ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΤΜΗΜΑ ΜΑΘΗΜΑΤΙΚΩΝ

Σπυρίδων Τζίμας

Το Προβλημα του Στοχευμενα Ακυκλου Επαγομενου Υπογραφηματος σε Γραφηματα Διαστηματων και σε Γραφηματα Μεταθεσεων

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΑΤΡΙΒΗ

ΙΩΑΝΝΙΝΑ, 2016

UNIVERSITY OF IOANNINA

DEPARTMENT OF MATHEMATICS

Spyridon Tzimas

Subset Feedback Vertex Set on Interval Graphs and Permutation Graphs

---

MASTER THESIS

IOANNINA, 2016

Dedicated to my parents,
who are ever by my side.

Η παρούσα Μεταπτυχιακή Διατριβή εκπονήθηκε στο πλαίσιο των σπουδών για την απόκτηση του Μεταπτυχιακού Διπλώματος Ειδίκευσης στα Υπολογιστικά Μαθηματικά και την Πληροφορική που απονέμει το Τμήμα Μαθηματικών του Πανεπιστημίου Ιωαννίνων.

Εγκρίθηκε την 20/10/2016 από την εξεταστική επιτροπή:

Ονοματεπώνυμο	**Βαθμίδα**
Χάρης Παπαδόπουλος	Επίκουρος Καθηγητής
Νικόλαος Γλυνός	Επίκουρος Καθηγητής
Σωκράτης Μπαλτζής	Λέκτορας

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

Περίληψη

Δοθέντος ενός γραφήματος με βάρη στις κορυφές του ως είσοδο, το (έμβαρο) πρόβλημα του Ακυκλου Επαγομενου Υπογραφηματος (FVS) ζητά την εύρεση ενός ελαχίστου βάρους υποσυνόλου των κορυφών του γραφήματος εισόδου των οποίων η αφαίρεση από αυτό έχει ως αποτέλεσμα αυτό να μην έχει πλέον επαγόμενο κύκλο. Το FVS βρίσκεται μεταξύ των κλασσικών προβλημάτων της Αλγοριθμικής Θεωρίας Γραφημάτων και έχει βρει πολλές εφαρμογές σε άλλα γνωστικά πεδία με το πέρασμα των χρόνων, με εφαρμογές στην ικανοποίηση περιορισμών και στην Μπεϊσιανή συμπερασματολογία, στην οπτική δικτύωση και στην υπολογιστική βιολογία να είναι μερικές πρόσφατες προσθήκες. Ως φυσικό επακόλουθο, οι αλγόριθμοι επίλυσης του FVS ήταν πάντα αντικείμενο ενεργής έρευνας. Τόσο ακριβείς όσο και προσεγγιστικοί αλγόριθμοι έχουν προταθεί για την επίλυση του FVS σε γενικά γραφήματα. Το FVS είναι $\mathcal{NP}$-πλήρες στα γενικά γραφήματα, στα επίπεδα γραφήματα, στα διμερή γραφήματα και στα επίπεδα διμερή γραφήματα. Συνεπώς, το FVS θεωρείται απίθανο να είναι πολυωνυμικά επιλύσιμο σε αυτές τις κλάσεις γραφημάτων. Αναφέρουμε επίσης πως το FVS είναι $\mathcal{FPT}$ στα γενικά γραφήματα. Το FVS είναι πολυωνυμικά επιλύσιμο στα γραφήματα διαστημάτων σε $\mathrm{O}(n+m)$ χρόνο, στα γραφήματα μεταθέσεων και στα γραφήματα τραπεζίων σε $\mathrm{O}(nm)$ χρόνο, στα συνσυγκρίσιμα γραφήματα και στα κυρτά διμερή γραφήματα σε $\mathrm{O}(n^2m)$, στα AT-ελεύθερα γραφήματα σε $\mathrm{O}(n^8m^2)$ χρόνο, στα χορδικά γραφήματα σε $\mathrm{O}(n^6)$ χρόνο και στα γραφήματα φραγμένου πλάτους κλίκας σε $\mathrm{O}(n)$ χρόνο, όπου n και m είναι το πλήθος των κορυφών και των ακμών του γραφήματος εισόδου αντίστοιχα.

Στην παρούσα διατριβή, μελετούμε μία γενίκευση του FVS που καλείται το πρόβλημα του Στοχευμενα Ακυκλου Επαγομενου Υπογραφηματος. Δοθέντων ενός γραφήματος με βάρη στις κορυφές του και ένα υποσύνολο S των κορυφών του ως είσοδο, το (έμβαρο) πρόβλημα του Στοχευμενα Ακυκλου Υπογραφηματος (SFVS) ζητά την εύρεση ενός ελαχίστου βάρους υποσυνόλου των κορυφών του γραφήματος εισόδου των οποίων η αφαίρεση από αυτό έχει ως αποτέλεσμα αυτό

να μην έχει πλέον επαγόμενο κύκλο που να περνά από κορυφή που ανήκει στο S. Εφόσον οι ενδεχόμενες γενικεύσεις των εφαρμογών του FVS μπορεί να απαιτούν την επίλυση μίας γενίκευσης του FVS στη θέση του ίδιου του FVS, η έλλειψη αποδοτικών αλγορίθμων για την επίλυση του SFVS μπορεί να αποτελεί τροχοπέδη στην πραγματοποίησή τους. Αυτή η παρατήρηση μας ωθεί να επιδιώξουμε την μελέτη του SFVS. Τόσο ακριβείς όσο και προσεγγιστικοί αλγόριθμοι έχουν επίσης προταθεί και για την επίλυση του SFVS σε γενικά γραφήματα. Το γεγονός ότι το SFVS αποτελεί γενίκευση του FVS συνεπάγεται ότι είναι και αυτό $\mathcal{NP}$-πλήρες στα γενικά γραφήματα, στα επίπεδα γραφήματα, στα διμερή γραφήματα και στα επίπεδα διμερή γραφήματα. Η πρώτη σημαντική διαφορά στην συμπεριφορά των δύο προβλημάτων είναι ότι, σε αντίθεση με το FVS το οποίο είναι $\mathcal{P}$ στα χορδικά γραφήματα, έχει δειχθεί ότι το SFVS είναι $\mathcal{NP}$-πλήρες στα διαχωρίσιμα γραφήματα, μία υποκλάση των χορδικών γραφημάτων. Επίσης σε αντίθεση με το FVS, δεν υπάρχουν πολυωνυμικά αποτέλεσματα όπου η είσοδος περιορίζεται σε κλάσεις γραφημάτων αναφορικά με το SFVS στη βιβλιογραφία. Στην παρούσα διατριβή, προτείνουμε νέους αλγορίθμους δυναμικού προγραμματισμού για την επίλυση του SFVS στα γραφήματα διαστημάτων σε $\mathrm{O}(n+m+l)$ χρόνο και στα γραφήματα μεταθέσεων σε $\mathrm{O}(m^3)$ χρόνο, όπου n, m και $l \in \mathrm{O}(n^3)$ είναι το πλήθος των κορυφών, των ακμών και των επαγόμενων τριγώνων του γραφήματος εισόδου αντίστοιχα—τα πρώτα πολυωνυμικά αποτελέσματα αναφορικά με το SFVS.

Η παρούσα διατριβή είναι δομημένη ως ακολούθως: Το Κεφάλαιο 1 είναι είναι εισαγωγικό κεφάλαιο που εφοδιάζει όλους τους απαραίτητους ορισμούς από την Θεωρία Πολυκλοκότητας και την Θεωρία Γραφημάτων. Επίσης φιλοξενεί πληροφορίες για τα FVS και SFVS. Στο Κεφάλαιο 2, δίνουμε τους καλύτερους πολυωνυμικούς αλγορίθμους δυναμικού προγραμματισμού για την επίλυση του FVS στα γραφήματα διαστημάτων και στα γραφήματα μεταθέσεων που υπάρχουν στη βιβλιογραφία και στη συνέχεια προτείνουμε νέους αλγορίθμους δυναμικού προγραμματισμού που παρουσιάζουν την ίδια χρονική πολυπλοκότητα και αποτελούν τους προπομπούς των αλγορίθμων μας για την επίλυση του SFVS στις ίδιες κλάσεις γραφημάτων. Το Κεφάλαιο 3 φιλοξενεί τους προαναφερθέντες πολυωνυμικούς αλγορίθμους δυναμικού προγραμματισμού μας για την επίλυση του SFVS στα γραφήματα διαστημάτων και στα γραφήματα μεταθέσεων. Το Κεφάλαιο 4 ολοκληρώνει την παρούσα διατριβή με μία ενημέρωση της κατάστασης των FVS και SFVS και μία συζήτηση πάνω σε μελλοντική έρευνα και ανοικτά προβλήματα αναφορικά με το SFVS. Τέλος, υπάρχει το Παράρτημα Α, ένα παράρτημα που φιλοξενεί ορισμούς μαθηματικών εννοιών που χρησιμοποιούνται στο κύριο μέρος της παρούσας διατριβής και οι οποίες μελετώνται σε πεδία των μαθηματικών εκτός της Θεωρίας Πολυπλοκότητας και της Θεωρίας Γραφημάτων.

Abstract

Given a graph with weights on its vertices as input, the (weighted) Feedback Vertex Set (FVS) problem asks for a minimum-weight subset of the input graph's vertices whose removal from it results in it no longer having an induced cycle. FVS finds itself among the classical problems of Algorithmic Graph Theory and has found many applications in other fields of study over the years, with applications in constraint satisfaction and Bayesian inference, optical networking and computational biology being some recent additions. As a natural consequence, FVS solving algorithms have always been a subject of active research. Both exact and approximation algorithms have been proposed for solving FVS on general graphs. FVS is $\mathcal{NP}$-complete on general graphs, planar graphs, bipartite graphs and planar bipartite graphs. Therefore, FVS is considered unlikely to be polynomially solvable on those graphs classes. We also mention that FVS is $\mathcal{FPT}$ on general graphs. FVS is polynomially solvable on interval graphs in $\mathrm{O}(n+m)$ time, on permutation graphs and trapezoid graphs in $\mathrm{O}(nm)$ time, on cocomparability graphs and convex bipartite graphs in $\mathrm{O}(n^2m)$ time, on AT-free graphs in $\mathrm{O}(n^8m^2)$ time, on chordal graphs in $\mathrm{O}(n^6)$ time and on graphs of bounded cliquewidth in $\mathrm{O}(n)$ time, where n and m are the number of vertices and edges of the input graph respectively.

In this thesis, we study a generalization of FVS called the Subset Feedback Vertex Set problem. Given a graph with weights on its vertices and a subset S of its vertices as input, the (weighted) Subset Feedback Vertex Set (SFVS) problem asks for a minimum-weight subset of the input graph's vertices whose removal from it results in it no longer having an induced cycle that passes through a vertex which is an element of S. Since potential generalizations of FVS applications may require solving a generalization of FVS in place of FVS itself, the absence of efficient SFVS solving algorithms may hamper their realisation. The above observation compels us to pursue the study

of SFVS. Both exact and approximation algorithms have also been proposed for solving SFVS on general graphs. The fact that SFVS is a generalization of FVS implies that it is $\mathcal{NP}$-complete on general graphs, planar graphs, bipartite graphs and planar bipartite graphs as well. The first significant difference in the behaviour of the two problems is that, unlike FVS which is $\mathcal{P}$ on chordal graphs, SFVS was shown to be $\mathcal{NP}$-complete on split graphs, a subclass of chordal graphs. Also unlike FVS, there is no polynomial result where the input is restricted to graph classes regarding SFVS to be found in literature. In this thesis, we propose novel dynamic programming algorithms for solving SFVS on interval graphs in $\mathrm{O}(n + m + l)$ time and on permutation graphs in $\mathrm{O}(m^3)$ time, where n, m and $l \in \mathrm{O}(n^3)$ are the number of vertices, edges and induced triangles of the input graph respectively—the first polynomial results regarding SFVS.

This thesis is structured as follows: Chapter 1 is an introductory chapter that supplies all the necessary definitions from Complexity Theory and Graph Theory. It also hosts information on FVS and SFVS. In Chapter 2, we give the best polynomial FVS solving dynamic programming algorithms on interval graphs and permutation graphs found in literature and subsequently propose novel dynamic programming algorithms which excibit the same time complexity and are the precursors of our algorithms for solving SFVS on the same graph classes. Chapter 3 hosts our aforementioned polynomial SFVS solving dynamic programming algorithms on interval graphs and permutation graphs. Chapter 4 concludes this thesis with an update on the state of FVS and SFVS and a talk on future work and open problems regarding SFVS. Lastly, there is Appendix A, an appendix hosting definitions of mathematical concepts that are used in the main matter of this thesis and which are subjects of fields of mathematics other that Complexity Theory and Graph Theory.

Contents

Chapter 1

Preliminaries

1.1 Complexity Theory

A central question in Computer Science is how to identify which is the "fastest" way to solve a problem. That is, given two algorithms A_1 and A_2 that solve the same problem P, can we say that A_1 is in some sense "faster" than A_2 in solving P for all its instances? Complexity Theory gives us a framework to answer this question mathematically.

In order to compare the speed of two algorithms, we first need to define how to determine an algorithm's speed. In Complexity Theory, we do not involve ourselves with the speed of algorithms per se; we use the amount of time that they require to be completed instead. To ensure that our measurements do not depend on the specifics of the hardware that we might be using at any given time, for otherwise our comparisons would be invalid, we consider the abstraction that performing an elementary operation requires exactly one time unit.

A *(computational) time complexity function* of an algorithm is a function from $\mathbb{N}$ to $\mathbb{N}$ that maps an input size to the amount of time the algorithm requires to be completed when given an input of that size *under a certain scenario*, that is, which satisfies a certain assumption.

- Under the *best-case* scenario, the input size is maped to the least amount of time the algorithm may require to be completed when given an input of that size.

- Under the *average-case* scenario, the input size is maped to the average amount of time the algorithm may require to be completed when given an input of that size.
- Under the *worst-case* scenario, the input size is maped to the greatest amount of time the algorithm may require to be completed when given an input of that size.

Even though all three aforementioned scenarios provide insight into an algorithm's performance, the worst-case scenario is the one that gives us a *performance guarantee*; an algorithm will always be completed faster than or as fast as it does under the worst-case scenario. In this thesis, we exclusively study the time complexity of algorithms under the worst case scenario.

Practice shows that being faster than others on small input sizes is irrelevant for an algorithm; algorithms are so fast on small input sizes that their performance differences in that regard are negligible. What we need is algorithms whose time complexity *scales* as best as possible with the input size, that is, it grows as slow as possible as the input size grows; given two algorithms A_1 and A_2 that solve the same problem P, if A_1's time complexity scales better with the input size than A_2's , A_1 will eventually outperform A_2 when the input size becomes sufficiently large—exactly where it counts.

1.1.1 Asymptotic Notation

We use $f(n)$ to denote a function $f : \mathbb{N} \to \mathbb{N}$, $n \mapsto f(n)$.

Definition 1.1.1 (Big O)**.** A function $f(n)$ is (in) $\mathrm{O}(g(n))$ if there are constants $N \in \mathbb{N}$ and $c \in \mathbb{R}$ such that $f(n) \leq cg(n)$ for all $n \in \mathbb{N}$ greater or equal to N or, equivalently, if

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty.$$

Definition 1.1.2 (Big Omega)**.** A function $f(n)$ is (in) $\Omega(g(n))$ if there are constants $N \in \mathbb{N}$ and $c \in \mathbb{R}$ such that $cg(n) \leq f(x)$ for all $n \in \mathbb{N}$ greater or equal to N or, equivalently, if

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} < \infty.$$

Definition 1.1.3 (Big Theta)**.** A function $f(n)$ is (in) $\Theta(g(n))$ if it is both $\mathrm{O}(g(n))$ and $\Omega(g(n))$.

The asymptotic notations of O, Ω and Θ essentially define classes of functions that asymptotically grow at most, at least and exactly as fast as $g(n)$ respectively. Thus, we may use asymptotic notation to compare algorithms in terms of time complexity. If the time complexity of an algorithm (as a function) is of a certain kind, then we may say that the algorithm itself is of the same kind. For example, we may say that an algorithm is *polynomial* if its time complexity is $O(p(n))$ for some polynomial function $p(n)$ where n denotes the size of its input.

1.1.2 Complexity Classes

For solving a particular problem, however, many algorithms excibiting many different time complexities may exist. This implies that the time complexity of a problem itself cannot not be determined from the time complexity of any single algorithm solving it. We define the time complexity of a problem to be the best among the time complexities of all algorithms that solve it. Since, of course, we may not already know all algorithms solving a particular problem, we consider its time complexity to be the best among the time complexities of all algorithms solving it that we already known.

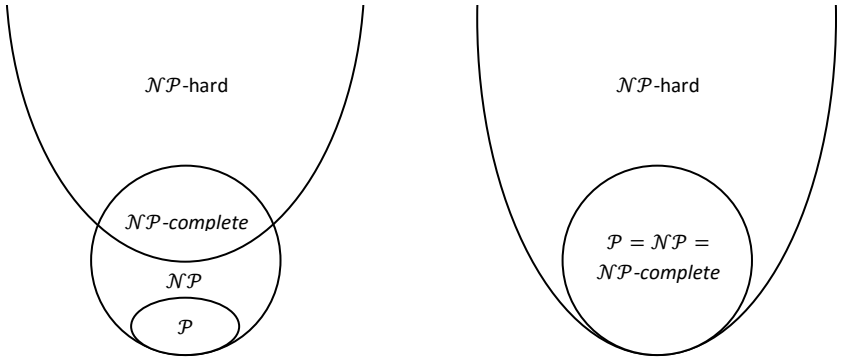
In order to take advantage of the fact that problems requiring the same time complexity to be solved may also excibit similar structure and vice versa, we define complexity classes of problems. Complexity classes are collections of problems and form a *hierarchy* according to the order of set inclusion. We subseqeuntly define all complexity classes mentioned in this thesis. For more information on complexity, complexity classes and where many problems are currently standing, the reader may refer to [20].

Polynomial

A problem is (in) $\mathcal{P}$*olynomial* if there is a polynomial algorithm solving it.

Non-deterministic Polynomial

A problem is (in) $\mathcal{N}$*on-deterministic* $\mathcal{P}$*olynomial* if there is a polynomial algorithm solving its *solution-verification* problem, that is, the problem of verifying whether a candidate solution indeed is a solution.



(a) (b)

Figure 1.1: Euler diagrams of the $\mathcal{P}$, $\mathcal{NP}$, $\mathcal{NP}$-hard and $\mathcal{NP}$-complete complexity classes (a) if $\mathcal{P} \neq \mathcal{NP}$ and (b) if $\mathcal{P} = \mathcal{NP}$.

A*polynomial reduction* of a problem P_1 to another problem P_2 is a polynomial algorithm transforming any instance of P_1 to an instance of P_2.

A problem is (in) $\mathcal{NP}$*-hard* if there is a polynomial reduction of P to it for all $P \in \mathcal{NP}$. A problem is (in) $\mathcal{NP}$*-complete* if it is both $\mathcal{NP}$ and $\mathcal{NP}$-hard.

The non-deterministic complexity classes play a central role in Complexity Theory, because many important problems were shown to be in them. What is arguably the most famour open problem in all Computer Science, the $\mathcal{P}$ versus $\mathcal{NP}$ problem, asks whether $\mathcal{P}$ is equal to $\mathcal{NP}$ or not. Figure 1.1 illustrates those two possibilities.

Fixed Parameter Tractable

We may choose to consider the time complexity of an algorithm (and consequently of a problem) to be a function of any number of parameters of its input instead of the typical choice of only the size of its input.

A problem is (in) $\mathcal{F}$*ixed* $\mathcal{P}$*arameter* $\mathcal{T}$*ractable* if there is an $\mathrm{O}(f(k)p(n))$ algorithm solving it for an arbitrary function $f(k)$ and a polynomial function $p(n)$ where n and k are the size and another parameter of its input respectively, that is, there is an algorithm solving it which is polynomial provided a certain parameter of its input other than its size is considered fixed.

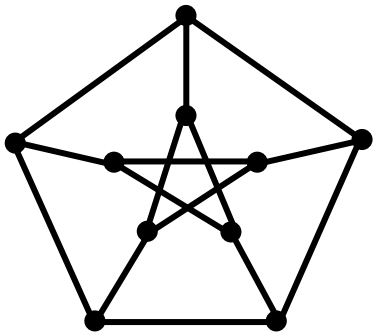


Figure 1.2: Petersen's graph.

1.2 Graph Theory

The term 'graph' in a mathematical context is most commonly used to refer to the **graph**ical representation of a function. In the specific context of Graph Theory, however, the term 'graph' is used to refer to objects like the one shown in Figure 1.2. In this section, we provide definitions regarding those objects including characteristics that they might excibit, relations that they may have to one another and collections that they might be part of.

1.2.1 Fundamental Concepts

A *(simple undirected) graph* is an ordered pair $G = (V, E)$ of sets such that $E \subseteq \{\{u, v\} : u, v \in V \text{ and } u \neq v\}$. V is called *the vertex set* (and its elements are called *vertices*) of G and E is called *the edge set* (and its elements are called *edges*) of G. We commonly use n and m to denote $|V|$ and $|E|$ respectively. The graphical representantion of a graph is shown in Figure 1.2; a vertex of the graph is represented by a point and an edge between two vertices by a line segment connecting the two points that represent the vertices.

We say that an edge between two vertices *directly connects* those vertices; the vertices it directly connects are called *adjacent* or *neighbouring. The neighbourhood* of a vertex i in a graph is the set of all other vertices that are adjacent to i in the graph. We use $N(i)$ to denote the neighbourhood of i. We say that a vertex that has an empty neighbourhood is *isolated*.

A *path* between two vertices in a graph is a (finite) sequence of vertices of the graph such that the two vertices occupy the first and last positions in the

sequence and all vertices that occupy subsequent positions in the sequence are adjacent. We say that a path between two vertices *connects* those vertices. A graph is *connected* if all its vertices are pairwise connected. For a connected graph, since no vertex is isolated, $m \in \Omega(n)$. *The length* of a path is its length as a sequence minus one. The distance of two vertices in a graph is the minimum among all the lengths of paths between them in the graph.

A *subgraph* of a graph $G = (V, E)$ is a graph $G' = (V', E')$ such that $V' \subseteq V$ and $E' \subseteq \{\{u, v\} \in E : u, v \in V'\}$. *The subgraph* of G *induced* by a set $X \subseteq V$ is the subgraph $G' = (V', E')$ of G such that $V' = X$ and $E' = \{\{u, v\} \in E : u, v \in V'\}$. We use $G[X]$ to denote the subgraph of G induced by X.

1.2.2 Graph Classes

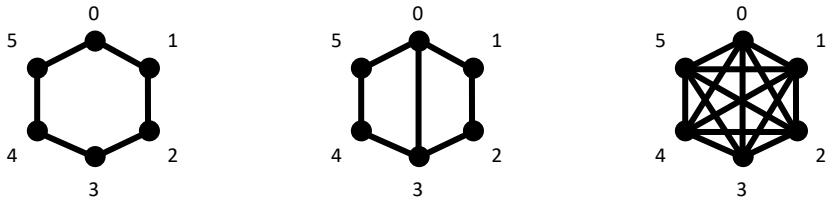
In order to handle graphs exhibiting common structure together, we define graph classes. Graph classes are collections of graphs and form an *hierarchy* according to the order of set inclusion.

Most graph classes have more that one equivalent definitions, called *characterizations*. Researchers are always on the lookout for new characterizations of known graph classes, because a problem may be easier and/or faster to solve on a graph class when using one of its characterizations instead of another. A common and much sought after type of characterization is for the members of a class to be $\mathcal{C}$*-free* where $\mathcal{C}$ is a collection of graphs. In that characterization, $\mathcal{C}$ acts as a list of *forbidden* induced subgraphs; members of the class have no induced subgraph listed in C. Another sought after characterization is that of an intesection model. An *intersection model* of a graph is a collection of sets for which there is a one-to-one correspondence from the sets of the collection to the vertices of the graph such that two vertices are adjacent if and only if their corresponding sets are intersecting.

We subsequently characterize all graph classes mentioned in this thesis. For more information on these graph classes and more, the reader may refer to [6].

Independent Sets

An *independent set* is a graph that has no edge.



