.

Lemma 6.2.9. *Let $ij \in E$ such that $i <_t j$. Then,*

$$A^{\varnothing}_{V_{ij}} \leftrightarrow \begin{cases} \max\limits_{\text{weight}} \left\{ A^{\varnothing}_{V_{\overline{\lhd} ij}}, A^{\varnothing}_{V_{\unlhd ij}}, A^{ii}_{V_{\lll jj}} \cup \{i, j\}, A^{jj}_{V_{\lll ii}} \cup \{i, j\} \right\}, & \text{if } i \in S \text{ or } j \in S \\ \max\limits_{\text{weight}} \left\{ A^{\varnothing}_{V_{\overline{\lhd} ij}}, A^{\varnothing}_{V_{\unlhd ij}}, A^{ij}_{V_{\lhd ij}} \cup \{i, j\} \right\} & , \text{if } i, j \notin S. \end{cases}$$

Proof. If $i \notin A^{\varnothing}_{V_{ij}}$, then it follows that $A^{\varnothing}_{V_{ij}} \leftrightarrow A^{\varnothing}_{V_{\bar{\lhd} ij}}$ by Observation 6.2.2 (1). Likewise, if $j \notin A^{\varnothing}_{V_{ij}}$, then it follows that $A^{\varnothing}_{V_{ij}} \leftrightarrow A^{\varnothing}_{V_{\trianglelefteq ij}}$. Let $i, j \in A^{\varnothing}_{V_{ij}}$. We distinguish two cases according to whether i or j belongs to S.

- Assume that $i, j \notin S$. Then $A^{\varnothing}_{V_{ij}} \leftrightarrow A^{ij}_{V_{\lhd ij}} \cup \{i, j\}$ by definition.

- Assume that $i \in S$ or $j \in S$. We show that the vertices of $A^{\varnothing}_{V_{ij}} \setminus \{i, j\}$ are all non-adjacent to i or all non-adjacent to j. Let $h \in A^{\varnothing}_{V_{ij}} \setminus \{i, j\}$ such that $hi, hj \in E$. Then $\langle h, i, j \rangle$ is an S-triangle of G, yielding a contradiction to $h, i, j \in A^{\varnothing}_{V_{ij}}$. Thus, for every $h \in A^{\varnothing}_{V_{ij}} \setminus \{i, j\}$, we have $hi \notin E$ or $hj \notin E$. Let $g, h \in A^{\varnothing}_{V_{ij}} \setminus \{i, j\}$ such that $gj, hi \in E$ and $gi, hj \notin E$. Then $g, h <_t j$ and $g, h <_b i$. For $gj, hi \in E$ and $gi, hj \notin E$ to hold, we get that $g <_t i <_t h$ and $h <_b j <_b g$, and in particular that $gh \in E$. Hence, $\langle g, h, i, j \rangle$ is an S-square of G, yielding a contradiction to $g, h, i, j \in A^{\varnothing}_{V_{ij}}$. Thus, if a vertex of $A^{\varnothing}_{V_{ij}} \setminus \{i, j\}$ is adjacent to i (resp. j), then every vertex of $A^{\varnothing}_{V_{ij}} \setminus \{i, j\}$ is non-adjacent to j (resp. i). By Observation 6.2.2 (2), it follows that $A^{\varnothing}_{V_{ij}} \setminus \{i, j\} \subseteq V_{\ll\!\lhd jj}$ or $A^{\varnothing}_{V_{ij}} \setminus \{i, j\} \subseteq V_{\ll\!\lhd ii}$. Suppose $A^{\varnothing}_{V_{ij}} \setminus \{i, j\} \subseteq V_{\ll\!\lhd jj}$. Observe that the neighborhood of j in $G[V_{\ll\!\lhd jj} \cup \{i, j\}]$ is $\{i\}$. This implies that no subset of $V_{\ll\!\lhd jj} \cup \{i, j\}$ containing j induces an S-cycle of G. We conclude that $A^{\varnothing}_{V_{ij}} \leftrightarrow A^{ii}_{V_{\ll\!\lhd jj}} \cup \{i, j\}$. Symmetrically, if $A^{\varnothing}_{V_{ij}} \setminus \{i, j\} \subseteq V_{\ll\!\lhd ii}$, then it follows that $A^{\varnothing}_{V_{ij}} \leftrightarrow A^{jj}_{V_{\ll\!\lhd ii}} \cup \{i, j\}$.

Therefore, the stated formula follows. □

Lemma 6.2.10. *Let $ij \in E$ such that $i <_t j$ and let $x \in [n] \setminus V_{ij}$. Moreover, let $\{i, j, x\} = \{x', y', z'\}$ such that $x'y' <_\ell z'z'$.*

(1) If $ix, jx \notin E$, then $A^{xx}_{V_{ij}} \leftrightarrow A^{\varnothing}_{V_{ij}}$.

(2) If $ix \in E$ and $jx \notin E$, then

$$A^{xx}_{V_{ij}} \leftrightarrow \begin{cases} \max\limits_{\text{weight}} \left\{ A^{xx}_{V_{\bar{\lhd} ij}}, A^{xx}_{V_{\trianglelefteq ij}}, A^{ii}_{V_{\ll\!\lhd jj}} \cup \{i, j\}, A^{jj}_{V_{\ll\!\lhd ix}} \cup \{i, j\} \right\}, & \textit{if } i \in S \\ & \textit{or } j \in S \\ \max\limits_{\text{weight}} \left\{ A^{xx}_{V_{\bar{\lhd} ij}}, A^{xx}_{V_{\trianglelefteq ij}}, A^{ij}_{V_{(\lhd ij) \bowtie (\ll\!\lhd xx)}} \cup \{i, j\} \right\} & , \textit{if } i, j \notin S \\ & \textit{and } x \in S \\ \max\limits_{\text{weight}} \left\{ A^{xx}_{V_{\bar{\lhd} ij}}, A^{xx}_{V_{\trianglelefteq ij}}, A^{x'y' \cup z'z'}_{V_{\lhd ij}} \cup \{i, j\} \right\} & , \textit{if } i, j, x \notin S. \end{cases}$$

(3) If $ix \notin E$ and $jx \in E$, then

$$A^{xx}_{V_{ij}} \leftrightarrow \begin{cases} \max\limits_{\text{weight}} \left\{ A^{xx}_{V_{\bar{\lhd} ij}}, A^{xx}_{V_{\unlhd ij}}, A^{ii}_{V_{\lll xj}} \cup \{i,j\}, A^{jj}_{V_{\lll ii}} \cup \{i,j\} \right\}, & \textit{if } i \in S \\ & \textit{or } j \in S \\ \max\limits_{\text{weight}} \left\{ A^{xx}_{V_{\bar{\lhd} ij}}, A^{xx}_{V_{\unlhd ij}}, A^{ij}_{V_{(\lhd ij) \bowtie (\lll xx)}} \cup \{i,j\} \right\} & , \textit{if } i,j \notin S \\ & \textit{and } x \in S \\ \max\limits_{\text{weight}} \left\{ A^{xx}_{V_{\bar{\lhd} ij}}, A^{xx}_{V_{\unlhd ij}}, A^{x'y' \cup z'z'}_{V_{\lhd ij}} \cup \{i,j\} \right\} & , \textit{if } i,j,x \notin S. \end{cases}$$

(4) If $ix, jx \in E$, then

$$A^{xx}_{V_{ij}} \leftrightarrow \begin{cases} \max\limits_{\text{weight}} \left\{ A^{xx}_{V_{\bar{\lhd} ij}}, A^{xx}_{V_{\unlhd ij}} \right\} & , \textit{if } i \in S \textit{ or } j \in S \textit{ or } x \in S \\ \max\limits_{\text{weight}} \left\{ A^{xx}_{V_{\bar{\lhd} ij}}, A^{xx}_{V_{\unlhd ij}}, A^{x'y' \cup z'z'}_{V_{\lhd ij}} \cup \{i,j\} \right\}, & \textit{if } i,j,x \notin S. \end{cases}$$

Proof. First assume that $ix, jx \notin E$. Since $ij \in E$ and $x \notin V_{ij}$, we have $i <_t j$ and $j <_b i$ as well as $j <_t x$ or $i <_b x$. For $ix, jx \notin E$ to hold, we get that $i <_t j <_t x$ and $j <_b i <_b x$. We also have $h \leq_t j$ and $h \leq_b i$ for every $h \in V_{ij}$ by definition. We conclude that $h <_t x$ and $h <_b x$ for every $h \in V_{ij}$, so x has no neighbor in $G[V_{ij} \cup \{x\}]$. This implies that no subset of $V_{ij} \cup \{x\}$ containing x induces an S-cycle of G. Hence, $A^{xx}_{V_{ij}} \leftrightarrow A^{\varnothing}_{V_{ij}}$ follows.

Next assume that $ix \in E$ or $jx \in E$. If $i \notin A^{xx}_{V_{ij}}$, then it follows that $A^{xx}_{V_{ij}} \leftrightarrow A^{xx}_{V_{\bar{\lhd} ij}}$ by Observation 6.2.2 (1). Likewise, if $j \notin A^{xx}_{V_{ij}}$, then it follows that $A^{xx}_{V_{ij}} \leftrightarrow A^{xx}_{V_{\unlhd ij}}$. Let $i, j \in A^{xx}_{V_{ij}}$. We distinguish the following cases according to whether ix or jx belongs to E.

- Assume that $ix \in E$ and $jx \notin E$. Since $x \notin V_{ij}$, we have $j <_t x$ or $i <_b x$. Let $i <_b x$. Then, for $ix \in E$ to hold, we get that $x <_t i <_t j$ and $j <_b i <_b x$, and in particular that $jx \in E$, contradicting our assumption. Thus, we have $j <_t x$. For $ix \in E$ and $jx \notin E$ to hold, we get that $i <_t j <_t x$ and $j <_b x <_b i$. We also have $h \leq_t j$ and $h \leq_b j$ for every $h \in V_{jj}$ by definition. We conclude that $h <_t x$ and $h <_b x$ for every $h \in V_{jj}$, so x has no neighbor in $G[V_{jj}]$. We further distinguish subcases depending on whether i, j or x belongs to S.
  - Let $i \in S$ or $j \in S$. We show that the vertices of $A^{xx}_{V_{ij}} \setminus \{i,j\}$ are all non-adjacent to i or all non-adjacent to j. Let $h \in A^{xx}_{V_{ij}} \setminus \{i,j\}$ such that $hi, hj \in E$. Then $\langle h, i, j \rangle$ is an S-triangle of G, yielding a

contradiction to $h, i, j \in A^{xx}_{ij}$. Thus, for every $h \in A^{xx}_{V_{ij}} \setminus \{i, j\}$, we have $hi \notin E$ or $hj \notin E$. Let $g, h \in A^{xx}_{V_{ij}} \setminus \{i, j\}$ such that $gj, hi \in E$ and $gi, hj \notin E$. Then $g, h <_t j$ and $g, h <_b i$. For $gj, hi \in E$ and $gi, hj \notin E$ to hold, we get that $g <_t i <_t h$ and $h <_b j <_b g$, and in particular that $gh \in E$. Hence, $\langle g, h, i, j\rangle$ is an S-square of G, yielding a contradiction to $g, h, i, j \in A^{xx}_{V_{ij}}$. Thus, if a vertex of $A^{\varnothing}_{V_{ij}} \setminus \{i, j\}$ is adjacent to i (resp. j), then every vertex of $A^{\varnothing}_{V_{ij}} \setminus \{i, j\}$ is non-adjacent to j (resp. i). By Observation 6.2.2 (2), it follows that $A^{xx}_{V_{ij}} \setminus \{i, j\} \subseteq V_{\ll\!\lhd jj}$ or $A^{xx}_{V_{ij}} \setminus \{i, j\} \subseteq V_{\ll\!\lhd ii}$.

First suppose $A^{xx}_{V_{ij}} \setminus \{i, j\} \subseteq V_{\ll\!\lhd jj}$. Observe that the neighborhoods of j and x in $G[V_{\ll\!\lhd jj} \cup \{i, j, x\}]$ are both $\{i\}$. This implies that no subset of $V_{\ll\!\lhd jj} \cup \{i, j, x\}$ containing j or x induces an S-cycle of G. We conclude that $A^{xx}_{V_{ij}} \leftrightarrow A^{ii}_{V_{\ll\!\lhd jj}} \cup \{i, j\}$.

Now suppose $A^{xx}_{V_{ij}} \setminus \{i, j\} \subseteq V_{\ll\!\lhd ii}$. Let $h \in A^{xx}_{V_{ij}} \setminus \{i, j\}$. Then $h <_t i$ and $h <_b i$. We show that $hx \notin E$. Assume for contradiction that $hx \in E$. Recall that $i <_t j <_t x$ and $j <_b x <_b i$. For $hx \in E$ to hold, we get that $h <_t i <_t j <_t x$ and $j <_b x <_b h <_b i$, and in particular that $hj \in E$. Hence, $\langle h, j, i, x\rangle$ is an S-square of G, yielding a contradiction to $h, i, j \in A^{xx}_{V_{ij}}$. Thus, for every $h \in A^{xx}_{V_{ij}} \setminus \{i, j\}$, we have $hx \notin E$. By Observation 6.2.2 (3), it follows that $A^{xx}_{V_{ij}} \setminus \{i, j\} \subseteq V_{\ll\!\lhd ix}$. Notice that the neighborhood of x in $G[V_{\ll\!\lhd ix} \cup \{i, j, x\}]$ is $\{i\}$, so no subset of $V_{\ll\!\lhd ix} \cup \{i, j, x\}$ containing x induces an S-cycle of G. Hence, $A^{xx}_{V_{ij}} \leftrightarrow A^{ij}_{V_{\ll\!\lhd ix}} \cup \{i, j\}$ follows. Also notice that the neighborhood of i in $G[V_{\ll\!\lhd ix} \cup \{i, j\}]$ is $\{j\}$, so no subset of $V_{\ll\!\lhd ix} \cup \{i, j\}$ containing i induces an S-cycle of G. We conclude that $A^{xx}_{V_{ij}} \leftrightarrow A^{jj}_{V_{\ll\!\lhd ix}} \cup \{i, j\}$.

– Let $i, j \notin S$ and $x \in S$. Let $h \in A^{xx}_{V_{ij}} \setminus \{i, j\}$. Then $h <_t j$ and $h <_b i$. We show that $hx \notin E$. Assume for contradiction that $hx \in E$. Recall that $i <_t j <_t x$ and $j <_b x <_b i$. For $hx \in E$ to hold, we get that $h, i <_t j <_t x$ and $j <_b x <_b h <_b i$, and in particular that $hj \in E$. If additionally $hi \in E$, then $\langle h, i, x\rangle$ is an S-triangle of G, which implies a contradiction to $h, i \in A^{xx}_{V_{ij}}$; otherwise, $\langle h, j, i, x\rangle$ is an S-square of G, which implies a contradiction to $h, i, j \in A^{xx}_{V_{ij}}$. Thus, for every $h \in A^{xx}_{V_{ij}} \setminus \{i, j\}$, we have $hx \notin E$. Notice that the neighborhood of x in $G[V_{(\lhd ij)\bowtie(\ll\!\lhd xx)} \cup \{i, j, x\}]$ is $\{i\}$, so no subset of $V_{(\lhd ij)\bowtie(\ll\!\lhd xx)} \cup \{i, j, x\}$ containing x induces an S-cycle of G. By Observation 6.2.2 (4), it follows that $A^{xx}_{V_{ij}} \leftrightarrow A^{ij}_{V_{(\lhd ij)\bowtie(\ll\!\lhd xx)}} \cup \{i, j\}$.

  - Let $i, j, x \notin S$. By Observation 6.2.2 (1), it follows that $A^{xx}_{V_{ij}} \leftrightarrow A^{x'y' \cup z'z'}_{V_{\lhd ij}} \cup \{i, j\}$.

- The case for $ix \notin E$ and $jx \in E$ is completely symmetric.
- Assume that $ix, jx \in E$. If a vertex of $\{i, x, y\}$ belongs to S, then $\langle i, j, x\rangle$ is an S-triangle of G, which implies a contradiction to $i, j \in A^{xx}_{V_{ij}}$. Thus, no vertex of $\{i, x, y\}$ belongs to S. By Observation 6.2.2 (1), it follows that $A^{xx}_{V_{ij}} \leftrightarrow A^{x'y' \cup z'z'}_{V_{\lhd ij}} \cup \{i, j\}$.

Therefore, we obtain the stated formulas in all cases. □

Lemma 6.2.11. *Let $ij \in E$ such that $i <_t j$ and let $x, y \in ([n] \setminus V_{ij}) \setminus S$ such that $xy \in E$, $x <_t y$ and $ij <_r xy$. Moreover, if $iy, jx \in E$, then let $\{i, j, x, y\} = \{x', y', z', w'\}$ such that $x'y' <_\ell z'w'$.*

(1) If $iy \notin E$, then $A^{xy}_{V_{ij}} \leftrightarrow A^{xx}_{V_{ij}}$.

(2) If $jx \notin E$, then $A^{xy}_{V_{ij}} \leftrightarrow A^{yy}_{V_{ij}}$.

(3) If $iy, jx \in E$, then

$$A^{xy}_{ij} \leftrightarrow \begin{cases} \max\limits_{\text{weight}} \left\{A^{xy}_{V_{\bar{\lhd} ij}}, A^{xy}_{V_{\unlhd ij}}\right\} & \text{, if } i \in S \text{ or } j \in S \\ \max\limits_{\text{weight}} \left\{A^{xy}_{V_{\bar{\lhd} ij}}, A^{xy}_{V_{\unlhd ij}}, A^{x'y' \cup z'w'}_{V_{\lhd ij}} \cup \{i, j\}\right\}, & \text{if } i, j \notin S. \end{cases}$$

Proof. Assume that $iy \notin E$. Since $ij <_r xy$, we have $j <_t y$ and $i <_b x$. For $iy \notin E$ to hold, we get that $i <_t j <_t y$ and $j <_b i <_b y$. We also have $h \leq_t j$ and $h \leq_b i$ for every $h \in V_{ij}$ by definition. We conclude that $h <_t y$ and $h <_b y$ for every $h \in V_{ij}$, so the neighborhood of y in $G[V_{ij} \cup \{x, y\}]$ is $\{x\}$. This implies that no subset of $V_{ij} \cup \{x, y\}$ containing y induces an S-cycle of G. Hence, $A^{xy}_{V_{ij}} \leftrightarrow A^{xx}_{V_{ij}}$ follows. If $jx \notin E$, then completely symmetrical arguments apply in showing that $A^{xy}_{V_{ij}} \leftrightarrow A^{yy}_{V_{ij}}$.

Assume that $iy, jx \in E$. If $i \notin A^{xy}_{V_{ij}}$, then it follows that $A^{xy}_{V_{ij}} \leftrightarrow A^{xy}_{V_{\bar{\lhd} ij}}$ by Observation 6.2.2 (1). Likewise, if $j \notin A^{xy}_{V_{ij}}$, then it follows that $A^{xy}_{V_{ij}} \leftrightarrow A^{xy}_{V_{\unlhd ij}}$. Let $i, j \in A^{xy}_{V_{ij}}$. We show that $i, j \notin S$. If $ix \in E$ or $jy \in E$, then $\langle i, j, x\rangle$ or $\langle i, j, y\rangle$ is a triangle, otherwise $\langle i, j, x, y\rangle$ is a square. Thus, if $i \in S$ or $j \in S$, then we get that a subset of $\{i, j, x, y\}$ induces an S-cycle of G, which is a contradiction to $i, j \in A^{xy}_{V_{ij}}$. Hence, we have $i, j \notin S$. By Observation 6.2.2 (1), it follows $A^{xy}_{V_{ij}} \leftrightarrow A^{x'y' \cup z'w'}_{V_{\lhd ij}} \cup \{i, j\}$. □

Lemma 6.2.12. *Let $ij \in E$ such that $i <_t j$ and let $x, y, z \in ([n] \setminus V_{ij}) \cup S$ such that $xy \in E$, $x <_t y$ and $ij <_r xy$ and $xy <_\ell zz$. Moreover, if $iz \in E$ or $jz \in E$, then let $\{i, j, x, y, z\} = \{x', y', z', w', a'\}$ such that $x'y' <_\ell z'w' <_\ell a'a'$.*

(1) If $iz, jz \notin E$, then $A^{xy \cup zz}_{V_{ij}} \leftrightarrow A^{xy}_{V_{ij}}$.

(2) If $iz \in E$ or $jz \in E$, then

$$A^{xy\cup zz}_{V_{ij}} \leftrightarrow \begin{cases} \max\limits_{\text{weight}} \left\{ A^{xy\cup zz}_{V_{\bar{\lhd} ij}}, A^{xy\cup zz}_{V_{\unlhd ij}} \right\} & \text{, if } i \in S \text{ or } j \in S \\ \max\limits_{\text{weight}} \left\{ A^{xy\cup zz}_{V_{\bar{\lhd} ij}}, A^{xy\cup zz}_{V_{\unlhd ij}}, A^{x'y'\cup z'w'}_{V_{\lhd ij}} \cup \{i,j\} \right\}, & \text{if } i, j \notin S. \end{cases}$$

Proof. First assume that $iz, jz \notin E$. Since $z \notin V_{ij}$, we get that $i <_t j <_t z$ and $j <_b i <_b z$. This implies that the neighborhood of z in $G[V_{ij} \cup \{x, y, z\}]$ is a subset of $\{x, y\}$. We show that no subset of $V_{ij} \cup \{x, y, z\}$ containing z induces an S-cycle of G. Assume for contradiction that a subset of $V_{ij} \cup \{x, y, z\}$ containing z induces an S-cycle C of G. If $C = \langle v_1, v_2, z\rangle$, then $\{v_1, v_2\} = \{x, y\}$, which implies a contradiction to $\{v_1, v_2, z\}$ containing a vertex that belongs to S, because $x, y, z \notin S$. If $C = \langle v_1, v_2, v_3, z\rangle$, then $\{v_1, v_3\} = \{x, y\}$, which implies a contradiction to $v_1v_3 \notin E$, because $xy \in E$. Thus, no subset of $V_{ij} \cup \{x, y, z\}$ containing z induces an S-cycle of G. Hence, $A^{xy\cup zz}_{V_{ij}} \leftrightarrow A^{xy}_{V_{ij}}$ follows.

Assume that $iz \in E$ or $jz \in E$. If $i \notin A^{xy\cup zz}_{V_{ij}}$, then it follows that $A^{xy\cup zz}_{V_{ij}} \leftrightarrow A^{xy\cup zz}_{V_{\bar{\lhd} ij}}$ by Observation 6.2.2 (1). Likewise, if $j \notin A^{xy\cup zz}_{V_{ij}}$, then it follows that $A^{xy\cup zz}_{V_{ij}} \leftrightarrow A^{xy\cup zz}_{V_{\unlhd ij}}$. Let $i, j \in A^{xy\cup zz}_{V_{ij}}$. We show that $i, j \notin S$. If $ix \in E$ or $jy \in E$, then $\langle i, j, x\rangle$ or $\langle i, j, y\rangle$ is a triangle, otherwise $\langle i, j, x, y\rangle$ is a square. Thus, if $i \in S$ or $j \in S$, then we get that a subset of $\{i, j, x, y\}$ induces an S-cycle of G, which is a contradiction to $i, j \in A^{xy\cup zz}_{V_{ij}}$. Hence, we have $i, j \notin S$. Observe that no vertex of $\{x', y', z', w', a'\}$ belongs to S. Applying Lemma 6.2.3 on $gh = \lhd ij$, $ab = x'y'$, $cd = z'w'$ and $ef = a'a'$, we get that for every subset U of $V_{\lhd ij}$, the set $U \cup \{x', y', z', w'\}$ induces an S-forest of G if and only if the set $U \cup \{x', y', z', w', a'\}$ induces an S-forest of G. By Observation 6.2.2 (1), it follows that $A^{xy\cup zz}_{V_{ij}} \leftrightarrow A^{x'y'\cup z'w'}_{V_{\lhd ij}} \cup \{i, j\}$. □

Lemma 6.2.13. *Let $ij \in E$ such that $i <_t j$ and let $x, y, z, w \in ([n] \setminus V_{ij}) \cup S$ such that $xy \in E$, $x <_t y$, $zw \in E$, $z <_t w$ and $ij <_r xy, zw$ and $xy <_\ell zw$. Moreover, if $iw, jz \in E$, then let $\{i, j, x, y, z, w\} = \{x', y', z', w', a', b'\}$ such that $x'y' <_\ell z'w' <_\ell a'b'$.*

(1) If $iw \notin E$*, then* $A^{xy \cup zw}_{V_{ij}} \leftrightarrow A^{xy \cup zz}_{V_{ij}}$.

(2) If $jz \notin E$*, then* $A^{xy \cup zw}_{V_{ij}} \leftrightarrow A^{xy \cup ww}_{V_{ij}}$.

(3) If $iw, jz \in E$*, then*

$$A^{xy \cup zw}_{V_{ij}} \leftrightarrow \begin{cases} \max\limits_{\text{weight}} \left\{ A^{xy \cup zw}_{V_{\bar{\lhd} ij}}, A^{xy \cup zw}_{V_{\unlhd ij}} \right\} & \text{, } \textit{if } i \in S \textit{ or } j \in S \\ \max\limits_{\text{weight}} \left\{ A^{xy \cup zw}_{V_{\bar{\lhd} ij}}, A^{xy \cup zw}_{V_{\unlhd ij}}, A^{x'y' \cup z'w'}_{V_{\lhd ij}} \cup \{i, j\} \right\}, & \textit{if } i, j \notin S. \end{cases}$$

Proof. First assume that $iw \notin E$. Since $ij \in E$, $i <_t j$, $xy \in E$, $x <_t y$, $zw \in E$, $z <_t w$ and $ij <_r xy, zw$ and $xy <_\ell zw$, we have $i <_t j <_t y, w$ and $j <_t i <_b x, z$ as well as $x <_t z <_t w$ and $y <_b w <_b z$. For $iw \notin E$ to hold, we get that $i <_b w$. This implies that the neighborhood of w in $G[V_{ij} \cup \{x, y, z, w\}]$ is a subset N of $\{x, y, z\}$. Given a subset U of V_{ij} such that $U \cup \{x, y, z, w\}$ induces an S-forest of G, it is clear that $U \cup \{x, y, z\}$ induces an S-forest of G as well. Let U be a subset of V_{ij} such that $U \cup \{x, y, z\}$ induces an S-forest of G. We show that $U \cup \{x, y, z, w\}$ induces an S-forest of G. Assume for contradiction that a subset of $U \cup \{x, y, z, w\}$ containing w induces an S-cycle C of G.

- Let $C = \langle v_1, v_2, w \rangle$. Then $\{v_1, v_2\} \subseteq N$, which implies a contradiction to $\{v_1, v_2, w\}$ containing a vertex that belongs to S, because $x, y, z, w \notin S$.
- Let $C = \langle v_1, v_2, v_3, w \rangle$. Then $\{v_1, v_3\} \subseteq N$. Since $x, y, z, w \notin S$, we get that v_2 must belong to S, so $v_2 \in U$.
  - The case for $\{v_1, v_3\} = \{x, y\}$ implies a contradiction to $v_1 v_3 \notin E$, because $xy \in E$.
  - Assume that $\{v_1, v_3\} = \{y, z\}$. Then $yz \notin E$ and $v_2 w \in E$ must hold. For $yz \notin E$ to hold, we get that $v_2 \leq_t j <_t y <_t z$ and $v_2 \leq_b i <_b w <_b z$, and in particular that $v_2 z \notin E$, which is a contradiction.
  - Assume that $\{v_1, v_3\} = \{x, z\}$. Then $xz \notin E$ and $v_2 z \in E$ must hold. For them to hold, we get that $x <_t z <_t v_2 \leq_t j <_t y$ and $v_2 \leq_b i <_b x <_b z$ and $y <_b x <_b z$, and in particular that $yz \in E$. If additionaly $v_2 y \in E$, then $\langle x, v_2, y \rangle$ is an S-triangle of G, which implies a contradiction to $v_2 \in U$; otherwise, $\langle x, v_2, z, y \rangle$ is an S-square of G, which implies a contradiction to $v_2 \in U$ as well.