(a) $\langle 0, 1, 2, 3, 4, 5 \rangle$.

(b) An even cycle having an odd chord.

(c) A clique of six vertices.

Figure 1.3

Cliques

A *clique* is a graph that has all possible edges between its vertices.

Cycles

A *(chordless) cycle* is a graph G for which there is a one-to-one correspondence from the integers of $\mathbb{Z}_n$ to the vertices of G such that two vertices are adjacent if and only if their corresponding integers differ by 1 modulo n. If $i \mapsto v_i$ for all $i \in \mathbb{Z}_n$, then $G = \langle v_0, v_1, \ldots, v_{n-1} \rangle$. An *even (odd)* cycle is a cycle of an even (odd) number of vertices. In this thesis, we call a cycle of three vertices a *triangle* and a cycle of four vertices a *square*.

A *chord* is an edge that G has in addition to those of the chordless cycle. An *odd (even) chord* is a chord between two vertices whose distance in the chordless cycle is odd (even).

Forests

The class of forests is the class of cycle-free graphs; that is, a forest is a graph that has no induced cycle.

Planar Graphs

A *planar graph* is a graph that can be graphically represented on a plane such that the line segments representing its edges are pairwise disjoint.

Bipartite Graphs

A *bipartite graph* is a graph $G = (V, E)$ for which there is a partition of V into A and B such that $G[A]$ and $G[B]$ are both independent sets.

Convex Bipartite Graphs

A *convex bipartite graph* is a bipartite graph $G = (A, B, E)$ for which there is a total order $<$ of A such that every B-vertice has all its neighbouring A-vertices be consecutive with respect to $<$.

Convex bipartite graphs form a subclass of bipartite graphs.

Chordal Graphs

The class of *chordal graphs* is the class of $\{\langle v_0, v_1, \ldots, v_{n-1}\rangle : n \geq 4\}$-free graphs; that is, a chordal graph is a graph whose induced cycles are all triangles, or, equivalently, a graph whose induced cycles that aren't triangles have a chord.

Strongly Chordal Graphs

A *strongly chordal graph* is a chordal graph whose induced even cycles have an odd chord.

Strongly chordal graphs form a subclass of chordal graphs.

Split graphs

A *split graph* is a graph $G = (V, E)$ for which there is a partition of V into A and B such that $G[A]$ is a clique and $G[B]$ is an independent sets.

Split graphs form a subclass of chordal graphs.

AT-free Graphs

An *asteroidal triple* of a graph G are three vertices of G for which there is a path in G between any two of them such that it does not pass through

the neighbourhood of the third one. An *AT-free graph* is a graph that has no asteroidal triple.

Cocomparability Graphs

A *cocomparability* graph is a graph for which there is an irreflexive and transitive relation $\mathcal{R}$ on its vertex set such that two vertices are adjacent if and only if they are not comparable with respect to $\mathcal{R}$.

Cocomparability graphs form a subclass of both $\{\langle v_0, v_1, \ldots, v_{n-1}\rangle : n \geq 5\}$-free graphs and AT-free graphs.

Trapezoid Graphs

A *trapezoid graph* is a graph G for which there is a collection $\mathcal{C}$ of trapezoids on the plane that all have two of their edges be line segments of the same two parallel lines and a one-to-one correspondence of trapezoids of $\mathcal{C}$ to vertices of G such that two vertices are adjacent if and only if their corresponding trapezoids are intersecting.

Trapezoid graphs form a subclass of cocomparability graphs.

Permutation Graphs

A *permutation graph* is a graph G for which there is a permutation $\pi : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ and a one-to-one correspondence of integers of $\{1, 2, \ldots, n\}$ to vertices of G such that two vertices are adjacent if and only if π reverses their corresponding integers' relative order.

Permutation graphs form a subclass of trapezoid graphs.

Interval Graphs

An *interval graph* is a graph G for which there is a collection $\mathcal{C}$ of closed intervals of $\mathbb{R}$ and a one-to-one correspondence of intervals of $\mathcal{C}$ to vertices of G such that two vertices are adjacent if and only if their corresponding intervals are intersecting.

Interval graphs form a subclass of both chordal graphs and permutation graphs.

1.2.3 Graph Parameters

Many important $\mathcal{NP}$-complete problems are graph-related. To help extend the study of those problems' time complexity to *complexity measures* beyond simply graph size, many *graph parameters*, measures of various aspects of a graph's structure, were devised. One such measure mentioned in this thesis is cliquewidth, which was first introduced by Courcelle et al. [9, 10].

Cliquewidth

The *cliquewidth* of a graph is the minimum number of distinct labels needed to construct a copy of it with labeled vertices when the only available operations to do so are

- creating a single-vertex graph with its vertex labeled i,
- the disjoint union of two thusly-constructed graphs with labeled vertices,
- changing the labels of all vertices labeled i on a thusly-constructed graph with labeled vertices from i to j and
- directly connecting all vertices labeled i on a thusly-constructed graph with labeled vertices to all its vertices labeled j.

1.3 The Problems

The two problems we involve ourselves with in this thesis are both special cases, called *variants* by some authors, of the Feedback Set (FS) problem. Given a graph G as input, all FS variants ask for a set of certain elements of G such that certain induced cycles of G are no longer induced cycles once those elements are removed. We state the formal definitions of the two aforementioned problems and information regarding their tractability below. For more information on recent developments in FS related research, the reader may consult [15].

1.3.1 Feedback Vertex Set

A *feedback vertex set* of a graph $G = (V, E)$ is a set $U \subseteq V$ such that $G[V \setminus U]$ is a forest.

Given a graph $G = (V, E)$ (along with a weight function $w : V \to \mathbb{R}$) as input, the (weighted) Feedback Vertex Set (FVS) problem asks for a feedback vertex set of G that has minimum cardinality (weight) among all feedback vertex sets of G.

FVS finds itself among the classical problems of Algorithmic Graph Theory and has found many applications in other fields of study over the years, with applications in constraint satisfaction and Bayesian inference [13, 12, 1], optical networking [22] and computational biology [18] being some recent additions. As a natural consequence, FVS solving algorithms have always been a subject of active research. Both exact [16] and approximation [2] algorithms have been proposed for solving FVS on general graphs.

FVS is $\mathcal{NP}$-complete on general graphs. In fact, it was one of the first problems that were shown to be $\mathcal{NP}$-complete; it was included in Karp's list of 21 $\mathcal{NP}$-complete problems [21]. It is also $\mathcal{NP}$-complete on planar graphs [20], bipartite graphs [30] and planar bipartite graphs [29]. Therefore, FVS is considered unlikely to be polynomially solvable on those graph classes. We also mention that FVS is $\mathcal{FPT}$ on general graphs [7].

When a problem is shown to be $\mathcal{NP}$-complete on general graphs, we start searching for graph classes on which it is $\mathcal{P}$. Research in that regard for FVS has been fruitious; the list of graph classes on which it was shown to be polynomially solvable includes interval graphs [27], permutation graphs [24], trapezoid graphs, cocomparability graphs and convex bipartite graphs [25], AT-free graphs [23], chordal graphs [8] and graphs of bounded cliquewidth [28, 19].

Figure 1.4 shows a graphical representation of the hierarchy of most of the graph classes that we mention in this thesis augmented with the best known results regarding FVS on them found in literature. For the behaviour of problems on graph classes, it is not difficult to show the following:

- A problem that is $\mathcal{NP}$-complete on a graph class is also $\mathcal{NP}$-complete on all superclasses of that graph class.
- An algorithm that solves a problem on a graph class in $O(g(n))$ time also solves it on all subclasses of that graph class in $O(g(n))$ time.

1.3.2 Subset Feedback Vertex Set

Definition 1.3.1. Let $G = (V, E)$ be a graph and $S \subseteq V$. Then

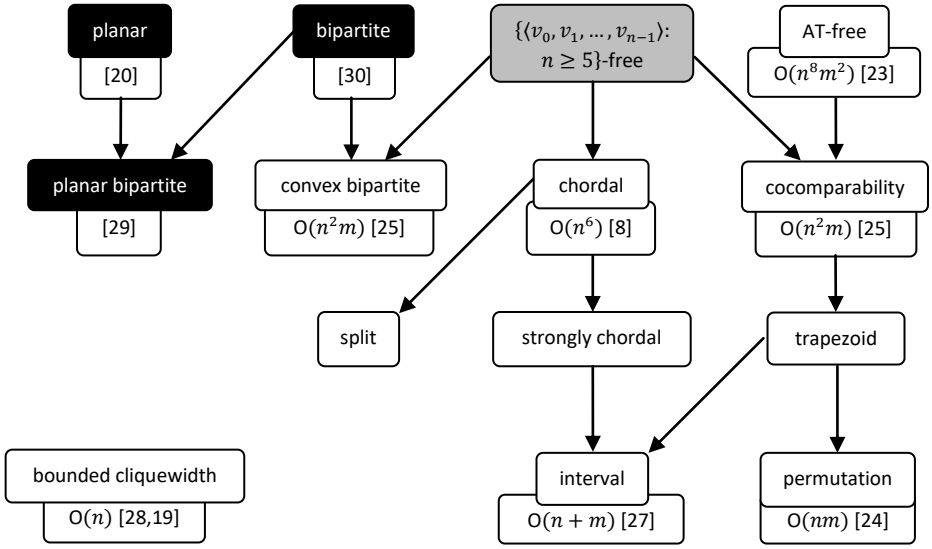


Figure 1.4: Best known results regarding FVS on the listed graph classes. An arrow between two graph classes indicates that the source is a superclass of the target. Graph classes are colored black if FVS is $\mathcal{NP}$-complete on them, gray if its behaviour on them is unknown and white if it is $\mathcal{P}$ on them. (We follow this style throughout this thesis.)

- an S-vertex is a vertex that is an element of S,
- an S-cycle is a cycle that has an S-vertex and
- an S-forest is an S-cycle-free graph; that is, an S-forest is a graph that has no induced S-cycle.

An *S-feedback vertex set* of a graph $G = (V, E)$ where $S \subseteq V$ is a set $U \subseteq V$ such that $G[V \setminus U]$ is an S-forest. Whevener the subset S is unspecified or not subject to confusion, we use the term *subset feedback vertex set* instead.

Given a graph $G = (V, E)$ along with a set $S \subseteq V$ (and a weight function $w : V \to \mathbb{R}$) as input, the (weighted) Subset Feedback Vertex Set (SFVS) problem asks for an S-feedback vertex set of G that has minimum cardinality (weight) among all S-feedback vertex sets of G.

Both exact [17] and approximation [14] algorithms have also been proposed for solving SFVS on general graphs. SFVS naturally generalises FVS; we may solve FVS on a graph $G = (V, E)$ by solving SFVS on G for $S = V$.

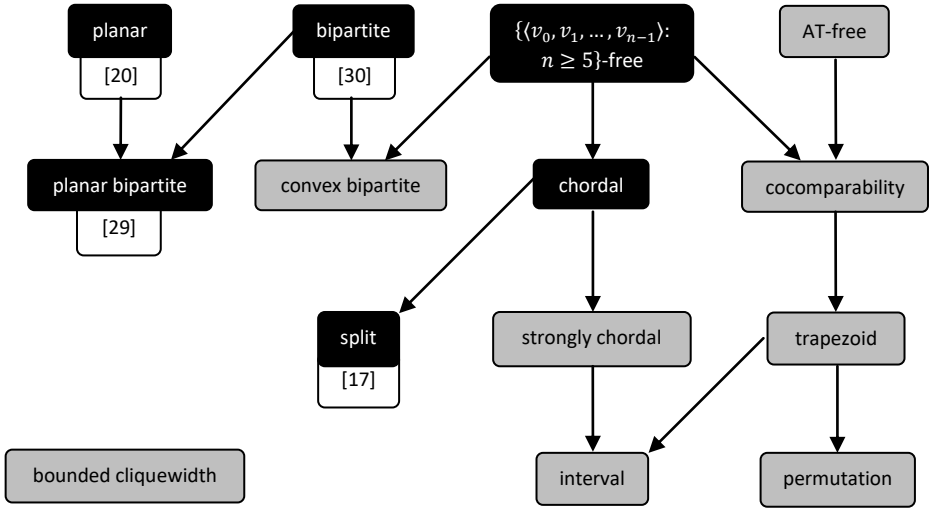


Figure 1.5: Best known results regarding SFVS on the listed graph classes.

This implies that SFVS is $\mathcal{NP}$-complete on general graphs, planar graphs, bipartite graphs and planar bipartite graphs as well. We also mention that SFVS is $\mathcal{FPT}$ on general graphs [11].

The first significant difference in the behaviour of the two problems is that, unlike FVS which is $\mathcal{P}$ on chordal graphs, SFVS was shown to be $\mathcal{NP}$-complete on split graphs, a subclass of chordal graphs [17]. Also unlike FVS, there is no polynomial result where the input is restricted to graph classes regarding SFVS to be found in literature.

Figure 1.5 shows a graphical representation of the same hierarchy of graph classes as Figure 1.4 augmented with the best known results regarding SFVS on them found in literature.

Chapter 2

Solving FVS in Polynomial Time

In this chapter, we present the best know dynamic programming algorithms for solving FVS on interval graphs [27] and permutation graphs [24] in polynomial time found in literature as well as our own novel dynamic programming algorithms for solving it on the same graph classes in the same times. These algorithms do not find a feedback vertex set of the input graph that has minimum weight directly and return it; they find a forest-inducing vertex set of the input graph that has maximum weight and return the set of all other vertices instead. Throughout this chapter, we use $\mathcal{F}$ to denote the collection of all induced forests of the input graph.

2.1 Best Known Algorithm on Interval Graphs

In this section we reproduce the $O(n+m)$ dynamic programming algorithm for solving FVS on interval graphs proposed by Lu and Tang in [27]. Minor adjustments to definitions, notation and presentation have been made to improve readability without alteration of the results. All proofs are omitted.

We assume that we are given an interval graph $G = (V, E)$ along with a corresponding collection of closed intervals of $\mathbb{R}$ where all endpoints are distinct and a weight function $w : V \to \mathbb{R}$ as input. Wherever $\max_w$ is called in this section, it returns an arbitrary choice that has maximum weight among all its operands. We add an isolated dummy vertex that has non-positive

weight to G and we add a dummy interval that has minimum right endpoint to the given collection as its corresponding interval. We consider vertices of G to be equivalent to their corresponding intervals and to the integers of $\{0, 1, \ldots, n\}$ that indicate their position when sorted in ascending order of their corresponding intervals' right endpoints.

For every vertex i of G, we use $a(i)$ and $b(i)$ to denote its corresponding interval's left and right endpoint respectively. We consider the relation on V that is defined by

$$i \leq_r j \iff b(i) \leq b(j)$$

for all $i, j \in V$. Since all endpoints of the collection's intervals are distinct, it is not difficult to show that $\leq_r$ is a total order on V. Wherever $\max_r$ is called in this section, it returns the maximum among its operands with respect to $\leq_r$. We define some predecessors with respect to $\leq_r$ and the V-sets, which correspond to the subproblems that Lu and Tang's [27] dynamic programming algorithm wants to solve.

Definition 2.1.1 (Predecessors)**.** Let $i \in V \setminus \{0\}$. Then

$$\ll i = \max_r\{h \in V : h <_r i \text{ and } \{h, i\} \notin E\}.$$

Example 2.1.1. For the interval graph that has

$$\mathcal{C} = \left\{ \begin{array}{c} I_1 = [2, 4], I_2 = [1, 6], I_3 = [7, 8], I_4 = [5, 10], \\ I_5 = [3, 11], I_6 = [9, 13], I_7 = [12, 14] \end{array} \right\}$$

as a corresponding collection of closed intervals of $\mathbb{R}$, $\ll 6 = 3$.

Definition 2.1.2 (V-sets)**.** Let $i, j \in V$ such that $i <_r j$. Then

$$V_{i,j} = \{h \in V : h \leq_r i\} \cup \{j\}.$$

Now, we define the sets that Lu and Tang's [27] dynamic programming algorithm computes in order to conpute the forest-inducing vertex set of G that has maximum weight.

Definition 2.1.3 ([27])**.** Let $i, j \in V$ such that $i <_b j$. Then

- $A_{i,j} = \max_w\{X \subseteq V_{i,j} : G[X] \in \mathcal{F}\}$,
- $B_{i,j} = \max_w\{X \subseteq V_{i,j} : G[X] \in \mathcal{F} \text{ and } i, j \in X\}$ and
- $C_{i,j} = \max_w\{X \subseteq V_{i,j} : G[X] \in \mathcal{F} \text{ and } j \in X\}$.

Observe that, since $V_{n-1,n} = V$, $A_{n-1,n} = \max_w\{X \subseteq V : G[X] \in \mathcal{F}\}$. The following lemmas state how to recursively compute all the sets of Definition 2.1.3.

Lemma 2.1.1 ([27]). *Let $i, j \in V$ such that $i <_r j$.*

(1) If $i = 0$, then $A_{0,j} = \max_w \{\varnothing, \{j\}\}$.

(2) If $i \neq 0$ and $b(i) < a(j)$, then $A_{ij} = \max_w \{A_{i-1,i}, A_{i-1,i} \cup \{j\}\}$.

(3) If $i \neq 0$ and $a(j) < b(i)$, then $A_{ij} = \max_w \{A_{i-1,i}, A_{i-1,j}, B_{i,j}\}$.

Lemma 2.1.2 ([27]). *Let $i, j \in V$ such that $i <_r j$.*

(1) If $i = 0$, then $B_{0,j} = \{0, j\}$.

(2) If $i \neq 0$ and $b(i) < a(j)$, then $B_{i,j} = C_{i-1,i} \cup \{j\}$.

(3) If $i \neq 0$ and $a(j) < a(i)$, then $B_{i,j} = C_{\ll i,j} \cup \{i\}$.

(4) If $i \neq 0$ and $a(i) < a(j) < b(i)$, then $B_{i,j} = C_{\ll j,i} \cup \{j\}$.

Lemma 2.1.3 ([27]). *Let $i, j \in V$ such that $i <_r j$.*

(1) If $i = 0$, then $C_{0,j} = \{j\}$.

(2) If $i \neq 0$ and $b(i) < a(j)$, then $C_{i,j} = A_{i-1,i} \cup \{j\}$.

(3) If $i \neq 0$ and $a(j) < b(i)$, then $C_{i,j} = \max_w \{C_{i-1,j}, B_{i,j}\}$.

Theorem 2.1.4 ([27]). *The dynamic programming algorithm shown in Figure 2.1 returns a feedback vertex set of G that has minimum weight in $O(n + m)$ time.*

2.2 Best Known Algorithm on Permutation Graphs

The first algorithm to solve FVS on permutation graphs was an $O(n^6)$ dynamic programming algorithm proposed by Brandstädt and Kratsch in [4, 5]. The time complexity was later improved to $O(nm\bar{m})$ where $\bar{m}$ is the number of edges in the input graph's complement by Brandstädt in [3] and subsequently to $O(nm)$ by Liang in [24]. In this section we reproduce the $O(nm)$ dynamic

```
for i := 0 to n − 1
  if i ≠ 0 then compute ≪ i of Definition 2.1.1;
  for j := i + 1 to n
    compute B_{i,j} according to Lemma 2.1.2;
    compute A_{i,j} according to Lemma 2.1.1;
    compute C_{i,j} according to Lemma 2.1.3;
    if {i, j} ∉ E then break;
  end for
end for

return V \ A_{n−1,n};
```

Figure 2.1: Lu and Tang's [27] dynamic programming algorithm for solving FVS on an interval graph.

programming algorithm proposed by Liang in [24]. Minor adjustments to definitions, notation and presentation have been made to improve readability without alteration of the results. All proofs are omitted.

We assume that we are given a connected permutation graph $G = (V, E)$ along with a corresponding permutation π of G and a weight function $w : V \to \mathbb{R}$ as input. Wherever $\max_w$ is called in this section, it returns an arbitrary choice that has maximum weight among all its operands. We add an isolated dummy vertex that has non-positive weight to G and we add $0 \mapsto 0$ to π as its first column where 0 is its corresponding integer. We consider the vertices of G to be equivalent to their corresponding integers in π's domain.

We consider the two relations on V defined by

$$\begin{aligned} i \leq_t j &\iff i \leq j \\ i \leq_b j &\iff \pi^{-1}(i) \leq \pi^{-1}(j) \end{aligned}$$

for all $i, j \in V$. It is not difficult to show that both $\leq_t$ and $\leq_b$ are total orders on V; they are exactly the orders in which the integers appear on the **top** and **bottom** row of π respectively.

Liang's [24] dynamic programming algorithm iterates on elements called *ordered crossing vertex pairs*. We define the following collections of such elements:

$$\mathcal{X} = \{ij \in V^2 : i \leq_t j \text{ and } j \leq_b i\}$$

$$\mathcal{I} = \{ij \in V^2 : i = j\} \subset \mathcal{X}$$

Observe that for every $ij \in \mathcal{X}$, if $i \neq j$, then $\{i, j\} \in E$. We consider the two relations on $\mathcal{X}$ defined by

$$\begin{array}{lcl} gh \leq_r ij & \iff & g \leq_b i \text{ and } h \leq_t j \\ gh \leq_r^{lex} ij & \iff & g <_b i \text{ or } g = i \text{ and } h \leq_t j \end{array}$$

for all $gh, ij \in \mathcal{X}$. It is not difficult to show that $\leq_r$ is a partial order on $\mathcal{X}$, which we call the **right-endpoints** *product* order, whereas $\leq_r^{lex}$ is a total order on $\mathcal{X}$, which we call the and **right-endpoints** *lexicographic* order. We also note that $\leq_r^{lex}$ is a linear extention of $\leq_r$. Wherever $\max_r$ is called in this section, it returns the maximum among its operands with respect to $\leq_r$. We define some predecessors with respect to $\leq_r$ and the V-sets, which correspond to the subproblems that Liang's [24] dynamic programming algorithm wants to solve.

Definition 2.2.1 (Predecessors)**.** Let $ij \in \mathcal{X} \setminus \{00\}$. Then

- $\lll ij = \max_r\{gh \in \mathcal{C} : gh <_r ij \text{ and } h \neq j\}$,
- $\ll ij = \max_r\{gh \in \mathcal{C} : gh <_r ij \text{ and } g \neq i\}$,
- $< ij = \max_r\{gh \in \mathcal{C} : gh <_r ij \text{ and } g \neq i \text{ and } h \neq j\}$ and
- $\lll ij = \max_r\{gh \in \mathcal{C} : gh <_r ij \text{ and } \{g, i\}, \{g, j\}, \{h, i\}, \{h, j\} \notin E\}$.

Example 2.2.1. For the permutation graph that has

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 2 & 8 & 6 & 3 & 1 & 7 & 4 & 5 \end{pmatrix}$$

as a corresponding permutation of $\{1, 2, 3, 4, 5, 6, 7, 8\}$, $\lll 57 = 56$, $\ll 57 = 47$, $< 57 = 46$, $\lll 57 = 13$ and $\lll 36, \ll 36, < 36, \lll 36 = 22$.

Definition 2.2.2 (V-sets)**.** Let $ij \in \mathcal{X}$. Then $V_{ij} = \{h \in V : hh \leq_r ij\}$.

Now, we define the sets that Liang's [24] dynamic programming algorithm computes in order to compute the forest-inducing vertex set of G that has maximum weight.

Definition 2.2.3 ([24])**.** Let $ij \in \mathcal{X}$ and $k \in V \setminus V_{ij}$. Then

- $A_{ij} = \max_w\{X \subseteq V_{ij} : G[X] \in \mathcal{F}\}$,

- $B_{ij} = \max_w\{X \subseteq V_{ij} : G[X] \in \mathcal{F} \text{ and } i,j \in X\}$
  provided $i \neq j$,
- $C_{ij,k} = \max_w\{X \subseteq V_{ij} \cup \{k\} : G[X] \in \mathcal{F} \text{ and } k \in X\}$
  provided $i <_b k$,
- $D_{ij,k} = \max_w\{X \subseteq V_{ij} \cup \{k\} : G[X] \in \mathcal{F} \text{ and } k \in X\}$
  provided $j <_t k$,
- $E_{ij,k} = \max_w\{X \subseteq V_{ij} \cup \{k\} : G[X] \in \mathcal{F} \text{ and } i,j,k \in X\}$
  provided $i \neq j$, $i <_b k$ and $k <_t j$ and
- $F_{ij,k} = \max_w\{X \subseteq V_{ij} \cup \{k\} : G[X] \in \mathcal{F} \text{ and } i,j,k \in X\}$
  provided $i \neq j$, $k <_b i$ and $j <_t k$.

Observe that, since $V_{00} = \{0\}$ and $w(0) \leq 0$, $A_{00} = \varnothing$ and $C_{00,k} = D_{00,k} = \{k\}$ for all $k \in V \setminus \{0\}$ and, since $V_{\pi(n)n} = V$, $A_{\pi(n)n} = \max_w\{X \subseteq V : G[X] \in \mathcal{F}\}$. The following lemmas state how to recursively compute all sets of Definition 2.2.3 other than A_{00} and $C_{00,k}$ and $D_{00,k}$ for all $k \in V \setminus \{0\}$.

Lemma 2.2.1 ([24]). *Let $ij \in \mathcal{X} \setminus \{00\}$. Then $A_{ij} = B_{gh}$ for some $gh \leq_r ij$.*

Lemma 2.2.2 ([24]). *Let $i \in V \setminus \{0\}$. Then $A_{ii} = \max_w\{A_{<ii}, A_{<ii} \cup \{i\}\}$.*

Lemma 2.2.3 ([24]). *Let $ij \in \mathcal{X} \setminus \mathcal{I}$. Then $A_{ij} = \max_w\{A_{\leqslant ij}, A_{\leqslant ij}, B_{ij}\}$.*

Lemma 2.2.4 ([24]). *Let $ij \in \mathcal{X} \setminus \mathcal{I}$. Then*

$$B_{ij} = \max_w\{A_{\ll ij} \cup \{i,j\}, C_{\ll jj,i} \cup \{j\}, D_{\ll ii,j} \cup \{i\}\}.$$

Lemma 2.2.5 ([24]). *Let $ij \in \mathcal{X} \setminus \{00\}$ and $k \in V \setminus V_{ij}$ such that $k <_t j$.*

(1) If $i = j$, then $C_{ii,k} = \max_w\{C_{<ii,k}, C_{<ii,k} \cup \{i\}\}$.

(2) If $i \neq j$, then $C_{ij,k} = \max_w\{C_{\leqslant ij,k}, C_{\leqslant ij,k}, E_{ij,k}\}$.

Lemma 2.2.6 ([24]). *Let $ij \in \mathcal{X} \setminus \{00\}$ and $k \in V \setminus V_{ij}$ such that $k <_b i$.*

(1) If $i = j$, then $D_{ii,k} = \max_w\{D_{<ii,k}, D_{<ii,k} \cup \{i\}\}$.

(2) If $i \neq j$, then $D_{ij,k} = \max_w\{D_{\leqslant ij,k}, D_{\leqslant ij,k}, F_{ij,k}\}$.

Lemma 2.2.7 ([24]). *Let $ij \in \mathcal{X} \setminus \mathcal{I}$ and $k \in V \setminus V_{ij}$ such that $i <_t k <_t j$. Then*

$$E_{ij,k} = \max_w\{C_{\ll kj,i} \cup \{j,k\}, D_{\ll ii,j} \cup \{i,k\}\}.$$

```
compute a list R containing all ordered crossing vertex pairs of X
    sorted in ascending order with respect to <_r^lex
    and all predecessors of Definition 2.2.1 according to [26];

for ij in R
  if i ≠ j then compute B_ij according to Lemma 2.2.4;
  compute A_ij according to Lemmas 2.2.2 and 2.2.3;
  for k := 1 to n
    if i ≠ j
      if i <_b k and k <_t j then compute E_ij,k according to Lemma 2.2.7;
      if k <_b i and j <_t k then compute F_ij,k according to Lemma 2.2.8;
    end if
    if i <_b k then compute C_ij,k according to Lemma 2.2.5;
    if j <_t k then compute D_ij,k according to Lemma 2.2.6;
  end for
end for

return V \ A_π(n)n;
```

Figure 2.2: Liang's [24] dynamic programming algorithm for solving FVS on a permutation graph.

Lemma 2.2.8 ([24])**.** *Let $ij \in \mathcal{X} \setminus \mathcal{I}$ and $k \in V \setminus V_{ij}$ such that $j <_b k <_b i$. Then*

$$F_{ij,k} = \max_{w} \{C_{\ll jj,i} \cup \{j,k\}, D_{\ll ik,j} \cup \{i,k\}\}.$$

Theorem 2.2.9 ([24])**.** *The dynamic programming algorithm shown in Figure 2.2 returns a feedback vertex set of G that has minimum weight in $O(nm)$ time.*

2.3 Our New Algorithm on Interval Graphs

In this section we propose a novel $O(n+m)$ dynamic programming algorithm for solving FVS on interval graphs as a precursor to the dynamic programming algorithm of Section 3.1 for solving SFVS on interval graphs.

We assume that we are given an interval graph $G = (V, E)$ along with a corresponding collection of closed intervals of $\mathbb{R}$ where all endpoints are distinct and a weight function $w : V \to \mathbb{R}$ as input. We extend w such that if it is

called with a set of vertices, then it returns the sum of their respective weights. Wherever $\max_w$ is called in this section, it returns an arbitrary choice that has maximum weight among all its operands. We add an isolated dummy vertex that has non-positive weight to G and we add a dummy interval that has minimum right endpoint to the given collection as its corresponding interval. We consider vertices of G to be equivalent to their corresponding intervals and to the integers of $\{0, 1, \ldots, n\}$ that indicate their position when sorted in ascending order of their corresponding intervals' right endpoints.

For every vertex i of G, we use $a(i)$ and $b(i)$ to denote its corresponding interval's left and right endpoint respectively. We consider the two relations on V that are defined by

$$\begin{aligned} i \leq_l j &\iff a(i) \leq a(j) \\ i \leq_r j &\iff b(i) \leq b(j) \end{aligned}$$

for all $i, j \in V$. Since all endpoints of the collection's intervals are distinct, it is not difficult to show that $\leq_l$ and $\leq_r$ are total orders on V. Wherever $\min_l$ is called in this section, it returns the minimum among its operands with respect to $\leq_l$; and wherever $\max_r$ is called in this section, it returns the maximum among its operands with respect to $\leq_r$. We define some predecessors with respect to $\leq_r$ and the V-sets, which correspond to the subproblems that our dynamic programming algorithm wants to solve.

Definition 2.3.1 (Predecessors)**.** Let $i \in V \setminus \{0\}$. Then

- $< i = \max_r\{h \in V : h <_r i\}$ and
- $\ll i = \max_r\{h \in V : h <_r i \text{ and } \{h, i\} \notin E\}$.

Example 2.3.1. For the interval graph that has

$$\mathcal{C} = \left\{ \begin{array}{c} I_1 = [2, 4], I_2 = [1, 6], I_3 = [7, 8], I_4 = [5, 10], \\ I_5 = [3, 11], I_6 = [9, 13], I_7 = [12, 14] \end{array} \right\}$$

as a corresponding collection of closed intervals of $\mathbb{R}$, $< 6 = 5$ and $\ll 6 = 3$.

Definition 2.3.2 (V-sets)**.** Let $i \in V$. Then $V_i = \{h \in V : h \leq_r i\}$.

Observation 2.3.1. *Let $i \in V \setminus \{0\}$ and $j \in V \setminus V_i$ such that $\{i, j\} \in E$. Then*

(1) $V_i = V_{<i} \cup \{i\}$ and

(2) $V_{<i} = V_{\ll j} \cup \{h \in V_{<i} : \{h, j\} \in E\}$.

Now, we define the sets that our dynamic programming algorithm computes in order to conpute the forest-inducing vertex set of G that has maximum weight.

Definition 2.3.3 (A-sets)**.** Let $i \in V$. Then,

$$A_i = \max_w \mathcal{A}_i = \max_w\{X \subseteq V_i : G[X] \in \mathcal{F}\}.$$

Definition 2.3.4 (B-sets)**.** Let $i \in V$ and $x \in V \setminus V_i$. Then,

$$B_i^x = \max_w \mathcal{B}_i^x = \max_w\{X \subseteq V_i : G[X \cup \{x\}] \in \mathcal{F}\}.$$

Observe that, since $V_0 = \{0\}$ and $w(0) \leq 0$, $A_0 = \varnothing$ and, since $V_n = V$, $A_n = \max_w\{X \subseteq V : G[X] \in \mathcal{F}\}$. The following lemmas state how to recursively compute all A-sets and B-sets other than A_0.

Lemma 2.3.2 (A-sets)**.** *Let $i \in V \setminus \{0\}$. Then $A_i = \max_w \left\{A_{<i}, B^i_{<i} \cup \{i\}\right\}$.*

Proof. Let $i \in V \setminus \{0\}$.

Let $i \notin A_i$. By this fact, Observation 2.3.1(1) and Definition 2.3.3, it follows that

$$\left.\begin{array}{l} A_i \in \mathcal{A}_{<i} \\ A_{<i} \in \mathcal{A}_i \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(A_i) \leq w(A_{<i}) \\ w(A_{<i}) \leq w(A_i) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(A_i) = w(A_{<i}) \Rightarrow A_i = A_{<i}.$$

Let $i \in A_i$. By this fact, Observation 2.3.1(1) and Definitions 2.3.3 and 2.3.4, it follows that

$$\left.\begin{array}{l} A_i \setminus \{i\} \in \mathcal{B}^i_{<i} \\ B^i_{<i} \cup \{i\} \in \mathcal{A}_i \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(A_i \setminus \{i\}) \leq w(B^i_{<i}) \\ w(B^i_{<i} \cup \{i\}) \leq w(A_i) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(A_i) = w(B^i_{<i} \cup \{i\}) \Rightarrow A_i = B^i_{<i} \cup \{i\}. \qquad \square$$

Lemma 2.3.3 (B-sets)**.** *Let $i \in V$ and $x \in V \setminus V_i$.*

(1) If $\{i, x\} \notin E$, then $B_i^x = A_i$.

(2) If $\{i, x\} \in E$, then $B_i^x = \max_w \left\{B^x_{<i}, B^{x'}_{\ll y'} \cup \{i\}\right\}$ where $x' = \min_l\{i, x\}$ and $y' = \min_l(\{i, x\} \setminus \{x'\})$.

Proof. Let $i \in V$ and $x \in V \setminus V_i$.

(1) Let $\{i, x\} \notin E$. Then $b(i) < a(x)$, so the neighbourhood of x in $G[V_i \cup \{x\}]$ is $\varnothing$. Consequently, no subset of $V_i \cup \{x\}$ that contains x induces a cycle of G. By this fact and Definitions 2.3.3 and 2.3.4, it follows that

$$\left.\begin{array}{l} B_i^x \in \mathcal{A}_i \\ A_i \in \mathcal{B}_i^x \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B_i^x) \leq w(A_i) \\ w(A_i) \leq w(B_i^x) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(B_i^x) = w(A_i) \Rightarrow B_i^x = A_i.$$

(2) Let $\{i, x\} \in E$.

Let $i \notin B_i^x$. By this fact, Observation 2.3.1(1) and Definition 2.3.4, it follows that

$$\left.\begin{array}{l} B_i^x \in \mathcal{B}_{<i}^x \\ B_{<i}^x \in \mathcal{B}_i^x \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B_i^x) \leq w(B_{<i}^x) \\ w(B_{<i}^x) \leq w(B_i^x) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(B_i^x) = w(B_{<i}^x) \Rightarrow B_i^x = B_{<i}^x.$$

Let $i \in B_i^x$. By this fact and Observation 2.3.1(1), it follows that (i) $B_i^x \setminus \{i\} \subseteq V_{<i}$. We define $x' = \min_l\{i, x\}$ and $y' = \min_l(\{i, x\} \setminus \{x'\})$. Let $h \in B_i^x \setminus \{i\}$ such that $\{h, y'\} \in E$. Then

$$a(x') < a(y') < b(h) < b(x'), b(y') \Rightarrow \{h, x'\} \in E.$$

Consequently, $\langle h, x', y' \rangle$ is an induced triangle of G, a contradiction, so $\{h, y'\} \notin E$ for all $h \in B_i^x \setminus \{i\}$. By this fact, (i) and Observation 2.3.1(2), it follows that (ii) $B_i^x \setminus \{i\} \subseteq V_{\ll y'}$. The neighbourhood of y' in $G[V_{\ll y'} \cup \{x', y'\}]$ is $\{x'\}$. Consequently, no subset of $V_{\ll y'} \cup \{x', y'\}$ that contains y' induces a cycle of G. By this fact, (ii) and Definition 2.3.4, it follows that

$$\left.\begin{array}{l} B_i^x \setminus \{i\} \in \mathcal{B}_{\ll y'}^{x'} \\ B_{\ll y'}^{x'} \cup \{i\} \in \mathcal{B}_i^x \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B_i^x \setminus \{i\}) \leq w(B_{\ll y'}^{x'}) \\ w(B_{\ll y'}^{x'} \cup \{i\}) \leq w(B_i^x) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(B_i^x) = w(B_{\ll y'}^{x'} \cup \{i\}) \Rightarrow B_i^x = B_{\ll y'}^{x'} \cup \{i\}. \qquad \square$$

Theorem 2.3.4. *The dynamic programming algorithm shown in Figure 2.3 computes a feedback vertex set of G that has minimum weight in $O(n + m)$ time.*

compute a list $\mathcal{L}$ containing all vertices of V
    sorted in descending order with respect to $<_l$ and
    a list $\mathcal{R}$ containing all vertices of V
    sorted in ascending order with respect to $<_r$;

$A_0 := \varnothing$;
$B_0^1 := \varnothing$;

for i **in** $\mathcal{R}$ **starting from** $i := 1$
  compute $< i$ and $\ll i$ of Definition 2.3.1;
  compute A_i according to Lemma 2.3.2;
  for x **in reverse** $\mathcal{L}$ **starting from** $x := i$
    if $x = i$ **then continue**;
    compute B_i^x according to Lemma 2.3.3;
    if $\{i, x\} \notin E$ **then break**;
  end for
end for

return $V \setminus A_n$;

Figure 2.3: Our dynamic programming algorithm for solving FVS on an interval graph.

Proof. The correctness of the algorithm follows from Lemmas 2.3.2 and 2.3.3. The computation of a single predecessor, A-set or B-set takes constant time. The number of iterations performed is

$$\mathrm{O}\left(\sum_{i \in V}\left(1 + \sum_{x \in N(i)} 1\right)\right) = \mathrm{O}(n+m).$$

Therefore, the total time complexity of the algorithm is $\mathrm{O}(n+m)$. □

2.4 Our New Algorithm on Permutation Graphs

In this section we propose a novel $\mathrm{O}(nm)$ dynamic programming algorithm for solving FVS on permutation graphs as a precursor to the dynamic programming algorithm of Section 3.2 for solving SFVS on permutation graphs.

We assume that we are given a connected permutation graph $G = (V, E)$ along with a corresponding permutation π of G and a weight function $w : V \to \mathbb{R}$ as input. As any vertex that has non-positive weight would be automatically included in a feedback vertex set that has minimum weight, we may also assume that all vertices have positive weight. We extend w such that if it is called with a set of vertices, then it returns the sum of their respective weights. Wherever $\max_w$ is called in this section, it returns an arbitrary choice that has maximum weight among all its operands. We add an isolated dummy vertex that has non-positive weight to G and we add $0 \mapsto 0$ to π as its first column where 0 is its corresponding integer. We consider the vertices of G to be equivalent to their corresponding integers in π's domain.

We consider the two relations on V defined by

$$\begin{aligned} i \leq_t j &\iff i \leq j \\ i \leq_b j &\iff \pi^{-1}(i) \leq \pi^{-1}(j) \end{aligned}$$

for all $i, j \in V$. It is not difficult to show that both $\leq_t$ and $\leq_b$ are total orders on V; they are exactly the orders in which the integers appear on the **top** and **bottom** row of π respectively.

Our dynamic programming algorithm iterates on ordered crossing vertex pairs. We define the following collections of ordered crossing vertex pairs:

$$\mathcal{X} = \{ij \in V^2 : i \leq_t j \text{ and } j \leq_b i\}$$

$$\mathcal{I} = \{ij \in V^2 : i = j\} \subset \mathcal{X}$$

Observe that for every $ij \in \mathcal{X}$, if $i \neq j$, then $\{i, j\} \in E$. We consider the two relations on $\mathcal{X}$ defined by

$$\begin{array}{lcl} gh \leq_r ij & \iff & g \leq_b i \text{ and } h \leq_t j \\ gh \leq_r^{lex} ij & \iff & g <_b i \text{ or } g = i \text{ and } h \leq_t j \end{array}$$

for all $gh, ij \in \mathcal{X}$ as well as the two relations on $\mathcal{X}$ defined by

$$\begin{array}{lcl} xy \leq_l zw & \iff & x \leq_t z \text{ and } y \leq_b w \\ xy \leq_l^{lex} zw & \iff & x <_t z \text{ or } x = z \text{ and } y \leq_b w \end{array}$$

for all $xy, zw \in \mathcal{X}$. It is not difficult to show that $\leq_l$ and $\leq_r$ are partial orders on $\mathcal{X}$, which we call the **left-endpoints** and **right-endpoints** *product* order respectively, whereas $\leq_l^{lex}$ and $\leq_r^{lex}$ are total orders on $\mathcal{X}$, which we call the **left-endpoints** and **right-endpoints** *lexicographic* order respectively. We also note that $\leq_l^{lex}$ and $\leq_r^{lex}$ are linear extentions of $\leq_l$ and $\leq_r$ respectively. Wherever $\min_l$ is called in this section, it returns the minimum among its operands with respect to $\leq_l$; and wherever $\max_r$ is called in this section, it returns the maximum among its operands with respect to $\leq_r$. We define some predecessors with respect to $\leq_r$ and the V-sets, which correspond to the subproblems that our dynamic programming algorithm wants to solve.

Definition 2.4.1 (Predecessors)**.** Let $ij \in \mathcal{X} \setminus \{00\}$. Then

- $\leqslant ij = \max_r\{gh \in \mathcal{C} : gh <_r ij \text{ and } h \neq j\}$,
- $\leqslant ij = \max_r\{gh \in \mathcal{C} : gh <_r ij \text{ and } g \neq i\}$,
- $< ij = \max_r\{gh \in \mathcal{C} : gh <_r ij \text{ and } g \neq i \text{ and } h \neq j\}$ and
- $\ll ij = \max_r\{gh \in \mathcal{C} : gh <_r ij \text{ and } \{g, i\}, \{g, j\}, \{h, i\}, \{h, j\} \notin E\}$.

Example 2.4.1. For the permutation graph that has

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 2 & 8 & 6 & 3 & 1 & 7 & 4 & 5 \end{pmatrix}$$

as a corresponding permutation of $\{1, 2, 3, 4, 5, 6, 7, 8\}$, $\leqslant 57 = 56$, $\leqslant 57 = 47$, $< 57 = 46$, $\ll 57 = 13$ and $\leqslant 36, \leqslant 36, < 36, \ll 36 = 22$.

Definition 2.4.2 (V-sets)**.** Let $ij \in \mathcal{X}$. Then $V_{ij} = \{h \in V : hh \leq_r ij\}$.

Observation 2.4.1. *Let $ij \in \mathcal{X}$. Then*

(1) $V_{ij} = V_{\leqslant ij} \cup \{j\}$,

(2) $V_{ij} = V_{\leqslant ij} \cup \{i\}$,

(3) $V_{ij} = V_{<ij} \cup \{i, j\}$,

(4) $V_{<ij} = V_{\ll jj} \cup \{h \in V_{<ij} : \{h, j\} \in E\}$,

(5) $V_{<ij} = V_{\ll ii} \cup \{h \in V_{<ij} : \{h, i\} \in E\}$,

(6) $V_{\ll ii} = V_{\ll ij} \cup \{h \in V_{\ll ii} : \{h, j\} \in E\}$ *and*

(7) $V_{\ll jj} = V_{\ll ij} \cup \{h \in V_{\ll jj} : \{h, i\} \in E\}$.

Now, we define the sets that our dynamic programming algorithm computes in order to compute the forest-inducing vertex set of G that has maximum weight.

Definition 2.4.3 (A-sets)**.** Let $ij \in \mathcal{X}$. Then,

$$A_{ij} = \max_w \mathcal{A}_{ij} = \max_w \{X \subseteq V_{ij} : G[X] \in \mathcal{F}\}.$$

Definition 2.4.4 (B-sets)**.** Let $ij \in \mathcal{X}$ and $x \in V \setminus V_{ij}$. Then,

$$B^{xx}_{ij} = \max_w \mathcal{B}^{xx}_{ij} = \max_w \{X \subseteq V_{ij} : G[X \cup \{x\}] \in \mathcal{F}\}.$$

Observe that, since $V_{00} = \{0\}$ and $w(0) \leq 0$, $A_{00} = \varnothing$ and, since $V_{\pi(n)n} = V$, $A_{\pi(n)n} = \max_w\{X \subseteq V : G[X] \in \mathcal{F}\}$. The following lemmas state how to recursively compute all A-sets and B-sets other than A_{00}.

Lemma 2.4.2. *Let $i \in V \setminus \{0\}$. Then $A_{ii} = A_{<ii} \cup \{i\}$.*

Proof. Let $i \in V \setminus \{0\}$. Then the neighbourhood of i in $G[V_{ii}]$ is $\varnothing$. Consequently, no subset of V_{ii} that contains i induces a cycle of G, so $i \in A_{ii}$. By these facts, Observation 2.4.1(3) and Definition 2.4.3, it follows that

$$\left.\begin{array}{l} A_{ii} \setminus \{i\} \in \mathcal{A}_{<ii} \\ A_{<ii} \cup \{i\} \in \mathcal{A}_{ii} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(A_{ii} \setminus \{i\}) \leq w(A_{<ii}) \\ w(A_{<ii} \cup \{i\}) \leq w(A_{ii}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(A_{ii}) = w(A_{<ii} \cup \{i\}) \Rightarrow A_{ii} = A_{<ii} \cup \{i\}. \qquad \square$$

Lemma 2.4.3. *Let $i \in V$ and $x \in V \setminus V_{ii}$.*

(1) If $\{i, x\} \notin E$, then $B^{xx}_{ii} = A_{ii}$.

(2) If $\{i,x\} \in E$, *then* $B^{xx}_{ii} = B^{xx}_{<ii} \cup \{i\}$.

Proof. Let $i \in V$ and $x \in V \setminus V_{ii}$.

(1) Let $\{i,x\} \notin E$. Then $i <_t x$ and $i <_b x$, so the neighbourhood of x in $G[V_{ii} \cup \{x\}]$ is $\varnothing$. Consequently, no subset of $V_{ii} \cup \{x\}$ that contains x induces a cycle of G. By this fact and Definitions 2.4.3 and 2.4.4, it follows that

$$\left.\begin{array}{l} B^{xx}_{ii} \in \mathcal{A}_{ii} \\ A_{ii} \in \mathcal{B}^{xx}_{ii} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B^{xx}_{ii}) \le w(A_{ii}) \\ w(A_{ii}) \le w(B^{xx}_{ii}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(B^{xx}_{ii}) = w(A_{ii}) \Rightarrow B^{xx}_{ii} = A_{ii}.$$

(2) Let $\{i,x\} \in E$. Then the neighbourhood of i in $G[V_{ii} \cup \{x\}]$ is $\{x\}$. Consequently, no subset of $V_{ii} \cup \{x\}$ that contains i induces a cycle of G, so $i \in B^{xx}_{ii}$. By these facts, Observation 2.4.1(3) and Definition 2.4.4, it follows that

$$\left.\begin{array}{l} B^{xx}_{ii} \setminus \{i\} \in \mathcal{B}^{xx}_{<ii} \\ B^{xx}_{<ii} \cup \{i\} \in \mathcal{B}^{xx}_{ii} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B^{xx}_{ii} \setminus \{i\}) \le w(B^{xx}_{<ii}) \\ w(B^{xx}_{<ii} \cup \{i\}) \le w(B^{xx}_{ii}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(B^{xx}_{ii}) = w(B^{xx}_{<ii} \cup \{i\}) \Rightarrow B^{xx}_{ii} = B^{xx}_{<ii} \cup \{i\}. \qquad \square$$

Lemma 2.4.4. *Let* $ij \in \mathcal{X} \setminus \mathcal{I}$. *Then*

$$A_{ij} = \max_w \left\{ A_{\leqslant ij}, A_{\leqslant ij}, A^{ii}_{\ll jj} \cup \{i,j\}, A^{jj}_{\ll ii} \cup \{i,j\} \right\}.$$

Proof. Let $ij \in \mathcal{X} \setminus \mathcal{I}$.