Therefore, we obtain a contradiction in all cases. We conclude that $A_{V_{ij}}^{xy \cup zw} \leftrightarrow A_{V_{ij}}^{xy \cup zz}$. If $jz \notin E$, then completely symmetrical arguments apply in showing that $A_{V_{ij}}^{xy \cup zw} \leftrightarrow A_{V_{ij}}^{xy \cup ww}$.

Now assume that $iw, jz \in E$. If $i \notin A_{V_{ij}}^{xy \cup zw}$, then it follows that $A_{V_{ij}}^{xy \cup zw} \leftrightarrow A_{V_{\overline{\triangleleft} ij}}^{xy \cup zw}$ by Observation 6.2.2 (1). Likewise, if $j \notin A_{V_{ij}}^{xy \cup zw}$, then it follows that $A_{V_{ij}}^{xy \cup zw} \leftrightarrow A_{V_{\trianglelefteq ij}}^{xy \cup zw}$. Let $i, j \in A_{V_{ij}}^{xy \cup zw}$. We show that $i, j \notin S$. If $iz \in E$ or $jw \in E$, then $\langle i, j, z \rangle$ or $\langle i, j, w \rangle$ is a triangle, otherwise $\langle i, j, z, w \rangle$ is a square. Thus, if $i \in S$ or $j \in S$, then we get that a subset of $\{i, j, z, w\}$ induces an S-cycle of G, which is a contradiction to $i, j \in A_{V_{ij}}^{xy \cup zw}$. Hence, we have $i, j \notin S$. Observe that no vertex of $\{x', y', z', w', a', b'\}$ belongs to S. Applying Lemma 6.2.3 to $gh = \triangleleft ij$, $ab = x'y'$, $cd = z'w'$ and $ef = a'b'$, we get that for every subset U of $V_{\triangleleft ij}$, the set $U \cup \{x', y', z', w'\}$ induces an S-forest of G if and only if the set $U \cup \{x', y', z', w', a', b'\}$ induces an S-forest of G. By Observation 6.2.2 (1), it follows that $A_{V_{ij}}^{xy \cup zw} \leftrightarrow A_{V_{\triangleleft ij}}^{x'y' \cup z'w'} \cup \{i, j\}$. □

Based on Lemmas 6.2.9–6.2.13, we can compute A_X^Y in the cases of $X = V_{ij}$, $ij \in E$ that are considered by our algorithm. Observe that all recursive formulas provided by Lemmas 6.2.4–6.2.13 reduce the computation of a single set A_X^Y to the computation of solutions to strictly smaller subproblems that can also be computed via these formulas. We are now ready to present our claimed polynomial-time algorithm for solving the SFVS optimization problem on permutation graphs.

Theorem 6.2.14. *The weighted SFVS optimization problem can be solved on permutation graphs in $\mathcal{O}(m^3)$ time.*

Proof. Let us describe such an algorithm. Recall that we consider connected graphs with $n \leq m$ for the analysis of its running time. Given a permutation π, that is, the ordering of the vertices of the input graph with respect to $\leq_b$, we first construct the set $\mathcal{X}$. Observe that $|\mathcal{X}| = 1 + |[n]| + |E| = 1 + n + m$. In a preprocessing step, for every $ij \in \mathcal{X}$, we compute $\overline{\triangleleft} ij$, $\trianglelefteq ij$, $\triangleleft ij$, $\lll ij$ and $(\triangleleft ij) \bowtie (\lll xx)$ for all $x \in [n] \setminus V_{ij}$. Note that such a simple application requires $\mathcal{O}(n^2)$ time for a single $ij \in \mathcal{X}$, yielding a total running time of $\mathcal{O}(n^2 m)$ for this step. Next, we iterate over the elements of $\mathcal{X}$ according to their ascending order with respect to $\leq_r$. For each $ij \in \mathcal{X}$, we compute $A_X^{\varnothing}$ according to Lemmas 6.2.4 and 6.2.9, and we iterate over the elements of $\mathcal{X}[[n] \setminus V_{ij}]$ in descending order with respect to $\leq_\ell$. For each $xy \in \mathcal{X}[[n] \setminus V_{ij}]$, we compute $A_{V_{ij}}^{xy}$ according to Lemmas 6.2.5, 6.2.6, 6.2.10 and 6.2.11, and we

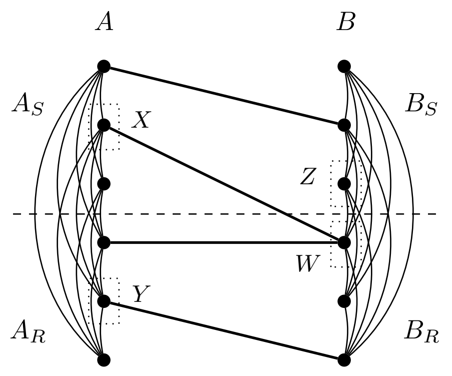


Figure 6.3: Illustrating the partition $\{X, Y, Z, W\}$ of a maximal S-forest F of the cobipartite graph G.

iterate over the elements of $\mathcal{X}[[n] \setminus V_{xy}]$ in descending order with respect to $\leq_\ell$. For each $zw \in \mathcal{X}[[n] \setminus V_{xy}]$, we compute $A^{xy \cup xw}_{V_{ij}}$ according to Lemmas 6.2.7, 6.2.8, 6.2.12 and 6.2.13. The set $A^{\varnothing}_{\pi(n)n}$ is a maximum weighted S-forest of G, so $[n] \setminus A^{\varnothing}_{\pi(n)n}$ is a minimum subset feedback vertex set of (G, S) and the output of the algorithm. Observe that the number of sets A^Y_X being computed is $\mathcal{O}(m^3)$ and the computation of a single set A^Y_X takes constant time. We conclude that the total running time of the algorithm is $\mathcal{O}(m^3)$. □

6.3 Cobipartite Graphs

Here we show that the number of minimal subset feedback vertex sets on a cobipartite graph is polynomial, which implies a polynomial-time algorithm for solving the SFVS optimization problem on the class of cobipartite graphs.

Theorem 6.3.1. *The number of maximal S-forests of a cobipartite graph is at most $16n^4$ and they can be enumerated in $\mathcal{O}(n^4)$ time.*

Proof. The key observation here is that for every clique C of G, an S-forest of G containing a vertex of $C \cap S$ contains at most two vertices of C. Let $G = (V, E)$ be a cobipartite graph and let $\{A, B\}$ be a partition of V into two cliques. We consider the partition $\{A_S, A_R, B_S, B_R\}$ of V into the four cliques $A_S = A \cap S$, $A_R = A \setminus S$, $B_S = B \cap S$ and $B_R = B \setminus S$. Let $\{X, Y, Z, W\}$ be the partition of the vertex set of a maximal S-forest F of G such that $X \subseteq A_S$, $Y \subseteq A_R$, $Z \subseteq B_S$ and $W \subseteq B_R$ (see Figure 6.3). Then we have $|X| \leq 2$ and

$|Z| \leq 2$. Moreover, if $|X| \geq 1$ (resp. $|Z| \geq 1$), then we also have $|X| + |Y| \leq 2$ (resp. $|Z| + |W| \leq 2$). By examining the nine cases corresponding to the value combinations of $|X|$ and $|Z|$, we show that there exist at most $22n^4$ maximal S-forests of G and they can be enumerated in $\mathcal{O}(n^4)$ time. For every two sets L and R, we denote by $L \triangle R$ their *symmetric difference*, which is the set $(L \setminus R) \cup (R \setminus L)$.

- Let $X = \varnothing$ and $Z = \varnothing$. Then F contains no vertex of $S = A_S \cup B_S$. No subset of $V \setminus S = A_R \cup B_R$ induces an S-cycle of G. Thus, the partition of the vertex set of F in this case is

  1. $\{X = \varnothing, Y = A_R, Z = \varnothing, W = B_R\}$.

- Let $X = \{a_S\}$ and $Z = \varnothing$. Then either $Y = \varnothing$ or $Y = \{a_R\}$. First assume that $Y = \varnothing$. Let U be a subset of B_R. Then $\{a_S\} \cup U$ induces an S-cycle of G if and only if U consists of two neighbors of a_S. We conclude that W must contain at most one neighbor of a_S, so the partition of the vertex set of F in this subcase is

  2. either $\{X = \{a_S\}, Y = \varnothing, Z = \varnothing, W = \{b_R\} \cup (B_R \setminus N(a_S))\}$,
     where $b_R \in N(a_S)$,
     or $\{X = \{a_S\}, Y = \varnothing, Z = \varnothing, W = B_R\}$,
     if no such b_R exists.

  Next assume that $Y = \{a_R\}$. Let U be a subset of B_R. Then $\{a_S\} \cup U$ induces an S-cycle of G if and only if U consists of two neighbors of a_S. Moveover, $\{a_S, a_R\} \cup U$ induces an S-cycle of G if and only if U consists of a neighbor of a_S and a (not necessarily distinct) neighbor of a_R. We conclude that W must contain at most one neighbor of a_S and that if W contains a neighbor of a_S, then W must contain no neighbor of a_R. Thus, the partition of the vertex set of F in this subcase is

  3. $\{X = \{a_S\}, Y = \{a_R\}, Z = \varnothing, W = \{b_R\} \cup (B_R \setminus N(\{a_S, a_R\}))\}$,
     where $b_R \in N(a_S) \setminus N(a_R)$;
  4. $\{X = \{a_S\}, Y = \{a_R\}, Z = \varnothing, W = B_R \setminus N(a_S)\}$.

- Let $X = \varnothing$ and $Z = \{b_S\}$. Completely symmetrical arguments apply in showing that the partition of the vertex set of F in this case is

  5. either $\{X = \varnothing, Y = \{a_R\} \cup (A_R \setminus N(b_S)), Z = \{b_S\}, W = \varnothing\}$,
     where $a_R \in N(b_S)$,
     or $\{X = \varnothing, Y = A_R, Z = \{b_S\}, W = \varnothing\}$,
     if no such a_R exists;

6. $\{X = \varnothing, Y = \{a_R\} \cup (A_R \setminus N(\{b_S, b_R\})), Z = \{b_S\}, W = \{b_R\}\}$,
   where $a_R \in N(b_S) \setminus N(b_R)$;
7. $\{X = \varnothing, Y = A_R \setminus N(b_S), Z = \{b_S\}, W = \{b_R\}\}$.

- Let $X = \{a_S\}$ and $Z = \{b_S\}$. Then either $Y = \varnothing$ or $Y = \{a_R\}$ holds as well as either $W = \varnothing$ or $W = \{b_R\}$ holds, so the partition of the vertex set of F in this case is

8. $\{X = \{a_S\}, Y = \{a_R\}, Z = \{b_S\}, W = \{b_R\}\}$,
   where a_R, b_R are such that $G[\{a_S, a_R, b_S, b_R\}]$ is an S-forest;
9. $\{X = \{a_S\}, Y = \{a_R\}, Z = \{b_S\}, W = \varnothing\}$,
   where a_R is such that $G[\{a_S, a_R, b_S\}]$ is an S-forest,
   if no $b_R \in B_R$ is such that $G[\{a_S, a_R, b_S, b_R\}]$ is an S-forest;
10. $\{X = \{a_S\}, Y = \varnothing, Z = \{b_S\}, W = \{b_R\}\}$,
    where b_R is such that $G[\{a_S, b_S, b_R\}]$ is an S-forest,
    if no $a_R \in A_R$ is such that $G[\{a_S, a_R, b_S, b_R\}]$ is an S-forest;
11. $\{X = \{a_S\}, Y = \varnothing, Z = \{b_S\}, W = \varnothing\}$,
    if no $c_R \in A_R \cup B_R$ is such that $G[\{a_S, b_S, c_R\}]$ is an S-forest.

- Let $X = \{a_S, a'_S\}$ and $Z = \varnothing$. Then $Y = \varnothing$. Let U be a subset of B_R. Then $\{a_S\} \cup U$ (resp. $\{a'_S\} \cup U$) induces an S-cycle of G if and only if U consists of two neighbors of a_S (resp. a'_S). Moreover, $\{a_S, a'_S\} \cup U$ induces an S-cycle of G if and only if U consists of a neighbor of a_S and a (not necessarily distinct) neighbor of a'_S. We conclude that W must contain at most one neighbor of a_S and at most one neighbor of a'_S, and that if W contains a neighbor of a_S, then W must contain no neighbor of a'_S, and vice versa. Thus, the partition of the vertex set of F in this case is

12. either $\{X = \{a_S, a'_S\}, Y = \varnothing, Z = \varnothing, W = \{b_R\} \cup (B_R \setminus N(\{a_S, a'_S\}))\}$,
    where $b_R \in N(a_S) \vartriangle N(a'_S)$,
    or $\{X = \{a_S, a'_S\}, Y = \varnothing, Z = \varnothing, W = B_R \setminus (N(a_S) \cap N(a'_S))\}$,
    if no such b_R exists.

- Let $X = \varnothing$ and $Z = \{b_S, b'_S\}$. Completely symmetrical arguments apply in showing that the partition of the vertex set of F in this case is

13. either $\{X = \varnothing, Y = \{a_R\} \cup (A_R \setminus N(\{b_S, b'_S\})), Z = \{b_S, b'_S\}, W = \varnothing\}$,
    where $a_R \in N(b_S) \vartriangle N(b'_S)$,
    or $\{X = \varnothing, Y = A_R \setminus (N(b_S) \cap N(b'_S)), Z = \{b_S, b'_S\}, W = \varnothing\}$,
    if no such a_R exists.

- Let $X = \{a_S, a'_S\}$ and $Z = \{b_S\}$. Then $Y = \varnothing$ holds as well as either $W = \varnothing$ or $W = \{b_R\}$ holds, so the partition of the vertex set of F in this case is

  14. either $\{X = \{a_S, a'_S\}, Y = \varnothing, Z = \{b_S\}, W = \{b_R\}\}$,
      where b_R is such that $G[\{a_S, a'_S, b_S, b_R\}]$ is an S-forest;
      or $\{X = \{a_S, a'_S\}, Y = \varnothing, Z = \{b_S\}, W = \varnothing\}$,
      if no such b_R exists.

- Let $X = \{a_S\}$ and $Z = \{b_S, b'_S\}$. Then either $Y = \varnothing$ or $Y = \{a_R\}$ holds as well as $W = \varnothing$ holds, so the partition of the vertex set of F in this case is

  15. either $\{X = \{a_S\}, Y = \{a_R\}, Z = \{b_S, b'_S\}, W = \varnothing\}$,
      where a_R is such that $G[\{a_S, a_R, b_S, b'_S\}]$ is an S-forest;
      or $\{X = \{a_S\}, Y = \varnothing, Z = \{b_S, b'_S\}, W = \varnothing\}$,
      if no such a_R exists.

- Let $X = \{a_S, a'_S\}$ and $Z = \{b_S, b'_S\}$. Then $Y = \varnothing$ and $W = \varnothing$, so the partition of the vertex set of F in this case is

  16. $\{X = \{a_S, a'_S\}, Y = \varnothing, Z = \{b_S, b'_S\}, W = \varnothing\}$.

Notice that every one of the above 16 forms of the partition $\{X, Y, Z, W\}$ of the maximal S-forest F of G involves choosing at most 4 distinct vertices of G. This implies that every one of these forms describes at most n^4 maximal S-forests of G. Therefore, the maximal S-forests of G are at most $16n^4$ in total. $\square$

7 SFVS on Subclasses of Chordal Graphs

In this chapter, we present our results for SFVS on particular subclasses of chordal graphs beyond interval graphs. The chapter is structured as follows: In Section 7.2, we make an important observation that we apply extensively in order to obtain the two polynomial-time algorithms that we provide in this chapter. In Section 7.3, we show how to transform a tree model of a chordal graph into one which satisfies a property that is required by our algorithms. In Section 7.4, we provide a polynomial-time algorithm for solving SFVS on graphs with bounded leafage and we establish W[1]-hardness of the weighted SFVS problem on chordal graphs parameterized by leafage. In Section 7.5, we provide a polynomial-time algorithm for solving SFVS on rooted path graphs and we establish NP-hardness of SFVS on undirected path graphs. The results presented in this chapter were published in the following works:

- Charis Papadopoulos and Spyridon Tzimas. Computing a Mininum Subset Feedback Vertex Set on Chordal Graphs Parameterized by Leafage. 33rd International Workshop on Combinatorial Algorithms (IWOCA 2022). *Lecture Notes in Computer Science (LNCS)*, 13270:466–479 (2022).
- Charis Papadopoulos and Spyridon Tzimas. Computing a Mininum Subset Feedback Vertex Set on Chordal Graphs Parameterized by Leafage. *Algorithmica* (2023).

7.1 Preliminaries on the Tree Model Structure

Let T be a rooted tree. We use $r(T)$ to denote its root. We assume that the edges of T are directed away from $r(T)$. We denote the unique directed path from a node s to a node t by $s \to t$. If $s \to t$ exists in T, we say that t is a *descendant* of s and that s is an *ancestor* of t. The leaves of an undirected tree T are exactly the nodes of T having degree at most one. The leaves of a rooted tree T are exactly the nodes of T having in-degree at most one and out-degree zero. For any tree T, we use $L(T)$ to denote the set of its leaves. Observe that for an undirected tree T we have $|L(T)| = 1$ if and only if T has no edges, whereas for a rooted tree T we have $|L(T)| = 1$ if and only if T is a directed path.

A binary relation defined on a set is called a *partial order* if it is transitive and anti-symmetric. Let X be a set and $\leq$ be a partial order on X. We say that two elements u and v of X are *comparable* with respect to $\leq$ if $u \leq v$ or $v \leq u$; otherwise, u and v are called *incomparable* with respect to $\leq$. If $u \leq v$ and $u \neq v$, then we simply write $u < v$. For all $X' \subseteq X$, we write $\min_{\leq} X'$ and $\max_{\leq} X'$ to denote the sets of all minimal and maximal elements of X' with respect to $\leq$ respectively. Given a rooted tree $T = (V_T, E_T)$, we define a partial order on the nodes of T as follows: for every $x, y \in V_T$, $x \leq_T y \Leftrightarrow y \to x$ exists in T. Regarding $\leq_T$ we make the following observations.

Observation 7.1.1. *Let $T = (V_T, E_T)$ be a rooted tree. For every $x, y, y' \in V_T$, if $x \leq_T y$ and $x \leq_T y'$, then y and y' are comparable with respect to $\leq_T$.*

Proof. Let $x \leq_T y$ and $x \leq_T y'$. Then by definition $y \to x$ and $y' \to x$ exist in T. Since T is a rooted tree, every node has at most one parent in T. By induction, for every $k \in \mathbb{N}$, every node has at most one ancestor at distance k in T. We set y_c and y_f to be the nodes among y, y' which are the closest and the farthest away from x respectively. Then observe that $y_f \to x$ contains y_c and in particular $y_f \to y_c$ exists in T, which implies that $y_c \leq_T y_f$ by definition, so y and y' are comparable with respect to $\leq_T$. □

Observation 7.1.2. *Let T be a rooted tree and let T' be a subtree of T. For every $l, r \in V(T')$ such that $l < r$, every node $b \in V(T)$ such that $l < b < r$ is also in $V(T')$.*

Proof. By definition, $l < b < r$ implies that $r \to b$ and $b \to l$ exist in T. In other words, $r \to l$ exists in T and contains b. Since T' is connected and

$l, r \in V(T')$, we conclude that all the nodes in $r \to l$ are in $V(T')$. In particular $b \in V(T')$. $\square$

Observation 7.1.3. *Let T be a rooted tree and let V be a set of pairwise incomparable nodes of T with respect to $\leq_T$. Then $|V| \leq |L(T)|$.*

Proof. Notice that for every node x of T, there exists a leaf l of T such that $l \leq_T x$. Assume that $|L(T)| < |V|$. Then there exists a leaf l of T and two distinct nodes $x, y \in V$ such that $l \leq_T x$, and $l \leq_T y$. By Observation 7.1.1, the nodes x and y are comparable with respect to $\leq_T$, a contradiction. We conclude that $|V| \leq |L(T)|$. $\square$

Leafage and vertex leafage A *tree model* of a graph $G = (V_G, E_G)$ is a pair $(T, \{T_v\}_{v \in V_G})$ such that (1) T is a tree, called a *host tree*[1], (2) for each $v \in V_G$, T_v is a subtree of T, and (3) for each $u, v \in V_G$ such that $u \neq v$, $uv \in E_G$ if and only if $V(T_u) \cap V(T_v) \neq \varnothing$. It is known that a graph is chordal if and only if it admits a tree model [16, 39]. The tree model of a chordal graph is not necessarily unique. The *leafage* of a chordal graph G, denoted by $\ell(G)$, is the minimum number of leaves of the host tree among all tree models of G, that is, $\ell(G)$ is the smallest integer ℓ such that there exists a tree model $(T, \{T_v\}_{v \in V_G})$ of G with $|L(T)| = \ell$ [59]. Moreover, every chordal graph G admits a tree model for which its host tree T has the minimum $|L(T)|$ and $|V(T)| \leq n$ [19, 43]; such a tree model can be constructed in $\mathcal{O}(n^3)$ time [43]. Thus the leafage $\ell(G)$ of a chordal graph G is computable in polynomial time.

A relaxation of the leafage is the *vertex leafage* introduced by Chaplick and Stacho [19]. The vertex leafage of a chordal graph G, denoted by $v\ell(G)$, is the smallest integer $v\ell$ such that there exists a tree model $(T, \{T_v\}_{v \in V_G})$ of G where $|L(T_v)| \leq v\ell$ for all $v \in V_G$. Clearly, we have $v\ell(G) \leq \ell(G)$. Unlike the leafage, deciding whether the vertex leafage of a chordal graph is at most $v\ell$ is NP-complete for every fixed integer $v\ell \geq 3$ [19].

Notice that for any tree model $(T, \{T_v\}_{v \in V_G})$ and for any $\ell, v\ell \in \mathbb{N}^*$ such that $|L(T)| \leq \ell$ and $|L(T_v)| \leq v\ell$ for all $v \in V_G$, after rooting T in an arbitrary node, the same conditions still hold. Henceforth, we will only consider tree models with host trees that are rooted trees unless otherwise stated. Under these terms, observe that (1) $\ell(G) \leq 1 \Leftrightarrow G$ is an interval graph[2], (2) $v\ell(G) \leq 1 \Leftrightarrow G$

[1]The host tree is also known as a *clique tree*, usually when we are concerned with the maximal cliques of a chordal graph [39].

[2]If the host tree T is an undirected tree, then $\ell(G) \leq 2 \Leftrightarrow G$ is an interval graph [19].

is a rooted path graph, and (3) $v\ell(G) \leq 2$ if G is an undirected path graph.

7.2 Preliminaries for Solving SFVS

Let $G = (V_G, E_G)$ be a chordal graph, let $S \subseteq V_G$ and let $X, Y \subseteq V_G$ such that $X \cap Y = \varnothing$ and $G[Y] \in \mathcal{F}_S$. A partition $\mathcal{P}$ of X is called *nice* if for any S-triangle S_t of $G[X \cup Y]$, there is a part $P \in \mathcal{P}$ such that $V(S_t) \cap X \subseteq P$. In other words, any S-triangle of $G[X \cup Y]$ is involved with at most one part of any nice partition of X. With respect to the defined optimal solutions A_X^Y, we observe the following:

Observation 7.2.1. *Let $G = (V_G, E_G)$ be a chordal graph, let $S \subseteq V_G$ and let $X, Y \subseteq V_G$ such that $X \cap Y = \varnothing$ and $G[Y] \in \mathcal{F}_S$. Then the following hold:*

(1) $A_X^Y \leftrightarrow \bigcup_{P \in \mathcal{P}} A_P^Y$ for any nice partition $\mathcal{P}$ of X.

(2) $A_X^Y \leftrightarrow A_X^{Y'}$ where $Y' = Y \cap N(X')$ for any $X \supseteq X' \supseteq X \setminus \{u \in X \setminus S \mid Y \cap N(u) \subseteq Y \setminus S\}$.

Proof. For the first statement, assume that there is an S-triangle S_t in $G[X \cup Y]$. Then it must contain a vertex of some part P of $\mathcal{P}$, as $G[Y]$ is an S-forest. By the definition of a nice partition, we have $V(S_t) \cap X \subseteq P$. Therefore, we deduce $A_X^Y \cap P \leftrightarrow A_P^Y$, which shows the claim.

For the second statement, observe that $G[Y'] \in \mathcal{F}_S$, as $Y' \subseteq Y$ and $G[Y] \in \mathcal{F}_S$. Also, notice that any S-triangle in $G[X \cup Y']$ remains an S-triangle in $G[X \cup Y]$. Consider an S-triangle in $G[X \cup Y]$ induced by $\{x, y, z\}$ where $x \in X$ and $y \in Y$. We show that $y \in Y'$ and $z \in X \cup Y'$. If $x \in X'$, then $y \in Y'$ and $z \in X \cup Y'$ by the fact that $Y' = Y \cap N(X')$. Suppose that $x \in X \setminus S$ such that $Y \cap N(x) \subseteq Y \setminus S$. Then $y \in Y \setminus S$ and $z \in X \cup (Y \setminus S)$. This means that z must be in S and in particular $z \in X \cap S \subseteq X'$. By the fact that $Y' = Y \cap N(X')$, we conclude that $y \in Y'$. Thus, any S-triangle in $G[X \cup Y]$ remains an S-triangle in $G[X \cup Y']$, which concludes the proof. $\square$

Observation 7.2.1 suggests how to reduce the computation of A_X^Y to the computation of optimal solutions to smaller instances. More precisely, by Observation 7.2.1 (1), if we obtain a nice partition $\mathcal{P}$ of the vertex set X, then we can reduce the computation of A_X^Y to the computation of A_P^Y for every $P \in$

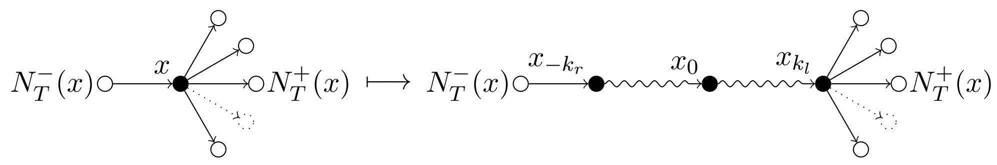


Figure 7.1: We replace node x of T by the directed path $\langle x_{-k_r}, \dots, x_0, \dots, x_{k_l}\rangle$ such that in T' the node of $N_T^-(x)$ points to x_{-k_r} and all nodes of $N_T^+(x)$ are pointed by x_{k_l} instead.

$\mathcal{P}$, which are optimal solutions to smaller and pairwise-independent instances, and Observation 7.2.1 (2) states that for computing A_X^Y, it is sufficient to consider only the vertices y of Y which have neighbours x in X such that at least one of x and y is in S.

7.3 Expanded Tree Model

Given a tree model of a chordal graph, we are interested in defining a partial order on the vertices of the graph that takes advantage of the underlying tree structure. For this purpose, it is necessary that each of the subtrees of the tree model corresponds to at most one vertex of the graph. Here we show how a tree model can be altered in order to obtain this property in a formal way. Assume that G is a chordal graph.