Let $j \notin A_{ij}$. By this fact, Observation 2.4.1(1) and Definition 2.4.3, it follows that

$$\left.\begin{array}{l} A_{ij} \in \mathcal{A}_{\leqslant ij} \\ A_{\leqslant ij} \in \mathcal{A}_{ij} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(A_{ij}) \le w(A_{\leqslant ij}) \\ w(A_{\leqslant ij}) \le w(A_{ij}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(A_{ij}) = w(A_{\leqslant ij}) \Rightarrow A_{ij} = A_{\leqslant ij}.$$

Let $i \notin A_{ij}$. Then one can as above show that $A_{ij} = A_{\leqslant ij}$.

Let $i,j \in A_{ij}$. By this fact and Observation 2.4.1(3), it follows that (i) $A_{ij} \setminus \{i,j\} \subseteq V_{<ij}$. Let $h \in A_{ij} \setminus \{i,j\}$ such that $\{h,i\}, \{h,j\} \in E$. Then $\langle h,i,j \rangle$ is an induced triangle of G, a contradiction, so either $\{h,i\} \notin E$ or $\{h,j\} \notin E$ for all $h \in A_{ij} \setminus \{i,j\}$. Let $g,h \in A_{ij} \setminus \{i,j\}$ such that $\{g,j\}, \{h,i\} \in E$. Then

$$\left.\begin{array}{l} g <_t i <_t h \\ h <_b j <_b g \end{array}\right\} \Rightarrow \{g,h\} \in E.$$

Consequently, $\langle g, h, i, j\rangle$ is an induced square of G, a contradiction, so either $\{h, i\} \notin E$ for all $h \in A_{ij} \setminus \{i, j\}$ or $\{h, j\} \notin E$ for all $h \in A_{ij} \setminus \{i, j\}$. By this fact, (i) and Observations 2.4.1(4)–(5), it follows that either (ii) $A_{ij} \setminus \{i, j\} \subseteq V_{\ll jj}$ or (iii) $A_{ij} \setminus \{i, j\} \subseteq V_{\ll ii}$. Assume that (ii) holds. The neighbourhood of j in $G[V_{\ll jj} \cup \{i, j\}]$ is $\{i\}$. Consequently, no subset of $V_{\ll jj} \cup \{i, j\}$ that contains j induces a cycle of G. By this fact, (ii) and Definitions 2.4.3 and 2.4.4, it follows that

$$\left.\begin{array}{l} A_{ij} \setminus \{i, j\} \in \mathcal{B}^{ii}_{\ll jj} \\ B^{ii}_{\ll jj} \cup \{i, j\} \in \mathcal{A}_{ij} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(A_{ij} \setminus \{i, j\}) \leq w(B^{ii}_{\ll jj}) \\ w(B^{ii}_{\ll jj} \cup \{i, j\}) \leq w(A_{ij}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(A_{ij}) = w(B^{ii}_{\ll jj} \cup \{i, j\}) \Rightarrow A_{ij} = B^{ii}_{\ll jj} \cup \{i, j\}.$$

Assuming that (iii) holds, one can as above show that $A_{ij} = B^{jj}_{\ll ii} \cup \{i, j\}$. □

Lemma 2.4.5. *Let $ij \in \mathcal{X} \setminus \mathcal{I}$ and $x \in V \setminus V_{ij}$.*

(1) If $\{i, x\}, \{j, x\} \notin E$, then $B^{xx}_{ij} = A_{ij}$.

(2) If $\{i, x\} \in E$ and $\{j, x\} \notin E$, then

$$B^{xx}_{ij} = \max_w \left\{ B^{xx}_{\leqslant ij}, B^{xx}_{\leqslant ij}, B^{ii}_{\ll jj} \cup \{i, j\}, B^{jj}_{\ll ix} \cup \{i, j\} \right\}.$$

(3) If $\{i, x\} \notin E$ and $\{j, x\} \in E$, then

$$B^{xx}_{ij} = \max_w \left\{ B^{xx}_{\leqslant ij}, B^{xx}_{\leqslant ij}, B^{ii}_{\ll xj} \cup \{i, j\}, B^{jj}_{\ll ii} \cup \{i, j\} \right\}.$$

(4) If $\{i, x\}, \{j, x\} \in E$, then $B^{xx}_{ij} = \max_w \left\{ B^{xx}_{\leqslant ij}, B^{xx}_{\leqslant ij} \right\}$.

Proof. Let $ij \in \mathcal{X} \setminus \mathcal{I}$ and $x \in V \setminus V_{ij}$.

(1) Let $\{i, x\}, \{j, x\} \notin E$. Then $i <_t j <_t x$ and $j <_b i <_b x$, so the neighbourhood of x in $G[V_{ij} \cup \{x\}]$ is $\varnothing$. Consequently, no subset of $V_{ij} \cup \{x\}$ that contains x induces a cycle of G. By this fact and Definitions 2.4.3 and 2.4.4, it follows that

$$\left.\begin{array}{l} B^{xx}_{ij} \in \mathcal{A}_{ij} \\ A_{ij} \in \mathcal{B}^{xx}_{ij} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B^{xx}_{ij}) \leq w(A_{ij}) \\ w(A_{ij}) \leq w(B^{xx}_{ij}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(B^{xx}_{ij}) = w(A_{ij}) \Rightarrow B^{xx}_{ij} = A_{ij}.$$

(2)–(4) Let $\{i,x\} \in E$ or $\{j,x\} \in E$.

Let $j \notin B^{xx}_{ij}$. By this fact, Observation 2.4.1(1) and Definition 2.4.4, it follows that

$$\left.\begin{array}{l} B^{xx}_{ij} \in \mathcal{B}^{xx}_{\leqslant ij} \\ B^{xx}_{\leqslant ij} \in \mathcal{B}^{xx}_{ij} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B^{xx}_{ij}) \leq w(B^{xx}_{\leqslant ij}) \\ w(B^{xx}_{\leqslant ij}) \leq w(B^{xx}_{ij}) \end{array}\right\} \Rightarrow$$
$$\Rightarrow w(B^{xx}_{ij}) = w(B^{xx}_{\leqslant ij}) \Rightarrow B^{xx}_{ij} = B^{xx}_{\leqslant ij}.$$

Let $i \notin B^{xx}_{ij}$. Then one can as above show that $B^{xx}_{ij} = B^{xx}_{\leqslant ij}$.

Let $i,j \in B^{xx}_{ij}$. By this fact and Observation 2.4.1(3), it follows that (i) $B^{xx}_{ij} \setminus \{i,j\} \subseteq V_{<ij}$.

(2) Let $\{i,x\} \in E$ and $\{j,x\} \notin E$. Let $h \in B^{xx}_{ij} \setminus \{i,j\}$ such that $\{h,i\},\{h,j\} \in E$. Then $\langle h,i,j\rangle$ is an induced triangle of G, a contradiction, so either $\{h,i\} \notin E$ or $\{h,j\} \notin E$ for all $h \in B^{xx}_{ij} \setminus \{i,j\}$. Let $g,h \in B^{xx}_{ij} \setminus \{i,j\}$ such that $\{g,j\},\{h,i\} \in E$. Then

$$\left.\begin{array}{l} g <_t i <_t h \\ h <_b j <_b g \end{array}\right\} \Rightarrow \{g,h\} \in E.$$

Consequently, $\langle g,h,i,j\rangle$ is an induced square of G, a contradiction, so either $\{h,i\} \notin E$ for all $h \in B^{xx}_{ij} \setminus \{i,j\}$ or $\{h,j\} \notin E$ for all $h \in B^{xx}_{ij} \setminus \{i,j\}$. By this fact, (i) and Observations 2.4.1(4)–(5), it follows that either (ii) $B^{xx}_{ij} \setminus \{i,j\} \subseteq V_{\ll jj}$ or (iii) $B^{xx}_{ij} \setminus \{i,j\} \subseteq V_{\ll ii}$. Assume that (ii) holds. The neighbourhoods of j and x in $G[V_{\ll jj} \cup \{i,j,x\}]$ are $\{i\}$. Consequently, no subset of $V_{\ll jj} \cup \{i,j,x\}$ that contains j and/or x induces a cycle of G. By this fact, (ii) and Definition 2.4.4, it follows that

$$\left.\begin{array}{l} B^{xx}_{ij} \setminus \{i,j\} \in \mathcal{B}^{ii}_{\ll jj} \\ B^{ii}_{\ll jj} \cup \{i,j\} \in \mathcal{B}^{xx}_{ij} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B^{xx}_{ij} \setminus \{i,j\}) \leq w(B^{ii}_{\ll jj}) \\ w(B^{ii}_{\ll jj} \cup \{i,j\}) \leq w(B^{xx}_{ij}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(B^{xx}_{ij}) = w(B^{ii}_{\ll jj} \cup \{i,j\}) \Rightarrow B^{xx}_{ij} = B^{ii}_{\ll jj} \cup \{i,j\}$$

Now, assume that (iii) holds. Let $h \in B^{xx}_{ij} \setminus \{i,j\}$ such that $\{h,x\} \in E$. Then

$$\left.\begin{array}{l} h <_t i <_t j <_t x \\ j <_b x <_b h <_b i \end{array}\right\} \Rightarrow \{h,j\} \in E.$$

Consequently, $\langle h,j,i,x\rangle$ is an induced square of G, a contradiction, so $\{h,x\} \notin E$ for all $h \in B^{xx}_{ij} \setminus \{i,j\}$. By this fact, (iii) and Observation 2.4.1(6), it follows that (iv) $B^{xx}_{ij} \setminus \{i,j\} \subseteq V_{\ll ix}$. The neighbourhoods of i and x in $G[V_{\ll ix} \cup \{i,j,x\}]$ are $\{j\}$. Consequently, no subset of $V_{\ll ix} \cup \{i,j,x\}$ that

compute a list $\mathcal{R}$ containing all ordered crossing vertex pairs of $\mathcal{X}$
    sorted in ascending order with respect to $<_r^{lex}$
    and all predecessors of Definition 2.4.1 according to [26];

for ij **in** $\mathcal{R}$
  if $ij = 00$
    $A_{00} := \varnothing$;
  else
    compute A_{ij} according to Lemmas 2.4.2 and 2.4.4;
  end if
  for $x := 1$ **to** n
    if $x \in V_{ij}$ **then continue**;
    compute B_{ij}^{xx} according to Lemmas 2.4.3 and 2.4.5;
  end for
end for

return $V \setminus A_{\pi(n)n}$;

Figure 2.4: Our dynamic programming algorithm for solving FVS on a permutation graph.

contains i and/or x induces a cycle of G. By this fact, (iv) and Definition 2.4.4, it follows that

$$\left.\begin{array}{l} B_{ij}^{xx} \setminus \{i,j\} \in \mathcal{B}_{\ll ix}^{jj} \\ B_{\ll ix}^{jj} \cup \{i,j\} \in \mathcal{B}_{ij}^{xx} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B_{ij}^{xx} \setminus \{i,j\}) \leq w(B_{\ll ix}^{jj}) \\ w(B_{\ll ix}^{jj} \cup \{i,j\}) \leq w(B_{ij}^{xx}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(B_{ij}^{xx}) = w(B_{\ll ix}^{jj} \cup \{i,j\}) \Rightarrow B_{ij}^{xx} = B_{\ll ix}^{jj} \cup \{i,j\}$$

(3) Let $\{i,x\} \notin E$ and $\{j,x\} \in E$. Then one can as in (2) show that either $B_{ij}^{xx} = B_{\ll xj}^{ii} \cup \{i,j\}$ or $B_{ij}^{xx} = B_{\ll ii}^{jj} \cup \{i,j\}$.

(4) Let $\{i,x\},\{j,x\} \in E$. Then $\langle i,j,x\rangle$ is an induced triangle of G, a contradiction. □

Theorem 2.4.6. *The dynamic programming algorithm shown in Figure 2.4 returns a feedback vertex set of G that has minimum weight in $O(nm)$ time.*

Proof. The correctness of the algorithm follows from Lemmas 2.4.2–2.4.5. The computation of $\mathcal{R}$ and all predecessors of Definition 2.4.1 takes $O(nm)$ time

according to [26]. The computation of a single A-set or B-set takes constant time. The number of iterations performed is

$$\mathrm{O}\left(\sum_{ij\in\mathcal{X}}\left(1+\sum_{x\in V}1\right)\right)=\mathrm{O}(nm).$$

Therefore, the total time complexity of the algorithm is $\mathrm{O}(nm)$. □

Chapter 3

Solving SFVS in Polynomial Time

As we have already mentioned, there is no known polynomial result where the input is restricted to a graph class regarding SFVS. In this chapter, we propose our novel dynamic programming algorithms for solving SFVS on interval graphs and permutation graphs in polynomial time. These algorithms do not find an S-feedback vertex set of the input graph that has minimum weight directly and return it; they find an S-forest-inducing vertex set of the input graph that has maximum weight and return the set of all other vertices instead. Throughout this chapter, we use $\mathcal{F}_S$ to denote the collection of all induced S-forests of the input graph.

3.1 Our Algorithm on Interval Graphs

In this section we propose a novel $\mathrm{O}(n + m + l)$ dynamic programming algorithm for solving SFVS on interval graphs where $l \in \mathrm{O}(n^3)$ is the number of induced triangles of the input graph.

We assume that we are given an interval graph $G = (V, E)$ along with a corresponding collection of closed intervals of $\mathbb{R}$ where all endpoints are distinct, a set $S \subseteq V$ and a weight function $w : V \to \mathbb{R}$ as input. We extend w such that if it is called with a set of vertices, then it returns the sum of their respective weights. Wherever $\max_w$ is called in this section, it returns an arbitrary choice that has maximum weight among all its operands. We add

an isolated dummy vertex that has non-positive weight to G and we add a dummy interval that has minimum right endpoint to the given collection as its corresponding interval. We consider vertices of G to be equivalent to their corresponding intervals and to the integers of $\{0, 1, \ldots, n\}$ that indicate their position when sorted in ascending order of their corresponding intervals' right endpoints.

For every vertex i of G, we use $a(i)$ and $b(i)$ to denote its corresponding interval's left and right endpoint respectively. We consider the two relations on V that are defined by

$$\begin{array}{rcl} i \leq_l j & \iff & a(i) \leq a(j) \\ i \leq_r j & \iff & b(i) \leq b(j) \end{array}$$

for all $i, j \in V$. Since all endpoints of the collection's intervals are distinct, it is not difficult to show that $\leq_l$ and $\leq_r$ are total orders on V. Wherever $\min_l$ is called in this section, it returns the minimum among its operands with respect to $\leq_l$; and wherever $\max_r$ is called in this section, it returns the maximum among its operands with respect to $\leq_r$. We define some predecessors with respect to $\leq_r$ and the V-sets, which correspond to the subproblems that our dynamic programming algorithm wants to solve.

Definition 3.1.1 (Predecessors)**.** Let $i \in V \setminus \{0\}$. Then

- $< i = \max_r\{h \in V : h <_r i\}$ and
- $\ll i = \max_r\{h \in V : h <_r i \text{ and } \{h, i\} \notin E\}$.

Example 3.1.1. For the interval graph that has

$$\mathcal{C} = \left\{ \begin{array}{c} I_1 = [2, 4], I_2 = [1, 6], I_3 = [7, 8], I_4 = [5, 10], \\ I_5 = [3, 11], I_6 = [9, 13], I_7 = [12, 14] \end{array} \right\}$$

as a corresponding collection of closed intervals of $\mathbb{R}$, $< 6 = 5$ and $\ll 6 = 3$.

Definition 3.1.2 (V-sets)**.** Let $i \in V$. Then $V_i = \{h \in V : h \leq_r i\}$.

Observation 3.1.1. *Let $i \in V \setminus \{0\}$ and $j \in V \setminus V_i$ such that $\{i, j\} \in E$. Then*

(1) $V_i = V_{<i} \cup \{i\}$ and

(2) $V_{<i} = V_{\ll j} \cup \{h \in V_{<i} : \{h, j\} \in E\}$.

Now, we define the sets that our dynamic programming algorithm computes in order to conpute the S-forest-inducing vertex set of G that has maximum weight.

Definition 3.1.3 (A-sets)**.** Let $i \in V$. Then,

$$A_i = \max_w \mathcal{A}_i = \max_w \{X \subseteq V_i : G[X] \in \mathcal{F}_S\}.$$

Definition 3.1.4 (B-sets)**.** Let $i \in V$ and $x \in V \setminus V_i$. Then,

$$B_i^x = \max_w \mathcal{B}_i^x = \max w\{X \subseteq V_i : G[X \cup \{x\}] \in \mathcal{F}_S\}.$$

Definition 3.1.5 (C-sets)**.** Let $i \in V$ and $x, y \in V \setminus (V_i \cup S)$ such that $x <_l y$ and $\{x, y\} \in E$. Then,

$$C_i^{x,y} = \max_w \mathcal{C}_i^{x,y} = \max_w \{X \subseteq V_i : G[X \cup \{x, y\}] \in \mathcal{F}_S\}.$$

Observe that, since $V_0 = \{0\}$ and $w(0) \leq 0$, $A_0 = \varnothing$ and, since $V_n = V$, $A_n = \max_w\{X \subseteq V : G[X] \in \mathcal{F}_S\}$. The following lemmas state how to recursively compute all A-sets, B-sets and C-sets other than A_0.

Lemma 3.1.2 (A-sets)**.** *Let $i \in V \setminus \{0\}$. Then $A_i = \max_w \left\{A_{<i}, B^i_{<i} \cup \{i\}\right\}$.*

Proof. Let $i \in V \setminus \{0\}$.

Let $i \notin A_i$. By this fact, Observation 3.1.1(1) and Definition 3.1.3, it follows that

$$\left.\begin{array}{l} A_i \in \mathcal{A}_{<i} \\ A_{<i} \in \mathcal{A}_i \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(A_i) \leq w(A_{<i}) \\ w(A_{<i}) \leq w(A_i) \end{array}\right\} \Rightarrow$$
$$\Rightarrow w(A_i) = w(A_{<i}) \Rightarrow A_i = A_{<i}.$$

Let $i \in A_i$. By this fact, Observation 3.1.1(1) and Definitions 3.1.3 and 3.1.4, it follows that

$$\left.\begin{array}{l} A_i \setminus \{i\} \in \mathcal{B}^i_{<i} \\ B^i_{<i} \cup \{i\} \in \mathcal{A}_i \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(A_i \setminus \{i\}) \leq w(B^i_{<i}) \\ w(B^i_{<i} \cup \{i\}) \leq w(A_i) \end{array}\right\} \Rightarrow$$
$$\Rightarrow w(A_i) = w(B^i_{<i} \cup \{i\}) \Rightarrow A_i = B^i_{<i} \cup \{i\}. \qquad \square$$

Lemma 3.1.3 (B-sets)**.** *Let $i \in V$ and $x \in V \setminus V_i$.*

(1) If $\{i, x\} \notin E$, then $B_i^x = A_i$.

(2) If $\{i,x\} \in E$*, then*

$$B_i^x = \begin{cases} \max_w \left\{ B_{<i}^x, B_{\ll y'}^{x'} \cup \{i\} \right\}, \textit{if } i \in S \textit{ or } x \in S \\ \max_w \left\{ B_{<i}^x, C_{<i}^{x',y'} \cup \{i\} \right\}, \textit{if } i,x \notin S \end{cases}$$

where $x' = \min_a\{i,x\}$ *and* $y' = \min_a(\{i,x\} \setminus \{x'\})$.

Proof. Let $i \in V$ and $x \in V \setminus V_i$.

(1) Let $\{i,x\} \notin E$. Then $b(i) < a(x)$, so the neighbourhood of x in $G[V_i \cup \{x\}]$ is $\varnothing$. Consequently, no subset of $V_i \cup \{x\}$ that contains x induces an S-cycle of G. By this fact and Definitions 3.1.3 and 3.1.4, it follows that

$$\left.\begin{matrix} B_i^x \in \mathcal{A}_i \\ A_i \in \mathcal{B}_i^x \end{matrix}\right\} \Rightarrow \left.\begin{matrix} w(B_i^x) \leq w(A_i) \\ w(A_i) \leq w(B_i^x) \end{matrix}\right\} \Rightarrow$$

$$\Rightarrow w(B_i^x) = w(A_i) \Rightarrow B_i^x = A_i.$$

(2) Let $\{i,x\} \in E$.

Let $i \notin B_i^x$. By this fact, Observation 3.1.1(1) and Definition 3.1.4, it follows that

$$\left.\begin{matrix} B_i^x \in \mathcal{B}_{<i}^x \\ B_{<i}^x \in \mathcal{B}_i^x \end{matrix}\right\} \Rightarrow \left.\begin{matrix} w(B_i^x) \leq w(B_{<i}^x) \\ w(B_{<i}^x) \leq w(B_i^x) \end{matrix}\right\} \Rightarrow$$

$$\Rightarrow w(B_i^x) = w(B_{<i}^x) \Rightarrow B_i^x = B_{<i}^x.$$

Let $i \in B_i^x$. By this fact and Observation 3.1.1(1), it follows that (i) $B_i^x \setminus \{i\} \subseteq V_{<i}$. We define $x' = \min_l\{i,x\}$ and $y' = \min_l(\{i,x\} \setminus \{x'\})$.

Case 1: Let $i \in S$ or $x \in S$. Let $h \in B_i^x \setminus \{i\}$ such that $\{h,y'\} \in E$. Then

$$a(x') < a(y') < b(h) < b(x'), b(y') \Rightarrow \{h,x'\} \in E.$$

Consequently, $\langle h,x',y' \rangle$ is an induced S-triangle of G, a contradiction, so $\{h,y'\} \notin E$ for all $h \in B_i^x \setminus \{i\}$. By this fact, (i) and Observation 3.1.1(2), it follows that (ii) $B_i^x \setminus \{i\} \subseteq V_{\ll y'}$. The neighbourhood of y' in $G[V_{\ll y'} \cup \{x',y'\}]$ is $\{x'\}$. Consequently, no subset of $V_{\ll y'} \cup \{x',y'\}$ that contains y' induces an S-cycle of G. By this fact, (ii) and Definition 3.1.4, it follows that

$$\left.\begin{matrix} B_i^x \setminus \{i\} \in \mathcal{B}_{\ll y'}^{x'} \\ B_{\ll y'}^{x'} \cup \{i\} \in \mathcal{B}_i^x \end{matrix}\right\} \Rightarrow \left.\begin{matrix} w(B_i^x \setminus \{i\}) \leq w(B_{\ll y'}^{x'}) \\ w(B_{\ll y'}^{x'} \cup \{i\}) \leq w(B_i^x) \end{matrix}\right\} \Rightarrow$$

$$\Rightarrow w(B_i^x) = w(B^{x'}_{\ll y'} \cup \{i\}) \Rightarrow B_i^x = B^{x'}_{\ll y'} \cup \{i\}.$$

Case 2: Let $i, x \notin S$. By (i) and Definitions 3.1.4 and 3.1.5, it follows that

$$\left.\begin{array}{l} B_i^x \setminus \{i\} \in \mathcal{C}_{<i}^{x',y'} \\ C_{<i}^{x',y'} \cup \{i\} \in \mathcal{B}_i^x \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B_i^x \setminus \{i\}) \leq w(C_{<i}^{x',y'}) \\ w(C_{<i}^{x',y'} \cup \{i\}) \leq w(B_i^x) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(B_i^x) = w(C_{<i}^{x',y'} \cup \{i\}) \Rightarrow B_i^x = C_{<i}^{x',y'} \cup \{i\}. \qquad \square$$

Lemma 3.1.4 (C-sets)**.** *Let $i \in V$ and $x, y \in V \setminus (V_i \cup S)$ such that $x <_a y$ and $\{x, y\} \in E$.*

(1) If $\{i, y\} \notin E$, then $C_i^{x,y} = B_i^x$.

(2) If $\{i, y\} \in E$, then $C_i^{x,y} = \begin{cases} C_{<i}^{x,y} & , \text{if } i \in S \\ \max_w \left\{C_{<i}^{x,y}, C_{<i}^{x',y'} \cup \{i\}\right\} & , \text{if } i \notin S \end{cases}$
where $x' = \min_l\{i, x, y\}$ and $y' = \min_l(\{i, x, y\} \setminus \{x'\})$.

Proof. Let $i \in V$ and $x, y \in V \setminus (V_i \cup S)$ such that $x <_a y$ and $\{x, y\} \in E$.

(1) Let $\{i, y\} \notin E$. Then $b(i), a(x) < a(y) < b(x)$, so the neighbourhood of y in $G[V_i \cup \{x, y\}]$ is $\{x\}$. Consequently, no subset of $V_i \cup \{x, y\}$ that contains y induces an S-cycle of G. By this fact and Definitions 3.1.4 and 3.1.5, it follows that

$$\left.\begin{array}{l} C_i^{x,y} \in \mathcal{B}_i^x \\ B_i^x \in \mathcal{C}_i^{x,y} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(C_i^{x,y}) \leq w(B_i^x) \\ w(B_i^x) \leq w(C_i^{x,y}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(C_i^{x,y}) = w(B_i^x) \Rightarrow C_i^{x,y} = B_i^x.$$

(2) Let $\{i, y\} \in E$. Then $a(x) < a(y) < b(i) < b(x), b(y)$, so $\langle i, x, y\rangle$ is an induced triangle of G.

Let $i \notin C_i^{x,y}$. By this fact, Observation 3.1.1(1) and Definition 3.1.5, it follows that

$$\left.\begin{array}{l} C_i^{x,y} \in \mathcal{C}_{<i}^{x,y} \\ C_{<i}^{x,y} \in \mathcal{C}_i^{x,y} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(C_i^{x,y}) \leq w(C_{<i}^{x,y}) \\ w(C_{<i}^{x,y}) \leq w(C_i^{x,y}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(C_i^{x,y}) = w(C_{<i}^{x,y}) \Rightarrow C_i^{x,y} = C_{<i}^{x,y}.$$

Let $i \in C_i^{x,y}$. By this fact and Observation 3.1.1(1), it follows that (i) $C_i^{x,y} \setminus \{i\} \subseteq V_{<i}$. We define $x' = \min_l\{i, x, y\}$, $y' = \min_l(\{i, x, y\} \setminus \{x'\})$ and $z' = \min_l(\{i, x, y\} \setminus \{x', y'\})$.

Case 1: Let $i \in S$. Then $\langle i, x, y\rangle$ is an induced S-triangle of G, a contradiction.

Case 2: Let $i \notin S$. We will show that if a subset of $V_{<i} \cup \{x', y', z'\}$ that contains z' induces an S-cycle of G, then its non-empty intersection with $V_{<i}$ is not a subset of $C_i^{x',y'}$.

- Let $v_1, v_2 \in V_{<i} \cup \{x', y'\}$ such that $\langle v_1, v_2, z'\rangle$ is an induced S-triangle of G. Since $x', y', z' \notin S$, without loss of generality, assume that $v_1 \in S$ $\Rightarrow v_1 \in V_{<i}$. Then

  $$a(x') < a(y') < a(z') < b(v_1) < b(x'), b(y'), b(z') \Rightarrow \{v_1, x'\}, \{v_1, y'\} \in E.$$

  Consequently, $\langle v_1, x', y'\rangle$ is an induced S-triangle of G, so $v_1 \notin C_i^{x',y'}$.

Therefore, if a subset of $V_{<i} \cup \{x', y', z'\}$ that contains z' induces an S-cycle of G, then its non-empty intersection with $V_{<i}$ is not a subset of $C_i^{x',y'}$. By this fact, (i) and Definition 3.1.5, it follows that

$$\left.\begin{array}{l} C_i^{x,y} \setminus \{i\} \in \mathcal{C}_{<i}^{x',y'} \\ C_{<i}^{x',y'} \cup \{i\} \in \mathcal{C}_i^{x,y} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(C_i^{x,y} \setminus \{i\}) \leq w(C_{<i}^{x',y'}) \\ w(C_{<i}^{x',y'} \cup \{i\}) \leq w(C_i^{x,y}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(C_i^{x,y}) = w(C_{<i}^{x',y'} \cup \{i\}) \Rightarrow C_i^{x,y} = C_{<i}^{x',y'} \cup \{i\}. \qquad \square$$

Theorem 3.1.5. *The dynamic programming algorithm shown in Figure 3.1 computes an S-feedback vertex set of G that has minimum weight in $O(n + m + l)$ time where $l \in O(n^3)$ is the number of induced triangles of G.*

Proof. The correctness of the algorithm follows from Lemmas 3.1.2–3.1.4. The computation of a single predecessor, A-set, B-set or C-set takes constant time. The number of iterations performed is

$$\mathrm{O}\left(\sum_{i \in V}\left(1 + \sum_{x \in N(i)}\left(1 + \sum_{y \in N(i) \cap N(x)} 1\right)\right)\right) = \mathrm{O}(n + m + l).$$

Therefore, the total time complexity of the algorithm is $\mathrm{O}(n + m + l)$. $\square$

```
compute a list 𝓛 containing all vertices of V
      sorted in descending order with respect to <_l and
      a list ℛ containing all vertices of V
      sorted in ascending order with respect to <_r;

A_0 := ∅;
B_0^1 := ∅;
if {1,2} ∈ E then C_0^{1,2} := ∅;

for i in ℛ starting from i := 1
  compute < i and ≪ i of Definition 3.1.1;
  compute A_i according to Lemma 3.1.2;
  for x in reverse 𝓛 starting from x := i
    if x = i then continue;
    compute B_i^x according to Lemma 3.1.3;
      for y in reverse 𝓛 starting from y := x
        if y = x then continue;
        if {x,y} ∈ E then compute C_i^{x,y} according to Lemma 3.1.4;
        if {i,y} ∉ E then break;
      end for
    if {i,x} ∉ E then break;
  end for
end for

return V \ A_n;
```

Figure 3.1: Our dynamic programming algorithm for solving SFVS on an interval graph.

3.2 Our Algorithm on Permutation Graphs

In this section we propose a novel $O(m^3)$ dynamic programming algorithm for solving SFVS on permutation graphs.

We assume that we are given a connected permutation graph $G = (V, E)$ along with a corresponding permutation π of G, a set $S \subseteq V$ and a weight function $w : V \to \mathbb{R}$ as input. As any vertex that has non-positive weight would be automatically included in an S-feedback vertex set that has minimum weight, we may also assume that all vertices have positive weight. We extend w such that if it is called with a set of vertices, then it returns the sum of their respective weights. Wherever $\max_w$ is called in this section, it returns an arbitrary choice that has maximum weight among all its operands. We add an isolated dummy vertex that has non-positive weight to G and we add $0 \mapsto 0$ to π as its first column where 0 is its corresponding integer. We consider the vertices of G to be equivalent to their corresponding integers in π's domain.

We consider the two relations on V defined by

$$\begin{array}{lcl} i \leq_t j & \iff & i \leq j \\ i \leq_b j & \iff & \pi^{-1}(i) \leq \pi^{-1}(j) \end{array}$$

for all $i, j \in V$. It is not difficult to show that both $\leq_t$ and $\leq_b$ are total orders on V; they are exactly the orders in which the integers appear on the **top** and **bottom** row of π respectively.

Our dynamic programming algorithm iterates on ordered crossing vertex pairs. We define the following collections of ordered crossing vertex pairs:

$$\mathcal{X} = \{ij \in V^2 : i \leq_t j \text{ and } j \leq_b i\}$$

$$\mathcal{I} = \{ij \in V^2 : i = j\} \subset \mathcal{X}$$

Observe that for every $ij \in \mathcal{X}$, if $i \neq j$, then $\{i, j\} \in E$. We consider the two relations on $\mathcal{X}$ defined by

$$\begin{array}{lcl} gh \leq_r ij & \iff & g \leq_b i \text{ and } h \leq_t j \\ gh \leq_r^{lex} ij & \iff & g <_b i \text{ or } g = i \text{ and } h \leq_t j \end{array}$$

for all $gh, ij \in \mathcal{X}$ as well as the two relations on $\mathcal{X}$ defined by

$$\begin{array}{lcl} xy \leq_l zw & \iff & x \leq_t z \text{ and } y \leq_b w \\ xy \leq_l^{lex} zw & \iff & x <_t z \text{ or } x = z \text{ and } y \leq_b w \end{array}$$

for all $xy, zw \in \mathcal{X}$. It is not difficult to show that $\leq_l$ and $\leq_r$ are partial orders on $\mathcal{X}$, which we call the **left-endpoints** and **right-endpoints** *product* order respectively, whereas $\leq_l^{lex}$ and $\leq_r^{lex}$ are total orders on $\mathcal{X}$, which we call the **left-endpoints** and **right-endpoints** *lexicographic* order respectively. We also note that $\leq_l^{lex}$ and $\leq_r^{lex}$ are linear extentions of $\leq_l$ and $\leq_r$ respectively. Wherever $\min_l$ is called in this section, it returns the minimum among its operands with respect to $\leq_l$; and wherever $\max_r$ is called in this section, it returns the maximum among its operands with respect to $\leq_r$. We define some predecessors with respect to $\leq_r$ and the V-sets, which correspond to the subproblems that our dynamic programming algorithm wants to solve.

Definition 3.2.1 (Predecessors)**.** Let $ij \in \mathcal{X} \setminus \{00\}$. Then

- $\leqslant ij = \max_r\{gh \in \mathcal{C} : gh <_r ij \text{ and } h \neq j\}$,
- $\leqslant ij = \max_r\{gh \in \mathcal{C} : gh <_r ij \text{ and } g \neq i\}$,
- $< ij = \max_r\{gh \in \mathcal{C} : gh <_r ij \text{ and } g \neq i \text{ and } h \neq j\}$,
- $\ll ij = \max_r\{gh \in \mathcal{C} : gh <_r ij \text{ and } \{g,i\}, \{g,j\}, \{h,i\}, \{h,j\} \notin E\}$ and
- $< ij \ll xx = \max_r\{gh \in C : gh <_r ij \text{ and } \{h,x\}, \{g,x\} \notin E\}$.

Example 3.2.1. For the permutation graph that has

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 2 & 8 & 6 & 3 & 1 & 7 & 4 & 5 \end{pmatrix}$$

as a corresponding permutation of $\{1,2,3,4,5,6,7,8\}$, $\leqslant 57 = 56$, $\leqslant 57 = 47$, $< 57 = 46$, $\ll 57 = 13$ and $\leqslant 36, \leqslant 36, < 36, \ll 36, < 36 \ll 88, < 57 \ll 88 = 22$.

Definition 3.2.2 (V-sets)**.** Let $ij \in \mathcal{X}$. Then $V_{ij} = \{h \in V : hh \leq_r ij\}$.

Observation 3.2.1. *Let $ij \in \mathcal{X}$ and $x \in V \setminus V_{ij}$. Then*

(1) $V_{ij} = V_{\leqslant ij} \cup \{j\}$,

(2) $V_{ij} = V_{\leqslant ij} \cup \{i\}$,

(3) $V_{ij} = V_{<ij} \cup \{i,j\}$,

(4) $V_{<ij} = V_{\ll jj} \cup \{h \in V_{<ij} : \{h,j\} \in E\}$,

(5) $V_{<ij} = V_{\ll ii} \cup \{h \in V_{<ij} : \{h,i\} \in E\}$,

(6) $V_{\ll ii} = V_{\ll ij} \cup \{h \in V_{\ll ii} : \{h, j\} \in E\}$,

(7) $V_{\ll jj} = V_{\ll ij} \cup \{h \in V_{\ll jj} : \{h, i\} \in E\}$ *and*

(8) $V_{<ij} = V_{<ij \ll xx} \cup \{h \in V_{<ij} : \{h, x\} \in E\}$.

Now, we define the sets that our dynamic programming algorithm computes in order to compute the S-forest-inducing vertex set of G that has maximum weight.

Definition 3.2.3 (A-sets)**.** Let $ij \in \mathcal{X}$. Then,

$$A_{ij} = \max_w \mathcal{A}_{ij} = \max_w \{X \subseteq V_{ij} : G[X] \in \mathcal{F}_S\}.$$

Definition 3.2.4 (B-sets)**.** Let $ij \in \mathcal{X}$ and $x \in V \setminus V_{ij}$. Then,

$$B_{ij}^{xx} = \max_w \mathcal{B}_{ij}^{xx} = \max_w \{X \subseteq V_{ij} : G[X \cup \{x\}] \in \mathcal{F}_S\}.$$

Definition 3.2.5 (B-sets)**.** Let $ij \in \mathcal{X}$ and $xy \in \mathcal{X} \setminus \mathcal{I}$ such that $j <_t y$, $i <_b x$ and $x, y \notin S$. Then,

$$B_{ij}^{xy} = \max_w \mathcal{B}_{ij}^{xy} = \max_w \{X \subseteq V_{ij} : G[X \cup \{x, y\}] \in \mathcal{F}_S\}.$$

Definition 3.2.6 (C-sets)**.** Let $ij \in \mathcal{X}$, $xy \in \mathcal{X} \setminus \mathcal{I}$ and $z \in V \setminus V_{ij}$ such that $xy <_l zz$, $x, y \neq z$, $\{x, z\} \in E \vee \{y, z\} \in E$, $j <_t y$, $i <_b x$ and $x, y, z \notin S$. Then,

$$C_{ij}^{xy,zz} = \max_w \mathcal{C}_{ij}^{xy,zz} = \max_w \{X \subseteq V_{ij} : G[X \cup \{x, y, z\}] \in \mathcal{F}_S\}.$$

Definition 3.2.7 (C-sets)**.** Let $ij \in \mathcal{X}$ and $xy, zw \in \mathcal{X} \setminus \mathcal{I}$ such that $xy <_l zw$, $x, y \neq z, w$, $\{x, w\}, \{y, z\} \in E$, $j <_t y, w$, $i <_b x, z$ and $x, y, z, w \notin S$. Then,

$$C_{ij}^{xy,zw} = \max_w \mathcal{C}_{ij}^{xy,zw} = \max_w \{X \subseteq V_{ij} : G[X \cup \{x, y, z, w\}] \in \mathcal{F}_S\}.$$

Observe that, since $V_{00} = \{0\}$ and $w(0) \leq 0$, $A_{00} = \varnothing$ and, since $V_{\pi(n)n} = V$, $A_{\pi(n)n} = \max_w\{X \subseteq V : G[X] \in \mathcal{F}_S\}$. The following lemmas state how to recursively compute all A-sets, B-sets, and C-sets other than A_{00}.

Lemma 3.2.2. *Let $i \in V \setminus \{0\}$. Then $A_{ii} = A_{<ii} \cup \{i\}$.*

Proof. Let $i \in V \setminus \{0\}$. Then the neighbourhood of i in $G[V_{ii}]$ is $\varnothing$. Consequently, no subset of V_{ii} that contains i induces an S-cycle of G, so $i \in A_{ii}$. By these facts, Observation 3.2.1(3) and Definition 3.2.3, it follows that

$$\left.\begin{array}{l} A_{ii} \setminus \{i\} \in \mathcal{A}_{<ii} \\ A_{<ii} \cup \{i\} \in \mathcal{A}_{ii} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(A_{ii} \setminus \{i\}) \leq w(A_{<ii}) \\ w(A_{<ii} \cup \{i\}) \leq w(A_{ii}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(A_{ii}) = w(A_{<ii} \cup \{i\}) \Rightarrow A_{ii} = A_{<ii} \cup \{i\}. \qquad \square$$

Lemma 3.2.3. *Let $i \in V$ and $x \in V \setminus V_{ii}$.*

(1) If $\{i, x\} \notin E$, then $B^{xx}_{ii} = A_{ii}$.

(2) If $\{i, x\} \in E$, then $B^{xx}_{ii} = B^{xx}_{<ii} \cup \{i\}$.

Proof. Let $i \in V$ and $x \in V \setminus \{V_{ii}\}$.

(1) Let $\{i, x\} \notin E$. Then $i <_t x$ and $i <_b x$, so the neighbourhood of x in $G[V_{ii} \cup \{x\}]$ is $\varnothing$. Consequently, no subset of $V_{ii} \cup \{x\}$ that contains x induces an S-cycle of G. By this fact and Definitions 3.2.3 and 3.2.4, it follows that

$$\left.\begin{array}{l} B^{xx}_{ii} \in \mathcal{A}_{ii} \\ A_{ii} \in \mathcal{B}^{xx}_{ii} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B^{xx}_{ii}) \leq w(A_{ii}) \\ w(A_{ii}) \leq w(B^{xx}_{ii}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(B^{xx}_{ii}) = w(A_{ii}) \Rightarrow B^{xx}_{ii} = A_{ii}.$$

(2) Let $\{i, x\} \in E$. Then the neighbourhood of i in $G[V_{ii} \cup \{x\}]$ is $\{x\}$. Consequently, no subset of $V_{ii} \cup \{x\}$ that contains i induces an S-cycle of G, so $i \in B^{xx}_{ii}$. By these facts, Observation 3.2.1(3) and Definition 3.2.4, it follows that

$$\left.\begin{array}{l} B^{xx}_{ii} \setminus \{i\} \in \mathcal{B}^{xx}_{<ii} \\ B^{xx}_{<ii} \cup \{i\} \in \mathcal{B}^{xx}_{ii} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B^{xx}_{ii} \setminus \{i\}) \leq w(B^{xx}_{<ii}) \\ w(B^{xx}_{<ii} \cup \{i\}) \leq w(B^{xx}_{ii}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(B^{xx}_{ii}) = w(B^{xx}_{<ii} \cup \{i\}) \Rightarrow B^{xx}_{ii} = B^{xx}_{<ii} \cup \{i\}. \qquad \square$$

Lemma 3.2.4. *Let $i \in V$ and $xy \in \mathcal{X} \setminus \mathcal{I}$ such that $i <_t y$, $i <_b x$ and $x, y \notin S$.*

(1) If $\{i, y\} \notin E$, then $B^{xy}_{ii} = B^{xx}_{ii}$.

(2) If $\{i, x\} \notin E$, then $B^{xy}_{ii} = B^{yy}_{ii}$.

(3) If $\{i, x\}, \{i, y\} \in E$, then $B^{xy}_{ii} = \begin{cases} B^{xy}_{<ii} & \text{, if } i \in S \\ B^{xy}_{<ii} \cup \{i\}, & \text{if } i \notin S. \end{cases}$

Proof. Let $i \in V$ and $xy \in \mathcal{X} \setminus \mathcal{I}$ such that $i <_t y$, $i <_b x$ and $x, y \notin S$.

(1) Let $\{i, y\} \notin E$. Then $i, x <_t y$ and $i <_b y <_b x$, so the neighbourhood of y in $G[V_{ii} \cup \{x, y\}]$ is $\{x\}$. Consequently, no subset of $V_{ii} \cup \{x, y\}$ that contains y induces an S-cycle of G. By this fact and Definitions 3.2.4 and 3.2.5, it follows that

$$\left.\begin{array}{l} B_{ii}^{xy} \in \mathcal{B}_{ii}^{xx} \\ B_{ii}^{xx} \in \mathcal{B}_{ii}^{xy} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B_{ii}^{xy}) \leq w(B_{ii}^{xx}) \\ w(B_{ii}^{xx}) \leq w(B_{ii}^{xy}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(B_{ii}^{xy}) = w(B_{ii}^{xx}) \Rightarrow B_{ii}^{xy} = B_{ii}^{xx}.$$

(2) Let $\{i, x\} \notin E$. Then one can as in (1) show that $B_{ii}^{xy} = B_{ii}^{yy}$.

(3) Let $\{i, x\}, \{i, y\} \in E$. Then $x <_t i <_t y$ and $y <_b i <_b x$, so the neighbourhood of i in $G[X \cup \{x, y\}]$ is $\{x, y\}$.

Case 1: Let $i \in S$. Then $\langle i, x, y\rangle$ is an induced S-triangle of G, so $i \notin B_{ii}^{xy}$. By this fact, Observation 3.2.1(3) and Definition 3.2.5, it follows that

$$\left.\begin{array}{l} B_{ii}^{xy} \in \mathcal{B}_{<ii}^{xy} \\ B_{<ii}^{xy} \in \mathcal{B}_{ii}^{xy} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B_{ii}^{xy}) \leq w(B_{<ii}^{xy}) \\ w(B_{<ii}^{xy}) \leq w(B_{ii}^{xy}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(B_{ii}^{xy}) = w(B_{<ii}^{xy}) \Rightarrow B_{ii}^{xy} = B_{<ii}^{xx}.$$

Case 2: Let $i \notin S$. We will show that no subset of $V_{ii} \cup \{x, y\}$ that contains i induces an S-cycle of G.

- Let $v_1, v_2 \in V_{<ii} \cup \{x, y\}$ such that $\langle v_1, v_2, i\rangle$ is an induced S-triangle of G. Then $\{v_1, v_2\} = \{x, y\}$, a contradiction, because $i, x, y \notin S$.
- Let $v_1, v_2, v_3 \in V_{<ii} \cup \{x, y\}$ such that $\langle v_1, v_2, v_3, i\rangle$ is an induced S-square of G. Then $\{v_1, v_3\} = \{x, y\}$, a contradiction, because $\{x, y\} \in E$.

Therefore, no subset of $V_{ii} \cup \{x, y\}$ that contains i induces an S-cycle of G, so $i \in B_{ii}^{xy}$. By these facts, Observation 3.2.1(3) and Definition 3.2.5, it follows that

$$\left.\begin{array}{l} B_{ii}^{xy} \setminus \{i\} \in \mathcal{B}_{<ii}^{xy} \\ B_{<ii}^{xy} \cup \{i\} \in \mathcal{B}_{ii}^{xy} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B_{ii}^{xy} \setminus \{i\}) \leq w(B_{<ii}^{xy}) \\ w(B_{<ii}^{xy} \cup \{i\}) \leq w(B_{ii}^{xy}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(B_{ii}^{xy}) = w(B_{<ii}^{xy} \cup \{i\}) \Rightarrow B_{ii}^{xy} = B_{<ii}^{xy} \cup \{i\}. \qquad \square$$

Lemma 3.2.5. *Let $i \in V$, $xy \in \mathcal{X} \setminus \mathcal{I}$ and $z \in V \setminus V_{ij}$ such that $xy <_l zz$, $x, y \neq z$, $\{x, z\} \in E \vee \{y, z\} \in E$, $i <_t y$, $i <_b x$ and $x, y, z \notin S$.*

(1) If $\{i,z\} \notin E$, *then* $C_{ii}^{xy,zz} = B_{ii}^{xy}$.

(2) If $\{i,z\} \in E$, *then* $C_{ii}^{xy,zz} = \begin{cases} C_{<ii}^{xy,zz} & , \text{if } i \in S \\ C_{<ii}^{xy,zz} \cup \{i\}, & \text{if } i \notin S. \end{cases}$

Proof. Let $i \in V$, $xy \in \mathcal{X} \setminus \mathcal{I}$ and $z \in V \setminus V_{ij}$ such that $xy <_l zz$, $x, y \neq z$, $\{x,z\} \in E \vee \{y,z\} \in E$, $i <_t y$, $i <_b x$ and $x, y, z \notin S$.

(1) Let $\{i,z\} \notin E$. Then $\left.\begin{matrix} i, x <_t y, z \\ i, y <_b z <_b x \end{matrix}\right\}$ or $\left.\begin{matrix} i, x <_t z <_t y \\ i, y <_b x, z \end{matrix}\right\}$, so the neighbourhood of z in $G[V_{ii} \cup \{x,y,z\}]$ is a subset of $\{x,y\}$. We will show that no subset of $V_{ii} \cup \{x,y,z\}$ that contains z induces an S-cycle of G.

- Let $v_1, v_2 \in V_{ii} \cup \{x,y\}$ such that $\langle v_1, v_2, z\rangle$ is an induced S-triangle of G. Then $\{v_1, v_2\} = \{x,y\}$, a contradiction, because $x, y, z \notin S$.
- Let $v_1, v_2, v_3 \in V_{ii} \cup \{x,y\}$ such that $\langle v_1, v_2, v_3, z\rangle$ is an induced S-square of G. Then $\{v_1, v_3\} = \{x,y\}$, a contradiction, because $\{x,y\} \in E$.