Definition 7.3.1. A tree model $(T, \{T_v\}_{v\in V_G})$ of G is called *expanded tree model* if the sets of the collection $\{\{r(T_v)\}\}_{v\in V_G} \cup \{L(T_v)\}_{v\in V_G}$ are paiwise-disjoint.

We show that for any tree model M of a chordal graph G, there exists an expanded tree model M' of G that is structurally close to M. In fact, we provide an algorithm that, given a tree model of G, constructs such an expanded tree model of G.

Lemma 7.3.1. *For any tree model $(T, \{T_v\}_{v\in V_G})$ of G and for any $\ell, v\ell \in \mathbb{N}^*$ such that $|L(T)| = \ell$ and $|L(T_v)| \le v\ell$ for all $v \in V_G$, there is an expanded tree model $(T', \{T'_v\}_{v\in V_G})$ of G such that:*

- $|L(T')| = |L(T)| = \ell$ *and* $|L(T'_v)| = |L(T_v)| \le v\ell$ *for all* $v \in V_G$*, and*
- $|V(T')| \le |V(T)| + (1 + v\ell)n.$

Moreover, given $(T, \{T_v\}_{v\in V_G})$, the expanded tree model can be constructed in time $\mathcal{O}(n^2)$.

Proof. Consider a node x of T. Assume that x is the root of k_r subtrees $T_{v_{-1}}, \dots, T_{v_{-k_r}}$ and a leaf of k_l subtrees $T_{v_1}, \dots, T_{v_{k_l}}$ of $\{T_v\}_{v\in V_G}$ where $k_r + k_l \geq 2$. We replace the node x in T by the gadget shown in Figure 7.1. We also modify every subtree T_v of $\{T_v\}_{v\in V_G}$ as follows:

- If there exists an $i \in -[k_r]$ such that $T_v = T_{v_i}$ and $T_v \neq T_{v_j}$ for all $j \in [k_l]$, then we replace x in T_v by the part of the gadget involving the vertices $x_i, \dots, x_0, \dots, x_{k_l}$.
- If $T_v \neq T_{v_i}$ for all $i \in -[k_r]$ and there exists a $j \in [k_l]$ such that $T_v = T_{v_j}$, then we replace x in T_v by the part of the gadget involving the vertices $x_{-k_r}, \dots, x_0, \dots, x_j$.
- If there exists an $i \in -[k_r]$ such that $T_v = T_{v_i}$ and a $j \in [k_l]$ such that $T_v = T_{v_j}$, then we replace x in T_v by the part of the gadget involving the vertices $x_i, \dots, x_0, \dots, x_j$.
- If $T_v \neq T_{v_i}$ for all $i \in -[k_r]$ and $T_v \neq T_{v_j}$ for all $j \in [k_l]$, then
  - if x is a (necessarilly internal) node of T_v, we replace x in T_v by the whole gadget, otherwise $T'_v = T_v$.

To see that $(T', \{T'_v\}_{v\in V_G})$ is indeed a tree model of G, observe that for every $T_u, T_w \in \{T_v\}_{v\in V_G}$:

- if $x \in V(T_u) \cap V(T_w)$, then $x_0 \in V(T'_u) \cap V(T'_w)$, and
- if $x \notin V(T_u) \cap V(T_w)$, then $x_{-k_l}, \dots, x_{k_r} \notin V(T'_u) \cap V(T'_w)$.

Thus the intersection graph of $(T', \{T'_v\}_{v\in V_G})$ is isomorphic to G. Observe that among the sets $\{r(T_{v_{-k_r}})\}, \dots, \{r(T_{v_{-1}})\}, L(T_{v_1}), \dots, L(T_{v_{k_l}})$, the node x_i is only in $\{r(T_{v_i})\}$ for all $i \in -[k_r]$ and only in $L(T_{v_i})$ for all $i \in [k_r]$. Iteratively applying the above modifications to $(T, \{T_v\}_{v\in V_G})$ results in an tree model of G that satisfies Definition 7.3.1 for being an expanded tree model.

Observe that the iterative procedure described above preserves the number of leaves of the tree and of all subtrees in the collection of the tree model. Let us now bound $|V(T')|$. Recall that every subtree in $\{T_v\}_{v\in V_G}$ has at most $v\ell$ leaves. In the worst case, every subtree in $\{T_v\}_{v\in V_G}$ has exactly $v\ell$ leaves and

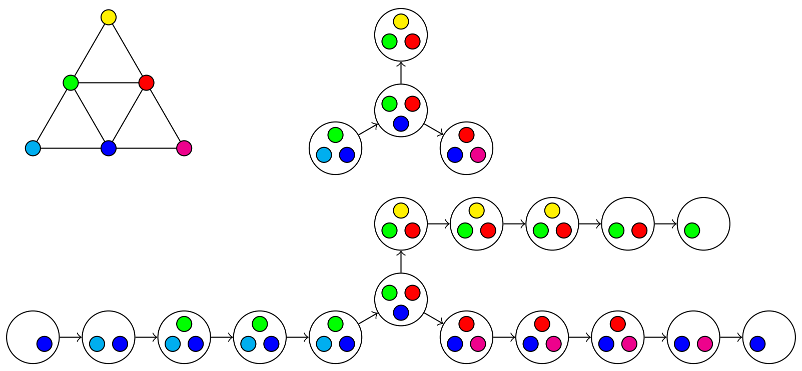


Figure 7.2: Illustration of a chordal graph G (top left), a tree model of G (top center) and an expanded tree model of G obtained via the iterative procedure described in the proof of Lemma 7.3.1 (bottom). The paths of the two models corresponding to each vertex of G are the ones formed from the nodes containing the color of that vertex.

no node of T is the root of one subtree and a leaf of no subtree of $\{T_v\}_{v \in V_G}$ or vice versa. In this case, the iterative procedure described above will add $\sum_{v \in V_G}(|\{r(T_v)\}| + |L(T_v)|) = (1 + v\ell)n$ nodes to T. We conclude that the procedure adds at most $(1 + v\ell)n$ nodes to T as well as to the n subtrees in $\{T_v\}_{v \in V_G}$, resulting in the total running time of $\mathcal{O}(n^2)$. □

An example of an expanded tree model produced by the iterative procedure described in the proof of Lemma 7.3.1 is shown in Figure 7.2. Hereafter we assume that $(T, \{T_v\}_{v \in V_G})$ is an expanded tree model of G. For any vertex u of G, we denote the node $r(T_u)$ by $r(u)$ for simplicity. We define the following partial order on the vertices of G: for all $u, v \in V_G$, $u \leq_G v \Leftrightarrow r(u) \leq_T r(v)$. In other words, two vertices of G are comparable with respect to $\leq_G$ if and only if there is a directed path between their corresponding roots in T. Since we defined $\leq_G$ and $\leq_T$ on disjoint sets, we will subsequently omit mentioning the relevant partial order explicitly.

Observation 7.3.2. *Let $u, v, w, z \in V_G$. Then, the following hold:*

(1) If $uv \in E_G$, then u and v are comparable.

(2) If $u \leq v$, $z \leq w$, and u and z are comparable, then v and w are comparable.

(3) If $u < v < w$ and $uw \in E_G$, then $vw \in E_G$.

Proof. For the first statement, assume that $x \in V(T_u) \cap V(T_v)$, which exists as $uv \in E_G$. Then the paths $r(u) \to x$ and $r(v) \to x$ exist in T. Equivalently, $x \leq r(u)$ and $x \leq r(v)$ hold. By Observation 7.1.1, we get that $r(u)$ and $r(v)$ are comparable, which implies that u and v are also comparable.

For the second statement, assume first that $u \leq z$. Then $r(u) \leq r(z) \leq r(w)$ because also $z \leq w$. Additionally $r(u) \leq r(v)$ because $u \leq v$. Just as before, by Observation 7.1.1, we get that $r(v)$ and $r(w)$ are comparable, which implies that v and w are also comparable. The case for $z \leq u$ is completely symmetrical.

For the third statement, observe that $u < v < w$ implies that $r(u) < r(v) < r(w)$. We show that $r(v) \in V(T_w)$. Since u and w are adjacent, there exists a node $x \in V(T_u) \cap V(T_w)$. Then the paths $r(u) \to x$ and $r(w) \to x$ exist in T, implying that $x \leq r(u)$ and $x \leq r(w)$. Since $x, r(w) \in V(T_w)$ and $x < r(v) < r(w)$, by Observation 7.1.2, we get that $r(v) \in V(T_w)$, so v and w are adjacent. $\square$

For all $u \in V_G$, we define V_u to be the set $\{u' \in V_G \mid u' \leq u\}$. We also define $\lhd u$ to be the set $\max\{u' \in V_G \mid u' < u\} = \max(V_u \setminus \{u\})$. Moreover, for all $uv \in E_G$, we define $\lhd uv$ to be the set $\max\{u' \in V_G \mid u' < u, v \text{ and } (u'u \notin E_G \text{ or } u'v \notin E_G)\} = \max((V_u \cap V_v) \setminus (N[u] \cap N[v]))$. Recall that for any edge $uv \in E_G$, either $u < v$ or $v < u$ by Observation 7.3.2 (1). If $u < v$ holds, then $\lhd uv = \max(V_u \setminus (N[u] \cap N(v)))$. For all $U \subseteq V_G$, we define $\mathcal{V}_U$ to be the collection $\{V_u\}_{u \in U}$. For the example of Figure 7.2, denoting the red, green, blue, cyan, magenta and yellow vertices by r, g, b, c, m and y respectively, the following hold:

$$\begin{array}{llll}
V_r = \{r, m, y\} & \lhd r = \{m, y\} & & \lhd gb = \{m, y\} \\
V_g = \{r, g, m, y\} & \lhd g = \{r\} & \lhd rg = \{m\} & \lhd gc = \{r\} \\
V_b = \{r, g, b, c, m, y\} & \lhd b = \{c\} & \lhd rb = \{y\} & \lhd gy = \varnothing \\
V_c = \{r, g, c, m, y\} & \lhd c = \{g\} & \lhd rm = \varnothing & \\
V_m = \{m\} & \lhd m = \varnothing & \lhd ry = \varnothing & \lhd bc = \{r\} \\
V_y = \{y\} & \lhd y = \varnothing & & \lhd bm = \varnothing
\end{array}$$

Having defined all the primary components, we can now provide a brief outline of our dynamic programming algorithms.

Step 1: Construction of expanded tree model. From a tree model of the chordal graph G that we are given as input, we produce an expanded tree model $(T, \mathcal{T})$ as described in the proof of Lemma 7.3.1.

Step 2: Computation of auxiliary vertex sets. Traversing T from its leaves to its root, upon reaching each node $v \in V_T$, if there exists a vertex $u \in V_G$ such that $r(u) = v$, then we compute the set $\lhd u$. Similarly for all other auxiliary vertex sets that are necessary for each algorithm.

Step 3: Computation of optimal solutions to subproblems. Traversing T again from its leaves to its root, upon reaching each node $v \in V_T$, if there exists a vertex $u \in V_G$ such that $r(u) = v$, then we compute the optimal solution $A^{\varnothing}_{V_u}$ from previously computed optimal solutions to smaller subproblems. Similarly for optimal solutions to all other subproblems that are necessary for each algorithm.

The following two lemmas provide nice partitions of X, to be used in the application of Observation 7.2.1 (1), in certain cases of X that are considered by both our algorithms.

Lemma 7.3.3. *For every $u \in V_G$, the collection $\mathcal{V}_{\lhd u}$ is a partition of $V_u \setminus \{u\}$ into pairwise disconnected sets. For every $u, v \in V_G$ such that $u < v$ and $uv \in E_G$, the collection $\mathcal{V}_{\lhd uv}$ is a partition of $V_u \setminus (N[u] \cap N(v))$ into pairwise disconnected sets.*

Proof. We prove the first statement. The proof of the second statement is completely analogous. Firstly notice that, by definition, the vertices of $\lhd u$ are pairwise incomparable. Consider two vertices u'_1 and u'_2 such that $u'_1 \leq u_1$ and $u'_2 \leq u_2$ where u_1 and u_2 are two vertices of $\lhd u$. Clearly, $u'_1 \in V_{u_1}$ and $u'_2 \in V_{u_2}$. By Observation 7.1.1 and Observation 7.3.2 (1–2), it follows that the vertices u'_1 and u'_2 are distinct and non-adjacent. □

Lemma 7.3.4. *For every $u \in V_G$, the collection $\mathcal{V}_{\lhd u}$ is a nice partition of $V_u \setminus \{u\}$. For every $u, v \in V_G$ such that $u < v$ and $uv \in E_G$, the collection $\mathcal{V}_{\lhd uv}$ is a nice partition of $V_u \setminus (N[u] \cap N(v))$.*

Proof. We prove the first statement. The proof of the second statement is completely analogous. Let $X = V_u \setminus \{u\}$ and $Y \subseteq V_G$ such that $X \cap Y = \varnothing$. Suppose that S_t is an S-triangle of $G[X \cup Y]$ for which the intersection of $V(S_t)$ and a part of $\mathcal{V}_{\lhd u}$ is non-empty for at least two such parts. Assume that P_1 and P_2 are two of those parts and let $u_1 \in V(S_t) \cap P_1$ and $u_2 \in V(S_t) \cap P_2$. Then u_1 and u_2 must be adjacent, which is in contradiction to Lemma 7.3.3. □

The following Lemma simplifies the calculation of Y' in Observation 7.2.1 (2) in the case of $X = V_u$ for some $u \in V_G$.

Lemma 7.3.5. *Let $u \in V_G$ and $Y \subseteq V_G \setminus V_u$. Then $Y \cap N(V_u) = Y \cap N(u)$.*

Proof. From the facts that $u \in V_u$ and $Y \subseteq V_G \setminus V_u$, it directly follows that $Y \cap N(u) \subseteq Y \cap N(V_u)$. We will now show that also $Y \cap N(V_u) \subseteq Y \cap N(u)$. It suffices to show that $N(V_u) \subseteq N(u)$. Let $w \in N(V_u)$. Then there exists a vertex $v \in V_u$ such that $vw \in E_G$. It suffices to show that $w \in N(u)$. Assume that $u \neq v$, as otherwise the claim trivially holds. Then $v < u$, because $v \in V_u$. Moreover, Observation 7.3.2 (1) implies that either $w < v$ or $v < w$. Since $w < v < u$ contradicts the fact that $w \notin V_u$, we conclude that $v < w$. Then by applying Observation 7.3.2 (2) we obtain that u and w are comparable. Since $w \notin V_u$, it must be that $v < u < w$ and by Observation 7.3.2 (3) we conclude that $uw \in E_G$. □

We are now ready to show the first recursive expressions of optimal solutions to subproblems, to be used for the computation of A_X^Y in certain cases of X and Y that are considered by both our algorithms. Both statements involve the application of Observation 7.2.1 in combination with Lemma 7.3.4 and Lemma 7.3.5.

Lemma 7.3.6. *Let $u \in V_G$ and $Y \subseteq V_G \setminus V_u$. If $u \notin A_{V_u}^Y$, then $A_{V_u}^Y \leftrightarrow \bigcup\limits_{u' \in \lhd u} A_{V_{u'}}^{Y \cap N(u')}$.*

Proof. Since $u \notin A_{V_u}^Y$, we have $A_{V_u}^Y \leftrightarrow A_{V_u \setminus \{u\}}^Y$. According to Lemma 7.3.4, the collection $\mathcal{V}_{\lhd u}$ is a nice partition of $V_u \setminus \{u\}$. By Observation 7.2.1 and Lemma 7.3.5, we get $A_{V_u \setminus \{u\}}^Y \leftrightarrow \bigcup_{u' \in \lhd u} A_{V_{u'}}^Y \leftrightarrow \bigcup_{u' \in \lhd u} A_{V_{u'}}^{Y \cap N(V_{u'})} \leftrightarrow \bigcup_{u' \in \lhd u} A_{V_{u'}}^{Y \cap N(u')}$. □

Lemma 7.3.7. *Let $u \in V_G$. If $u \in A_{V_u}^{\varnothing}$, then $A_{V_u}^{\varnothing} \leftrightarrow \{u\} \cup \bigcup\limits_{u' \in \lhd u} A_{V_{u'}}^{\{u\} \cap N(u')}$.*

Proof. Assume that $u \in A_{V_u}^{\varnothing}$. Then $A_{V_u}^{\varnothing} \leftrightarrow \{u\} \cup A_{V_u \setminus \{u\}}^{\{u\}}$. Recall that the collection $\mathcal{V}_{\lhd u}$ is a nice partition of $V_u \setminus \{u\}$. By Observation 7.2.1 and Lemma 7.3.5, we get $A_{V_u \setminus \{u\}}^{\{u\}} \leftrightarrow \bigcup_{u' \in \lhd u} A_{V_{u'}}^{\{u\}} \leftrightarrow \bigcup_{u' \in \lhd u} A_{V_{u'}}^{\{u\} \cap N(V_{u'})} \leftrightarrow \bigcup_{u' \in \lhd u} A_{V_{u'}}^{\{u\} \cap N(u')}$. □

7.4 SFVS on Graphs with Bounded Leafage

In this section our goal is to show that SFVS can be solved in polynomial time on chordal graphs with bounded leafage. In particular, we consider chordal graphs that have an intersection model tree with at most ℓ leaves and we show that SFVS can be solved in $n^{\mathcal{O}(\ell)}$ time. We subsequently assume that we are given a chordal graph G that admits an expanded tree model $(T, \{T_v\}_{v \in V_G})$ with $|L(T)| = \ell$ due to Lemma 7.3.1.

Given a set of vertices of G, we collect the nodes and the leaves and of their corresponding subtrees: for every $U \subseteq V_G$, we define $V(U) = \bigcup_{u \in U} V(T_u)$ and $L(U) = \bigcup_{u \in U} L(T_u)$. Notice that for any non-empty $U \subseteq V_G$, the sets $V(U)$ and $L(U)$ are also non-empty, and the nodes of $L(U)$ admit a partial order $\leq_T$. Moreover, given a set of nodes of T, we collect the vertices corresponding to the subtrees of which they are leaves: for every $V \subseteq V_T$, we define $L^{-1}(V)$ to be the set $\{u \in V_G \mid L(T_u) \cap V \neq \varnothing\}$.

Observation 7.4.1. *Let $U \subseteq V_G$ and $V \subseteq L(U)$. Then $L^{-1}(V) \subseteq U$.*

Proof. The fact that $V \subseteq L(U)$ yields $L^{-1}(V) \subseteq L^{-1}(L(U))$. We will show that $L^{-1}(L(U)) \subseteq U$. Let u be a vertex of G such that $u \notin U$. Then, since $(T, \{T_v\}_{v \in V_G})$ is an expanded tree model, Definition 7.3.1 implies that $L(T_u) \cap L(U) = \varnothing$. Thus $u \notin L^{-1}(L(U))$. □

For every $U \subseteq V_G$, we define the *representation* of U to be the set $R_{\leq 2}(U) = R_1(U) \cup R_2(U)$ where $R_1(U) = L^{-1}(\min L(U))$ and $R_2(U) =$ $L^{-1}(\min L(U \setminus R_1(U)))$. Observation 7.4.1 implies that $R_{\leq 2}(U) \subseteq U$ for every $U \subseteq V_G$. Observe that for any $V \subseteq V_T$, the set $\min V$ of minimal nodes of V is a set of pairwise-incomparable nodes, so $|\min V| \leq |L(T)| = \ell$ by Observation 7.1.3. This implies that $|R_{\leq 2}(U)| \leq 2\ell$ for all $U \subseteq V_G$. Representations have the following property.

Observation 7.4.2. *Let $u \in V_G$ and let $Y \subseteq V_G \setminus V_u$. Then $V(Y) \cap V(V_u) =$ $V(R_1(Y)) \cap V(V_u)$ and $V(Y \setminus R_1(Y)) \cap V(V_u) = V(R_2(Y)) \cap V(V_u)$.*

Proof. We show that the first equation holds. Showing that the second equation holds is completely analogous. By Observation 7.4.1, we get $R_1(Y) \subseteq Y \Rightarrow V(R_1(Y)) \subseteq V(Y) \Rightarrow V(R_1(Y)) \cap V(V_u) \subseteq V(Y) \cap V(V_u)$. We will show that also $V(Y) \cap V(V_u) \subseteq V(R_1(Y)) \cap V(V_u)$. Let $b \in V(Y) \cap V(V_u)$. Then there exist $u' \in V_u$ and $v \in Y$ such that $b \in V(T_{u'}) \cap V(T_v)$. Since $b \in V(T_{u'})$, we get $b \leq r(u')$. Since $b \in V(T_v)$, there exists an $l \in L(T_v) \subseteq L(Y)$ such that $l \leq b$. Then there exists an $l' \in \min L(Y)$ such that $l' \leq l$. By definition of $R_1(Y)$, there exists a vertex $v' \in R_1(Y)$ such that $l' \in L(T_{v'})$. Putting it all together yields $l' \leq l \leq b \leq r(u') \leq r(u) < r(v')$. By Observation 7.1.2, we conclude that $b \in V(T_{v'}) \subseteq V(R_1(Y))$. □

We subsequently show that for computing A_X^Y, the vertex set Y can be substituted by its representation $R_{\leq 2}(Y)$ in certain cases of X and Y that are considered by the algorithm of this section.

Lemma 7.4.3. *Let $u \in V_G$ and $W \subseteq V_G \setminus V_u$ such that $W \neq \varnothing$, $G[W] \in \mathcal{F}_S$ and $\{u\} \cup W$ is a clique, and let $u \in A_{V_u}^W$.*

- *If $(\{u\} \cup W) \cap S \neq \varnothing$, then $W = \{w\}$ and no vertex of $V_u \cap N(u) \cap N(w)$ belongs to $A_{V_u}^{\{w\}}$.*
- *If $(\{u\} \cup W) \cap S = \varnothing$, then $A_{V_{u'}}^{(\{u\} \cup W) \cap N(u')} \leftrightarrow A_{V_{u'}}^{R_{\leq 2}((\{u\} \cup W) \cap N(u'))}$ for every vertex $u' \in \lhd u$.*

Proof. Assume that some vertex of $\{u\} \cup W$ is in S. Further assume that $|W| \geq 2$. Then there are $w_1, w_2 \in W$ such that $\{u, w_1, w_2\} \cap S \neq \varnothing$. Since $\{u\} \cup W$ is a clique, we have that $\{u, w_1, w_2\}$ induces an S-triangle, contradicting the fact that u belongs to $A_{V_u}^W$. We deduce that $W = \{w\}$ because $W \neq \varnothing$. Observe that for any $u' \in V_u \cap N(u) \cap N(w)$, the vertex set $\{u', u, w\}$ induces an S-triangle, since u and w are adjacent. Thus, no vertex of $V_u \cap N(u) \cap N(w)$ is in $A_{V_u}^{\{w\}}$.

Assume that no vertex of $\{u\} \cup W$ is in S. Consider a vertex $u' \in \lhd u$. Observe that for any two vertices $a \in V_{u'}$ and $b \in V_G \setminus V_{u'}$ to be adjacent, since $r(a) \leq r(u') < r(b)$ already holds, there must exist an $l \in L(T_b)$ such that $l < r(a)$ also holds. Let $W' = (\{u\} \cup W) \cap N(u')$ and $R = R_{\leq 2}(W')$. We will show that $A_{V_{u'}}^{W'} \leftrightarrow A_{V_{u'}}^{R}$.

- Assume there are two vertices $u_1'', u_2'' \in V_{u'}$ and a vertex $w' \in W'$ such that $\{u_1'', u_2'', w'\}$ induces an S-triangle. Then u_1'', u_2'' are adjacent and consequently, by Observation 7.3.2 (1), comparable, so without loss of generality we may assume that $r(u_1'') < r(u_2'')$. Let $l' \in L(T_{w'})$ such that $l' < r(u_1'')$. By definition of R, there is a vertex $w'' \in R$ for which there is a node $l'' \in L(T_{w''})$ such that $l'' \leq l'$. This implies that the set $\{u_1'', u_2'', w''\}$ also induces an S-triangle.

- Assume there is a vertex $u'' \in V_{u'}$ and two vertices $w'_1, w'_2 \in W'$ such that $\{u'', w'_1, w'_2\}$ induces an S-triangle. Let $l'_1 \in L(T_{w'_1})$ and $l'_2 \in L(T_{w'_2})$ such that $l'_1, l'_2 < r(u'')$. By definition of R, there are two distinct vertices $w''_1, w''_2 \in R$ for which there are nodes $l''_1 \in L(T_{w''_1})$ and $l''_2 \in L(T_{w''_2})$ such that $l''_1 \leq l'_1$ and $l''_2 \leq l'_2$. This implies that the set $\{u'', w''_1, w''_2\}$ also induces an S-triangle. □

We next show that Lemma 7.3.6, Lemma 7.3.7 and Lemma 7.4.3 suffice for the development of a dynamic programming scheme. As the size of the representation of any subset of V_G is bounded by 2ℓ, we need to store only a bounded number of optimal solutions to subproblems. In particular, we show that we need to compute A_X^Y only for $\mathcal{O}(n)$ cases of X and only for cases of Y such that $|Y| \leq 2\ell$ holds.

Theorem 7.4.4. *There is an algorithm that, given a connected chordal graph G and an expanded tree model $(T, \mathcal{T})$ of G with $|L(T)| = \ell$, solves the weighted* Subset Feedback Vertex Set *problem in $\mathcal{O}(n^{2\ell+1})$ time.*

Proof. Let $u_{\max}$ denote the (unique) vertex of $\max V_G$. Our task is to solve SFVS on G by computing $A^{\varnothing}_{V_{u_{\max}}}$. For doing so, we device a dynamic programming algorithm that visits the nodes of T in a bottom-up fashion starting from its leaves and moving towards its root. At each node $v \in V_T$, if there exists a vertex $u \in V_G$ such that $r(u) = v$, we store the values of $A^{\varnothing}_{V_u}$ and $A^{W}_{V_u}$ for every $W \subseteq V_G \setminus V_u$ such that $W \neq \varnothing$, $G[W] \in \mathcal{F}_S$ and $\{u\} \cup W$ is a clique. In order to compute $A^{\varnothing}_{V_u}$, we apply Lemma 7.3.6 and Lemma 7.3.7. In particular, after retrieving all necessary values being stored on the corresponding descendants of v, we apply the formula

$$A^{\varnothing}_{V_u} = \max_{\text{weight}} \left\{ \bigcup_{u' \in \lhd u} A^{\varnothing}_{V_{u'}}, \quad \{u\} \cup \bigcup_{u' \in \lhd u} A^{\{u\} \cap N(u')}_{V_{u'}} \right\}.$$

Proof: The first case in the above formula is the case of $u \notin A^{\varnothing}_{V_u}$ and is due to Lemma 7.3.6, whereas the second case is the case of $u \in A^{\varnothing}_{V_u}$ and is due to Lemma 7.3.7. ⌟

For computing $A^{W}_{V_u}$, we apply Lemma 7.3.6 and Lemma 7.4.3. In particular, depending on W, after retrieving all necessary values being stored on the corresponding descendants of v, we apply the appropriate formula, as follows:

- If $(\{u\} \cup W) \cap S \neq \varnothing$ and $|W| \geq 2$, then we apply $A^W_{V_u} = \bigcup_{u' \in \lhd u} A^{W \cap N(u')}_{V_{u'}}$.