Therefore, no subset of $V_{ii} \cup \{x,y,z\}$ that contains z induces an S-cycle of G. By this fact and Definitions 3.2.5 and 3.2.6, it follows that

$$\left.\begin{matrix} C_{ii}^{xy,zz} \in \mathcal{B}_{ii}^{xy} \\ B_{ii}^{xy} \in \mathcal{C}_{ii}^{xy,zz} \end{matrix}\right\} \Rightarrow \left.\begin{matrix} w(C_{ii}^{xy,zz}) \leq w(B_{ii}^{xy}) \\ w(B_{ii}^{xy}) \leq w(C_{ii}^{xy,zz}) \end{matrix}\right\} \Rightarrow$$

$$\Rightarrow w(C_{ii}^{xy,zz}) = w(B_{ii}^{xy}) \Rightarrow C_{ii}^{xy,zz} = B_{ii}^{xy}.$$

(2) Let $\{i,z\} \in E$. Then either $\left.\begin{matrix} i, x <_t y, z \\ y <_b z <_b i <_b x \end{matrix}\right\}$ or $\left.\begin{matrix} x <_t z <_t i <_t y \\ i, y <_b x, z \end{matrix}\right\}$, so (i) the neighbourhood of i in $G[V_{ii} \cup \{x,y,z\}]$ is a superset of $\{y,z\}$ or (ii) it is a superset of $\{x,z\}$.

Case 1: Let $i \in S$. We will show that $i \notin C_{ii}^{xy,zz}$. Let $i \in C_{ii}^{xy,zz}$. Assume that (i) holds.

- If $\{y,z\} \in E$, then $\langle i, y, z\rangle$ is an induced S-triangle of S, a contradiction.
- If $\{y,z\} \notin E$, then $\langle i, y, x, z\rangle$ is an induced S-square of S, a contradiction.

Likewise, assuming that (ii) holds. Therefore, $i \notin C_{ii}^{xy,zz}$. By this fact, Observation 3.2.1 and Definition and 3.2.6, it follows that

$$\left.\begin{matrix} C_{ii}^{xy,zz} \in \mathcal{C}_{<ii}^{xy,zz} \\ C_{<ii}^{xy,zz} \in \mathcal{C}_{ii}^{xy,zz} \end{matrix}\right\} \Rightarrow \left.\begin{matrix} w(C_{ii}^{xy,zz}) \leq w(C_{<ii}^{xy,zz}) \\ w(C_{<ii}^{xy,zz}) \leq w(C_{ii}^{xy,zz}) \end{matrix}\right\} \Rightarrow$$

$$\Rightarrow w(C_{ii}^{xy,zz}) = w(C_{<ii}^{xy,zz}) \Rightarrow C_{ii}^{xy,zz} = C_{<ii}^{xy,zz}.$$

Case 2: Let $i \notin S$. We will show that if a subset of $V_{ii} \cup \{x, y, z\}$ that contains i induces an S-cycle of G, then its non-empty intersection with $V_{<ii}$ is not a subset of $C_{ii}^{xy,zz}$.

- Let $v_1, v_2 \in V_{<ii} \cup \{x, y, z\}$ such that $\langle v_1, v_2, i\rangle$ is an induced S-triangle of G. Then $\{v_1, v_2\} \subset \{x, y, z\}$, a contradiction, because $i, x, y, z \notin S$.
- Let $v_1, v_2, v_3 \in V_{<ii} \cup \{x, y, z\}$ such that $\langle v_1, v_2, v_3, i\rangle$ is an induced S-square of G. Then $\{v_1, v_3\} \subset \{x, y, z\}$ and, since $i, x, y, z \notin S$, $v_2 \in S \Rightarrow v_2 \in V_{<ii}$.
  - Assuming that $\{v_1, v_3\} = \{x, y\}$ yields a contradiction, because $\{x, y\} \in E$.
  - Assume that $\{v_1, v_3\} = \{y, z\}$. If $\{y, z\} \in E$, this yields a contradiction. If $\{y, z\} \notin E$, then $\langle y, v_2, z, x\rangle$ is an induced S-square of G, so $v_2 \notin C_{ii}^{xy,zz}$.
  - Likewise, assuming that $\{v_1, v_3\} = \{x, z\}$.

Therefore, if a subset of $V_{ii} \cup \{x, y, z\}$ that contains i induces an S-cycle of G, then its non-empty intersection with $V_{<ii}$ is not a subset of $C_{ii}^{xy,zz}$, so $i \in C_{ii}^{xy,zz}$. By these facts, Observation 3.2.1(3) and Definition 3.2.6, it follows that

$$\left.\begin{array}{l} C_{ii}^{xy,zz} \setminus \{i\} \in \mathcal{C}_{<ii}^{xy,zz} \\ C_{<ii}^{xy,zz} \cup \{i\} \in \mathcal{C}_{ii}^{xy,zz} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(C_{ii}^{xy,zz} \setminus \{i\}) \leq w(C_{<ii}^{xy,zz}) \\ w(C_{<ii}^{xy,zz} \cup \{i\}) \leq w(C_{ii}^{xy,zz}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(C_{ii}^{xy,zz}) = w(C_{<ii}^{xy,zz} \cup \{i\}) \Rightarrow C_{ii}^{xy,zz} = C_{<ii}^{xy,zz} \cup \{i\}. \qquad \square$$

Lemma 3.2.6. *Let $i \in V$ and $xy, zw \in \mathcal{X} \setminus \mathcal{I}$ such that $xy <_l zw$, $x, y \neq z, w$, $\{x, w\}, \{y, z\} \in E$, $i <_t y, w$, $i <_b x, z$ and $x, y, z, w \notin S$.*

(1) If $\{i, w\} \notin E$, then $C_{ii}^{xy,zw} = C_{ii}^{xy,zz}$.

(2) If $\{i, z\} \notin E$, then $C_{ii}^{xy,zw} = C_{ii}^{xy,ww}$.

(3) If $\{i, z\}, \{i, w\} \in E$, then $C_{ii}^{xy,zw} = \left\{\begin{array}{l} C_{<ii}^{xy,zw} \quad\quad\quad\quad\quad\quad , \text{if } i \in S \\ C_{<ii}^{xy,zw} \cup \{i\}, \text{if } i \notin S. \end{array}\right.$

Proof. Let $i \in V$ and $xy, zw \in \mathcal{X} \setminus \mathcal{I}$ such that $xy <_l zw$, $x, y \neq z, w$, $\{x, w\}, \{y, z\} \in E$, $i <_t y, w$, $i <_b x, z$ and $x, y, z, w \notin S$.

(1) Let $\{i, w\} \notin E$. Then $i, x <_t y, z, w$, $z <_t w$, $w <_b z$ and $i, y <_b x, z, w$, so the neighbourhood of w in $G[V_{ii} \cup \{x, y, z, w\}]$ is a subset of $\{x, y, z\}$. We will show that if a subset of $V_{ii} \cup \{x, y, z, w\}$ that contains w induces an S-cycle of G, then its non-empty intersection with V_{ii} is not a subset of any $X \in \mathcal{C}_{ii}^{xy,zz}$.

- Let $v_1, v_2 \in V_{ii} \cup \{x, y, z\}$ such that $\langle v_1, v_2, w \rangle$ is an induced S-triangle of G. Then $\{v_1, v_2\} \subset \{x, y, z\}$, a contradiction, because $x, y, z, w \notin S$.
- Let $v_1, v_2, v_3 \in V_{ii} \cup \{x, y, z\}$ such that $\langle v_1, v_2, v_3, w \rangle$ is an induced S-square of G. Then $\{v_1, v_3\} \subset \{x, y, z\}$ and, since $x, y, z, w \notin S$, $v_2 \in S \Rightarrow v_2 \in V_{ii}$.
  - Assuming that $\{v_1, v_3\} = \{x, y\}$ or $\{y, z\}$ yields a contradiction, because $\{x, y\}, \{y, z\} \in E$.
  - Assume that $\{v_1, v_3\} = \{x, z\}$. If $\{x, z\} \in E$, this yields a contradiction. If $\{x, z\} \notin E$, then $\langle x, v_2, z, y \rangle$ is an induced S-square of G, so $v_2 \notin C_{ii}^{xy,zz}$.

Therefore, if a subset of $V_{ii} \cup \{x, y, z, w\}$ that contains w induces an S-cycle of G, then its non-empty intersection with V_{ii} is not a subset of $C_{ii}^{xy,zz}$. By this fact and Definitions 3.2.6 and 3.2.7, it follows that

$$\left.\begin{array}{l} C_{ii}^{xy,zw} \in \mathcal{C}_{ii}^{xy,zz} \\ C_{ii}^{xy,zz} \in \mathcal{C}_{ii}^{xy,zw} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(C_{ii}^{xy,zw}) \leq w(C_{ii}^{xy,zz}) \\ w(C_{ii}^{xy,zz}) \leq w(C_{ii}^{xy,zw}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(C_{ii}^{xy,zw}) = w(C_{ii}^{xy,zz}) \Rightarrow C_{ii}^{xy,zw} = C_{ii}^{xy,zz}.$$

(2) Let $\{i, z\} \notin E$. Then one can as in (1) show that $C_{ii}^{xy,zw} = C_{ii}^{xy,ww}$.

(3) Let $\{i, z\}, \{i, w\} \in E$. Then $x <_t z <_t i <_t y, w$ and $y <_b w <_b i <_b x, z$, so the neighbourhood of i in $G[V_{ii} \cup \{x, y, z, w\}]$ is $\{x, y, z, w\}$.

Case 1: Let $i \in S$. Then $\langle i, x, y \rangle$ is an S-triangle of G, so $i \notin C_{ii}^{xy,zw}$. By this fact, Observation 3.2.1 and Definition 3.2.7, it follows that

$$\left.\begin{array}{l} C_{ii}^{xy,zw} \in \mathcal{C}_{<ii}^{xy,zw} \\ C_{<ii}^{xy,zw} \in \mathcal{C}_{ii}^{xy,zw} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(C_{ii}^{xy,zw}) \leq w(C_{<ii}^{xy,zw}) \\ w(C_{<ii}^{xy,zw}) \leq w(C_{ii}^{xy,zw}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(C_{ii}^{xy,zw}) = w(C_{<ii}^{xy,zw}) \Rightarrow C_{ii}^{xy,zw} = C_{<ii}^{xy,zw}.$$

Case 2: Let $i \notin S$. We will show that if a subset of $V_{ii} \cup \{x, y, z, w\}$ that contains i induces an S-cycle of G, then its non-empty intersection with $V_{<ii}$ is not a subset of $C_{ii}^{xy,zw}$.

- Let $v_1, v_2 \in V_{<ii} \cup \{x, y, z, w\}$ such that $\langle v_1, v_2, i\rangle$ is an induced S-triangle of G. Then $\{v_1, v_2\} \subset \{x, y, z, w\}$, a contradiction, because $i, x, y, z, w \notin S$.

- Let $v_1, v_2, v_3 \in V_{<ii} \cup \{x, y, z, w\}$ such that $\langle v_1, v_2, v_3, i\rangle$ is an induced S-square of G. Then $\{v_1, v_3\} \subset \{x, y, z, w\}$ and, since $i, x, y, z, w \notin S$, $v_2 \in S \Rightarrow v_2 \in V_{<ii}$.

  - Assuming that $\{v_1, v_3\} = \{x, y\}$, $\{y, z\}$, $\{z, w\}$ or $\{w, x\}$ yields a contradiction, because $\{x, y\}, \{y, z\}, \{z, w\}, \{w, x\} \in E$.
  - Assume that $\{v_1, v_3\} = \{x, z\}$. If $\{x, z\} \in E$, this yields a contradiction. If $\{x, z\} \notin E$, then $\langle x, v_2, z, y\rangle$ is an induced S-square of G, so $v_2 \notin C_{ii}^{xy,zw}$.
  - Likewise, assuming that $\{v_1, v_3\} = \{y, w\}$.

Therefore, if a subset of $V_{ii} \cup \{x, y, z, w\}$ that contains i induces an S-cycle of G, then its non-empty intersection with $V_{<ii}$ is not a subset of $C_{ii}^{xy,zw}$, so $i \in C_{ii}^{xy,zw}$. By these facts, Observation 3.2.1(3) and Definition 3.2.7, it follows that

$$\left.\begin{array}{l} C_{ii}^{xy,zw} \setminus \{i\} \in \mathcal{C}_{<ii}^{xy,zw} \\ C_{<ii}^{xy,zw} \cup \{i\} \in \mathcal{C}_{ii}^{xy,zw} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(C_{ii}^{xy,zw} \setminus \{i\}) \leq w(C_{<ii}^{xy,zw}) \\ w(C_{<ii}^{xy,zw} \cup \{i\}) \leq w(C_{ii}^{xy,zw}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(C_{ii}^{xy,zw}) = w(C_{<ii}^{xy,zw} \cup \{i\}) \Rightarrow C_{ii}^{xy,zw} = C_{<ii}^{xy,zw} \cup \{i\}. \qquad \square$$

Lemma 3.2.7. *Let $ij \in \mathcal{X} \setminus \mathcal{I}$. Then*

$$A_{ij} = \begin{cases} \max_w \left\{ A_{\leqslant ij}, A_{\leqslant ij}, B^{ii}_{\ll jj} \cup \{i, j\}, B^{jj}_{\ll ii} \cup \{i, j\} \right\}, \\ \textit{if } i \in S \textit{ or } j \in S \\ \\ \max_w \left\{ A_{\leqslant ij}, A_{\leqslant ij}, B^{ij}_{<ij} \cup \{i, j\} \right\}, \\ \textit{if } i, j \notin S. \end{cases}$$

Proof. Let $ij \in \mathcal{X} \setminus \mathcal{I}$.

Let $j \notin A_{ij}$. By this fact, Observation 3.2.1(1) and Definition 3.2.3, it follows that

$$\left.\begin{array}{l} A_{ij} \in \mathcal{A}_{\leqslant ij} \\ A_{\leqslant ij} \in \mathcal{A}_{ij} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(A_{ij}) \leq w(A_{\leqslant ij}) \\ w(A_{\leqslant ij}) \leq w(A_{ij}) \end{array}\right\} \Rightarrow$$
$$\Rightarrow w(A_{ij}) = w(A_{\leqslant ij}) \Rightarrow A_{ij} = A_{\leqslant ij}.$$

Let $i \notin A_{ij}$. Then one can as above show that $A_{ij} = A_{\leqslant ij}$.

Let $i, j \in A_{ij}$. By this fact and Observation 3.2.1(3), it follows that (i) $A_{ij} \setminus \{i, j\} \subseteq V_{<ij}$.

Case 1: Let $i \in S$ or $j \in S$. Let $h \in A_{ij} \setminus \{i, j\}$ such that $\{h, i\}, \{h, j\} \in E$. Then $\langle h, i, j\rangle$ is an induced S-triangle of G, a contradiction, so either $\{h, i\} \notin E$ or $\{h, j\} \notin E$ for all $h \in A_{ij} \setminus \{i, j\}$. Let $g, h \in A_{ij} \setminus \{i, j\}$ such that $\{g, j\}, \{h, i\} \in E$. Then

$$\left.\begin{array}{l} g <_t i <_t h \\ h <_b j <_b g \end{array}\right\} \Rightarrow \{g, h\} \in E.$$

Consequently, $\langle g, h, i, j\rangle$ is an induced S-square of G, a contradiction, so either $\{h, i\} \notin E$ for all $h \in A_{ij} \setminus \{i, j\}$ or $\{h, j\} \notin E$ for all $h \in A_{ij} \setminus \{i, j\}$. By this fact and Observations 3.2.1(4)–(5), it follows that either (ii) $A_{ij} \setminus \{i, j\} \subseteq V_{\ll jj}$ or (iii) $A_{ij} \setminus \{i, j\} \subseteq V_{\ll ii}$. Assume that (ii) holds. The neighbourhood of j in $G[V_{\ll jj} \cup \{i, j\}]$ is $\{i\}$. Consequently, no subset of $V_{\ll jj} \cup \{i, j\}$ that contains j induces an S-cycle of G. By this fact, (ii) and Definitions 3.2.3 and 3.2.4, it follows that

$$\left.\begin{array}{l} A_{ij} \setminus \{i, j\} \in \mathcal{B}^{ii}_{\ll jj} \\ B^{ii}_{\ll jj} \cup \{i, j\} \in \mathcal{A}_{ij} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(A_{ij} \setminus \{i, j\}) \leq w(B^{ii}_{\ll jj}) \\ w(B^{ii}_{\ll jj} \cup \{i, j\}) \leq w(A_{ij}) \end{array}\right\} \Rightarrow$$
$$\Rightarrow w(A_{ij}) = w(B^{ii}_{\ll jj} \cup \{i, j\}) \Rightarrow A_{ij} = B^{ii}_{\ll jj} \cup \{i, j\}.$$

Assuming that (iii) holds, one can as above show that $A_{ij} = B^{jj}_{\ll ii} \cup \{i, j\}$.

Case 2: Let $i, j \notin S$. By (i), Observation 3.2.1(3) and Definitions 3.2.3 and 3.2.5, it follows that

$$\left.\begin{array}{l} A_{ij} \setminus \{i, j\} \in \mathcal{B}^{ij}_{<ij} \\ B^{ij}_{<ij} \cup \{i, j\} \in \mathcal{A}_{ij} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(A_{ij} \setminus \{i, j\}) \leq w(B^{ij}_{<ij}) \\ w(B^{ij}_{<ij} \cup \{i, j\}) \leq w(A_{ij}) \end{array}\right\} \Rightarrow$$
$$\Rightarrow w(A_{ij}) = w(B^{ij}_{<ij} \cup \{i, j\}) \Rightarrow A_{ij} = B^{ij}_{<ij} \cup \{i, j\}. \qquad \square$$

Lemma 3.2.8. *Let $ij \in \mathcal{X} \setminus \mathcal{I}$ and $x \in V \setminus V_{ij}$.*

(1) If $\{i,x\},\{j,x\} \notin E$*, then* $B^{xx}_{ij} = A_{ij}$.

(2) If $\{i,x\} \in E$ *and* $\{j,x\} \notin E$*, then*

$$B^{xx}_{ij} = \begin{cases} \max_w \left\{ B^{xx}_{\leqslant ij}, B^{xx}_{\leqslant ij}, B^{ii}_{\ll jj} \cup \{i,j\}, B^{jj}_{\ll ix} \cup \{i,j\} \right\}, \\ \text{if } i \in S \text{ or } j \in S \\ \\ \max_w \left\{ B^{xx}_{\leqslant ij}, B^{xx}_{\leqslant ij}, B^{ij}_{<ij\ll xx} \cup \{i,j\} \right\}, \\ \text{if } i,j \notin S \text{ and } x \in S \\ \\ \max_w \left\{ B^{xx}_{\leqslant ij}, B^{xx}_{\leqslant ij}, C^{x'y',z'z'}_{<ij} \cup \{i,j\} \right\}, \\ \text{if } i,j,x \notin S \end{cases}$$

(3) If $\{i,x\} \notin E$ *and* $\{j,x\} \in E$*, then*

$$B^{xx}_{ij} = \begin{cases} \max_w \left\{ B^{xx}_{\leqslant ij}, B^{xx}_{\leqslant ij}, B^{ii}_{\ll xj} \cup \{i,j\}, B^{jj}_{\ll ii} \cup \{i,j\} \right\}, \\ \text{if } i \in S \text{ or } j \in S \\ \\ \max_w \left\{ B^{xx}_{\leqslant ij}, B^{xx}_{\leqslant ij}, B^{ij}_{<ij\ll xx} \cup \{i,j\} \right\}, \\ \text{if } i,j \notin S \text{ and } x \in S \\ \\ \max_w \left\{ B^{xx}_{\leqslant ij}, B^{xx}_{\leqslant ij}, C^{x'y',z'z'}_{<ij} \cup \{i,j\} \right\}, \\ \text{if } i,j,x \notin S \end{cases}$$

(4) If $\{i,x\},\{j,x\} \in E$*, then*

$$B^{xx}_{ij} = \begin{cases} \max_w \left\{ B^{xx}_{\leqslant ij}, B^{xx}_{\leqslant ij} \right\}, \\ \text{if } i \in S \text{ or } j \in S \text{ or } x \in S \\ \\ \max_w \left\{ B^{xx}_{\leqslant ij}, B^{xx}_{\leqslant ij}, C^{x'y',z'z'}_{<ij} \cup \{i,j\} \right\}, \\ \text{if } i,j,x \notin S \end{cases}$$

In all cases

- $x'y' = \min_l\{uv \in \mathcal{X} \setminus \mathcal{I} : u,v \in \{i,j,x\}\}$ *and*
- $z'z' = \min_l\{uu \in \mathcal{I} : u \in \{i,j,x\} \setminus \{x',y'\}\}$.

Proof. Let $ij \in \mathcal{X} \setminus \mathcal{I}$ and $x \in V \setminus V_{ij}$.

(1) Let $\{i,x\},\{j,x\} \notin E$. Then $i <_t j <_t x$ and $j <_b i <_b x$, so the neighbourhood of x in $G[V_{ij} \cup \{x\}]$ is $\varnothing$. Consequently, no subset of $V_{ij} \cup \{x\}$ that contains x induces an S-cycle of G. By this fact and Definitions 3.2.3 and 3.2.4, it follows that

$$\left.\begin{array}{l} B_{ij}^{xx} \in \mathcal{A}_{ij} \\ A_{ij} \in \mathcal{B}_{ij}^{xx} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B_{ij}^{xx}) \leq w(A_{ij}) \\ w(A_{ij}) \leq w(B_{ij}^{xx}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(B_{ij}^{xx}) = w(A_{ij}) \Rightarrow B_{ij}^{xx} = A_{ij}.$$

(2)–(4) Let $\{i,x\} \in E$ or $\{j,x\} \in E$.

Let $j \notin B_{ij}^{xx}$. By this fact, Observation 3.2.1(1) and Definition 3.2.4, it follows that

$$\left.\begin{array}{l} B_{ij}^{xx} \in \mathcal{B}_{\leqslant ij}^{xx} \\ B_{\leqslant ij}^{xx} \in \mathcal{B}_{ij}^{xx} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B_{ij}^{xx}) \leq w(B_{\leqslant ij}^{xx}) \\ w(B_{\leqslant ij}^{xx}) \leq w(B_{ij}^{xx}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(B_{ij}^{xx}) = w(B_{\leqslant ij}^{xx}) \Rightarrow B_{ij}^{xx} = B_{\leqslant ij}^{xx}.$$

Let $i \notin B_{ij}^{xx}$. Then one can as above show that $B_{ij}^{xx} = B_{\leqslant ij}^{xx}$.

Let $i,j \in B_{ij}^{xx}$. By this fact and Observation 3.2.1(3), it follows that (i) $B_{ij}^{xx} \setminus \{i,j\} \subseteq V_{<ij}$. We define the following ordered crossing vertex pairs:

- $x'y' = \min_l\{uv \in \mathcal{X} \setminus \mathcal{I} : u,v \in \{i,j,x\}\}$ and
- $z'z' = \min_l\{uu \in \mathcal{I} : u \in \{i,j,x\} \setminus \{x',y'\}\}$.

(2) Let $\{i,x\} \in E$ and $\{j,x\} \notin E$.

Case 1: Let $i \in S$ or $j \in S$. Let $h \in B_{ij}^{xx} \setminus \{i,j\}$ such that $\{h,i\},\{h,j\} \in E$. Then $\langle h,i,j\rangle$ is an induced S-triangle of G, a contradiction, so either $\{h,i\} \notin E$ or $\{h,j\} \notin E$ for all $h \in B_{ij}^{xx} \setminus \{i,j\}$. Let $g,h \in B_{ij}^{xx} \setminus \{i,j\}$ such that $\{g,j\},\{h,i\} \in E$. Then

$$\left.\begin{array}{l} g <_t i <_t h \\ h <_b j <_b g \end{array}\right\} \Rightarrow \{g,h\} \in E.$$

Consequently, $\langle g,h,i,j\rangle$ is an induced S-square of G, a contradiction, so either $\{h,i\} \notin E$ for all $h \in B_{ij}^{xx} \setminus \{i,j\}$ or $\{h,j\} \notin E$ for all $h \in B_{ij}^{xx} \setminus \{i,j\}$. By this fact, (i) and Observations 3.2.1(4)–(5), it follows that either (ii) $B_{ij}^{xx} \setminus \{i,j\} \subseteq V_{\ll jj}$ or (iii) $B_{ij}^{xx} \setminus \{i,j\} \subseteq V_{\ll ii}$. Assume that (ii) holds. The neighbourhoods of

j and x in $G[V_{\ll jj} \cup \{i,j,x\}]$ are $\{i\}$. Consequently, no subset of $V_{\ll jj} \cup \{i,j,x\}$ that contains j and/or x induces an S-cycle of G. By this fact, (ii) and Definition 3.2.4, it follows that

$$\left.\begin{array}{l} B^{xx}_{ij} \setminus \{i,j\} \in \mathcal{B}^{ii}_{\ll jj} \\ B^{ii}_{\ll jj} \cup \{i,j\} \in \mathcal{B}^{xx}_{ij} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B^{xx}_{ij} \setminus \{i,j\}) \leq w(B^{ii}_{\ll jj}) \\ w(B^{ii}_{\ll jj} \cup \{i,j\}) \leq w(B^{xx}_{ij}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(B^{xx}_{ij}) = w(B^{ii}_{\ll jj} \cup \{i,j\}) \Rightarrow B^{xx}_{ij} = B^{ii}_{\ll jj} \cup \{i,j\}$$

Now, assume that (iii) holds. Let $h \in B^{xx}_{ij} \setminus \{i,j\}$ such that $\{h,x\} \in E$. Then

$$\left.\begin{array}{l} h <_t i <_t j <_t x \\ j <_b x <_b h <_b i \end{array}\right\} \Rightarrow \{h,j\} \in E.$$

Consequently, $\langle h,j,i,x\rangle$ is an induced S-square of G, a contradiction, so $\{h,x\} \notin E$ for all $h \in B^{xx}_{ij} \setminus \{i,j\}$. By this fact, (iii) and Observation 3.2.1(6), it follows that (iv) $B^{xx}_{ij} \setminus \{i,j\} \subseteq V_{\ll ix}$. The neighbourhoods of i and x in $G[V_{\ll ix} \cup \{i,j,x\}]$ are $\{j\}$. Consequently, no subset of $V_{\ll ix} \cup \{i,j,x\}$ that contains i and/or x induces an S-cycle of G. By this fact, (iv) and Definition 3.2.4, it follows that

$$\left.\begin{array}{l} B^{xx}_{ij} \setminus \{i,j\} \in \mathcal{B}^{jj}_{\ll ix} \\ B^{jj}_{\ll ix} \cup \{i,j\} \in \mathcal{B}^{xx}_{ij} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B^{xx}_{ij} \setminus \{i,j\}) \leq w(B^{jj}_{\ll ix}) \\ w(B^{jj}_{\ll ix} \cup \{i,j\}) \leq w(B^{xx}_{ij}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(B^{xx}_{ij}) = w(B^{jj}_{\ll ix} \cup \{i,j\}) \Rightarrow B^{xx}_{ij} = B^{jj}_{\ll ix} \cup \{i,j\}$$

Case 2: Let $i,j \notin S$ and $x \in S$. Let $h \in B^{xx}_{ij} \setminus \{i,j\}$ such that $\{h,x\} \in E$. Then

$$\left.\begin{array}{l} h,i <_t j <_t x \\ j <_b x <_b h <_b i \end{array}\right\} \Rightarrow \{h,j\} \in E.$$

Consequently, if $\{h,i\} \in E$, then $\langle h,i,j\rangle$ is an induced S-triangle of G, a contradiction, and if $\{h,i\} \notin E$, then $\langle h,j,i,x\rangle$ is an induced S-square of G, also a contradiction, so $\{h,x\} \notin E$ for all $h \in B^{xx}_{ij} \setminus \{i,j\}$. By this fact, (i) and Observation 3.2.1(8), it follows that

$$\left.\begin{array}{l} B^{xx}_{ij} \setminus \{i,j\} \in \mathcal{B}^{ij}_{<ij\ll xx} \\ B^{ij}_{<ij\ll xx} \cup \{i,j\} \in \mathcal{B}^{xx}_{ij} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B^{xx}_{ij} \setminus \{i,j\}) \leq w(B^{ij}_{<ij\ll xx}) \\ w(B^{ij}_{<ij\ll xx} \cup \{i,j\}) \leq w(B^{xx}_{ij}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(B^{xx}_{ij}) = w(B^{ij}_{<ij\ll xx} \cup \{i,j\}) \Rightarrow B^{xx}_{ij} = B^{ij}_{<ij\ll xx} \cup \{i,j\}$$

Case 3: Let $i, j, x \notin S$. By (i) and Definitions 3.2.4 and 3.2.6, it follows that

$$\left.\begin{array}{l} B_{ij}^{xx} \setminus \{i,j\} \in \mathcal{B}_{<ij}^{x'y',z'z'} \\ B_{<ij}^{x'y',z'z'} \cup \{i,j\} \in \mathcal{B}_{ij}^{xx} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B_{ij}^{xx} \setminus \{i,j\}) \leq w(B_{<ij}^{x'y',z'z'}) \\ w(B_{<ij}^{x'y',z'z'} \cup \{i,j\}) \leq w(B_{ij}^{xx}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(B_{ij}^{xx}) = w(B_{<ij}^{x'y',z'z'} \cup \{i,j\}) \Rightarrow B_{ij}^{xx} = B_{<ij}^{x'y',z'z'} \cup \{i,j\}.$$

(3) Let $\{i,x\} \notin E$ and $\{j,x\} \in E$. Then one can as above show the following:

- If $i \in S$ or $j \in S$, then either $B_{ij}^{xx} = B_{\ll xj}^{ii} \cup \{i,j\}$ or $B_{ij}^{xx} = B_{\ll ii}^{jj} \cup \{i,j\}$.
- If $x \in S$, then $B_{ij}^{xx} = B_{<ij \ll xx}^{ij} \cup \{i,j\}$.
- If $i, j, x \notin S$, then $B_{ij}^{xx} = B_{<ij}^{x'y',z'z'}$.

(4) Let $\{i,x\}, \{j,x\} \in E$.

Case 1: Let $i \in S$ or $j \in S$ or $x \in S$. Then $\langle i, j, x \rangle$ is an induced S-triangle of G, a contradiction.

Case 2: Let $i, j, x \notin S$. By (i) and Definitions 3.2.4 and 3.2.6, it follows that

$$\left.\begin{array}{l} B_{ij}^{xx} \setminus \{i,j\} \in \mathcal{B}_{<ij}^{x'y',z'z'} \\ B_{<ij}^{x'y',z'z'} \cup \{i,j\} \in \mathcal{B}_{ij}^{xx} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B_{ij}^{xx} \setminus \{i,j\}) \leq w(B_{<ij}^{x'y',z'z'}) \\ w(B_{<ij}^{x'y',z'z'} \cup \{i,j\}) \leq w(B_{ij}^{xx}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(B_{ij}^{xx}) = w(B_{<ij}^{x'y',z'z'} \cup \{i,j\}) \Rightarrow B_{ij}^{xx} = B_{<ij}^{x'y',z'z'} \cup \{i,j\}. \qquad \square$$

Lemma 3.2.9. *Let $ij, xy \in \mathcal{X} \setminus \mathcal{I}$ such that $j <_t y$, $i <_b x$ and $x, y \notin S$.*

(1) If $\{i,y\}, \{j,y\} \notin E$, then $B_{ij}^{xy} = B_{ij}^{xx}$.

(2) If $\{i,x\}, \{j,x\} \notin E$, then $B_{ij}^{xy} = B_{ij}^{yy}$.

(3) If $\{i,y\}, \{j,x\} \in E$, then

$$B_{ij}^{xy} = \begin{cases} \max_w \left\{ B_{\leqslant ij}^{xy}, B_{\leqslant ij}^{xy} \right\}, \\ \textit{if } i \in S \textit{ or } j \in S \\ \\ \max_w \left\{ B_{\leqslant ij}^{xy}, B_{\leqslant ij}^{xy}, C_{<ij}^{x'y',z'w'} \cup \{i,j\} \right\}, \\ \textit{if } i, j \notin S \end{cases}$$

where

- $x'y' = \min_l\{uv \in \mathcal{X} \setminus \mathcal{I} : u, v \in \{i, j, x, y\}\}$ *and*
- $z'w' = \min_l\{uv \in \mathcal{X} \setminus \mathcal{I} : u, v \in \{i, j, x, y\} \setminus \{x', y'\}\}$.

Proof. Let $ij, xy \in \mathcal{X} \setminus \mathcal{I}$ such that $j <_t y$, $i <_b x$ and $x, y \notin S$.

(1) Let $\{i, y\} \notin E$. Then $i <_t j$, $j, x <_t y$ and $j <_b i <_b y <_b x$, so the neighbourhood of y in $G[V_{ij} \cup \{x, y\}]$ is $\{x\}$. Consequently, no subset of $V_{ij} \cup \{x, y\}$ that contains y induces an S-cycle of G. By this fact and Definitions 3.2.4 and 3.2.5, it follows that

$$\left.\begin{array}{l} B^{xy}_{ij} \in \mathcal{B}^{xx}_{ij} \\ B^{xx}_{ij} \in \mathcal{B}^{xy}_{ij} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B^{xy}_{ij}) \leq w(B^{xx}_{ij}) \\ w(B^{xx}_{ij}) \leq w(B^{xy}_{ij}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(B^{xy}_{ij}) = w(B^{xx}_{ij}) \Rightarrow B^{xy}_{ij} = B^{xx}_{ij}.$$

(2) Let $\{j, x\} \notin E$. Then one can as in (1) show that $B^{xy}_{ij} = B^{yy}_{ij}$.

(3) Let $\{i, y\}, \{j, x\} \in E$.

Let $j \notin B^{xy}_{ij}$. By this fact, Observation 3.2.1(1) and Definition 3.2.5, it follows that

$$\left.\begin{array}{l} B^{xy}_{ij} \in \mathcal{B}^{xy}_{\leqslant ij} \\ B^{xy}_{\leqslant ij} \in \mathcal{B}^{xy}_{ij} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B^{xy}_{ij}) \leq w(B^{xy}_{\leqslant ij}) \\ w(B^{xy}_{\leqslant ij}) \leq w(B^{xy}_{ij}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(B^{xy}_{ij}) = w(B^{xy}_{\leqslant ij}) \Rightarrow B^{xy}_{ij} = B^{xy}_{\leqslant ij}.$$

Let $i \notin B^{xy}_{ij}$. Then one can as above show that $B^{xy}_{ij} = B^{xy}_{\leqslant ij}$.

Let $i, j \in B^{xy}_{ij}$. By this fact and Observation 3.2.1(3), it follows that (i) $B^{xy}_{ij} \setminus \{i, j\} \subseteq V_{<ij}$.

Case 1: Let $i \in S$ or $j \in S$.

- If $\{i, x\} \in E$, then $\langle i, x, y\rangle$ is an induced S-triangle of G, a contradiction.
- If $\{j, y\} \in E$, then $\langle j, x, y\rangle$ is an induced S-triangle of G, a contradiction.
- If $\{i, x\}, \{j, y\} \notin E$, then $\langle i, j, x, y\rangle$ is an induced S-square of G, a contradiction.

Case 2: Let $i, j \notin S$. We define the following ordered crossing vertex pairs:

- $x'y' = \min_l\{uv \in \mathcal{X} \setminus \mathcal{I} : u, v \in \{i, j, x, y\}\}$ and

- $z'w' = \min_l\{uv \in \mathcal{X} \setminus \mathcal{I} : u, v \in \{i, j, x, y\} \setminus \{x', y'\}\}$.

By (i) and Definitions 3.2.5 and 3.2.7, it follows that

$$\left.\begin{array}{l} B_{ij}^{xy} \setminus \{i,j\} \in \mathcal{C}_{<ij}^{x'y',z'w'} \\ C_{<ij}^{x'y',z'w'} \cup \{i,j\} \in \mathcal{B}_{ij}^{xy} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(B_{ij}^{xy} \setminus \{i,j\}) \leq w(C_{<ij}^{x'y',z'w'}) \\ w(C_{<ij}^{x'y',z'w'} \cup \{i,j\}) \leq w(B_{ij}^{xy}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(B_{ij}^{xy}) = w(C_{<ij}^{x'y',z'w'} \cup \{i,j\}) \Rightarrow B_{ij}^{xy} = C_{<ij}^{x'y',z'w'} \cup \{i,j\}. \qquad \square$$

Lemma 3.2.10. *Let $ij, xy \in \mathcal{X} \setminus \mathcal{I}$ and $z \in V \setminus V_{ij}$ such that $xy <_l zz$, $x, y \neq z$, $\{x, z\} \in E \vee \{y, z\} \in E$, $j <_t y$, $i <_b x$ and $x, y, z \notin S$.*

(1) If $\{i, z\}, \{j, z\} \notin E$, then $C_{ij}^{xy,zz} = B_{ij}^{xy}$.

(2) If $\{i, z\} \in E$ or $\{j, z\} \in E$, then

$$C_{ij}^{xy,zz} = \begin{cases} \max_w \left\{C_{\leqslant ij}^{xy,zz}, C_{\leqslant ij}^{xy,zz}\right\}, \\ \text{if } i \in S \text{ or } j \in S \\ \\ \max_w \left\{C_{\leqslant ij}^{xy,zz}, C_{\leqslant ij}^{xy,zz}, C_{<ij}^{x'y',z'w'} \cup \{i,j\}\right\}, \\ \text{if } i, j \notin S \end{cases}$$

where

- $x'y' = \min_l\{uv \in \mathcal{X} \setminus \mathcal{I} : u, v \in \{i, j, x, y, z\}\}$ *and*
- $z'w' = \min_l\{uv \in \mathcal{X} \setminus \mathcal{I} : u, v \in \{i, j, x, y, z\} \setminus \{x', y'\}\}$.

Proof. Let $ij, xy \in \mathcal{X} \setminus \mathcal{I}$ and $z \in V \setminus V_{ij}$ such that $xy <_l zz$, $x, y \neq z$, $\{x, z\} \in E \vee \{y, z\} \in E$, $j <_t y$, $i <_b x$ and $x, y, z \notin S$.

(1) Let $\{i, z\}, \{j, z\} \notin E$. Then $i <_t j$, $j, x <_t y, z$, $i, y <_b x, z$ and $j <_b i$, so the neighbourhood of z in $G[V_{ij} \cup \{x, y, z\}]$ is a subset of $\{x, y\}$. We will show that no subset of $V_{ij} \cup \{x, y, z\}$ that contains z induces an S-cycle of G.

- Let $v_1, v_2 \in V_{ij} \cup \{x, y\}$ such that $\langle v_1, v_2, z\rangle$ is an induced S-triangle of G. Then $\{v_1, v_2\} = \{x, y\}$, a contradiction, because $x, y, z \notin S$.
- Let $v_1, v_2, v_3 \in V_{ij} \cup \{x, y\}$ such that $\langle v_1, v_2, v_3, z\rangle$ is an induced S-square of G. Then $\{v_1, v_3\} = \{x, y\}$, a contradiction, because $\{x, y\} \in E$.

Therefore, no subset of $V_{ij} \cup \{x, y, z\}$ that contains z induces an S-cycle of G. By this fact and Definitions 3.2.5 and 3.2.6, it follows that

$$\left.\begin{array}{l} C_{ij}^{xy,zz} \in \mathcal{B}_{ij}^{xy} \\ B_{ij}^{xy} \in \mathcal{C}_{ij}^{xy,zz} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(C_{ij}^{xy,zz}) \leq w(B_{ij}^{xy}) \\ w(B_{ij}^{xy}) \leq w(C_{ij}^{xy,zz}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(C_{ij}^{xy,zz}) = w(B_{ij}^{xy}) \Rightarrow C_{ij}^{xy,zz} = B_{ij}^{xy}.$$

(2) Let $\{i, z\} \in E$ or $\{j, z\} \in E$.

Let $j \notin C_{ij}^{xy,zz}$. By this fact, Observation 3.2.1(1) and Definition 3.2.6, it follows that

$$\left.\begin{array}{l} C_{ij}^{xy,zz} \in \mathcal{C}_{\leqslant ij}^{xy,zz} \\ C_{\leqslant ij}^{xy,zz} \in \mathcal{C}_{ij}^{xy,zz} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(C_{ij}^{xy,zz}) \leq w(C_{\leqslant ij}^{xy,zz}) \\ w(C_{\leqslant ij}^{xy,zz}) \leq w(C_{ij}^{xy,zz}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(C_{ij}^{xy,zz}) = w(C_{\leqslant ij}^{xy,zz}) \Rightarrow C_{ij}^{xy,zz} = C_{\leqslant ij}^{xy,zz}.$$

Let $i \notin C_{ij}^{xy,zz}$. Then one can as above show that $C_{ij}^{xy,zz} = C_{\leqslant ij}^{xy,zz}$.

Let $i, j \in C_{ij}^{xy,zz}$. By this fact and Observation 3.2.1(3), it follows that (i) $C_{ij}^{xy,zz} \setminus \{i, j\} \subseteq V_{<ij}$.

Case 1: Let $i \in S$ or $j \in S$.

- Let $\{i, z\} \in E$. Then either

$$\left.\begin{array}{c} x <_t z <_t< i <_t j <_t y \\ i, y <_b x, z \end{array}\right\} \Rightarrow \{i, x\}, \{y, z\} \in E$$

or

$$\left.\begin{array}{c} i <_t j <_t y, z \\ x <_t z \\ y <_b z <_b< i <_b x \end{array}\right\} \Rightarrow \{i, y\}, \{x, z\} \in E.$$

Assume that the former inequalities hold.

  - If $\{i, y\} \in E$, then $\langle i, x, y\rangle$ is an induced S-triangle of G, a contradiction.
  - If $\{x, z\} \in E$, then $\langle i, x, z\rangle$ is an induced S-triangle of G, a contradiction.
  - If $\{i, y\}, \{x, z\} \notin E$, then $\langle i, x, y, z\rangle$ is an induced S-square of G, a contradiction.

Likewise, assuming that the latter inequalities hold.

Case 2: Let $i, j \notin S$. We define the following ordered crossing vertex pairs:

- $x'y' = \min_l\{uv \in \mathcal{X} \setminus \mathcal{I} : u, v \in \{i, j, x, y, z, w\}\}$,
- $z'w' = \min_l\{uv \in \mathcal{X} \setminus \mathcal{I} : u, v \in \{i, j, x, y, z, w\} \setminus \{x', y'\}\}$ and
- $a'a' = \min_l\{uu \in \mathcal{I} : u \in \{i, j, x, y, z\} \setminus \{x', y', z', w'\}\}$.

We will show that if a subset of $V_{<ij} \cup \{x', y', z', w', a'\}$ that contains a' induces an S-cycle of G, then its non-empty intersection with $V_{<ij}$ is not a subset of $C_{<ij}^{x'y',z'w'}$.

- Let $v_1, v_2 \in V_{<ij} \cup \{x', y', z', w'\}$ such that $\langle v_1, v_2, a'\rangle$ is an induced S-triangle of G. Since $x', y', z', w' \notin S$, without loss of generality, assume that $v_1 \in S \Rightarrow v_1 \in V_{<ij}$. Then either

$$\left.\begin{array}{c} x' <_t z' <_t< a' <_t v_1 <_t y' \\ v_1, y' <_b x', z', a' \end{array}\right\} \Rightarrow \{v_1, x'\}, \{v_1, z'\} \in E$$

  or

$$\left.\begin{array}{c} v_1, x' <_t y', w', a' \\ y' <_b w' <_b< a' <_b v_1 <_b x' \end{array}\right\} \Rightarrow \{v_1, y'\}, \{v_1, w'\} \in E.$$

  Assume that the former inequalities hold.

  - If $\{x', z'\} \in E$, then $\langle v_1, x', z'\rangle$ is an induced S-triangle of G.
  - If $\{v_1, y'\} \in E$, then $\langle v_1, x', y'\rangle$ is an induced S-triangle of G.
  - If $\{x', z'\}, \{v_1, y'\} \notin E$, then $\langle v_1, x', y', z'\rangle$ is induced an S-square of G.

  Therefore, $v_1 \notin C_{<ij}^{x'y',z'w'}$. Likewise, assuming that the latter inequalities hold.

- Let $v_1, v_2, v_3 \in V_{<ij} \cup \{x', y', z', w'\}$ such that $\langle v_1, v_2, v_3, a'\rangle$ is an induced S-square of G. Assuming that $v_1 \in S$ or $v_3 \in S$, one can as above show that the S-vertex is not an element of $C_{<ij}^{x'y',z'w'}$. Assume that $v_2 \in S$. Since $x', y', z', w' \notin S$, $v_2 \in V_{<ij}$. Then either

$$\left.\begin{array}{c} v_2, x' <_t a' <_t v_1, v_3 \\ v_1, v_3 <_b v_2 <_b x', a' \end{array}\right\} \Rightarrow \{v_1, x'\}, \{v_3, x'\} \in E$$

  or

$$\left.\begin{array}{c} v_1, v_3 <_b v_2 <_b y', a' \\ v_2, y' <_t a' <_t v_1, v_3 \end{array}\right\} \Rightarrow \{v_1, y'\}, \{v_3, y'\} \in E.$$

  Assume that the former inequalities hold.

– If $\{v_2, x'\} \in E$, then $\langle v_1, v_2, x' \rangle$ is an induced S-triangle of G.
– If $\{v_2, x'\} \notin E$, then $\langle v_1, v_2, v_3, x' \rangle$ is an induced S-square of G.

Therefore, $\{v_1, v_2, v_3\} \cap V_{<ij}$ is not a subset of $C_{<ij}^{x'y',z'w'}$. Likewise, assuming that the latter inequalities hold.

Therefore, if a subset of $V_{<ij} \cup \{x', y', z', w', a'\}$ that contains a' induces an S-cycle of G, then its non-empty intersection with $V_{<ij}$ is not a subset of $C_{<ij}^{x'y',z'w'}$. By this fact, (i) and Definitions 3.2.6 and 3.2.7, it follows that

$$\left.\begin{array}{l} C_{ij}^{xy,zz} \setminus \{i,j\} \in \mathcal{C}_{<ij}^{x'y',z'w'} \\ C_{<ij}^{x'y',z'w'} \cup \{i,j\} \in \mathcal{C}_{ij}^{xy,zz} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(C_{ij}^{xy,zz} \setminus \{i,j\}) \leq w(C_{<ij}^{x'y',z'w'}) \\ w(C_{<ij}^{x'y',z'w'} \cup \{i,j\}) \leq w(C_{ij}^{xy,zz}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(C_{ij}^{xy,zz}) = w(C_{<ij}^{x'y',z'w'} \cup \{i,j\}) \Rightarrow C_{ij}^{xy,zz} = C_{<ij}^{x'y',z'w'} \cup \{i,j\}. \qquad \square$$

Lemma 3.2.11. *Let $ij, xy, zw \in \mathcal{X} \setminus \mathcal{I}$ such that $xy <_l zw$, $x, y \neq z, w$, $\{x,w\}, \{y,z\} \in E$, $j <_t y, w$, $i <_b x, z$ and $x, y, z, w \notin S$.*

(1) If $\{i,w\} \notin E$, then $C_{ij}^{xy,zw} = C_{ij}^{xy,zz}$.

(2) If $\{j,z\} \notin E$, then $C_{ij}^{xy,zw} = C_{ij}^{xy,ww}$.

(3) If $\{i,w\}, \{j,z\} \in E$, then

$$C_{ij}^{xy,zw} = \begin{cases} \max_w \left\{ C_{\leqslant ij}^{xy,zw}, C_{\leqslant ij}^{xy,zw} \right\}, \\ \text{if } i \in S \text{ or } j \in S \\ \\ \max_w \left\{ C_{\leqslant ij}^{xy,zw}, A_{\leqslant ij}^{xy,zw}, A_{<ij}^{x'y',z'w'} \cup \{i,j\} \right\}, \\ \text{if } i,j \notin S \end{cases}$$

where

- $x'y' = \min_l\{uv \in \mathcal{X} \setminus \mathcal{I} : u, v \in \{i,j,x,y,z,w\}\}$ *and*
- $z'w' = \min_l\{uv \in \mathcal{X} \setminus \mathcal{I} : u, v \in \{i,j,x,y,z,w\} \setminus \{x', y'\}\}$.