  Proof: Lemma 7.4.3 implies that $u \notin A^W_{V_u}$. By Lemma 7.3.6 we get the above formula. ⌟

- If $(\{u\} \cup W) \cap S \neq \varnothing$ and $W = \{w\}$, then we apply
$$A^W_{V_u} = \max_{\text{weight}} \left\{ \bigcup_{u' \in \lhd u} A^{\{w\} \cap N(u')}_{V_{u'}}, \quad \{u\} \cup \bigcup_{u' \in \lhd uw} A^{\{u,w\} \cap N(u')}_{V_{u'}} \right\}.$$

  Proof: The first case in the above formula is the case of $u \notin A^{\{w\}}_{V_u}$ and is due to Lemma 7.3.6. For the second case, assume that $u \in A^{\{w\}}_{V_u}$. Lemma 7.4.3 implies that $A^{\{w\}}_{V_u} \leftrightarrow \{u\} \cup A^{\{u,w\}}_{V_u \setminus (N[u] \cap N(w))}$. According to Lemma 7.3.4, the collection $\mathcal{V}_{\lhd uw}$ is a nice partition of $V_u \setminus (N[u] \cap N(w))$. By Observation 7.2.1 and Lemma 7.3.5 we get $A^{\{u,w\}}_{V_u \setminus (N[u] \cap N(w))} \leftrightarrow \bigcup_{u' \in \lhd uw} A^{\{u,w\}}_{V_{u'}} \leftrightarrow \bigcup_{u' \in \lhd uw} A^{\{u,w\} \cap N(V_{u'})}_{V_{u'}} \leftrightarrow \bigcup_{u' \in \lhd uw} A^{\{u,w\} \cap N(u')}_{V_{u'}}$. ⌟

- If $(\{u\} \cup W) \cap S = \varnothing$, then we apply
$$A^W_{V_u} = \max_{\text{weight}} \left\{ \bigcup_{u' \in \lhd u} A^{W \cap N(u')}_{V_{u'}}, \quad \{u\} \cup \bigcup_{u' \in \lhd u} A^{R_{\leq 2}((\{u\} \cup W) \cap N(u'))}_{V_{u'}} \right\}.$$

  Proof: The first case in the above formula is the case of $u \notin A^{\{w\}}_{V_u}$ and is due to Lemma 7.3.6. For the second case, assume that $u \in A^W_{V_u}$. Then $A^W_{V_u} \leftrightarrow \{u\} \cup A^{\{u\} \cup W}_{V_u \setminus \{u\}}$. According to Lemma 7.3.4, the collection $\mathcal{V}_{\lhd u}$ is a nice partition of $V_u \setminus \{u\}$. By Observation 7.2.1, Lemma 7.3.5 and Lemma 7.4.3 we get $A^{\{u\} \cup W}_{V_u \setminus \{u\}} \leftrightarrow \bigcup_{u' \in \lhd u} A^{\{u\} \cup W}_{V_{u'}} \leftrightarrow \bigcup_{u' \in \lhd u} A^{(\{u\} \cup W) \cap N(V_{u'})}_{V_{u'}} \leftrightarrow \bigcup_{u' \in \lhd u} A^{(\{u\} \cup W) \cap N(u')}_{V_{u'}} \leftrightarrow \bigcup_{u' \in \lhd u} A^{R_{\leq 2}((\{u\} \cup W) \cap N(u'))}_{V_{u'}}$. ⌟

Regarding the correctness of the algorithm, we show that applying any of the above recursive formulas requires only sets that can also be computed via these formulas. Notice that an induced subgraph of a graph in $\mathcal{F}_S$ is also a graph in $\mathcal{F}_S$ and that a subset of a clique is also a clique. Now observe that applying any of the above recursive formulas requires only sets $A^{\varnothing}_{V_{u'}}$ and $A^{W'}_{V_{u'}}$ where $u' \in V_G$ and $W' \subseteq V_G \setminus V_{u'}$ such that $W' \neq \varnothing$, $G[W'] \in \mathcal{F}_S$ and $\{u'\} \cup W'$ is a clique. We conclude that all sets A^Y_X that are required for the application of any of these formulas can also be computed via these formulas.

We now analyze the running time of our algorithm. We begin by determining for every pair (x, y) of distinct nodes of the host tree T whether $x < y$ or not. As the act of discovering all nodes x that precede a node y in $\leq_T$ takes $\mathcal{O}(n)$ time by traversing T once, we complete this task in $\mathcal{O}(n^2)$ time. We then compute the sets $\lhd u$ and $\lhd uv$ for all $u \in V_G$ and for all $v \in V_G$ such that $uv \in E_G$. Since any such set can be computed in $\mathcal{O}(n)$ time by traversing T once, we compute all such sets in $\mathcal{O}(n(n+m))$ time, which is simply $\mathcal{O}(nm)$ time as G is connected. Let us also bound the size of such sets. Observe that by definition any such set U is a set of pairwise-incomparable vertices. By definition of $\leq_G$, this implies that the set $V = \{r(u) \mid u \in U\}$ is a set of pairwise-incomparable nodes, yielding $|V| \leq |L(T)| = \ell$ by Observation 7.1.3. We conclude that any such set U contains at most ℓ vertices. We proceed with the computation of the sets A_X^Y where $X, Y \subseteq V_G$ such that $X \cap Y = \varnothing$ and $G[Y] \in \mathcal{F}_S$. According to the recursive formulas shown above and due to the fact that $|R_{\leq 2}(U)| \leq 2\ell$ for all $U \subseteq V_G$, it is sufficient to compute for every $u \in V_G$ the sets A_X^Y where $X = V_u$ and either $Y = \varnothing$ or $Y \subseteq N(u) \backslash V_u$ such that $G[Y] \in \mathcal{F}_S$, $\{u\} \cup Y$ is a clique and $|Y| \leq 2\ell$. Therefore, it suffices to compute $\mathcal{O}(n^{2\ell+1})$ sets A_X^Y. Now consider such a set A_X^Y. Its computation requires the retrieval of a number of stored values. Due to the bound on the size of the auxiliary sets shown above, that number is at most 2ℓ. If the set A_X^Y is computed via the last of the formulas shown above, we must also compute at most ℓ representations $R_{\leq 2}(U) = R_1(U) \cup R_2(U)$ of sets $U \subseteq V_G$ such that $|U| \leq 2\ell + 1$. Consider one such set U. Computing $R_1(U)$ (resp. $R_2(U)$) requires the computation of $\min L(U)$ (resp. $\min L(U \setminus R_1(U))$), which in turn requires determining for every pair (x, y) of distinct nodes in $L(U)$ (resp. $L(U \setminus R_1(U))$) whether $x < y$ or not. Since all these are predetermined, it suffices to retrieve a number of stored values. Recall that $|L(T_v)| \leq \ell$ for all $v \in V_G$. We get that $|L(U \setminus R_1(U))| \leq |L(U)| = \sum_{v \in U} |L(T_v)| \leq (2\ell + 1)\ell$, which implies that the number of stored values to be retrieved is $\mathcal{O}(\ell^4)$. We conclude that the running time for computing the set A_X^Y is $\mathcal{O}(\ell^5)$, which is constant time. Thus the total running time of our algorithm is $\mathcal{O}(n^{2\ell+1})$. □

Notice that in the special case of $\ell = 1$, the number of sets A_X^Y that the algorithm of Theorem 7.4.4 computes is actually $\mathcal{O}(nm)$ and consequently its total running time is $\mathcal{O}(nm)$. If we let the leafage of a chordal graph be the maximum leafage over all of its connected components, then we obtain the following result.

Corollary 7.4.5. *The weighted* Subset Feedback Vertex Set *problem can be solved on chordal graphs with leafage at most ℓ in $n^{\mathcal{O}(\ell)}$ time.*

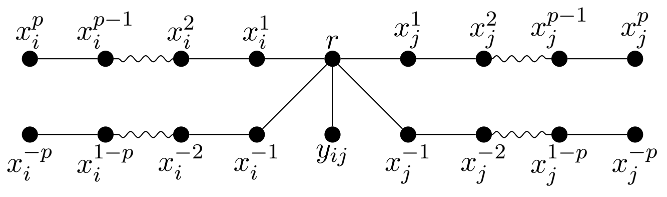


Figure 7.3: The subtree $T(\{x_i^+, x_i^-, y_{ij}, x_j^+, x_j^-\})$ of T for some $i, j \in [k]$ such that $i < j$.

Proof. Let G be a chordal graph. For every connected component C of G, we first recongize if C is an interval graph and if so construct a tree model $M(C)$ of C in linear time [9], otherwise we determine the leafage of C and construct a tree model $M(C)$ of C via the $\mathcal{O}(n^3)$-time algorithm of Habib and Stacho [43]. We then construct an expanded tree model $M'(C)$ from $M(C)$ in $\mathcal{O}(n^2)$ time by Lemma 7.3.1. Applying Theorem 7.4.4 on $M'(C)$, we compute $A^{\varnothing}_{V(C)}$ in $n^{\mathcal{O}(\ell)}$ time. It is not difficult to see that the collection $\{V(C) \mid C \text{ is a connected component of } G\}$ is a nice partition of $V(G)$. Thus by Observation 7.2.1 the set $A^{\varnothing}_{V(G)}$ is the union of $A^{\varnothing}_{V(C)}$ over all of the connected components C of G. Therefore, all above steps result in solving SFVS on G and can be carried out in $n^{\mathcal{O}(\ell)}$ time. □

Notice that in the special case of the input graph being an interval graph, the algorithm of Corollary 7.4.5 actually runs in $\mathcal{O}(nm)$ time, which is also the running time of the previously known algorithm for solving SFVS on interval graphs [69]. We next prove that we can hardly avoid the dependence of the exponent in the stated running time, since we show that weighted SUBSET FEEDBACK VERTEX SET is W[1]-hard parameterized by the leafage of a chordal graph. Our reduction is inspired by the W[1]-hardness of FEEDBACK VERTEX SET parameterized by the mim-width given by Jaffke et al. [50].

Theorem 7.4.6. *The weighted* SUBSET FEEDBACK VERTEX SET *decision problem on chordal graphs is W[1]-hard when parameterized by its leafage.*

Proof. We provide a reduction from the MULTICOLORED CLIQUE problem. Given a graph $G = (V, E)$ and a partition $\{V_i\}_{i \in [k]}$ of V into k parts, the MULTICOLORED CLIQUE (MCC) problem asks whether G has a clique that contains exactly one vertex of V_i for every $i \in [k]$. It is known that MCC is W[1]-hard when parameterized by k [31, 72].

Let $(G = (V, E), \{V_i\}_{i \in [k]})$ be an instance of MCC. We assume that $k \geq 10$ and without loss of generality that there exists $p \in \mathbb{N}$ such that $V_i = \{v_i^j\}_{j \in [p]}$ for every $i \in [k]$. We consider the $\frac{k}{2}(k+3)$-star T with internal node r and leaves x_i^+, x_i^- for every $i \in [k]$ and y_{ij} for every $i, j \in [k]$ such that $i < j$. We modify the star T as follows: for every $i \in [k]$, through a series of edge subdivisions, we replace the edge $\langle r, x_i^+ \rangle$ by the path $\langle r = x_i^0, x_i^1, \ldots, x_i^p = x_i^+ \rangle$ and the edge $\langle r, x_i^- \rangle$ by the path $\langle r = x_i^0, x_i^{-1}, \ldots, x_i^{-p} = x_i^- \rangle$. Given a set X of nodes of T, we write $T(X)$ to denote the minimal subtree of T containing all nodes of X. The subtree $T(X)$ for a particular choice of X is depicted in Figure 7.3. We define the following subtrees of T:

- For every $i, j \in [k]$ such that $i < j$ and for every $a, b \in [p]$ such that $v_i^a v_j^b \in E$, we define $e_{ij}^{ab} = T(\{x_i^a, x_i^{a-p}, y_{ij}, x_j^b, x_j^{b-p}\})$. We denote by R the collection of all these subtrees.
  - For all $i \in [k]$, we denote by R_i the collection $\{e_{ij}^{ab} \in R \mid j \in [k] \text{ and } a, b \in [p]\} \cup \{e_{ji}^{ba} \in R \mid j \in [k] \text{ and } a, b \in [p]\}$.
  - For all $i \in [k]$ and for all $a \in [p]$, we denote by R_i^a the collection $\{e_{ij}^{ab} \in R \mid j \in [k] \text{ and } b \in [p]\} \cup \{e_{ji}^{ba} \in R \mid j \in [k] \text{ and } b \in [p]\}$.
  - For all $i, j \in [k]$ such that $i < j$, we denote by R_{ij} the collection $\{e_{ij}^{ab} \in R \mid a, b \in [p]\}$.
- For every $i \in [k]$ and $a \in [p]$, we define $s_i^{a,1} = s_i^{a,2} = T[\{x_i^a\}]$ and $s_i^{-a,1} = s_i^{-a,2} = T[\{x_i^{-a}\}]$. We denote by S_V the collection of all these subtrees.
  - For all $i \in [k]$, we denote by S_i the collection $\{s_i^{a,c} \in S_V \mid a \in -[p] \cup [p] \text{ and } c \in \{1, 2\}\}$.
  - For all $i \in [k]$ and for all $a \in [p]$, we denote by S_i^a the collection $\{s_i^{a',c} \in S_V \mid a' \in -[p-a] \cup [a] \text{ and } c \in \{1, 2\}\}$.
- For every $i, j \in [k]$ such that $i < j$, we define $s_{ij} = T[\{y_{ij}\}]$. We denote by S_E the collection of all these subtrees.

We further denote by S the collection $S_V \cup S_E$ and by $\mathcal{T}$ the collection $R \cup S$. We construct a graph G' that is the intersection graph of the undirected tree model $(T, \mathcal{T})$. Notice that G' is a chordal graph of leafage at most $\frac{k}{2}(k+3)$. We identify the vertices of G' with their corresponding subtrees in $\mathcal{T}$. By the construction of $(T, \mathcal{T})$, regarding the adjacencies between vertices of G' we observe the following:

- R is a clique, because all its elements contain the node r.
- For every $i \in [k]$ and $a \in -[p] \cup [p]$, we have $N(s_i^{a,1}) \cap S = \{s_i^{a,2}\}$ and $N(s_i^{a,2}) \cap S = \{s_i^{a,1}\}$.
- For every $i \in [k]$ and $a \in [p]$, we have $N(e) \cap S_i = S_i^a$ for all $e \in R_i^a$.
- For every $i, j \in [k]$ such that $i < j$, we have $N(s_{ij}) = R_{ij}$.

We set the weight of all vertices of R, S_V and S_E to be $\frac{p}{2}$, 1 and $\frac{p}{2}m$, respectively. We will show that $(G, \{V_i\}_{i \in [k]})$ is a YES-instance of MCC if and only if there exists a solution to SFVS on (G', S) having weight $\frac{p}{2}(m - \frac{k}{2}(k-9))$.

For the forward direction, let $\{v_1^{a_1}, \dots, v_k^{a_k}\}$ be a solution of MCC on $(G, \{V_i\}_{i \in [k]})$. We set R_C to be the collection $\{e_{ij}^{a_i a_j} \in R \mid i, j \in [k]\}$. Observe that R_C contains exactly one element of R_{ij} for each $i, j \in [k]$ such that $i < j$. We further set $U = (R \setminus R_C) \cup \bigcup_{i \in [k]} S_i^{a_i}$. Now observe that in $G - U$ each of the remaining vertices of S has exactly one neighbour. Thus U is a solution to SFVS on (G', S) having weight $\frac{p}{2}(m - \frac{k}{2}(k-1)) + 2pk = \frac{p}{2}(m - \frac{k}{2}(k-9))$.

For the reverse direction, let U be a solution to SFVS on (G', S) having weight $\frac{p}{2}(m - \frac{k}{2}(k-9))$. Notice that no element of S_E can be in U. Consequently, for every $i, j \in [k]$ such that $i < j$, we have $|R_{ij} \setminus U| \leq 1$, since any two elements of R_{ij} along with s_{ij} form an S-triangle of G'. Any remaining S-triangle of G' is formed by either

- an element of R_i^a and two adjacent elements of S_i^a or
- an element of R_i^a, an element of $R_i^{a'}$ and an element of $S_i^a \cap S_i^{a'}$

for a particular choice of $i \in [k]$ and $a, a' \in [p]$. Let $i \in [k]$.

Claim 7.4.7. *If $|R_i \setminus U| \geq 1$, then $|S_i \cap U| \geq p$.*

Proof: Assume that $e \in R_i \setminus U$. Then there exists an $a \in [p]$ such that $e \in R_i^a$. We conclude that for every $a' \in -[p-a] \cup [a]$, at least one of $s_i^{a',1}, s_i^{a',2}$ must be in U, yielding $|S_i \cap U| \geq p$. ⌟

Claim 7.4.8. *If $|R_i \setminus U| \geq 2$, then $|S_i \cap U| \geq 2p$.*

Proof: Assume that e, e' are two distinct elements of $R_i \setminus U$. Then there exist $a, a' \in [p]$ such that $e \in R_i^a$ and $e' \in R_i^{a'}$. Without loss of generality, assume that $a \leq a'$. We conclude that

- for every $a'' \in -[p-a'] \cup [a]$ both $s_i^{a'',1}$ and $s_i^{a'',2}$ must be in U and
- for every $a'' \in (-[p-a] \cup [a']) \setminus (-[p-a'] \cup [a])$ at least one of $s_i^{a'',1}, s_i^{a'',2}$ must be in U,

yielding $|S_i \cap U| \geq 2((p-a')+a) + 1(((p-a)+a') - ((p-a')+a)) = 2p$. $\lrcorner$

Claim 7.4.9. *If $|R_i \setminus U| \geq 3$, then $|S_i \cap U| = 2p$ only if there exists $a \in [p]$ such that $R_i \setminus U \subseteq R_i^a$.*

Proof: Assume that e, e', e'' are three distinct elements of $R_i \setminus U$. Then there exist $a, a', a'' \in [p]$ such that and $e \in R_i^a$, $e' \in R_i^{a'}$ and $e'' \in R_i^{a''}$. Without loss of generality, assume that $a \leq a' \leq a''$. We conclude that

- for every $a''' \in -[p-a'] \cup [a']$ both $s_i^{a''',1}$ and $s_i^{a''',2}$ must be in U and
- for every $a''' \in (-[p-a] \cup [a'']) \setminus (-[p-a'] \cup [a'])$ at least one of $s_i^{a''',1}, s_i^{a''',2}$ must be in U,

yielding $|S_i \cap U| \geq 2p + 1(((p-a)+a'') - ((p-a')+a')) = 2p + (a''-a)$. Therefore, for $|S_i \cap U|$ to be $2p$, it must hold that $a = a' = a''$, so it must hold that $e, e', e'' \in R_i^a$. $\lrcorner$

Assume that $|\{i \in [k] \mid |R_i \setminus U| = 1\}| = k'$ and $|\{i \in [k] \mid |R_i \setminus U| \geq 2\}| = k''$. Then notice that $|R \setminus U| \leq k' + \frac{k''}{2}(k''-1)$, so $|R \cap U| \geq m - k' - \frac{k''}{2}(k''-1)$. Also, according to Claims 7.4.7 and 7.4.8, we have $|S_V \cap U| = \sum_{i \in [k]} |S_i \cap U| \geq p(k'+2k'')$. Lastly, recall that $|S_E \cap U| = \varnothing$. Consequently, the weight of U must be at least

$$\frac{p}{2}\left(m - k' - \frac{k''}{2}(k''-1)\right) + p(k'+2k'') = \frac{p}{2}\left(m + k' - \frac{k''}{2}(k''-9)\right) = B(k', k'').$$

Clearly, $k', k'' \in \{0, 1, \ldots, k\}$. Regarding the values of B, we observe the following:

- $B(k', k'') < B(k'+1, k'')$ for all $k' \in \{0, 1, \ldots, k-1\}$ and for all $k'' \in \{0, 1, \ldots, k\}$,
- $B(k', k'') \geq B(k', 9)$ for all $k' \in \{0, 1, \ldots, k\}$ and for all $k'' \in \{0, 1, \ldots, 8\}$, and

- $B(k', k'') > B(k', k''+1)$ for all $k' \in \{0, 1, \ldots, k\}$ and for all $k'' \in \{9, 10, \ldots, k-1\}$.

These imply that $B(k', k'')$ is minimum if and only if $k' = 0$ and $k'' = k$. Therefore, a weight of $\frac{p}{2}(m - \frac{k}{2}(k-9)) = B(0, k)$ is within bounds for U only if $k' = 0$ and $k'' = k$. Furthermore, the weight of U is $B(0, k)$ only if $|R \setminus U| = \frac{k}{2}(k-1)$ and $|S_i \cap U| = 2p$ for all $i \in [k]$. Now recall that $|R_{ij} \setminus U| \leq 1$ for all $i, j \in [k]$ such that $i < j$. We deduce that $|R_{ij} \setminus U| = 1$ for all $i, j \in [k]$ such that $i < j$, which implies that $|R_i \setminus U| = k - 1$ for all $i \in [k]$. Then, by Claim 7.4.9, for every $i \in [k]$, there exists an $a_i \in [p]$ such that $R_i \setminus U \subseteq R_i^{a_i}$. We conclude that the set $\{v_1^{a_1}, \ldots, v_k^{a_k}\}$ is a solution to MCC on $(G, \{V_i\}_{i \in [k]})$. □

7.5 SFVS on Graphs with Bounded Vertex Leafage

7.5.1 Rooted Path Graphs

Here we show how to extend our previous approach to solving SFVS to rooted path graphs. Recall that rooted path graphs are exactly the intersection graphs of directed paths on a rooted tree. We observe that rooted path graphs are a graph class of unbounded leafage.

Proposition 7.5.1. *There are rooted path graphs on n vertices having leafage $\Theta(n)$.*

Proof. Let G be the graph obtained from the ℓ-star with $\ell \geq 2$ by subdividing all of its edges. Notice that $n = 2\ell + 1$. We show that G is a rooted path graph having leafage $\ell - 1$.

Let $v_1, \ldots, v_\ell$ be the leaf vertices of G, let $u_1, \ldots, u_\ell$ be the vertices of G such that $u_i v_i \in E(G)$ for every $i \in [\ell]$ and let t be the remaining vertex of G. To show that G is a rooted path graph, we construct a tree model $(T, \{T_v\}_{v \in V(G)})$ of G as follows. We obtain the host tree T as the union of paths $P_0 = (x_\ell, x_{\ell-1}, \ldots, x_1)$, $P_\ell = (y_\ell, x_\ell)$ and $P_i = (x_i, y_i)$, $i \in [\ell - 1]$. We choose the subtrees to be $T_t = P_0$, $T_{u_i} = P_i$ and $T_{v_i} = (y_i)$, $i \in [\ell]$. Notice that T is a tree rooted on y_ℓ such that $L(T) = \ell - 1$ and all T_v, $v \in V(G)$ are directed subpaths of T. It is not difficult to see that $(T, \{T_v\}_{v \in V(G)})$ is a tree model of G, which shows that G is indeed a rooted path graph.

Let us now show that for every tree model $(T', \{T'_v\}_{v \in V(G)})$ of G, it holds that $|L(T')| \geq \ell - 1$. The maximal cliques of G are $C_i = \{t, u_i\}$ and $D_i = \{u_i, v_i\}$, $i \in [\ell]$. For every maximal clique C of G, there exists a node $c \in V(T')$ such that for every $v \in V(G)$, it holds that $c \in V(T'_v) \Leftrightarrow v \in C$. Moreover, all these nodes are pairwise distinct. For every C_i and for every D_i, we select one such node and denote it by c_i and d_i respectively. For all $V \subseteq V(T')$, we define $\rhd V$ to be the (unique) node of the set $\min\{y \in V(T') \mid \forall x \in V : x \leq y\}$. Observe that for every $v \in V(G)$, for every $V \subseteq V(T'_v)$, the node $\rhd V$ is also in $V(T'_v)$ because T'_v is connected. For all $i \in [\ell]$, we denote by b_i the node $\rhd\{c_i, d_i\}$ where $\{c_i, d_i\} \subseteq V(T'_{u_i})$. For all $i, j \in [\ell]$, we denote by a_{ij} the node $\rhd\{c_i, c_j\}$ where $\{c_i, c_j\} \subseteq V(T'_t)$.

Claim 7.5.2. *For every $i, j \in [\ell]$, if $d_i < d_j$ holds, then $d_i < a_{ij} < d_j$ holds.*

Proof: For every $i, j \in [\ell]$, we have that the following relations hold:

$$c_i \leq b_i \qquad d_i \leq b_i \qquad c_j \leq b_j \qquad d_j \leq b_j \qquad c_i \leq a_{ij} \qquad c_j \leq a_{ij}$$

By Observation 7.1.1, we obtain that a_{ij} is comparable to both b_i and b_j. Assume that $d_i < d_j$ holds. By Observation 7.1.1, we obtain that b_i is comparable to both b_j and d_j. If $d_i < d_j \leq b_i$ holds, then by Observation 7.1.2, we obtain that $d_j \in V(T_{u_i})$, which is a contradiction, so $d_i \leq b_i < d_j \leq b_j$ must hold. Now by Observation 7.1.1, we obtain that a_{ij} is comparable to d_j. If $c_i \leq b_i < d_j \leq a_{ij}$ holds, then by Observation 7.1.2, we obtain that $d_j \in V(T'_t)$, which is a contradiction, so $a_{ij} < d_j$ must hold. Assume $a_{ij} \leq b_i$ holds. Then both $c_i \leq a_{ij} \leq b_i$ and $c_j \leq a_{ij} < b_j$ hold. By Observation 7.1.2, we obtain that $a_{ij} \in V(T'_{u_i}) \cap V(T'_{u_j})$, which is a contradiction to $(T', \{T'_v\}_{v \in V(G)})$ being a tree model of G. We conclude that $d_i < a_{ij} < d_j$ holds. ⌟

We will show that there are at least $\ell - 1$ pairwise incomparable nodes in $D = \{d_i\}_{i \in [\ell]}$. Assume that there exist $i, j, k \in [\ell]$ such that $d_i < d_j < d_k$ holds. Then by Claim 7.5.2, we obtain that $d_i < a_{ij} < d_j < a_{jk} < d_k$ holds, and by Observation 7.1.2, we obtain that $d_j \in V(T'_t)$, a contradiction. Now assume that there exist $i, j, k, l \in [\ell]$ such that both $d_i < d_j$ and $d_k < d_l$ hold and d_j and d_l are incomparable. Then by Claim 7.5.2, we obtain that both $d_i < a_{ij} < d_j$ and $d_k < a_{kl} < d_l$ hold. Let a be the node $\rhd\{a_{ij}, a_{kl}\}$ where $\{a_{ij}, a_{kl}\} \subseteq V(T'_t)$. By Observation 7.1.1, we obtain that a is comparable to both d_j and d_l. If both $a \leq d_j$ and $a \leq d_l$ hold, then by Observation 7.1.1 we obtain that d_j and d_l are comparable, a contradiction, so at least one of $d_j < a$ and $d_l < a$ holds. Without loss of generality, assume that $d_j < a$ holds. Then by Observation 7.1.2, we obtain that $d_j \in V(T'_t)$, a contradiction. We conclude that there exists at most one node in D that succeeds another node in D, which implies that at least $\ell - 1$ nodes in D are pairwise incomparable. By Observation 7.1.3, we obtain that $|L(T')| \geq \ell - 1$, which concludes the proof. □

Our goal in this section is to device an algorithm for solving SFVS on rooted path graphs in polynomial time. Just as for the algorithm of Section 7.4, we will derive recursive formulas for optimal solutions A_X^Y and subsequently bound the number of optimal solutions to subproblems that the algorithm requires for solving the complete problem.

Assume that $G = (V_G, E_G)$ is a rooted path graph, $S \subseteq V_G$ and $(T, \{T_v\}_{v \in V_G})$ is an expanded tree model of G. For every $u \in V_G$, we denote by $l(u)$ the (unique) leaf of its corresponding directed path T_u. For devicing the algorithm of this section, further special vertices and subsets are required. For every $u, v \in V_G$ such that $u < v$, we define $u \lhd v$ to be the (unique) vertex of $\max(\{u\} \cup \{u' \in V_G \setminus S \mid u < u' < v \text{ and } u' \in N(u)\}) = \max(\{u\} \cup \{u' \in V_G \setminus S \mid l(u') < r(u) < r(u') < r(v)\}$. Moreover, for every $V_1, V_2, V_3 \subseteq V_G$, we define the following subsets of $V_G \setminus S$:

$$\begin{aligned} V[V_1;;] &= \{u \in V_G \setminus S \mid \nexists v_1 \in V_1 : l(u) < r(v_1)\} \\ V[;V_2;] &= \{u \in V_G \setminus S \mid \exists v_2 \in V_2 : l(u) < r(v_2) < r(u)\} \\ V[;;V_3] &= \{u \in V_G \setminus S \mid \exists v_3 \in V_3 : r(u) \leq r(v_3)\} \end{aligned}$$

$$\begin{aligned} V[V_1;V_2;] &= V[V_1;;] \cap V[;V_2;] \\ V[V_1;;V_3] &= V[V_1;;] \cap V[;;V_3] \\ V[;V_2;V_3] &= V[;V_2;] \cap V[;;V_3] \end{aligned}$$

$$V[V_1;V_2;V_3] = V[V_1;;] \cap V[;V_2;] \cap V[;;V_3]$$

For any $u, v \in V_G$, we denote the set $V_u \cup V[;\{u\};\{v\}]$ by $V_{u,v}$ for simplicity. Observe that the set $V_{u,u}$ is simply V_u.

Lemma 7.5.3. *Let $u, w \in V_G$ such that $u < w$ and $uw \in E_G$. Then the collection $\mathcal{V} = \{V[\lhd uw;;\lhd u]\} \cup \{V_{u',u' \lhd u}\}_{u' \in \lhd uw}$ is a nice partition of $X = (V_u \setminus \{u\}) \setminus (N(u) \cap N(w) \cap S)$ for every $Y \subseteq V_G \setminus X$ such that $Y \cap S = \varnothing$.*

Proof. We first show that $\mathcal{V}$ is a partition of X. Recall that $\lhd uw$ is a set of pairwise incomparable vertices by definition. Consider a vertex $u'' \in X$. Then exactly one of the following statements holds:

$$\begin{cases} \exists u' \in \lhd uw : r(u'') \leq r(u') \\ \exists u' \in \lhd uw : l(u'') < r(u') < r(u'') \\ \nexists u' \in \lhd uw : l(u'') < r(u') \end{cases} \Leftrightarrow \begin{cases} \exists u' \in \lhd uw : u'' \in V_{u'} \\ \exists u' \in \lhd uw : u'' \in V[;\{u'\};\{u' \lhd u\}] \\ u'' \in V[\lhd uw;;\lhd u] \end{cases}$$

By Observation 7.1.1 and the definition of $\leq_G$, the vertex $u' \in \lhd uw$ in the first two cases above is unique, otherwise we obtain a contradiction to the vertices of $\lhd uw$ being pairwise incomparable. This fact implies our claim.

Now let $Y \subseteq V_G \setminus X$ such that $Y \cap S = \varnothing$ and consider an S-triangle S_t of $G[X \cup Y]$. Then there exists a vertex $u' \in \lhd uw$ such that $V(S_t) \cap V_{u'} \cap S \neq \varnothing$. Since S_t is a triangle, every vertex in $V(S_t) \setminus V_{u'}$ is adjacent to every vertex in $V(S_t) \cap V_{u'}$. Then by Lemma 7.3.5, for every vertex $u'' \in V(S_t) \setminus V_{u'}$, the vertex u'' is adjacent to u', which implies that $l(u'') < r(u') < r(u'')$ holds. We conclude that $V(S_t) \subseteq V_{u',u' \lhd u}$. $\square$

Observation 7.5.4. *Let $X, Y \subseteq V_G$ such that $X \cap Y = \varnothing$ and $G[Y] \in \mathcal{F}_S$ and let $X' \subseteq X$. If $\mathcal{P}$ is a nice partition of X for Y, then $\mathcal{P}' = \{P \cap X' \mid P \in \mathcal{P} : P \cap X' \neq \varnothing\}$ is a nice partition of X' for Y.*

Proof. The fact that $\mathcal{P}'$ is a partition of X' follows trivially from the definition. For showing that $\mathcal{P}'$ is a nice partition of X' for Y, it suffices to notice that any S-triangle in $G[X' \cup Y]$ remains an S-triangle in $G[X \cup Y]$. $\square$

We are now ready to show the recursive expressions that hold exclusively for G being a rooted path graph and are required by the algorithm of this section. First we obtain an expression to be used for the computation of sets A_X^Y in case of $X = V_u$ and $Y = \{w\}$ where $u, w \in V_G$ such that $u < w$ and $uw \in E_G$.

Lemma 7.5.5. *Let $u, w \in V_G$ such that $u < w$ and $uw \in E_G$, and let $u \in A_{V_u}^{\{w\}}$.*

- *If $u \in S$ or $w \in S$, then* $A_{V_u}^{\{w\}} \leftrightarrow \{u\} \cup \bigcup\limits_{u' \in \lhd uw} A_{V_{u'}}^{\{u,w\} \cap N(u')}$.
- *If $u, w \notin S$, then* $A_{V_u}^{\{w\}} \leftrightarrow \{u\} \cup V[\lhd uw; ; \lhd u] \cup \bigcup\limits_{u' \in \lhd uw} A_{V_{u',u' \lhd u}}^{\{u,w\} \cap N(u')}$.

Proof. Observe that $A_{V_u}^{\{w\}} \leftrightarrow \{u\} \cup A_{V_u \setminus \{u\}}^{\{u,w\}}$ by definition. Regarding triangles of $G[V_u \cup \{w\}]$, we observe the following property:

(P1) By the hypothesis, the vertices u and w are adjacent. Thus, for any $u' \in V_u \cap N(u) \cap N(w)$, the vertex set $\{u', u, w\}$ induces a triangle.

If $u \in S$ or $w \in S$, then no vertex of $V_u \cap N(u) \cap N(w)$ is in $A^{\{w\}}_{V_u}$ because of (P1). By definition, we get $A^{\{u,w\}}_{V_u \setminus \{u\}} \leftrightarrow A^{\{u,w\}}_{V_u \setminus (N[u] \cap N(w))}$. According to Lemma 7.3.4, the collection $\mathcal{V}_{\lhd uw}$ is a nice partition of $V_u \setminus (N[u] \cap N(w))$. By Observation 7.2.1 and Lemma 7.3.5, we get $A^{\{u,w\}}_{V_u \setminus (N[u] \cap N(w))} \leftrightarrow \bigcup_{u' \in \lhd uw} A^{\{u,w\}}_{V_{u'}} \leftrightarrow \bigcup_{u' \in \lhd uw} A^{\{u,w\} \cap N(V_{u'})}_{V_{u'}} \leftrightarrow \bigcup_{u' \in \lhd uw} A^{\{u,w\} \cap N(u')}_{V_{u'}}$.

If $u, w \notin S$, then no vertex of $V_u \cap N(u) \cap N(w) \cap S$ is in $A^{\{w\}}_{V_u}$ because of (P1). Let $X = (V_u \setminus \{u\}) \setminus (N(u) \cap N(w) \cap S)$. By definition, we get $A^{\{u,w\}}_{V_u \setminus \{u\}} \leftrightarrow A^{\{u,w\}}_{X}$. According to Lemma 7.5.3, the collection $\{V[\lhd uw; ; \lhd u]\} \cup \{V_{u',u' \lhd u}\}_{u' \in \lhd uw}$ is a nice partition of X for $Y = \{u, w\}$. By Observation 7.2.1 and Lemma 7.3.5, we get $A^{\{u,w\}}_{X} \leftrightarrow A^{\{u,w\}}_{V[\lhd uw;;\lhd u]} \cup \bigcup_{u' \in \lhd uw} A^{\{u,w\}}_{V_{u',u' \lhd u}} \leftrightarrow$
$V[\lhd uw; ; \lhd u] \cup \bigcup_{u' \in \lhd uw} A^{\{u,w\} \cap N(V_{u'})}_{V_{u',u' \lhd u}} \leftrightarrow$
$V[\lhd uw; ; \lhd u] \cup \bigcup_{u' \in \lhd uw} A^{\{u,w\} \cap N(u')}_{V_{u',u' \lhd u}}$. □

We next obtain recursive expressions to be used for the computation of sets A^Y_X in case of $X = V_{u,v}$ where $u \in V_G$ and $v \in V_G \setminus S$ such that $u < v$ and $uv \in E_G$. The first two Lemmas follow directly from the definition of sets A^Y_X and the fact that $V_{u,v} \setminus \{v\} = V_{u,u \lhd v}$ for every $u, v \in V_G$ such that $u < v$.

Lemma 7.5.6. *Let $u \in V_G$ and $v \in V_G \setminus S$ such that $u < v$ and $uv \in E_G$ and let $Y \subseteq V_G \setminus V_{u,v}$. If $v \notin A^Y_{V_{u,v}}$, then $A^Y_{V_{u,v}} \leftrightarrow A^Y_{V_{u,u \lhd v}}$.*

Lemma 7.5.7. *Let $u \in V_G$ and $v \in V_G \setminus S$ such that $u < v$ and $uv \in E_G$. If $v \in A^{\varnothing}_{V_{u,v}}$, then $A^{\varnothing}_{V_{u,v}} \leftrightarrow \{v\} \cup A^{\{v\}}_{V_{u,u \lhd v}}$.*

Lemma 7.5.8. *Let $u \in V_G$ and $v, w \in V_G \setminus S$ such that $u < v < w$ and $\{u, v, w\}$ is a clique and let $v \in A^{\{w\}}_{V_{u,v}}$. Then $A^{\{w\}}_{V_{u,v}} \leftrightarrow \{v\} \cup V[\lhd vw; \{u\}; \{u \lhd v\}] \cup \bigcup\limits_{u' \in V_u \cap \lhd vw} A^{\{v,w\} \cap N(u')}_{V_{u',u' \lhd v}}$.*

Proof. Observe that $A^{\{w\}}_{V_{u,v}} \leftrightarrow \{v\} \cup A^{\{v,w\}}_{V_{u,v} \setminus \{v\}} \leftrightarrow \{v\} \cup A^{\{v,w\}}_{V_{u,u \lhd v}}$ by definition. Regarding triangles of $G[V_{u,v} \cup \{w\}]$, we observe the following property:

(P2) By the hypothesis, the vertices v and w are adjacent. Thus, for any $u' \in V_{u,v} \cap N(v) \cap N(w)$, the vertex set $\{u', v, w\}$ induces a triangle.

Since $v, w \notin S$, no vertex of $V_{u,v} \cap N(v) \cap N(w) \cap S$ is in $A^{\{w\}}_{V_{u,v}}$ because of (P2). Let $X = (V_v \setminus \{v\}) \setminus (N(v) \cap N(w) \cap S)$ and $X' = V_{u,u \lhd v} \setminus (N(v) \cap N(w) \cap S)$.

By definition, we get $A^{\{v,w\}}_{V_{u,u\lhd v}} \leftrightarrow A^{\{v,w\}}_{X'}$. Now notice that $X' \subseteq X$. According to Lemma 7.5.3 and Observation 7.5.4, the collection $\{V[\lhd vw;\{u\};\{u\lhd v\}]\} \cup \{V_{u',u'\lhd v}\}_{u'\in V_u \cap \lhd vw}$ is a nice partition of X' for $Y = \{v,w\}$. By Observation 7.2.1 and Lemma 7.3.5, we get $A^{\{v,w\}}_{X'} \leftrightarrow$
$A^{\{v,w\}}_{V[\lhd vw;\{u\};\{u\lhd v\}]} \cup \bigcup_{u'\in V_u\cap\lhd vw} A^{\{v,w\}}_{V_{u',u'\lhd v}} \leftrightarrow$
$V[\lhd vw;\{u\};\{u\lhd v\}] \cup \bigcup_{u'\in V_u\cap\lhd vw} A^{\{v,w\}\cap N(V_{u'})}_{V_{u',u'\lhd v}} \leftrightarrow$
$V[\lhd vw;\{u\};\{u\lhd v\}] \cup \bigcup_{u'\in V_u\cap\lhd vw} A^{\{v,w\}\cap N(u')}_{V_{u',u'\lhd v}}$. □

Now we are in position to state our claimed result, which is obtained via an algorithm similar to the one in the proof of Theorem 7.4.4.

Theorem 7.5.9. *The weighted* Subset Feedback Vertex Set *problem can be solved on rooted path graphs in $\mathcal{O}(n^2 m)$ time.*

Proof. We first describe the algorithm. Given a rooted path graph $G = (V_G, E_G)$, we construct a tree model $(T, \{T_v\}_{v\in V_G})$ of G such that all subtrees T_v, $v \in V_G$ are directed paths in $\mathcal{O}(n+m)$ time [29, 40]. We apply the iterative procedure described in the proof of Lemma 7.3.1 and obtain an expanded tree model $(T', \{T'_v\}_{v\in V_G})$ of G such that all subtrees T'_v, $v \in V_G$ are directed paths in $\mathcal{O}(n^2)$ time. As the host tree T of G has at most n nodes [19, 43], the expanded host tree T' has $\mathcal{O}(n)$ nodes. If G is an interval graph, then SFVS can be solved via the algorithm described in the proof of Corollary 7.4.5 in $\mathcal{O}(nm)$ time. Otherwise, we solve SFVS by computing $A^{\varnothing}_{V_{u_{\max}}}$ where $u_{\max}$ is the (unique) vertex of $\max V_G$.

For this purpose, we device a dynamic programming algorithm for computing $A^{\varnothing}_{V_{u_{\max}}}$. The algorithm works on T' traversing it in a bottom-up fashion starting from its leaves and moving towards its root. It maintains tables for storing the values of computed sets A^Y_X in the following four cases of X and Y:

- $X = V_u$ and $Y = \varnothing$ for every $u \in V_G$.
- $X = V_u$ and $Y = \{w\}$ for every $u, w \in V_G$ such that $u < w$ and $uw \in E_G$.
- $X = V_{u,v}$ and $Y = \varnothing$ for every $u \in V_G$ and $v \in V_G \setminus S$ such that $u < v$ and $uv \in E_G$.
- $X = V_{u,v}$ and $Y = \{w\}$ for every $u \in V_G$ and $v, w \in V_G \setminus S$ such that $u < v < w$ and $\{u, v, w\}$ is a clique.

For computing these sets, we derive the following recursive formulas:

- Let $u \in V_G$. Lemma 7.3.6 and Lemma 7.3.7 imply that

$$A^{\varnothing}_{V_u} = \max_{\text{weight}} \left\{ \bigcup_{u' \in \lhd u} A^{\varnothing}_{V_{u'}}, \quad \{u\} \cup \bigcup_{u' \in \lhd u} A^{\{u\} \cap N(u')}_{V_{u'}} \right\}.$$

- Let $u, w \in V_G$ such that $u < w$ and $uw \in E_G$. Lemma 7.3.6 and Lemma 7.5.5 imply the following:
  - If $u \in S$ or $w \in S$, then
  $$A^{\{w\}}_{V_u} = \max_{\text{weight}} \left\{ \bigcup_{u' \in \lhd u} A^{\{w\} \cap N(u')}_{V_{u'}}, \quad \{u\} \cup \bigcup_{u' \in \lhd uw} A^{\{u,w\} \cap N(u')}_{V_{u'}} \right\}.$$
  - If $u, w \notin S$, then
  $$A^{\{w\}}_{V_u} = \max_{\text{weight}} \left\{ \bigcup_{u' \in \lhd u} A^{\{w\} \cap N(u')}_{V_{u'}}, \right.$$
  $$\left. \{u\} \cup V[\lhd uw; ; \lhd u] \cup \bigcup_{u' \in \lhd uw} A^{\{u,w\} \cap N(u')}_{V_{u', u' \lhd u}} \right\}.$$

- Let $u \in V_G$ and $v \in V_G \setminus S$ such that $u < v$ and $uv \in E_G$. Lemma 7.5.6 and Lemma 7.5.7 imply that

$$A^{\varnothing}_{V_{u,v}} = \max_{\text{weight}} \left\{ A^{\varnothing}_{V_{u, u \lhd v}}, \quad \{v\} \cup A^{\{v\}}_{V_{u, u \lhd v}} \right\}.$$

- Let $u \in V_G$ and $v, w \in V_G \setminus S$ such that $u < v < w$ and $\{u, v, w\}$ is a clique. Lemma 7.5.6 and Lemma 7.5.8 imply that

$$A^{\{w\}}_{V_{u,v}} = \max_{\text{weight}} \left\{ A^{\{w\}}_{V_{u, u \lhd v}}, \right.$$
$$\left. \{v\} \cup V[\lhd vw; \{u\}; \{u \lhd v\}] \cup \bigcup_{u' \in V_u \cap \lhd vw} A^{\{v,w\} \cap N(u')}_{V_{u', u' \lhd v}} \right\}.$$

Regarding the correctness of the algorithm, observe that applying any of the above recursive formulas requires only sets A^Y_X that can also be computed via these formulas.

To evaluate the running time of the algorithm, we assume that the input graph is a connected rooted path graph. If not, observe that we can simply run our algorithm on each connected component of the input graph and subsequently combine all output solutions into a solution for the input graph. Notice that the number of sets A_X^Y described above that the algorithm computes and subsequently stores their values in corresponding table entries is $\mathcal{O}(nm)$. Computing a single such set A_X^Y via any of the recursive formulas shown above requires the retrieval of stored values from $\mathcal{O}(n)$ entries. These entries are determined via precomputed auxiliary objects. For some of these formulas, their application additionally requires the computation of a set $V[V_1; V_2; V_3]$. It is not difficult to see that any such set can be computed via a single transversal of the host tree T'. As there are $\mathcal{O}(n)$ nodes in T', traversing it once takes $\mathcal{O}(n)$ time. Thus the total processing time is $\mathcal{O}(n^2m)$. Now consider the aforementioned auxiliary objects: they are the vertex sets $\lhd v$, $\lhd vw$ and $V_u \cap \lhd vw$ and the vertices $u \lhd v$ for appropriate $u, v, w \in V_G$. For computing the vertex sets $\lhd v$, it is sufficient the traverse T' once for every $v \in V_G$. Similarly, for computing the vertex sets $\lhd vw$, it is sufficient to traverse T' once for every $v, w \in V_G$ such that $v < w$ and $vw \in E_G$, and for computing the vertices $u \lhd v$, it is sufficient to traverse T' once for every $u, v \in V_G$ such that $u < v$ and $uv \in E_G$. We also determine for every pair (x, y) of distinct nodes of T' whether $x < y$ or not. As mentioned in the proof of Theorem 7.4.4, this can be accomplished in $\mathcal{O}(n^2)$ time. Then for every $u, v, w \in V_G$ such that $u < v < w$ and $\{u, v, w\}$ is a clique, we compute the vertex set $V_u \cap \lhd vw$ in $\mathcal{O}(n)$ time by checking for every $u' \in \lhd vw$ whether $u' \leq u$. Thus the total preprocessing time is $\mathcal{O}(n^2m)$. Therefore, the total running time of our algorithm is $\mathcal{O}(n^2m)$. $\square$

7.5.2 Undirected Path Graphs

The results of Theorem 7.4.4 and Corollary 7.4.5 motivate us to investigate whether our approach can be further extended to provide similar results on larger classes of chordal graphs. The class of graphs with bounded vertex leafage is a natural candidate to consider for such an investigation. However we show that Subset Feedback Vertex Set is NP-complete on undirected path graphs which are a subclass of graphs with vertex leafage at most two. In particular, we provide a polynomial reduction from the NP-complete Max Cut problem. Given a graph G, the Max Cut problem concerns the finding of a partition of $V(G)$ into two sets A and $\overline{A}$ such that the number of edges with one endpoint in A and the other one in $\overline{A}$ is maximum among all such partitions. For two disjoint sets of vertices X and Y, we denote by $E(X, Y)$ the set $\{\{x, y\} \mid x \in X \text{ and } y \in Y\}$. The *cut-set* of a set $A \subseteq V(G)$ in G is the set of edges of G with exactly one endpoint in A, which is $E(A, V(G) \setminus A) \cap E(G)$. In such terminology, Max Cut concerns the finding of a set $A \subseteq V(G)$ such that its cut-set in G is of maximum size. The Max Cut problem is known to be NP-hard on general graphs [52] and to remain NP-hard even when the input graph is restricted to be a split or 3-colorable or undirected path graph [7]. We mention that our reduction is based on Max Cut on general graphs.

Towards the claimed reduction, to any graph G on n vertices and m edges, we will associate a graph H_G on $12n^2 + 4n + 2m$ vertices. First we describe the vertex set of H_G. For every vertex $v \in V(G)$, we consider the following sets:

- $X(v) = \{x_v^1, x_v^2, \dots, x_v^{2n}\}$ and $\overline{X}(v) = \{\overline{x}_v^1, \overline{x}_v^2, \dots, \overline{x}_v^{2n}\}$,
- $Y(v) = \{y_v^1, y_v^2, \dots, y_v^{2n+1}\}$ and $\overline{Y}(v) = \{\overline{y}_v^1, \overline{y}_v^2, \dots, \overline{y}_v^{2n+1}\}$,
- $Z(v) = \{z_v^1, z_v^2, \dots, z_v^{2n+1}, \overline{z}_v^1, \overline{z}_v^2, \dots, \overline{z}_v^{2n+1}\}$, and
- $W(v) = \{(v, v') \mid \{v, v'\} \in E(G)\}$.

We consider all these sets to be pairwise-disjoint. The vertex set of H_G is precisely the union of all these sets. Notice that for every edge $\{u, v\} \in E(G)$, the ordered pairs (u, v) and (v, u) are both vertices of H_G. We denote by $\overline{W}(v)$ the set $\{(v', v) \mid \{v', v\} \in E(G)\}$. The edge set of H_G contains precisely the following:

- all edges required for the set $\bigcup_{v \in V(G)} (Y(v) \cup \overline{Y}(v) \cup W(v))$ to be a clique and
- for every vertex $v \in V(G)$:
  - all elements of the sets $E(X(v), Y(v))$, $E(\overline{X}(v), \overline{Y}(v))$, $E(X(v), W(v))$, $E(\overline{X}(v), \overline{W}(v))$,
  - for every $i \in [n]$, the edges $\{x_v^i, x_v^{n+i}\}, \{\overline{x}_v^i, \overline{x}_v^{n+i}\}$, and
  - for every $j \in [2n+1]$, the edges $\{y_v^j, z_v^j\}, \{y_v^j, \overline{z}_v^j\}, \{\overline{y}_v^j, z_v^j\}, \{\overline{y}_v^j, \overline{z}_v^j\}$.

Observe that x_v^i, x_v^{n+i} are true twins and $\overline{x}_v^i, \overline{x}_v^{n+i}$ are true twins, whereas $z_v^j, \overline{z}_v^j$ are false twins. This completes the construction of H_G. An example of a graph G and its associated graph H_G is given in Figure 7.4.

Lemma 7.5.10. *For any graph G, the graph H_G is an undirected path graph.*

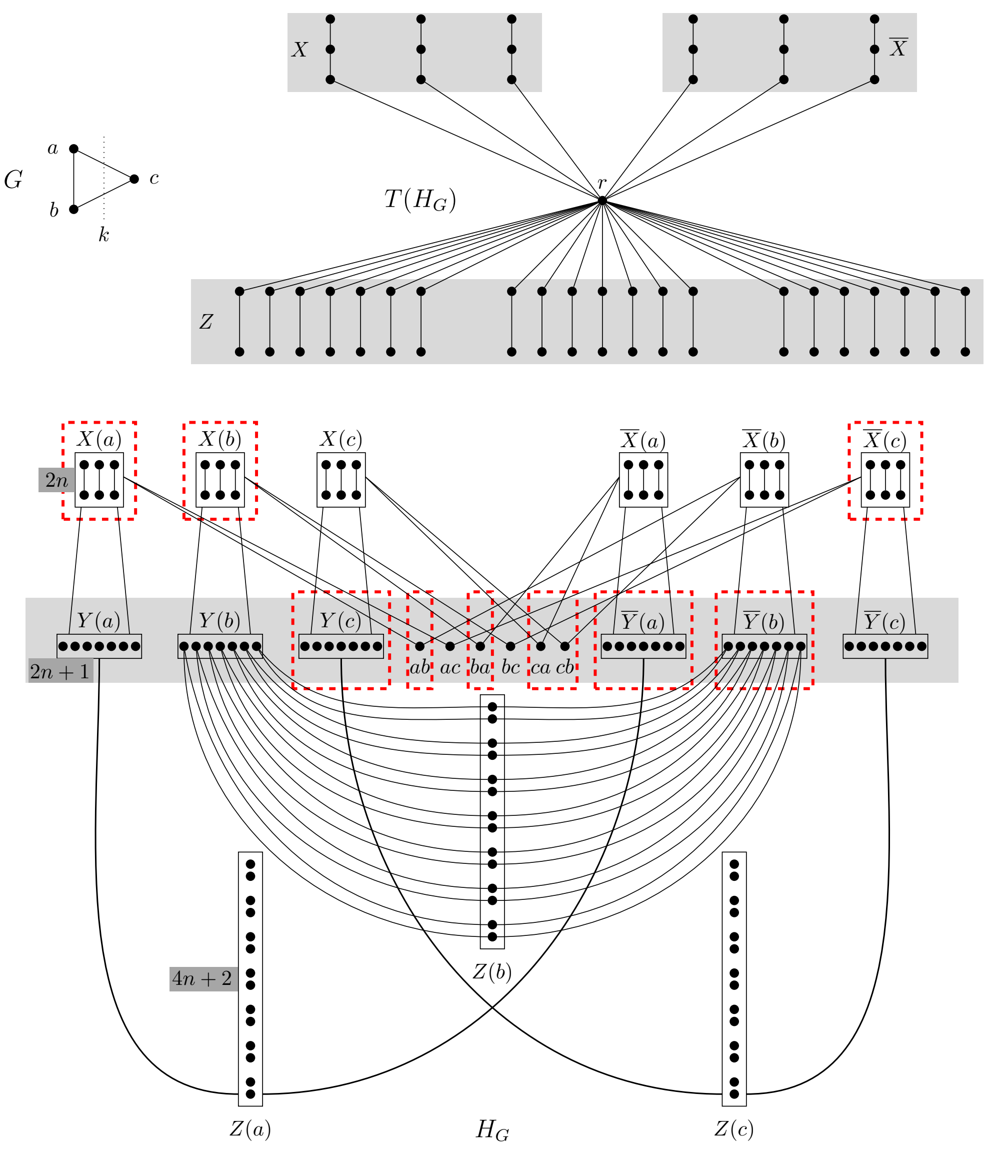


Figure 7.4: Illustration of a graph G on three vertices (top left) and its associated undirected path graph H_G (bottom). We also show the tree $T(H_G)$ described in the proof of Lemma 7.5.10 (top right). The vertices of H_G that lie in the gray area form a clique.

Proof. In order to show that H_G is an undirected path graph, we construct an undirected tree $T(H_G)$ such that the vertices of H_G correspond to particular paths of $T(H_G)$. To distinguish the vertex sets between G and $T(H_G)$, we refer to the vertices of $T(H_G)$ as nodes. In order to construct $T(H_G)$, starting from a particular node r, for every vertex $v \in V(G)$, we consider the following paths:

- $P_X(v) = \langle r, x_1^{(v)}, \dots, x_n^{(v)} \rangle$ and $P_{\overline{X}}(v) = \langle r, \overline{x}_1^{(v)}, \dots, \overline{x}_n^{(v)} \rangle$ and
- for every $j \in [2n+1]$, $P_Z(v,j) = \langle r, z_1^{(v,j)}, z_2^{(v,j)} \rangle$.

Notice that the initial node r is contained in all these paths. We consider them to be otherwise pairwise-disjoint. The tree $T(H_G)$ is precisely the union of all these paths. Next, we describe the paths of $T(H_G)$ that correspond to the vertices of H_G.

- For every $v \in V(G)$ and $i \in [n]$, to each of the vertices x_v^i, x_v^{n+i} (resp. $\overline{x}_v^i$, $\overline{x}_v^{n+i}$) we correspond the path $\langle x_i^{(v)} \rangle$ (resp. $\langle \overline{x}_i^{(v)} \rangle$).
- For every $v \in V(G)$ and $j \in [2n+1]$,
  - to the vertices y_v^j and $\overline{y}_v^j$ we correspond the paths $\langle x_n^{(v)}, \dots, x_1^{(v)}, r, z_1^{(v,j)}, z_2^{(v,j)} \rangle$ and $\langle \overline{x}_n^{(v)}, \dots, \overline{x}_1^{(v)}, r, z_1^{(v,j)}, z_2^{(v,j)} \rangle$) respectively, and
  - to the vertices z_v^j and $\overline{z}_v^j$ we correspond the paths $\langle z_1^{(v,j)} \rangle$ and $\langle z_2^{(v,j)} \rangle$ respectively.
- For every $\{u,v\} \in E(G)$, to the vertices (u,v) and (v,u) we correspond the paths $\langle x_n^{(u)}, \dots, x_1^{(u)}, r, \overline{x}_1^{(v)}, \dots, \overline{x}_n^{(v)} \rangle$ and $\langle x_n^{(v)}, \dots, x_1^{(v)}, r, \overline{x}_1^{(u)}, \dots, \overline{x}_n^{(u)} \rangle$ respectively.

Now it is not difficult to see that the intersection graph of the collection that contains precisely all these paths is isomorphic to H_G. Observe that all paths containing node r correspond to the vertices of the clique $\bigcup_{v \in V(G)} Y(v) \cup \overline{Y}(v) \cup W(v)$ and for every $v \in V(G)$, all paths that are subpaths of $P_X(v)$ and $P_{\overline{X}}(v)$ correspond to the vertices of $X(v)$ and $\overline{X}(v)$ respectively, and all paths that are subpaths of $P_Z(v,j)$, $j \in [2n+1]$ correspond to the vertices of $Z(v)$. Therefore, H_G is an undirected path graph. □

Let us now show that to any cut-set in G, there is an associated subset feedback vertex set in H_G. We first introduce some additional notation: $X =$

$\bigcup_{v\in V(G)} X(v)$, $\overline{X} = \bigcup_{v\in V(G)} \overline{X}(v)$, $Y = \bigcup_{v\in V(G)} Y(v)$, $\overline{Y} = \bigcup_{v\in V(G)} \overline{Y}(v)$, $Z = \bigcup_{v\in V(G)} Z(v)$, $W = \bigcup_{v\in V(G)} W(v) = \bigcup_{v\in V(G)} \overline{W}(v)$ and for every $A, B \subseteq V(G)$, $W(A,B) = (\bigcup_{a\in A} W(a)) \cap (\bigcup_{b\in B} \overline{W}(b))$. For every $A \subseteq V(G)$, we denote by $\overline{A}$ the set $V(G) \setminus A$ for simplicity. We also define the vertex set

$$U(A) = \left(\bigcup_{v\in A} (X(v) \cup \overline{Y}(v))\right) \cup \left(\bigcup_{v\in \overline{A}} (\overline{X}(v) \cup Y(v))\right) \cup (W \setminus W(A, \overline{A})).$$

Observe that $|U(A)| = n(2n + (2n+1)) + (2m - |W(A,\overline{A})|) = 4n^2 + n + 2m - |W(A,\overline{A})|$ and $|W(A,\overline{A})|$ is the size of the cut-set of A in G.

Lemma 7.5.11. *Let G be a graph and let $A \subseteq V(G)$. Then $U(A)$ is a subset feedback vertex set of (H_G, S) where $S = X \cup \overline{X} \cup Z$.*

Proof. We show that the undirected path graph $H_G - U(A)$ is an S-forest. Assume for contradiction that there is an S-triangle S_t in $H_G - U(A)$. Then S_t contains at least one vertex of S. We consider the following three cases:

- Let $x \in V(S_t) \cap X$. Then there exists $v \in V(G)$ such that $x \in X(v)$. Since $X(v') \subseteq U(A)$ for every $v' \in A$, the vertex v must be in $\overline{A}$. By the construction of H_G, any vertex of $X(v)$ has exactly one neighbor in $X(v)$. Thus there exists $y \in V(S_t)$ such that $y \notin X(v)$. Again by the construction of H_G, the neighborhood of any vertex of $X(v)$ in $H_G - X(v)$ is $Y(v) \cup W(v)$, which implies that $y \in Y(v) \cup W(v)$. Hence we reach a contradiction to the definition of $U(A)$, since $Y(v) \subseteq U(A)$ and $W(v) \subseteq W \setminus W(A,\overline{A}) \subseteq U(A)$.
- Let $\overline{x} \in V(S_t) \cap \overline{X}$. Arguments that are completely symmetrical to the ones employed in the previous case yield a contradiction to the definition of $U(A)$.
- Let $z \in V(S_t) \cap Z$. Then there exists $v \in V(G)$ such that $z \in Z(v)$ and by the construction of H_G, there exist $y \in Y(v)$ and $\overline{y} \in \overline{Y}(v)$ such that $N(z) = \{y, \overline{y}\}$. Thus $V(S_t)$ must be $\{z, y, \overline{y}\}$, which implies that $y, \overline{y} \notin U(A)$. Hence we reach a contradiction to the definition of $U(A)$, since either $Y(v) \subseteq U(A)$ and $\overline{Y}(v) \cap U(A) = \varnothing$ or vice versa.

Since we obtained a contradiction in all three cases, we conclude that there is no S-triangle in $H_G - U(A)$. $\square$

Now we are ready to show the main result of this section. Its forward direction follows from the previous lemma. Its reverse direction is obtained through a series of claims. These claims imply a procedure which progressively reconfigures any arbitrary initial subset feedback vertex set of H_G until it becomes one that is associated to a cut-set of G.

Theorem 7.5.12. *The unweighted* Subset Feedback Vertex Set *decision problem is* NP*-complete on undirected path graphs.*

Proof. We provide a polynomial reduction from the NP-complete Max Cut problem. Given a graph G on n vertices and m edges for the Max Cut problem, we construct the graph H_G. Observe that the size of H_G is polynomial and the construction of H_G can be done in polynomial time. By Lemma 7.5.10, H_G is an undirected path graph. We set $S = X \cup \overline{X} \cup Z$. We claim that G admits a cut-set of size at least k if and only if (H_G, S) admits a subset feedback vertex set of size at most $4n^2 + n + 2m - k$.

Lemma 7.5.11 provides the forward direction. Here we show the reverse direction. For every $U \subseteq V(H_G)$, we denote by $\overline{U}$ the set $V(H_G) \setminus U$ for simplicity. We also define the vertex set $A_U = \{v \in V(G) \mid X(v) \subseteq U\}$. Let U be a subset feedback vertex set of (H_G, S). Then it is not difficult to see that $U(A_U) = U$ holds if and only if U satisfies the following four properties:

(1) $Z \subseteq \overline{U}$.

(2) For all $v \in V(G)$, either $X(v) \subseteq U$ or $X(v) \subseteq \overline{U}$, and either $\overline{X}(v) \subseteq U$ or $\overline{X}(v) \subseteq \overline{U}$.

(3) For all $v \in V(G)$, either $X(v) \cup \overline{Y}(v) \subseteq U$ and $\overline{X}(v) \cup Y(v) \subseteq \overline{U}$, or vice versa.

(4) $W \setminus W(A_U, \overline{A_U}) \subseteq U$ and $W(A_U, \overline{A_U}) \subseteq \overline{U}$.

For every $i \in \{0, 1, \ldots, 4\}$, we say that a subset of $V(H_G)$ is a *tier-i sfvs* if it is a subset feedback vertex set of (H_G, S) that satisfies the first i properties listed above. Notice that if a subset of $V(H_G)$ is a tier-i sfvs, then it is a tier-j sfvs for all $j \in \{0, 1, \ldots, i\}$. Also notice that any subset feedback vertex set of (H_G, S) is a tier-0 sfvs. We will now show through a series of claims that for every tier-0 sfvs, there exists a tier-4 sfvs of at most equal size.

Claim 7.5.13. *For every tier-*0 *sfvs U, there exists a tier-*1 *sfvs U' such that $|U'| \leq |U|$.*

Proof: Let U be a tier-0 sfvs. If $Z \cap U = \varnothing$, then U is already a tier-1 sfvs. Assume otherwise. Then there exist $v \in V(G)$ and $j \in [2n+1]$ such that $\{z_v^j, \overline{z}_v^j\} \cap U \neq \varnothing$. We construct the set $U' = (U \setminus \{z_v^j, \overline{z}_v^j\}) \cup \{\overline{y}_v^j\}$. Notice that $|\{z_v^j, \overline{z}_v^j\} \cap U| \geq 1$ and $|\{\overline{y}_v^j\} \setminus U| \leq 1$, yielding $|U'| \leq |U|$. Observe that by the construction of H_G, the neighborhoods of z_v^j and $\overline{z}_v^j$ in $H_G - U'$ are the same subset of $\{y_v^j\}$, thus neither z_v^j nor $\overline{z}_v^j$ is a vertex of any triangle of $H_G - U'$. As z_v^j and $\overline{z}_v^j$ are the only vertices being removed from a tier-0 sfvs, this implies that U' is also a tier-0 sfvs. Iteratively following this argumentation for every $v \in V(G)$ and $j \in [2n+1]$ such that $\{z_v^j, \overline{z}_v^j\} \cap U \neq \varnothing$, we obtain a tier-1 sfvs U' such that $|U'| \leq |U|$. ⌟

Claim 7.5.14. *For every tier-*1 *sfvs U and $v \in V(G)$, it holds that $|(Y(v) \cup \overline{Y}(v)) \cap U| \geq 2n+1$.*

Proof: Let U be a tier-1 sfvs. Consider a vertex $v \in V(G)$. Then $Z(v) \subseteq \overline{U}$. Since $Z(v) \subset S$ also holds, the triangles induced by the sets $\{y_v^j, \overline{y}_v^j, \overline{z}_v^j\}$, $j \in [2n+1]$ are $2n+1$ vertex-disjoint S-triangles of H_G. Therefore, at least $2n+1$ vertices of $Y(v) \cup \overline{Y}(v)$ must be in U, because $H_G - U$ is an S-forest. ⌟

Claim 7.5.15. *For every tier-*1 *sfvs U, there exists a tier-*2 *sfvs U' such that $|U'| \leq |U|$.*

Proof: Let U be a tier-1 sfvs. Consider a vertex $v \in V(G)$ such that both $X(v) \cap U \neq \varnothing$ and $X(v) \cap \overline{U} \neq \varnothing$. Assume that $x \in X(v) \cap \overline{U}$. By the construction of H_G, we have that $N(x) \setminus X(v) = Y(v) \cup W(v)$ is a clique. Since $X(v) \subset S$ and $H_G - U$ is an S-forest, at most one vertex of $Y(v) \cup W(v)$ is in $\overline{U}$. We construct the set $U' = (U \setminus X(v)) \cup (Y(v) \cup W(v))$. Notice that $|X(v) \cap U| \geq 1$ and $|(Y(v) \cup W(v)) \setminus U| \leq 1$, yielding $|U'| \leq |U|$. The set U' is a tier-1 sfvs. This follows from the fact that by the construction of H_G, there are no triangles in $H_G[X(v)]$ and $N(X(v)) = Y(v) \cup W(v) \subseteq U'$. Now consider a vertex $v \in V(G)$ such that both $\overline{X}(v) \cap U \neq \varnothing$ and $\overline{X}(v) \cap \overline{U} \neq \varnothing$. Via completely symmetrical arguments, the set $U'' = (U' \setminus \overline{X}(v)) \cup (\overline{Y}(v) \cup \overline{W}(v))$ is a tier-1 sfvs such that $|U''| \leq |U'|$. Iteratively following the argumentations regarding all applicable cases for every $v \in V(G)$, we obtain a tier-2 sfvs U' such that $|U'| \leq |U|$. ⌟

Before we continue with our claims, we observe that for every tier-0 sfvs U and $v \in V(G)$, if $X(v) \subseteq \overline{U}$, then $Y(v) \cup W(v) \subseteq U$, and if $\overline{X}(v) \subseteq \overline{U}$, then $\overline{Y}(v) \cup \overline{W}(v) \subseteq U$. This follows from the facts that $H_G - U$ is an S-forest, $X(v) \cup \overline{X}(v) \subset S$ and by the construction of H_G, for every $y \in Y(v) \cup W(v)$

(resp. $\overline{y} \in \overline{Y}(v) \cup \overline{W}(v)$), the set $\{x_v^1, x_v^{n+1}, y\}$ (resp. $\{\overline{x}_v^1, \overline{x}_v^{n+1}, \overline{y}\}$) induces a triangle of H_G. In proving our remaining claims, we implicitly apply this observation.

Claim 7.5.16. *For every tier-*2 *sfvs U, there exists a tier-*3 *sfvs U' such that $|U'| \leq |U|$.*

Proof: Let U be a tier-2 sfvs. Consider a vertex $v \in V(G)$. Then exactly one of the following holds:

(1) $X(v) \cup \overline{X}(v) \subseteq \overline{U}$

(2) $X(v) \subseteq U$ and $\overline{X}(v) \subseteq \overline{U}$

(3) $X(v) \subseteq \overline{U}$ and $\overline{X}(v) \subseteq U$

(4) $X(v) \cup \overline{X}(v) \subseteq U$

Assume that (1) holds. Then $Y(v) \cup \overline{Y}(v) \subseteq U$ holds. We construct the set $U' = (U \setminus Y(v)) \cup X(v)$. Notice that $|Y(v) \cap U| = 2n+1$ and $|X(v) \setminus U| = 2n$, yielding $|U'| < |U|$. It is not difficult to show that the set U' is a tier-2 sfvs.

Now assume that (2) holds. Then $\overline{Y}(v) \subseteq U$ holds. We construct the set $U' = U \setminus Y(v)$. Clearly, $|U'| \leq |U|$. It is not difficult to show that the set U' is a tier-2 sfvs. Assuming that (3) holds, completely symmetrical arguments yield that the set $U' = U \setminus \overline{Y}(v)$ is a tier-2 sfvs such that $|U'| \leq |U|$.

Lastly assume that (4) holds. By Claim 7.5.14, we have $|(Y(v) \cup \overline{Y}(v)) \setminus U| \leq 2n+1$. Without loss of generality, assume that $|\overline{Y}(v) \setminus U| \leq n$. We construct the set $U' = (U \setminus (\overline{X}(v) \cup Y(v))) \cup (\overline{Y}(v) \cup \overline{W}(v))$. Notice that $|(\overline{X}(v) \cup Y(v)) \cap U| \geq 2n+0 = 2n$ and $|(\overline{Y}(v) \cup \overline{W}(v)) \setminus U| \leq n+(n-1) < 2n$, yielding $|U'| < |U|$. It is not difficult to show that the set U' is a tier-2 sfvs.

Iteratively following the argumentation regarding the appropriate case for every $v \in V(G)$, we obtain a tier-3 sfvs $|U'|$ such that $|U'| \leq |U|$. ⌟

Claim 7.5.17. *For every tier-*3 *sfvs U, there exists a tier-*4 *sfvs U' such that $|U'| \leq |U|$.*

Proof: Let U be a tier-3 sfvs. Consider a vertex $w \in W$. Then there exist $u, v \in V(G)$ such that $w = (u, v)$ is the (unique) vertex of $W(u) \cap \overline{W}(v)$. Assume that $w \in W \setminus W(A_U, \overline{A_U})$. Then $u \notin A_U$ or $v \notin \overline{A_U}$, which implies that $X(u) \subseteq \overline{U}$ or $\overline{X}(v) \subseteq \overline{U}$. In both cases, it follows that $w \in U$. Now assume that $w \in W(A_U, \overline{A_U})$. Then $u \in A_U$ and $v \in \overline{A_U}$, which implies that $X(u) \subseteq U$ and $\overline{X}(v) \subseteq U$. By the construction of H_G, we have that $N(w) \cap S = X(u) \cup \overline{X}(v)$. It follows that the set $U' = U \setminus W(A_U, \overline{A_U})$ is a tier-4 sfvs such that $|U'| \leq |U|$. ⌟

To conclude our proof, we assume that U is a tier-0 sfvs such that $|U| \leq 4n^2+n+2m-k$. Due to Claims 7.5.13, 7.5.15–7.5.17, there exists a tier-4 sfvs U' such that $|U'| \leq |U|$. Then $U(A_{U'}) = U'$ holds. Recall that $|U(A_{U'})| = 4n^2+n+2m - |W(A_{U'}, \overline{A_{U'}})|$ and $|W(A_{U'}, \overline{A_{U'}})|$ is the size of the cut-set of $A_{U'}$ in G. All of the above imply that the size of the cut-set of $A_{U'}$ in G is at least k. □

8 SFVS, FVS, NMC and UNMC on H-free Graphs

In this chapter, we present our results for SFVS, FVS, NMC and UNMC on H-free graphs and in particular on $(\alpha+1)K_1$-free graphs, $\alpha \in \mathbb{N}$. Section 8.1 contains our results for SFVS: a polynomial-time algorithm for solving the weighted SFVS problem on $4K_1$-free graphs, NP-hardness of the weighted SFVS problem on $5K_1$-free graphs and a polynomial-time algorithm for solving the unweighted SFVS problem on $(\alpha+1)K_1$-free graphs. In Section 8.2, we provide a polynomial-time algorithm for solving the weighted FVS problem on $(\alpha+1)K_1$-free graphs and an alternative reduction for showing a matching $\mathsf{W}[1]$-hardness of the unweighted FVS problem. Our results for the unweighted NMC and weighted UNMC problems which are corollaries of the results in Section 8.1 are contained in Sections 8.3 and 8.4 respectively. In Section 8.4, we also provide a polynomial-time algorithm for solving the unweighted UNMC problem on $(\alpha+1)K_1$-free graphs. All of the above are results of a study of these problems on graphs with bounded independent set number. Recall that:

- graphs with independent set number at most $\alpha \equiv (\alpha+1)K_1$-free graphs
- graphs with clique cover number at most $\alpha \equiv$ co-α-partite graphs

Also observe that the relations illustrated in Figure 8.1 hold. The results presented in this chapter were published in the following works:

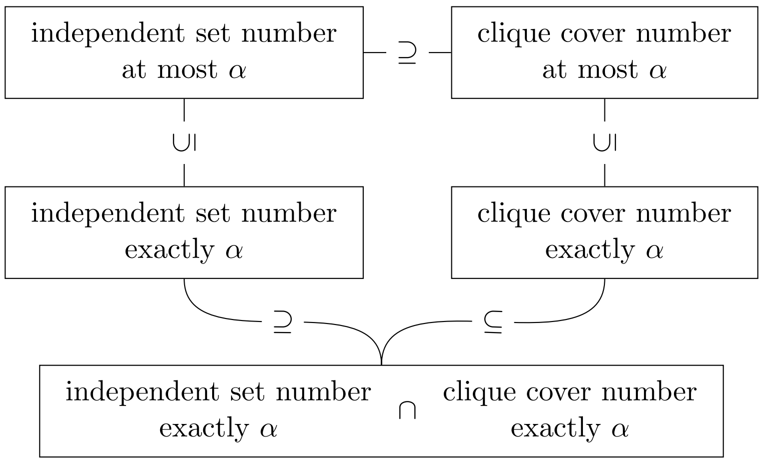

independent set number at most α
$\supseteq$
clique cover number at most α
$\cup|$
$\cup|$
independent set number exactly α
clique cover number exactly α
$\supseteq$
$\subseteq$
independent set number exactly α
$\cap$
clique cover number exactly α

Figure 8.1: Relations between the graph classes appearing in this chapter.

- Charis Papadopoulos and Spyridon Tzimas. Subset Feedback Vertex Set on Graphs of Bounded Independent Set Size. 13th International Symposium on Parameterized and Exact Computation (IPEC 2018). *Leibniz International Proceedings in Informatics (LIPIcs)*, 115:20:1–20:14 (2019).
- Charis Papadopoulos and Spyridon Tzimas. Subset feedback vertex set on graphs of bounded independent set size. *Theoretical Computer Science*, 814:177–188 (2020).

However, we note that the results are presented in reanalysed form in light of the aforementioned relations between the graph classes considered in this chapter.

8.1 SFVS on Graphs with Bounded Independent Set Number

Let us give a couple of observations on the nature of SFVS on graphs with bounded independent set number. Firstly note that the bound on the size of an independent set is a hereditary property; for every induced subgraph H of G, we have $\alpha(H) \leq \alpha(G)$. Moreover for any clique C of G, any S-forest of G contains at most two vertices of $S \cap C$.

Observation 8.1.1. *Let G be a graph with $\alpha(G) \leq \alpha$ and let $S \subseteq V$.*

(1) For every set X of at least $2\alpha + 1$ vertices, there exists a cycle in $G[X]$.

(2) Every S-forest of G has at most 2α vertices from S.

Proof. Let X be a set of at least $2\alpha+1$ vertices. Assume that $G[X]$ is a forest. As an induced subgraph of G, any independent set of $G[X]$ has size at most α. Since $G[X]$ is acyclic, there is a proper 2-coloring A, B of the vertices of $G[X]$ such that $|A| \geq |B|$. By the fact that $|A| \leq \alpha$, we conclude that $|A|+|B| \leq 2\alpha$, leading to a contradiction that $|X| \geq 2\alpha + 1$. Thus $G[X]$ contains a cycle.

For the second statement, let $F = (V_F, E_F)$ be an S-forest of G. By the first statement, if $S \cap V_F$ has at least $2\alpha + 1$ vertices then there is a cycle in $F[S \cap V_F]$, which implies an S-cycle in F. Thus $|S \cap V_F| \leq 2\alpha$. □

We note that Observation 8.1.1 allows us to construct by brute force all possible subsets of S belonging to any S-forest in $n^{\mathcal{O}(\alpha)}$ time.

8.1.1 Weighted SFVS

Here we consider the weighted SFVS problem and we show a complexity dichotomy result with respect to the independent set number. We first provide a polynomial-time algorithm for solving the weighted SFVS optimization problem on graphs with independent set number at most three and then we show that the weighted SFVS decision problem is NP-complete on graphs with independent set number at least four.

Let (G, S) be an instance of the weighted SFVS optimization problem such that $\alpha(G) = \alpha$. Let $H = (V_H, E_H)$ be an induced subgraph of G. Let $S_0 = S \cap V_H$ and let $S_1 = N_H(S_0)$. Furthermore, we denote by $S_{\leq 1}$ the set $S_0 \cup S_1$. We partition the graph H into two induced subgraphs $H_{\leq 1}$ and $H_{>1}$ as follows:

- $H_{\leq 1}$ is the subgraph $H[S_{\leq 1}]$ of H that is induced by the vertices that are at distance at most one from the vertices of S_0.
- $H_{>1}$ is the subgraph $H - S_{\leq 1}$ of H that is induced by the vertices that are at distance at least two from the vertices of S_0.

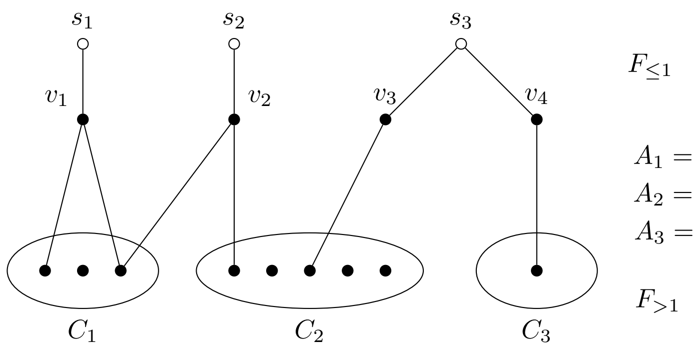


Figure 8.2: Illustrating an S-distance partition $(F_{\leq 1}, F_{>1})$ of an S-forest F with $S = \{s_1, s_2, s_3\}$ that shows the connected components C_1, C_2, C_3 of $F_{>1}$. The edges inside $F_{>1}$ are not drawn in order to highlight that the cut satisfies the given tuple (A_1, A_2, A_3).

Such a partition is called the S*-distance partition* of H, denoted by $(H_{\leq 1}, H_{>1})$. The set of edges of H having one endpoint in $H_{\leq 1}$ and the other in $H_{>1}$ is called *the cut* of the partition $(H_{\leq 1}, H_{>1})$. Notice that a vertex of $H_{\leq 1}$ that is adjacent to a vertex of $H_{>1}$ belongs to S_1.

Let $(C_1, \ldots, C_d)$ be an ordering of the partition of the vertices of $H_{>1}$ such that each C_i, $1 \leq i \leq d$, induces a connected component in $H_{>1}$. Because $H_{>1}$ is an induced subgraph of G, it is clear that $d \leq \alpha$. Let $(A_1, \ldots, A_d)$ be a tuple of d subsets of S_1, i.e., each $A_i \subseteq S_1$ holds. Observe that $(A_1, \ldots, A_d)$ neither partitions nor covers the set S_1. We say that *the cut satisfies* the tuple $(A_1, \ldots, A_d)$ if for any vertex $v \in C_i$, we have $(N(v) \cap S_{\leq 1}) \subseteq A_i$. Recall that an S-forest is an induced subgraph of G. Thus, an S-forest F of G admits an S-distance partition $(F_{\leq 1}, F_{>1})$. The notion of an S-distance partition of F with the corresponding cut is illustrated in Figure 8.2.

We now utilize the S-distance partition of H in order to construct an algorithm that solves the weighted SFVS optimization problem on graphs with independent set number α and subsequently show that this algorithm is efficient for $\alpha \leq 3$. Our general approach relies on the following facts:

- By Observation 8.1.1 (2), we try all subsets S' of S with at most 2α vertices and keep those sets that induce a forest. This step is used in constructing the vertices of S within the graph $H_{\leq 1}$. In particular, for each such set S', we construct all $H_{\leq 1}$ such that $S_0 = S'$. We will show that the number of such subsets produced is bounded by $n^{\mathcal{O}(\alpha)}$.

- For each of the potential subgraphs $H_{\leq 1}$ constructed in the previous step, and for each $d \leq \alpha$, we determine all possible tuples $(A_1, \ldots, A_d)$ with $A_i \subseteq S_1$ having the following property: every induced subgraph of G whose S-distance partition's first part is $H_{\leq 1}$ and its cut satisfies the tuple $(A_1, \ldots, A_d)$ is indeed an S-forest F of G. We show that considering only these tuples is sufficient in Lemma 8.1.2.

- Up to that point, we can show that all steps can be executed in time $n^{\mathcal{O}(\alpha)}$. However for the next and final step we can only achieve polynomial running time if we restrict ourselves to $\alpha \leq 3$ due to the number of connected components of $F_{>1}$. For each tuple computed in the previous step, we find connected components $C_1, \ldots, C_d$ of maximum cumulative weight such that the cut of $(F_{\leq 1}, G[C_1 \cup \cdots \cup C_d])$ satisfies the tuple. For doing so, we take advantage of the small number of connected components ($d \leq 3$) and an efficient way of computing a vertex-cut between such components.

We begin by showing that the S-distance partition of H provides a useful tool towards computing a maximum S-forest. Given a set of vertices $X \subseteq N[S]$ and d subsets A_i of $X \setminus S$, we construct the graph $\mathsf{aux}_{A_1,\ldots,A_d}(X)$ that is obtained from $G[X]$ by adding d vertices $w_1, \ldots, w_d$ such that every vertex w_i is adjacent to all the vertices of A_i. In what follows, we always assume that G is a graph having independent set size α.

Lemma 8.1.2. *Let F be an S-forest of G with S-distance partition $(F_{\leq 1}, F_{>1})$ such that $S_0 \neq \varnothing$. Then for some $d \leq \alpha$, there is a tuple $(A_1, \ldots, A_d)$ with $A_i \subseteq S_1$ such that*

(i) the cut of $(F_{\leq 1}, F_{>1})$ satisfies $(A_1, \ldots, A_d)$ and

(ii) every induced subgraph H of G with S-distance partition $(H[S_{\leq 1}], H - S_{\leq 1})$ that satisfies $(A_1, \ldots, A_d)$ is an S-forest.

Proof. Let $(C_1, \ldots, C_d)$ be an ordering of the partition of the vertices of $F_{>1}$ such that every C_i induces a connected component in $F_{>1}$. We define a tuple $(A_1, \ldots, A_d)$ in which every $A_i = N(C_i) \cap S_{\leq 1}$, for $1 \leq i \leq d$. Clearly $A_i \subseteq S_1$ since every vertex $F_{>1}$ is at distance at least two from S_0. Thus, by construction, the cut of $(F_{\leq 1}, F_{>1})$ satisfies the tuple $(A_1, \ldots, A_d)$.

For the next claim, we first show that $\widehat{G} := \mathsf{aux}_{A_1,\ldots,A_d}(S_{\leq 1})$ is an S-forest. Assume for contradiction that there is an S-cycle $\widehat{C}$ in $\widehat{G}$. Since $F_{\leq 1}$ does not contain any S-cycle, $\widehat{C}$ contains a vertex w_i and at least two vertices u_i, v_i from A_i, for some $1 \leq i \leq d$. By the fact that $A_i = N(C_i) \cap S_{\leq 1}$, there are vertices x_i and y_i, not necessarily distinct, in C_i that are adjacent to u_i and v_i, respectively. Since C_i induces a connected component of $F_{>1}$, there is a path between x_i and y_i that lies entirely in $F_{>1}[C_i]$. This means that we can replace the vertex w_i of $\hat{C}$ by a path between x_i and y_i for every i, to obtain an S-cycle in F, leading to a contradiction. Thus, $\widehat{G}$ is an S-forest.

Let H be an induced subgraph of G with S-distance partition $(H[S_{\leq 1}], H - S_{\leq 1})$ that satisfies $(A_1, \ldots, A_d)$. Observe that $H[S_{\leq 1}] = F_{\leq 1}$ as they are induced subgraphs of the same vertex set of G. Thus $H[S_{\leq 1}]$ does not contain any S-cycle, because F is an S-forest. Since the cut of $(H[S_{\leq 1}], H - S_{\leq 1})$ satisfies $(A_1, \ldots, A_d)$, there is a partition $(T_1, \ldots, T_d)$ of the vertices of $H - S_{\leq 1}$ such that T_i is a connected component of $H - S_{\leq 1}$ and $N(T_i) \subseteq A_i$, for $1 \leq i \leq d$. We show that H is indeed an S-forest. For contradiction, assume an S-cycle C in H. There are no S-cycles in $H[S_{\leq 1}]$ which implies that $C \cap T_i \neq \varnothing$, for some $1 \leq i \leq d$. For every such set, we replace the part $C \cap T_i$ by a vertex w'_i. Denote by H' the resulting graph. Notice that $H'[C]$ is a subgraph of $\widehat{G}[C]$ because $N_{H'}(w'_i) \subseteq N_{\widehat{G}}(w_i)$. This, however, implies an S-cycle in $\widehat{G}$, which gives the desired contradiction. Therefore, H is an S-forest. □

Notice that $G - S$ is trivially an S-forest of G. Moreover, $G - S$ is maximal among all S-forests of G such that $S_0 = \varnothing$. In what follows, we assume that $S_0 \neq \varnothing$ and show how to bound the vertex set $S_{\leq 1}$ of $F_{\leq 1}$.

Lemma 8.1.3. *Let F be an S-forest of G such that $S_0 \neq \varnothing$.*

1. *If $|S_0| \leq 2\alpha - 2$ then $|S_{\leq 1}| \leq 4\alpha - 2$.*
2. *If $|S_0| \geq 2\alpha - 1$ then $|S_{\leq 1}| \leq 2\alpha$.*

Proof. Let F be such an S-forest of G with $|S_0| \geq 1$. By Observation 8.1.1 (2), we know that $|S_0| \leq 2\alpha$. Notice that $|S_{\leq 1}| = |S_0| + |S_1|$. We consider separately the two cases of the claim.

Case 1. Let $1 \leq |S_0| \leq 2\alpha - 2$. Assume for contradiction that $|S_{\leq 1}| > 4\alpha - 2$. We show that $F[S_1]$ contains a matching with at least α edges. Applying Observation 8.1.1 (1) shows that there is a cycle C in $F[S_{\leq 1}]$. Since F is an S-forest, this is not an S-cycle, so all vertices contained in C are vertices of S_1. Let $V_M = \varnothing$. Iteratively adding the two endpoints of an edge of C to V_M

and applying Observation 8.1.1 (1) to $F - V_M$ as long as $|S_{\leq 1} \setminus V_M| > 2\alpha$, we identify α edges of S_1 such that all their endpoints are distinct. Thus, $F[S_1]$, and in particular $F[V_M]$, contains a matching M with at least α edges.

Let $C_1, \ldots, C_d$ be the connected components of $F[S_0]$. Notice that $d \leq \alpha$ because $F[S_0]$ is an induced subgraph of a graph with maximum independent set size α. By construction, every vertex of S_1 is adjacent to at least one vertex of S_0. If the endpoints of an edge of M in S_1 are adjacent to vertices of the same component C_i, $1 \leq i \leq d$, then there is an S-cycle in F since every vertex of C_i belongs to S. Thus the endpoints of every edge of M are adjacent to different connected components of $F[S_0]$. Now obtain a bipartite graph by contracting every component C_i into a single vertex and every edge of M into a single vertex and keep only the adjacencies between the components and the edges of M. Let (A, B) be the bipartition of the resulting bipartite graph such that A contains the components of $F[S_0]$ and B contains the edges of M. Since $|A| \leq |B|$ and every vertex of B is adjacent to at least two vertices of A, there is a cycle in the bipartite graph. Then, it is not difficult to see that the cycle of the contracted vertices corresponds to an S-cycle in F. Therefore there is an S-cycle in an S-forest, leading to a contradiction.

Case 2. Let $2\alpha - 1 \leq |S_0| \leq 2\alpha$. Assume for contradiction that $|S_{\leq 1}| > 2\alpha$. We pick a subset W of S_1 such that $|S_0| + |W| = 2\alpha + 1$. Notice that $1 \leq |W| \leq 2$. Then Observation 8.1.1 (1) implies that there is a cycle in $F[S_0 \cup W]$. Since W has at most two vertices, we conclude that the induced cycle of $F[S_0 \cup W]$ has at least one vertex from S, hence it is an S-cycle in F. Therefore, we reach a contradiction which implies that $|S_{\leq 1}| \leq 2\alpha$. $\square$

Lemma 8.1.3 shows that we can compute all possible candidates for $S_{\leq 1}$ in polynomial time as follows.

- We first construct, by brute force, all subsets S' of S having at most 2α vertices, according to Observation 8.1.1 (2).
- Then, for each such subset S', we incorporate a set $Y \subseteq N(S') \setminus S$ for which either $|S'| + |Y| \leq 4\alpha - 2$, or $|S'| + |Y| \leq 2\alpha$, according to Lemma 8.1.3.
- Given the described sets S' and Y, we check if $G[S' \cup Y]$ induces an S-forest and, if so, we include $S' \cup Y$ into a list L_1 containing all candidates for $S_{\leq 1}$.

The correctness follows from Observation 8.1.1 and Lemma 8.1.3. Regarding the running time, notice that we create at most $n^{\mathcal{O}(\alpha)}$ subsets for each of S' and $Y \subseteq N(S') \setminus S$. Thus, in $n^{\mathcal{O}(\alpha)}$ time we can compute a list L_1 that contains all possible candidates for (the solution's) set $S_{\leq 1}$.

Let $S_{\leq 1}$ be a set of L_1. We now focus on the graph $G' = G - (S_{\leq 1} \cup S)$ that contains the vertices that are at distance of at least two from S_0. Recall that we assume $S_0 \neq \varnothing$, by the discussion prior to Lemma 8.1.3. Let d be the number of connected components of G'. It is clear that $d \leq \alpha$. In fact, since $|S_0| \geq 1$ and the vertices of G' are at distance of at least two from S_0, we have $d < \alpha$.

By brute force, we find all tuples $(A_1, \ldots, A_d)$ such that the following hold:

(i) $A_i \subseteq S_1$, for every $1 \leq i \leq d$, and

(ii) the graph $\mathsf{aux}_{A_1,\ldots,A_d}(S_{\leq 1})$ is an S-forest.

Notice that by the proof of Lemma 8.1.2 (ii) it is sufficient to consider only such tuples. Since $A_i \subseteq S_{\leq 1}$, $d < \alpha$, and $|S_{\leq 1}| \leq 4\alpha$, the number of tuples is $\alpha^{\mathcal{O}(\alpha)}$, so that we can obtain the desired set of tuples that satisfy both conditions in polynomial time.

In what follows, we consider the case for $\alpha \leq 3$. By the previous arguments, we are given a set $S_{\leq 1} \subseteq N[S]$ and tuples of the form A_1 or (A_1, A_2) which are subsets of S_1. Our task is to compute a subset V' of the vertices of G' such that the vertices of $S_{\leq 1} \cup V'$ induce a maximum S-forest and the cut of $(G[S_{\leq 1}], G[V'])$ satisfies A_1 or (A_1, A_2), respectively. We distinguish the two cases with the following two lemmas.

Lemma 8.1.4. *Let $X \subseteq N[S]$ and let A_1 be a subset of $X \setminus S$ such that both $G[X]$ and $\mathsf{aux}_{A_1}(X)$ are S-forests. There exists a polynomial-time algorithm that computes a maximum S-forest F such that $S_{\leq 1} = X$ and the cut of its S-distance partition $(F_{\leq 1}, F_{>1})$ satisfies A_1.*

Proof. Since $F_{\leq 1}$ is a fixed S-forest of F, we need to determine the vertices of $V \setminus (X \cup S)$ that are included in $F_{>1}$. By the desired cut of $(F_{\leq 1}, F_{>1})$, we are restricted to the vertices of $V \setminus (X \cup S)$ whose neighbors in $F_{\leq 1}$ are only vertices of A_1. Those vertices can be described as follows:

$$B_1 = \{w \in V \setminus (X \cup S) \mid N(w) \cap S_{\leq 1} \subseteq A_1\}.$$

Since the cut satisfies a single subset A_1, we have at most one connected component of $G[B_1]$ in $F_{>1}$. In order to choose the correct connected component of $G[B_1]$, we try to include each of them in $F_{>1}$ and select the one having the maximum total weight. Notice that adding any component of $G[B_1]$ into $F_{>1}$ cannot create any S-cycle, because $\mathsf{aux}_{A_1}(X)$ is an S-forest. Thus, by Lemma 8.1.2, we correctly compute a maximum S-forest with the desired properties. Clearly the set B_1 can be constructed in polynomial time. Since the number of connected components $G[B_1]$ is at most two, all steps can be executed in polynomial time. $\square$

Next, we consider the case when we have a tuple (A_1, A_2).

Lemma 8.1.5. *Let $X \subseteq N[S]$ and let A_1, A_2 be subsets of $X \setminus S$ such that both $G[X]$ and $\mathsf{aux}_{A_1,A_2}(X)$ are S-forests. There exists a polynomial-time algorithm that computes a maximum S-forest F such that $S_{\leq 1} = X$ and the cut of its S-distance partition $(F_{\leq 1}, F_{>1})$ satisfies (A_1, A_2).*

Proof. Similar to the proof of Lemma 8.1.4, we first construct the sets B_1 and B_2 that contain all vertices of $V \setminus (X \cup S)$ whose neighbors in $F_{\leq 1}$ are only vertices of A_1 and A_2 respectively:

$$B_1 = \{w \in V \setminus (X \cup S) \mid N(w) \cap S_{\leq 1} \subseteq A_1\} \quad \text{and}$$
$$B_2 = \{w \in V \setminus (X \cup S) \mid N(w) \cap S_{\leq 1} \subseteq A_2\}\,.$$

As the desired cut of $(F_{\leq 1}, F_{>1})$ satisfies (A_1, A_2), there are two connected components of $F_{>1}$ which are subsets of the two sets B_1 and B_2, respectively. Let C_1 and C_2 be the connected components of $F_{>1}$ such that $C_1 \subseteq B_1$ and $C_2 \subseteq B_2$. Now observe that there should be two non-adjacent vertices $c_1 \in B_1$ and $c_2 \in B_2$ that belong to C_1 and C_2, respectively. We iterate over all possible pairs of non-adjacent vertices $c_1 \in B_1 \cap C_1$ and $c_2 \in B_2 \cap C_2$ in $\mathcal{O}(n^2)$ time. Assuming a given choice for c_1 and c_2, observe the following:

- Since c_1 and c_2 are vertices of different connected components of $F_{>1}$, the components themselves are further restricted to be subsets of $B_1 \setminus N[c_2]$ and $B_2 \setminus N[c_1]$, respectively. That is, $C_1 \subseteq (B_1 \setminus N[c_2])$ and $C_2 \subseteq (B_2 \setminus N[c_1])$.
- Since F has at least one vertex of S, $c_1, c_2 \in V \setminus (X \cup S)$ are non-adjacent, and by the fact $d \leq 3$, we have that $B_1 \setminus N[c_2]$ and $B_2 \setminus N[c_1]$ induce cliques in G. Thus $B_1 \setminus N[c_2] \subseteq N[c_1]$ and $B_2 \setminus N[c_1] \subseteq N[c_2]$, respectively.

Then by the second statement it is not difficult to see that $B_1 \setminus N[c_2]$ and $B_2 \setminus N[c_1]$ are disjoint. Let $B_1' = (B_1 \setminus N[c_2]) \setminus \{c_1\}$ and $B_2' = (B_2 \setminus N[c_1]) \setminus \{c_2\}$. Now in order to find the maximum induced S-forest under the stated conditions and our assumption that c_1 and c_2 belong to the two connected components of $F_{>1}$, it suffices to find the maximum subset $C_1 \cup C_2$ of $B_1' \cup B_2'$ such that there are no edges between the vertices of $C_1 \cap B_1'$ and the vertices of $C_2 \cap B_2'$. This boils down to computing a minimum weighted vertex cover on the bipartite graph G' obtained from $G[B_1' \cup B_2']$ and removing the edges inside $G[B_1']$ and $G[B_2']$. By standard techniques using maximum flow arguments, we compute a minimum weighted vertex cover U on G' in polynomial time [64, 67]. Therefore, $G[B_1' \cup B_2'] - U$ contains the connected components $C_1 \setminus \{c_1\}$ and $C_1 \setminus \{c_2\}$, as required. □

Now we are equipped with the necessary tools in order to obtain our main result, namely a polynomial-time algorithm that solves weighted SFVS on graphs with independent set number at most 3.

Theorem 8.1.6. *The weighted SFVS optimization problem can be solved on graphs with independent set number at most* 3 *in* $n^{\mathcal{O}(1)}$ *time.*

Proof. Let us briefly explain such an algorithm for computing a maximum S-forest F of a graph G having independent set size at most three. Initially we set $F^* = G - S$. Then, for every set $X \subseteq N[S]$ with $|X| \leq 4 \cdot 3$ such that $G[X]$ is an S-forest, we try by brute force every tuple A_1 and (A_1, A_2) with $A_i \subseteq (X \setminus S)$ and check whether $\mathsf{aux}_{A_1}(X)$ or $\mathsf{aux}_{A_1,A_2}(X)$ is an S-forest. For each of such subsets, we find a maximum S-forest F with an S-distance partition $(G[X], F_{>1})$ having a cut satisfying A_1 or (A_1, A_2), respectively, by applying the algorithms described in Lemma 8.1.4 and Lemma 8.1.5. At each step, we maintain the maximum weighted S-forest F^* by comparing F with F^*. Finally we provide the vertices $V \setminus V(F^*)$ as the set with the minimum total weight that are removed from G.

By Lemma 8.1.3, it is sufficient to consider the described subsets X. Since every induced subgraph of $G - X$ contains at most two connected components, Lemma 8.1.2 implies that all possible subsets A_1 or (A_1, A_2) with the described properties are enough to consider. Thus, the correctness follows from Lemmata 8.1.3–8.1.5. Regarding the running time, notice that whether a graph contains an S-cycle can be tested in polynomial time. Thus, we can construct all described and valid subsets in $n^{\mathcal{O}(1)}$ time. Therefore the total running time of the algorithm is $n^{\mathcal{O}(1)}$, since each of the algorithms given in Lemma 8.1.4 and Lemma 8.1.5, respectively, requires polynomial time. □

Let us now show that extending Theorem 8.1.6 to graphs with larger independent set number is not possible. More precisely, with the following result we show that weighted SFVS is para-NP-complete parameterized by the independent set number.

Theorem 8.1.7. *The weighted SFVS decision problem on graphs with independent set number exactly α and clique cover number exactly α is* NP*-complete if $a \geq 4$.*

Proof. We will a provide a polynomial reduction for $\alpha = 4$ which can be trivially generalized to any $\alpha \geq 4$. The reduction is from the unweighted VC decision problem on tripartite graphs which is NP-complete [37]. Let G be a tripartite graph on n vertices and let $\{A, B, C\}$ be a partition of $V(G)$ into three cliques. We construct a weighted graph G' from G in polynomial time as follows.

- We assign unary weight to all vertices of G.
- We add a vertex s and two vertices r_I, t_I for every $I \in \{A, B, C\}$. We assign weight n to all added vertices.
- For every $I \in \{A, B, C\}$, we add the edge $\{r_I, s\}$ and all edges necessary for turning $\{r_I, t_I\} \cup I$ into a clique.

This completes the construction of G'. Since $\{s, t_A, t_B, t_C\}$ is an independent set and $\{\{s\}, \{r_A, t_A\} \cup A, \{r_B, t_B\} \cup B, \{r_C, t_C\} \cup C\}$ is a partition of the $V(G')$ into cliques, the constructed graph G' is such that $\alpha(G') = \kappa(G') = 4$.

Next we set $S = \{s\}$ and we claim that G has a vertex cover of size at most $k < n$ if and only if G' has an S-fvs of weight at most k. Consider a vertex cover U of G of size at most k. By definition, U hits all edges of G, so $G - U$ is an independent set. It follows that $\{r_A, t_A\} \cup (A \setminus U)$, $\{r_B, t_B\} \cup (B \setminus U)$ and $\{r_C, t_C\} \cup (C \setminus U)$ are the connected components of $G' - (\{s\} \cup U)$. Since s is only adjacent to r_A, r_B and r_C, no vertex set containing s induces a cycle of $G' - U$. Thus $G' - U$ is a connected S-forest. Therefore U is an S-fvs of G' of weight at most k, because all vertices of G have unary weight in G'.

For the opposite direction, consider an S-fvs U of G' of weight at most $k < n$. If $U \nsubseteq V(G)$, then its weight is at least n. Thus $U \subseteq V(G)$. Assume that U is not a vertex cover of G. By definition, there is an edge of G that U does not hit. Without loss of generality, assume that the endpoints of this edge are the vertices $a \in A$ and $b \in B$. Then $\langle s, r_A, a, b, r_B \rangle$ is an S-cycle in

$G' - U$, which contradicts the fact that U is an S-fvs of G'. Therefore, U is a vertex cover of G of size at most k, because all vertices of G have unary weight in G'. □

Combining the results of Theorems 8.1.6–8.1.7, we obtain the following complexity dichotomy results for weighted SFVS.

Corollary 8.1.8. *Weighted SFVS on graphs with independent set number at most α is in* P *if $\alpha \leq 3$ and* NP*-complete otherwise. The same holds for weighted SFVS on graphs with independent set number exactly α, on graphs with clique cover number at most α, on graphs with clique cover number exactly α, and on graphs with independent set number exactly α and clique cover number exactly α.*

8.1.2 Unweighted SFVS

Here we show that despite the existence of the above complexity dichotomy results for weighted SFVS, unweighted SFVS can be solved in polynomial time on graphs with bounded independent set number.

Theorem 8.1.9. *The unweighted SFVS optimization problem can be solved on graphs with independent set number at most α in $n^{\mathcal{O}(\alpha)}$ time.*

Proof. Let G be a graph such that $\alpha(G) \leq \alpha$ and let $S \subseteq V(G)$. Consider a minimum S-fvs U of G. Then $F = G - U$ is a maximum S-forest of G. By Observation 8.1.1 (2), the set $S \cap V(F) = S \setminus U$ contains at most 2α vertices. We now claim that the set $U \setminus S$ also contains at most 2α vertices. To see this, observe that if $U \setminus S$ would contain more than 2α vertices, then S would be an S-fvs of G of size smaller than the size of U, leading to a contradiction to the optimality of U. Thus both $S \setminus U$ and $U \setminus S$ contain at most 2α vertices.

We conclude that in order to find such a set U, it suffices to consider all sets $S' \subseteq S$ and $U' \subseteq V(G) \setminus S$ containing at most 2α vertices as candidates for $S \setminus U$ and $U \setminus S$ respectively. To see this, observe that $U \cap S = S \setminus (S \setminus U)$. The number of such sets S' and U' is at most $2n^{2\alpha+1}$. Moreover, checking whether an induced subgraph of G is an S-forest takes $\mathcal{O}(n + m)$ time [14]. Therefore, in $n^{\mathcal{O}(\alpha)}$ time we compute a minimum S-fvs showing the claimed result. □

8.2 FVS on Graphs with Bounded Independent Set Number

Here we show that in addition to unweighted SFVS, weighted FVS can also be solved in polynomial time on graphs with bounded independent set number.

Theorem 8.2.1. *The weighted FVS optimization problem can be solved on graphs with independent set number at most α in $n^{\mathcal{O}(\alpha)}$ time.*

Proof. Let G be a graph such that $\alpha(G) \leq \alpha$. By Observation 8.1.1 (1), every forest of G has at most 2α vertices. Thus, in order to compute a minimum-weighted fvs of G, it suffices that we proceed as follows: We construct all sets $V \subseteq V(G)$ of size at most 2α. Then, for every such set V, we check whether it induces a forest in $\mathcal{O}(n+m)$ time and if so, we construct the set $U = V(G) \setminus V$. Finally, we pick a set with the smallest weight among the constructed sets U. Therefore, the total running time of our algorithm is $n^{\mathcal{O}(\alpha)}$. □

Regarding the dependence of the exponent on the independent set number in the running time of the algorithms given in Theorems 8.1.9 and 8.2.1, note that we can hardly avoid this fact, since unweighted FVS is W[1]-hard parameterized by the clique cover number as explicitly shown in [51] via a reduction from the INDEPENDENT SET problem. The same W[1]-hardness can also be shown via an appropriate modification of the reduction from the MULTICOLORED CLIQUE problem given in [48] for showing that unweighted FVS on H-graphs is W[1]-hard parameterized by mim-width. However, this latter reduction is quadratic in the parameter k. In the following result, we provide a simpler reduction which is linear in the parameter k.

Theorem 8.2.2. *The unweighted FVS decision problem is* W*[1]-hard parameterized by the clique cover number.*

Proof. The reduction is from the MULTICOLORED INDEPENDENT SET problem: given a graph G and a partition $\mathcal{V} = \{V_i\}_{i \in [k]}$ of $V(G)$ into k parts, decide whether G has an independent set of size k containing exactly one vertex from each part of $\mathcal{V}$. We call such a set a *multicolored independent set* of G. It is known that MULTICOLORED INDEPENDENT SET is W[1]-hard parameterized by k [31, 72]. Let $(G, \{V_i\}_{i \in [k]})$ be an instance of MULTICOLORED INDEPENDENT SET such that $|V(G)| = n$. We construct a graph G' from G as follows.

- For every $i \in [k]$, we add a vertex l_i and all edges necessary for turning the set $V_i \cup \{l_i\}$ into a clique.
- We add a vertex r and make it adjacent to every vertex of $V(G)$.

This completes the construction of G'. Note that $|V(G')| = n + k + 1$. We define the sets $L = \{l_i\}_{i \in [k]}$ and $I = \{r\} \cup L$. Observe that I is an independent set of G' of size $k+1$ and $\{\{r\}\} \cup \{V_i \cup \{l_i\}\}_{i \in [k]}$ is a partition of $V(G')$ into $k+1$ cliques. We conclude that $\alpha(G') = \kappa(G') = k+1$. We claim that G has a multicolored independent set if and only if G' has a fvs of size at most $n-k$.

Let $I_{mis} = \{v_i \in V_i\}_{i \in [k]}$ be a multicolored independent set of G. Consider the graph $F = G'[I \cup I_{mis}]$. By the construction of G', the graph F is a tree. Therefore, the set $V(G') \setminus V(F) = V(G) \setminus I_{mis}$ constitutes a fvs of G' of $n-k$ size.

For the opposite direction, let U be a fvs of G' of size at most $n-k$. Then $F = G' - U$ is a forest of G' that has at least $2k+1$ vertices. We claim that the forest F contains exactly one vertex from each set V_i, $i \in [k]$. Recall that a forest contains at most two vertices from any particular clique. Since the sets $V_i \cup \{l_i\}$, $i \in [k]$ are cliques of G', the forest F must contain at most two vertices from each one of these cliques. Then it is not difficult to see that F must contain exactly two vertices from each set $V_i \cup \{l_i\}$, $i \in [k]$ as well as the vertex r, because otherwise it would contain less than $2k+1$ vertices. In particular, the forest F must contain at least one vertex from each set V_i, $i \in [k]$. Since the sets $\{r\} \cup V_i$, $i \in [k]$ are also cliques of G', the forest F must also contain at most two vertices from each one of these cliques. In particular, the forest F must contain at most one vertex from each set V_i, $i \in [k]$. We conclude that the forest F contains exactly one vertex from each set V_i, $i \in [k]$. Let $I_{mis} = \{v_i \in V_i\}_{i \in [k]}$ be the set $V(F) \setminus I$. Without loss of generality, assume that v_1 and v_2 are adjacent. Then $\langle r, v_1, v_2 \rangle$ is a cycle in F, which is a contradiction to F being a forest. Therefore, the set I_{mis} is a multicolored independent set of G. □

Chen et al. [20] showed that Multicolored Independent Set admits no $f(k) \cdot n^{o(k)}$-time algorithm under the ETH. Since the reduction provided in the proof of Theorem 8.2.2 is linear in the parameter k, we get the following result conditioned on the ETH:

Corollary 8.2.3. *Unweighted FVS on graphs with independent set number exactly k and clique cover number exactly k cannot be solved in $f(k) \cdot n^{o(k)}$ time, unless the ETH fails.*

The lower bound stated in Corollary 8.2.3 implies that if the ETH holds, then the running times of the algorithms given in Theorems 8.1.9 and 8.2.1 are tight for all graph classes considered in this chapter.

8.3 NMC on Graphs with Bounded Independent Set Number

Here we completely characterize the complexity of NMC on graphs with bounded independent set number. In particular, for graphs with independent set number exactly α and clique cover number exactly $\alpha \geq 3$, we adapt the reduction given in Theorem 8.1.7.

Theorem 8.3.1. *NMC on graphs with independent set number at most α is in* P *if $\alpha \leq 2$ and* NP*-complete otherwise. The same holds for NMC on graphs with independent set number exactly α, on graphs with clique cover number at most α, on graphs with clique cover number exactly α and on graphs with independent set number exactly α and clique cover number exactly α.*

Proof. Let (G, T) be an instance of the weighted NMC optimization problem such that $\alpha(G) = \alpha \leq 2$. If $G[T]$ contains an edge, then we conclude that (G, T) is a `no`-instance of the problem, since we are not allowed to include any vertex of T in its solution. Otherwise, T is an independent set, so $|T| \leq \alpha \leq 2$. If $|T| \leq 1$, then the solution to the problem is $\varnothing$, and if $|T| = 2$, then we can solve the problem by standard maximum flow techniques [67].

Now we will provide a polynomial reduction to the unweighted NMC decision problem on graphs with independent set number exactly α and clique cover number exactly α for $\alpha = 3$ which can be trivially generalized to any $\alpha \geq 3$. The reduction is from the NP-complete unweighted VC decision problem on tripartite graphs and is similar to the one given in Theorem 8.1.7. Let G be a tripartite graph and let $\{A, B, C\}$ be a partition of $V(G)$ into three cliques. We construct a graph G' from G by adding three vertices t_A, t_B and t_C and turning the three vertex sets $\{t_A\} \cup A$, $\{t_A\} \cup B$, and $\{t_A\} \cup C$ into cliques. Since $\{t_A, t_B, t_C\}$ is an independent set and $\{\{t_A\} \cup A, \{t_A\} \cup B, \{t_A\} \cup C\}$ is a partition of $V(G')$ into cliques, the constructed graph G' is such that $\alpha(G') = \kappa(G') = 3$. We set $T = \{t_A, t_B, t_C\}$ and claim that G has a vertex cover of size at most k if and only if G' has a vertex subset of $V(G') \setminus T = V(G)$ of size at most k which hits every vertex set that induces a path in G' between vertices of T. Consider a vertex cover U of G of size at most k. By definition, U hits all edges of G, so $G - U$ is an independent set. It follows that $\{t_A\} \cup (A \setminus U)$, $\{t_B\} \cup (B \setminus U)$ and $\{t_C\} \cup (C \setminus U)$ are the connected components of $G' - U$. Thus U is a vertex subset of $V(G)$ with the desired property. For the opposite direction, consider a set $U \subseteq V(G)$ with the desired property. Assume that U is not a vertex cover of G. By definition, there is an edge of G that U does not hit. Without loss of generality, assume that the endpoints of this edge are the vertices $a \in A$ and $b \in B$. Then $\langle t_A, a, b, t_B \rangle$ is a path in $G' - U$ between vertices of T, leading to a contradiction. Therefore, U is a vertex cover of G of size at most k. □

8.4 UNMC on Graphs with Bounded Independent Set Number

Due to the difficulty of NMC even in its unweighted version and on graphs with small independent set number, we consider UNMC as a relaxed variant of NMC.

8.4.1 Weighted UNMC

Regarding the weighted version of UNMC, we provide a dichotomy result with respect to the independent set number. In particular, for graphs with independent set number at most $\alpha \leq 2$, we add a vertex of appropriately large weight, we make it adjacent to all terminals and we invoke the algorithm for solving the weighted SFVS optimization problem given in Theorem 8.1.6; and for graphs with independent set number exactly α and clique cover number exactly $\alpha \geq 3$, we assign appropriately large weight to all terminals and we reduce from the closely related unweighted NMC decision problem.

Theorem 8.4.1. *Weighted UNMC on graphs with independent set number at most α is in* P *if $\alpha \leq 2$ and* NP*-complete otherwise. The same holds for weighted UNMC on graphs with independent set number exactly α, on graphs with clique cover number at most α, on graphs with clique cover number exactly α and on graphs with independent set number exactly α and clique cover number exactly α.*

Proof. Let (G, T) be an instance of the weighted UNMC optimization problem such that $\alpha(G) \leq \alpha \leq 2$. We create an instance (G', S) for the weighted SFVS optimization problem as follows. We construct a graph G' from G by adding a vertex s of weight $w(T) + 1$ and making s adjacent to all vertices of T. Since we added exactly one vertex to G, the constructed graph G' is such that $\alpha(G') \leq \alpha + 1 \leq 3$. We also set $S = \{s\}$. Next we claim that every solution to the weighted UNMC optimization problem on (G, T) is also a solution to the weighted SFVS optimization problem on (G', S) and vice versa. Observe that a set $P \subseteq V(G)$ induces a path in G between vertices of T if and only if $\{s\} \cup P$ induces an S-cycle in G'. It directly follows that a set $U \subseteq V(G)$ hits all paths in G between vertices of T if and only if U is an S-fvs of G'. Notice that $G' - T$ is an S-forest of G'. This implies that $w(U) \leq w(T)$ for every minimum S-fvs U of G'. Thus no minimum S-fvs of G' contains s. We conclude that $U \subseteq V(G)$ for every minimum S-fvs U of G'. Therefore, by running the algorithm for solving the weighted SFVS optimization problem given in Theorem 8.1.6 on (G', S), we obtain a solution to the weighted UNMC optimization problem on (G, T) in $n^{\mathcal{O}(1)}$ time.

Now let (G, T, k) be an instance of the unweighted NMC decision problem such that $\alpha(G) = \kappa(G) = \alpha \geq 3$ and $k < n$. We assign weight n to all vertices of T and unary weight to all remaining vertices. Thus no solution to the weighted UNMC decision problem on (G, T, k) contains vertices of T, which implies that the unweighted NMC and weighted UNMC problems are equivalent on (G, T, k). Since the unweighted NMC decision problem on graphs with independent set number exactly α and clique cover number exactly α is NP-complete if $\alpha \geq 3$ by Theorem 8.3.1, it follows that the same holds for the weighted UNMC decision problem on the same graph class. $\square$

8.4.2 Unweighted UNMC

Next we show that the unweighted UNMC optimization problem can be solved in polynomial time on graphs with bounded independent set number using an idea which is similar to the one that we used in proving Theorem 8.1.9.

Theorem 8.4.2. *The unweighted UNMC optimization problem can be solved on graphs with independent set number at most α in $n^{\mathcal{O}(\alpha)}$ time.*

Proof. Let (G, T) be an instance of the unweighted UNMC optimization problem such that $\alpha(G) \leq \alpha$. Notice that $G - T$ trivially contains no path between vertices of T. This implies that $|U| \leq |T|$ for every solution U to the problem. Assume that $|T| \leq \alpha$. We construct all sets $U \subseteq V(G)$ of size at most $|T|$. Then, for every such set U, we check whether $G - U$ contains a path between vertices of T in $n^{\mathcal{O}(1)}$ time and if so, we discard the set U. Finally, we pick one with the smallest size among the remaining constructed sets. Thus the total running time in this case is $n^{\mathcal{O}(|T|)}$, which is bounded by $n^{\mathcal{O}(\alpha)}$.

Now assume that $|T| > \alpha$. Recall that for every induced subgraph H of G, $\alpha(H) \leq \alpha(G)$. We consider the graph $G[T]$. Since $\alpha(G[T]) \leq \alpha(G) \leq \alpha < |T|$, the set T is not an independent set. Thus $G[T]$ contains at least one edge. Since every edge of $G[T]$ is trivially a path between vertices of T, every solution to the problem must hit all edges of $G[T]$ or, equivalently, every solution to the problem must be a superset of a vertex cover of $G[T]$. We construct all sets $T' \subseteq T$ of size at most α. Then, for every such set T', we check whether it is an independent set in $n^{\mathcal{O}(1)}$ time and if so,

Step 1. we construct the vertex cover $U_{vc} = T \setminus T'$ of $G[T]$,

Step 2. we consider the graph $G' = G - U_{vc}$,

Step 3. we obtain a solution U' to the problem on the instance (G', T') as described in the previous case, because $\alpha(G') \leq \alpha(G) \leq \alpha$ and $|T'| \leq \alpha$, and

Step 4. we construct the set $U = U_{vc} \cup U'$.

Finally, we pick a set with the smallest size among the sets U constructed in the previous step. Therefore, the total running time in this case is $|T|^{\mathcal{O}(\alpha)} \cdot n^{\mathcal{O}(\alpha)}$, which is bounded by $n^{\mathcal{O}(\alpha)}$. Thus in both cases we output a solution to the problem in $n^{\mathcal{O}(\alpha)}$ time. □

Let us also show that we can hardly avoid the dependence of the exponent on the independent set number in the running time of the algorithm given in Theorem 8.4.2. Observe that the unweighted UNMC decision problem on an instance $(G, T = V(G), k)$ is equivalent to asking whether the graph G contains an independent set of size at least $n - k$. That is, it is equivalent to the unweighted INDEPENDENT SET decision problem on the instance $(G, n-k)$.

Theorem 8.4.3. *The unweighted* INDEPENDENT SET *decision problem is* W[1]*-hard parameterized by* $\kappa + k$ *where* κ *is the clique cover number and* k *is the solution size.*

Proof. The reduction is from the MULTICOLORED INDEPENDENT SET problem. Let $(G, \{V_i\}_{i \in [k]})$ be an instance of MULTICOLORED INDEPENDENT SET. We construct a graph G' from G by simply adding all edges necessary for turning the set V_i into a clique for every $i \in [k]$. Observe that $\alpha(G') \leq \kappa(G') \leq k$.

Now it is not difficult to see that a set $U \subseteq V(G)$ is a solution to Multicolored Independent Set on $(G, \{V_i\}_{i\in[k]})$ if and only if U is a solution to the unweighted Independent Set decision problem on the instance (G', k). $\square$

Notice that the reduction provided in the proof of Theorem 8.4.3 is linear in the parameter k. Thus, we get the following result conditioned on the ETH:

Corollary 8.4.4. *Unweighted* Independent Set *on graphs with clique cover number at most k cannot be solved in $f(k) \cdot n^{o(k)}$ time, unless the ETH fails.*

The lower bound stated in Corollary 8.4.4 implies the same lower bound for unweighted Independent Set on all graph classes considered in this chapter and that if the ETH holds, then the running time of the algorithm given in Theorem 8.4.2 is tight for all these graph classes.

9 Concluding Remarks

In this thesis, we provided a systematic study of the complexity of SFVS on subclasses of AT-free graphs and on subclasses of chordal graphs, and of the complexity SFVS, FVS, NMC and UNMC on $(k+1)K_1$-free graphs. Towards tackling the complexity of SFVS on AT-free graphs, we proposed algorithms for solving SFVS on interval graphs, on permutation graphs and on cobipartite graphs which were the first positive results regarding the complexity of SFVS restricted to graph classes appearing in the literature. In order to cope with the known NP-hardness of SFVS on chordal graphs, we considered the structural parameters of leafage and vertex leafage as natural tools to exploit insights of the corresponding tree representation. We also considered the structural parameter of independent set number and we completely characterized the complexity of all studied problems with respect to this parameter.

9.1 Open Problems

Recalling the graph classes and the relationships between them illustrated in Figure 3.1, we observe that the complexity status of SFVS is still open in a few of them. There are also parameterized hierarchies of graph classes for which conditional lower bounds have not yet been established, thus it is not yet known whether their respective known complexities are tight. Let us state all these open problem explicitly.

Subclasses of AT-free Graphs

Regarding subclasses of AT-free graphs, due to the algorithm proposed by Bergougnoux et al. [5], weighted SFVS is polynomial-time solvable on every graph class depicted in Figure 3.1 which we did not address in this thesis except for cocomparability graphs and AT-free graphs thermselves, which do not have bounded mim-width. Thus we ask the following:

Question 9.1.1. What is the complexity of SFVS on cocomparability graphs?

Question 9.1.2. What is the complexity of SFVS on AT-free graphs?

Moreover, for the hierarchies of k-permutation graphs and k-trapezoid graphs, which are known to be graphs with mim-width $\mathcal{O}(k)$ [3], SFVS may be solvable faster, as no conditional lower bound with respect to k has yet been established, motivating us to ask the following:

Question 9.1.3. Is SFVS on k-permutation graphs solvable in $n^{o(k^2)}$ time?

Question 9.1.4. Is SFVS on k-trapezoid graphs solvable in $n^{o(k^2)}$ time?

Subclasses of Chordal Graphs

Regarding subclasses of chordal graphs, the graph classes depicted in Figure 3.1 which we did not address in this thesis are directed path graphs and (split $\cap$) strongly chordal graphs. Thus we ask the following:

Question 9.1.5. What is the complexity of SFVS on directed path graphs?

Question 9.1.6. What is the complexity of SFVS on (split $\cap$) strongly chordal graphs?

Moreover, on the complexity of SFVS on graphs with leafage ℓ no conditional lower bound with respect to ℓ has yet been established, and in particular for its unweighted version not even W[1]-hardness has yet been shown, motivating us to ask the following:

Question 9.1.7. Is weighted SFVS on graphs with leafage ℓ solvable in $n^{o(\ell)}$ time?

Question 9.1.8. Is unweighted SFVS on graphs with leafage ℓ solvable in $f(\ell) \cdot n^{\mathcal{O}(1)}$ time?

Structural Parameters

Regarding the hierarchies of co-k-partite graphs and $(k+1)K_1$-free graphs, we have completely characterized the complexity of the SFVS, FVS, NMC and UNMC problems, including showing W[1]-hardness results and lower bounds conditioned on the ETH, thus there are no open problems.

We close this section, chapter and thesis with two open problems that are not evident from the illustration of Figure 3.1. On graphs with mim-width w, weighted FVS is known to be solvable in $n^{\mathcal{O}(w)}$ time [4, 50], whereas weighted SFVS is known to be solvable in $n^{\mathcal{O}(w^2)}$ time [5]. As no respective conditional lower bounds have yet been established:

Question 9.1.9. Is weighted FVS on graphs with mim-width w solvable in $n^{o(w)}$ time?

Question 9.1.10. Is weighted SFVS on graphs with mim-width w solvable in $n^{o(w^2)}$ time?

Bibliography

[1] Bar-Yehuda, R., Geiger, D., Naor, J., and Roth, R. M. Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and bayesian inference. *SIAM Journal on Computing 27*, 4 (1998), 942–959.

[2] Becker, A., and Geiger, D. Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. *Artificial Intelligence 83*, 1 (1996), 167 – 188.

[3] Belmonte, R., and Vatshelle, M. Graph classes with structured neighborhoods and algorithmic applications. *Theor. Comput. Sci. 511* (2013), 54–65.

[4] Bergougnoux, B., and Kanté, M. M. More applications of the d-neighbor equivalence: Connectivity and acyclicity constraints. In *Proceedings of ESA 2019* (2019), pp. 17:1–17:14.

[5] Bergougnoux, B., Papadopoulos, C., and Telle, J. A. Node multiway cut and subset feedback vertex set on graphs of bounded mim-width. *Algorithmica 84*, 5 (2022), 1385–1417.

[6] Bertossi, A. A., and Bonuccelli, M. A. Hamiltonian circuits in interval graph generalizations. *Inf. Process. Lett. 23*, 4 (1986), 195–200.

[7] Bodlaender, H. L., and Jansen, K. On the complexity of the maximum cut problem. *Nord. J. Comput. 7*, 1 (2000), 14–31.

[8] Booth, K. S., and Johnson, J. H. Dominating sets in chordal graphs. *SIAM J. Comput. 11* (1982), 191–199.

[9] Booth, K. S., and Lueker, G. S. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *J. Comput. Syst. Sci. 13*, 3 (1976), 335–379.

[10] Brandstädt, A. On improved time bounds for permutation graph problems. In *Proceedings of WG 1992* (1985), pp. 1–10.

[11] Brandstädt, A., and Kratsch, D. On the restriction of some NP-complete graph problems to permutation graphs. In *Proceedings of FCT 1985* (1985), pp. 53–62.

[12] Brandstädt, A., and Kratsch, D. On domination problems for permutation and other graphs. *Theoretical Computer Science 54*, 2 (1987), 181 – 198.

[13] Brandstädt, A., Le, V. B., and Spinrad, J. *Graph Classes: A Survey.* Society for Industrial and Applied Mathematics, 1999.

[14] Brettell, N., Johnson, M., Paesani, G., and Paulusma, D. Computing subset transversals in H-free graphs. In *Proceedings of WG 2020* (2020), vol. 12301, pp. 187–199.

[15] Bui-Xuan, B., Suchý, O., Telle, J. A., and Vatshelle, M. Feedback vertex set on graphs of low clique-width. *Eur. Journal of Combinatorics 34*, 3 (2013), 666–679.

[16] Buneman, P. A characterization of rigid circuit graphs. *Discret. Math. 9* (1974), 205–212.

[17] Calinescu, G. Multiway cut. In *Encyclopedia of Algorithms.* Springer, 2008.

[18] Chaplick, S. Intersection graphs of non-crossing paths. In *Graph-Theoretic Concepts in Computer Science - 45th International Workshop, WG 2019, Revised Papers* (2019), vol. 11789 of *Lecture Notes in Computer Science*, Springer, pp. 311–324.

[19] Chaplick, S., and Stacho, J. The vertex leafage of chordal graphs. *Discret. Appl. Math. 168* (2014), 14–25.

[20] Chen, J., Huang, X., Kanj, I. A., and Xia, G. Strong computational lower bounds via parameterized complexity. *J. Comput. Syst. Sci. 72* (2006), 1346–1367.

[21] Chen, J., Liu, Y., and Lu, S. An improved parameterized algorithm for the minimum node multiway cut problem. *Algorithmica 55* (2009), 1–13.

[22] Chitnis, R. H., Fomin, F. V., Lokshtanov, D., Misra, P., Ramanujan, M. S., and Saurabh, S. Faster exact algorithms for some terminal set problems. *Journal of Computer and System Sciences 88* (2017), 195–207.

[23] Cook, S. A. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing* (New York, NY, USA, 1971), STOC '71, Association for Computing Machinery, pp. 151–158.

[24] Corneil, D. G., and Fonlupt, J. The complexity of generalized clique covering. *Discret. Appl. Math. 22*, 2 (1988), 109–118.

[25] Corneil, D. G., and Perl, Y. Clustering and domination in perfect graphs. *Discret. Appl. Math. 9*, 1 (1984), 27–39.

[26] Cygan, M., Nederlof, J., Pilipczuk, M., Pilipczuk, M., van Rooij, J. M. M., and Wojtaszczyk, J. O. Solving connectivity problems parameterized by treewidth in single exponential time. In *Proceedings of FOCS 2011* (2011), pp. 150–159.

[27] Cygan, M., Pilipczuk, M., Pilipczuk, M., and Wojtaszczyk, J. O. On multiway cut parameterized above lower bounds. *ACM Trans. Comput. Theory 5*, 1 (2013), 3:1–3:11.

[28] Cygan, M., Pilipczuk, M., Pilipczuk, M., and Wojtaszczyk, J. O. Subset feedback vertex set is fixed-parameter tractable. *SIAM J. Discrete Math. 27*, 1 (2013), 290–309.

[29] Dietz, P. F. *Intersection graph algorithms.* PhD thesis, Cornell University, 1984.

[30] Even, G., Naor, J., and Zosin, L. An 8-approximation algorithm for the subset feedback vertex set problem. *SIAM J. Comput. 30*, 4 (2000), 1231–1252.

[31] Fellows, M. R., Hermelin, D., Rosamond, F. A., and Vialette, S. On the parameterized complexity of multiple-interval graph problems. *Theor. Comput. Sci. 410*, 1 (2009), 53–61.

[32] Festa, P., Pardalos, P. M., and Resende, M. G. C. Feedback set problems. In *Encyclopedia of Optimization*. Springer, 2009, pp. 1005–1016.

[33] Fomin, F. V., Gaspers, S., Lokshtanov, D., and Saurabh, S. Exact algorithms via monotone local search. In *Proceedings of STOC 2016* (2016), pp. 764–775.

[34] Fomin, F. V., Golovach, P. A., and Raymond, J. On the tractability of optimization problems on h-graphs. *Algorithmica 82*, 9 (2020), 2432–2473.

[35] Fomin, F. V., Heggernes, P., Kratsch, D., Papadopoulos, C., and Villanger, Y. Enumerating minimal subset feedback vertex sets. *Algorithmica 69*, 1 (2014), 216–231.

[36] Fulkerson, D., and Gross, O. Incidence matrices and interval graphs. *Pacific Journal of Mathematics 15* (1965), 835–855.

[37] Garey, M. R., and Johnson, D. S. *Computers and Intractability.* W. H. Freeman and Co., 1978.

[38] Garg, N., Vazirani, V. V., and Yannakakis, M. Multiway cuts in node weighted graphs. *J. Algorithms 50*, 1 (2004), 49–61.

[39] Gavril, F. The intersection graphs of subtrees in trees are exactly the chordal graphs. *J. Comb. Theory, Ser. B 16*, 1 (1974), 47–56.

[40] Gavril, F. A recognition algorithm for the intersection graphs of directed paths in directed trees. *Discret. Math. 13*, 3 (1975), 237–249.

[41] Golovach, P. A., Heggernes, P., Kratsch, D., and Saei, R. Subset feedback vertex sets in chordal graphs. *J. Discrete Algorithms 26* (2014), 7–15.

[42] Golumbic, M. C. *Algorithmic Graph Theory and Perfect Graphs.* Annals of Discrete Mathematics 57, Elsevier, 2004.

[43] Habib, M., and Stacho, J. Polynomial-time algorithm for the leafage of chordal graphs. In *Algorithms - ESA 2009, 17th Annual European Symposium, Proceedings* (2009), vol. 5757 of *Lecture Notes in Computer Science*, Springer, pp. 290–300.

[44] Heggernes, P., van 't Hof, P., van Leeuwen, E. J., and Saei, R. Finding disjoint paths in split graphs. *Theory Comput. Syst. 57*, 1 (2015), 140–159.

[45] Hols, E. C., and Kratsch, S. A randomized polynomial kernel for subset feedback vertex set. *Theory Comput. Syst. 62* (2018), 54–65.

[46] Ioannidou, K., Mertzios, G. B., and Nikolopoulos, S. D. The longest path problem has a polynomial solution on interval graphs. *Algorithmica 61*, 2 (2011), 320–341.

[47] Jaffke, L., Kwon, O., Strømme, T. J. F., and Telle, J. A. Mim-width III. graph powers and generalized distance domination problems. *Theor. Comput. Sci. 796* (2019), 216–236.

[48] Jaffke, L., Kwon, O., and Telle, J. A. A note on the complexity of feedback vertex set parameterized by mim-width. *CoRR abs/1711.05157* (2017).

[49] Jaffke, L., Kwon, O., and Telle, J. A. A unified polynomial-time algorithm for feedback vertex set on graphs of bounded mim-width. In *Proceedings of STACS 2018* (2018), pp. 42:1–42:14.

[50] Jaffke, L., Kwon, O., and Telle, J. A. Mim-width II. the feedback vertex set problem. *Algorithmica 82*, 1 (2020), 118–145.

[51] Jansen, B., Raman, V., and Vatshelle, M. Parameter ecology for feedback vertex set. *Tsinghua Sci. and Technol. 19*, 4 (2014), 387–409.

[52] Karp, R. M. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations* (1972), The IBM Research Symposia Series, Plenum Press, New York, pp. 85–103.

[53] Kawarabayashi, K., and Kobayashi, Y. Fixed-parameter tractability for the subset feedback set problem and the s-cycle packing problem. *J. Comb. Theory, Ser. B 102*, 4 (2012), 1020–1034.

[54] Kratsch, D., Müller, H., and Todinca, I. Feedback vertex set on AT-free graphs. *Discret. Appl. Math. 156*, 10 (2008), 1936–1947.

[55] Kratsch, S., and Wahlstrom, M. Representative sets and irrelevant vertices: new tools for kernelization. In *Proceedings of FOCS 2012* (2012), pp. 450–459.

[56] Levin, L. A. Universal sequential search problems. *Problems Inform. Transmission 9*, 3 (1973), 265–266.

[57] Liang, Y. D. On the feedback vertex set problem in permutation graphs. *Inf. Process. Lett. 52*, 3 (1994), 123 – 129.

[58] Liang, Y. D., and Chang, M.-S. Minimum feedback vertex sets in cocomparability graphs and convex bipartite graphs. *Acta Informatica 34*, 5 (1997), 337–346.

[59] Lin, I., McKee, T. A., and West, D. B. The leafage of a chordal graph. *Discuss. Math. Graph Theory 18*, 1 (1998), 23–48.

[60] Lu, C. L., and Tang, C. Y. A linear-time algorithm for the weighted feedback vertex problem on interval graphs. *Inf. Process. Lett. 61*, 2 (1997), 107–111.

[61] Marx, D. Parameterized graph separation problems. *Theor. Comput. Sci. 351* (2006), 399–406.

[62] Maw-Shang, C. Weighted domination of cocomparability graphs. *Discret. Appl. Math. 80*, 2 (1997), 135 – 148.

[63] McConnell, R. M., and Spinrad, J. P. Modular decomposition and transitive orientation. *Discrete Mathematics 201* (1999), 189–241.

[64] Micali, S., and Vazirani, V. V. An O(sqrt(—v—) —e—) algorithm for finding maximum matching in general graphs. In *Proceedings of FOCS 1980* (1980), pp. 17–27.

[65] Monma, C. L., and Wei, V. K. Intersection graphs of paths in a tree. *J. Comb. Theory, Ser. B 41*, 2 (1986), 141–181.

[66] Natarajan, S., and Sprague, A. P. Disjoint paths in circular arc graphs. *Nord. J. Comput. 3*, 3 (1996), 256–270.

[67] Orlin, J. B. Max flows in O(nm) time, or better. In *Proceedings of STOC 2013* (2013), pp. 765–774.

[68] Panda, B. S. The separator theorem for rooted directed vertex graphs. *J. Comb. Theory, Ser. B 81*, 1 (2001), 156–162.

[69] Papadopoulos, C., and Tzimas, S. Polynomial-time algorithms for the subset feedback vertex set problem on interval graphs and permutation graphs. *Discret. Appl. Math. 258* (2019), 204–221.

[70] Papadopoulos, C., and Tzimas, S. Subset feedback vertex set on graphs of bounded independent set size. *Theor. Comput. Sci. 814* (2020), 177–188.

[71] Philip, G., Rajan, V., Saurabh, S., and Tale, P. Subset feedback vertex set in chordal and split graphs. *Algorithmica 81*, 9 (2019), 3586–3629.

[72] Pietrzak, K. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *J. Comput. Syst. Sci. 67*, 4 (2003), 757–771.

[73] Spinrad, J. P. *Efficient graph representations*, vol. 19 of *Fields Institute monographs.* American Mathematical Society, 2003.

[74] Vatshelle, M. *New Width Parameters of Graphs.* PhD thesis, University of Bergen, Norway, 2012.

[75] Yannakakis, M. Node-deletion problems on bipartite graphs. *SIAM J. Comput. 10*, 2 (1981), 310–327.