Proof. Let $ij, xy, zw \in \mathcal{X} \setminus \mathcal{I}$ such that $xy <_l zw$, $x, y \neq z, w$, $\{x,w\}, \{y,z\} \in E$, $j <_t y, w$, $i <_b x, z$ and $x, y, z, w \notin S$.

(1) Let $\{i,w\} \notin E$. Then $i <_t j <_t y, w$, $x <_t y, z$, $z <_t w$, $j <_b i <_b w <_b z$ and $i, y <_b x, w$, so the neighbourhood of w in $G[V_{ij} \cup \{x,y,z,w\}]$ is a subset of $\{x,y,z\}$. We will show that if a subset of $V_{ij} \cup \{x,y,z,w\}$ that contains w induces an S-cycle of G, then its non-empty intersection with V_{ij} is not a subset of $C_{ij}^{xy,zz}$.

- Let $v_1, v_2 \in V_{ij} \cup \{x, y, z\}$ such that $\langle v_1, v_2, w\rangle$ is an induced S-triangle of G. Then $\{v_1, v_2\} \subset \{x, y, z\}$, a contradiction, because $x, y, z, w \notin S$.
- Let $v_1, v_2, v_3 \in V_{ij} \cup \{x, y, z\}$ such that $\langle v_1, v_2, v_3, w\rangle$ is an induced S-square of G. Then $\{v_1, v_3\} \subset \{x, y, z\}$ and, since $x, y, z, w \notin S$, $v_2 \in S \Rightarrow v_2 \in V_{ii}$.
  - Assuming that $\{v_1, v_3\} = \{x, y\}$ or $\{y, z\}$ yields a contradiction, because $\{x, y\}, \{y, z\} \in E$.
  - Assume that $\{v_1, v_3\} = \{x, z\}$. If $\{x, z\} \in E$, this yields a contradiction. If $\{x, z\} \notin E$, then $\langle x, v_2, z, y\rangle$ is an induced S-square of G, so $v_2 \notin C_{ij}^{xy,zz}$.

Therefore, if a subset of $V_{ij} \cup \{x, y, z, w\}$ that contains w induces an S-cycle of G, then its non-empty intersection with V_{ij} is not a subset of $C_{ij}^{xy,zz}$. By this fact and Definitions 3.2.6 and 3.2.7, it follows that

$$\left.\begin{array}{l} C_{ij}^{xy,zw} \in \mathcal{C}_{ij}^{xy,zz} \\ C_{ij}^{xy,zz} \in \mathcal{C}_{ij}^{xy,zw} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(C_{ij}^{xy,zw}) \leq w(C_{ij}^{xy,zz}) \\ w(C_{ij}^{xy,zz}) \leq w(C_{ij}^{xy,zw}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(C_{ij}^{xy,zw}) = w(C_{ij}^{xy,zz}) \Rightarrow C_{ij}^{xy,zw} = C_{ij}^{xy,zz}.$$

(2) Let $\{j, z\} \notin E$. Then one can as in (1) show that $C_{ij}^{xy,zw} = C_{ij}^{xy,ww}$.

(3) Let $\{i, w\}, \{j, z\} \in E$.

Let $j \notin C_{ij}^{xy,zw}$. By this fact, Observation 3.2.1(1) and Definition 3.2.7, it follows that

$$\left.\begin{array}{l} C_{ij}^{xy,zw} \in \mathcal{C}_{\leqslant ij}^{xy,zw} \\ C_{\leqslant ij}^{xy,zw} \in \mathcal{C}_{ij}^{xy,zw} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(C_{ij}^{xy,zw}) \leq w(C_{\leqslant ij}^{xy,zw}) \\ w(C_{\leqslant ij}^{xy,zw}) \leq w(C_{ij}^{xy,zw}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(C_{ij}^{xy,zw}) = w(C_{\leqslant ij}^{xy,zw}) \Rightarrow C_{ij}^{xy,zw} = C_{\leqslant ij}^{xy,zw}.$$

Let $i \notin C_{ij}^{xy,zw}$. Then one can as above show that $C_{ij}^{xy,zw} = C_{\leqslant ij}^{xy,zw}$.

Let $i, j \in C_{ij}^{xy,zw}$. By this fact and Observation 3.2.1(3), it follows that (i) $C_{ij}^{xy,zw} \setminus \{i, j\} \subseteq V_{<ij}$.

Case 1: Let $i \in S$ or $j \in S$.

- If $\{i, z\} \in E$, then $\langle i, j, z\rangle$ is an induced S-triangle of G, a contradiction.
- If $\{j, w\} \in E$, then $\langle i, j, w\rangle$ is an induced S-triangle of G, a contradiction.

- If $\{i,z\},\{j,w\} \notin E$, then $\langle i,j,z,w\rangle$ is an induced S-square of G, a contradiction.

Case 2: Let $i,j \notin S$. We define the following ordered crossing vertex pairs:

- $x'y' = \min_l\{uv \in \mathcal{X} \setminus \mathcal{I} : u,v \in \{i,j,x,y,z,w\}\}$,
- $z'w' = \min_l\{uv \in \mathcal{X} \setminus \mathcal{I} : u,v \in \{i,j,x,y,z,w\} \setminus \{x',y'\}\}$ and
- $a'b' = \min_l\{uv \in \mathcal{X} \setminus \mathcal{I} : u,v \in \{i,j,x,y,z,w\} \setminus \{x',y',z',w'\}\}$.

We will show that if a subset of $V_{<ij} \cup \{x',y',z',w',a',b'\}$ that contains a' and/or b' induces an S-cycle of G, then its non-empty intersection with $V_{<ij}$ is not a subset of $C_{<ij}^{x'y',z'w'}$.

- Let $v_1,v_2 \in (X \setminus \{i,j\}) \cup \{x',y',z',w',a'\}$ such that $\langle v_1,v_2,b'\rangle$ is an induced S-triangle of G. Since $x',y',z',w',a',b' \notin S$, without loss of generality, assume that $v_1 \in S \Rightarrow v_1 \in V_{<ij}$. Then

$$\left.\begin{array}{c} v_1,x' <_t y',w',b' \\ y' <_b w' <_b b' <_b v_1 <_b x' \end{array}\right\} \Rightarrow \{v_1,y'\},\{v_1,w'\} \in E.$$

  - If $\{v_1,x'\} \in E$, then $\langle v_1,x',y'\rangle$ is an induced S-triangle of G.
  - If $\{y',w'\} \in E$, then $\langle v_1,y',w'\rangle$ is an induced S-triangle of G.
  - If $\{v_1,x'\},\{y',w'\} \notin E$, then $\langle v_1,y',x',w'\rangle$ is an induced S-square of G.

- Let $v_1,v_2,v_3 \in V_{<ij} \cup \{x',y',z',w',a'\}$ such that $\langle v_1,v_2,v_3,b'\rangle$ is an S-square of G. Assuming that $v_1 \in S$ or $v_3 \in S$, one can as above show that the S-vertex is not an element of $C_{<ij}^{x'y',z'w'}$. Assume that $v_2 \in S$. Since $x',y',z',w',a',b' \notin S$, $v_2 \in V_{<ij}$. Then

$$\left.\begin{array}{c} v_1,v_3 <_t v_2 <_b y',b' \\ v_2,y' <_b b' <_b v_1,v_3 \end{array}\right\} \Rightarrow \{v_1,y'\},\{v_3,y'\} \in E.$$

  - If $\{v_2,y'\} \in E$, then $\langle v_1,v_2,y'\rangle$ is an induced S-triangle of G. If $v_1 = a'$, then

$$\left.\begin{array}{c} x' <_t a' <_t v_2 <_t y' \\ y' <_b v_2 <_b x',a' \end{array}\right\} \Rightarrow \{v_2,x'\} \in E.$$

    Consequently, $\langle v_2,x',y'\rangle$ is an induced S-triangle of G.

– If $\{v_2, y'\} \notin E$, then $\langle v_1, v_2, v_3, y' \rangle$ is an induced S-square of G. If, without loss of generality, $v_1 = a'$, then

$$\left.\begin{array}{c} x' <_t < z' <_t a' <_t v_2 <_t y' \\ v_2 <_b y <_b x', z', a' \end{array}\right\} \Rightarrow \{v_2, x'\}, \{v_2, z'\} \in E.$$

Consequently, if $\{x', z'\} \in E$, then $\langle v_2, x', z' \rangle$ is an induced S-triangle of G and if $\{x', z'\} \notin E$, then $\langle v_2, x', y', z' \rangle$ is an induced S-square of G.

- One can as above show that if a subset of $V_{<ij} \cup \{x', y', z', w', a', b'\}$ that contains a' induces an S-cycle of G, then its non-empty intersection with $V_{<ij}$ is not a subset of $C_{<ij}^{x'y',z'w'}$.

Therefore, if a subset of $V_{<ij} \cup \{x', y', z', w', a', b'\}$ that contains a' and/or b' induces an S-cycle of G, then its non-empty intersection with $V_{<ij}$ is not a subset of $C_{<ij}^{x'y',z'w'}$. By this fact, (i) and Definition 3.2.7, it follows that

$$\left.\begin{array}{l} C_{ij}^{xy,zw} \setminus \{i,j\} \in \mathcal{C}_{<ij}^{x'y',z'w'} \\ C_{<ij}^{x'y',z'w'} \cup \{i,j\} \in \mathcal{C}_{ij}^{xy,zw} \end{array}\right\} \Rightarrow \left.\begin{array}{l} w(C_{ij}^{xy,zw} \setminus \{i,j\}) \leq w(C_{<ij}^{x'y',z'w'}) \\ w(C_{<ij}^{x'y',z'w'} \cup \{i,j\}) \leq w(C_{ij}^{xy,zw}) \end{array}\right\} \Rightarrow$$

$$\Rightarrow w(C_{ij}^{xy,zw}) = w(C_{<ij}^{x'y',z'w'} \cup \{i,j\}) \Rightarrow C_{ij}^{xy,zw} = C_{<ij}^{x'y',z'w'} \cup \{i,j\}. \quad \square$$

Theorem 3.2.12. *The dynamic programming algorithm shown in Figure 3.2 returns an S-feedback vertex set of G that has minimum weight in $O(m^3)$ time.*

Proof. The correctness of the algorithm follows from Lemmas 3.2.2–3.2.11. The computation of $\mathcal{L}$, $\mathcal{R}$ and all predecessors of Definition 3.2.1 takes $\mathrm{O}(n^2 m)$ time. The computation of a single A-set, B-set or C-set takes constant time. The number of iterations performed is

$$\mathrm{O}\left(\sum_{ij \in \mathcal{X}} \left(1 + \sum_{xy \in \mathcal{X}} \left(1 + \sum_{zw \in \mathcal{X}} 1\right)\right)\right) = \mathrm{O}(m^3).$$

Therefore, the total time complexity of the algorithm is $\mathrm{O}(m^3)$. $\square$

compute a list $\mathcal{L}$ containing all ordered crossing vertex pairs of $\mathcal{X}$
    sorted in descending order with respect to $<_l^{lex}$,
    a list $\mathcal{R}$ containing all ordered crossing vertex pairs of $\mathcal{X}$
    sorted in ascending order with respect to $<_r^{lex}$
    and all predecessors of Definition 3.2.1;

for ij **in** $\mathcal{R}$
  compute A_{ij} according to Lemmas 3.2.2 and 3.2.7;
  for xy **in** reverse $\mathcal{L}$
    if $x \in V_{ij}$ or $y \in V_{ij}$ **then continue**;
    compute B_{ij}^{xy} according to Lemmas 3.2.3, 3.2.4, 3.2.8 and 3.2.9;
    if $x \neq y$
      for zw **in** reverse $\mathcal{L}$ starting from $zw := xy$
        if not all the conditions of the definition of $C_{ij}^{xy,zw}$ are met **then continue**;
        compute $C_{ij}^{xy,zw}$ according to Lemmas 3.2.5, 3.2.6, 3.2.10 and 3.2.11;
      end for
    end if
  end for
end for

return $V \setminus A_{\pi(n)n}$;

Figure 3.2: Our dynamic programming algorithm for solving SFVS on a permutation graph.

Chapter 4

Concluding Remarks

4.1 The New State of FVS and SFVS

FVS is a well-known and well-studied problem of Algorithmic Graph Theory. It is $\mathcal{NP}$-complete on general graphs, planar graphs, bipartite graphs and planar bipartite graphs. The list of graph classes on which it is $\mathcal{P}$ includes interval graphs, permutation graphs, trapezoid graphs, cocomparability graphs and convex bipartite graphs, AT-free graphs, chordal graphs and graphs of bounded cliquewidth. In Chapter 2, we proposed novel dynamic programming algorithms for solving FVS on interval graphs and permutation graphs that excibited the same time complexity as their respective best known counterparts found in literature. Figure 4.1 is an updated version of Figure 1.4 that includes our aforementioned results.

In this thesis we studied SFVS, a generalization of FVS that is not as well-studied as FVS. The fact that SFVS is a generalization of FVS implies that it is $\mathcal{NP}$-complete on general graphs, planar graphs, bipartite graphs and planar bipartite graphs as well. The first significant difference in the behaviour of the two problems is that, unlike FVS which is $\mathcal{P}$ on chordal graphs, SFVS was shown to be $\mathcal{NP}$-complete on split graphs, a subclass of chordal graphs. Also unlike FVS, there is no polynomial result where the input is restricted to graph classes regarding SFVS to be found in literature. In Chapter 3, we proposed novel dynamic programming algorithms for solving SFVS on interval graphs in $\mathrm{O}(n + m + l)$ time where $l \in \mathrm{O}(n^3)$ is the number of triangles in the input graph and on permutation graphs in $\mathrm{O}(m^3)$ time—the first polynomial results regarding SFVS. Figure 4.2 is an updated version of Figure 1.5 that includes

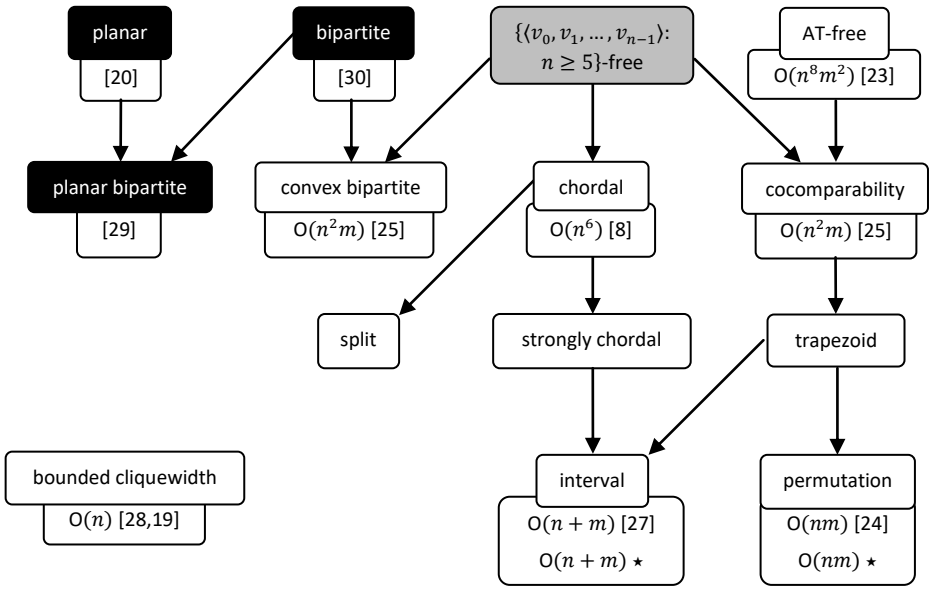


Figure 4.1: Best known results regarding FVS on the listed graph classes. Our results are indicated by a star ($\star$).

our aforementioned results.

4.2 Future Work and Open Problems

Trapezoid graphs are a superclass of permutation graphs whose intersection model naturally generalizes the intersection model of permutation graphs. Thus, we believe that, by defining an appropriate notion of ordered crossing vertex pairs, our novel dynamic programming approach can be adjusted to operate on trapezoid graphs and we will be pursuing this in the immediate future.

Graph classes lying on the hierarchy path from permutation graphs to AT-free graphs are not the only ones on which FVS behaviour is known and SFVS behaviour remains unknown, however. For example, even though SFVS was shown to be $\mathcal{NP}$-complete on split graphs, another well-studied subclass of chordal graphs is the one of strongly chordal graphs. Is SFVS $\mathcal{NP}$-complete on strongly chordal graphs as well or can a polynomial algorithm solving it on them be found?

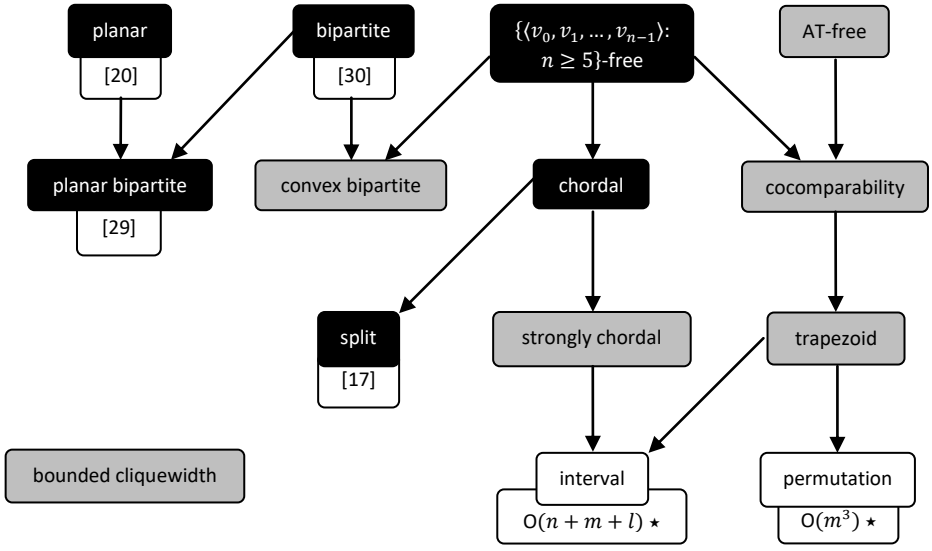


Figure 4.2: Best known results regarding SFVS on the listed graph classes. Our results are indicated by a star ($\star$).

Graphs of bounded cliquewidth are yet another graph class that is worth investigating. FVS is $\mathcal{P}$ on graphs of bounded cliquewidth. Many classical graph classes that we do not mention in this thesis are classes of graphs of bounded cliquewidth. This implies that FVS is $\mathcal{P}$ on all those graph classes. Is SFVS $\mathcal{P}$ on graphs of bounded cliquewidth as well? If not, is it perhaps $\mathcal{P}$ on graphs of a bounded graph parameter stricter than cliquewidth?

Appendix A

Miscellaneous Mathematical Concepts

partition

A *partition* of a set S is a collection of pairwise disjoint subsets of S that its union is S.

permutation

A *permutation* of a set S is a one-to-one correspondence of elements of S to elements of S. A permutation is commonly presented as a $2 \times |S|$ matrix where for each column the permutation maps the element of the first row to the element of the second row. For example, the permutation π of $\{1, 2, 3, 4, 5, 6, 7, 8\}$ where $\pi(1) = 2$, $\pi(2) = 8$, $\pi(3) = 6$, $\pi(4) = 3$, $\pi(5) = 1$, $\pi(6) = 7$, $\pi(7) = 4$ and $\pi(8) = 5$ is

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 2 & 8 & 6 & 3 & 1 & 7 & 4 & 5 \end{pmatrix}.$$

relation, partial order, total order, linear extention

A *(binary) relation* on a set S is a set $\mathcal{R} \subseteq S^2$. We commonly denote an inclusion in a relation as we would denote an action of a binary operation; that is, for a relation $\mathcal{R}$ on S, we consider $x\mathcal{R}y$ to be equivalent to $(x, y) \in \mathcal{R}$ for all $x, y \in S$.

A relation $\mathcal{R}$ on a set S may satisfy any number of the properties listed below:

- $\forall\ x \in S : x\mathcal{R}x$ (*reflexivity*)
- $\forall\ x \in S : \neg x\mathcal{R}x$ (*irreflexivity*)
- $\forall\ x, y, z \in S : (x\mathcal{R}y \wedge y\mathcal{R}z) \rightarrow x\mathcal{R}z$ (*transitivity*)
- $\forall\ x, y \in S : (x\mathcal{R}y \wedge y\mathcal{R}x) \rightarrow x = y$ (*antisymmetry*)
- $\forall\ x, y \in S : x\mathcal{R}y \vee y\mathcal{R}x$ (*totality*)

We say that two elements x, y of S for which $x\mathcal{R}y \vee y\mathcal{R}x$ holds are *comparable* (with respect to $\mathcal{R}$).

A *partial order* on a set S is a relation on S that is reflexive, transitive and antisymmetric. A *total order* on S is a partial order on S that is also total.

A *linear extention* of a partial order $\mathcal{R}$ on a set S is a total order $\mathcal{L}$ on S such that

$$x\mathcal{R}y \implies x\mathcal{L}y$$

for all $x, y \in S$.

Bibliography

[1] Bar-Yehuda, R., Geiger, D., Naor, J. S., 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] Brandstädt, A. *On improved time bounds for permutation graph problems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1993, pp. 1–10.

[4] Brandstädt, A., and Kratsch, D. *On the restriction of some NP-complete graph problems to permutation graphs*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1985, pp. 53–62.

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

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

[7] Chen, J., Fomin, F. V., Liu, Y., Lu, S., and Villanger, Y. Improved algorithms for feedback vertex set problems. *Journal of Computer and System Sciences 74*, 7 (2008), 1188 – 1198.

[8] Corneil, D., and Fonlupt, J. The complexity of generalized clique covering. *Discrete Applied Mathematics 22*, 2 (1988), 109 – 118.

[9] Courcelle, B., Engelfriet, J., and Rozenberg, G. Handle-rewriting hypergraph grammars. *Journal of Computer and System Sciences 46*, 2 (1993), 218 – 270.

[10] Courcelle, B., and Olariu, S. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics 101*, 1–3 (2000), 77 – 114.

[11] Cygan, M., Pilipczuk, M., Pilipczuk, M., and Wojtaszczyk, J. O. *Subset Feedback Vertex Set Is Fixed-Parameter Tractable.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 449–461.

[12] Dechter, R. Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. *Artif. Intell. 41*, 3 (Jan. 1990), 273–312.

[13] Dechter, R., and Pearl, J. *The cycle-cutset method for improving search performance in AI applications.* Orlando, FL, 1987, pp. 224–230.

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

[15] Festa, P., Pardalos, P. M., and Resende, M. G. *Feedback Set Problems.* Springer US, Boston, MA, 2009, pp. 1005–1016.

[16] Fomin, F. V., Gaspers, S., Pyatkin, A. V., and Razgon, I. On the minimum feedback vertex set problem: Exact and enumeration algorithms. *Algorithmica 52*, 2 (2008), 293–307.

[17] Fomin, F. V., Heggernes, P., Kratsch, D., Papadopoulos, C., and Villanger, Y. *Enumerating Minimal Subset Feedback Vertex Sets.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 399–410.

[18] Fried, C., Hordijk, W., Prohaska, S. J., Stadler, C. R., , and Stadler, P. F. The footprint sorting problem. *Journal of Chemical Information and Computer Sciences 44*, 2 (2004), 332–338. PMID: 15032508.

[19] Ganian, R., and Hliněný, P. On parse trees and myhill–nerode-type tools for handling graphs of bounded rank-width. *Discrete Applied Mathematics 158*, 7 (2010), 851 – 867. Third Workshop on Graph Classes, Optimization, and Width Parameters Eugene, Oregon, USA, October 2007.

[20] Garey, M. R., and Johnson, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA, 1979.

[21] Karp, R. M. *Reducibility among Combinatorial Problems.* Springer US, Boston, MA, 1972, pp. 85–103.

[22] Kleinberg, J., and Kumar, A. Wavelength conversion in optical networks. *Journal of Algorithms 38*, 1 (2001), 25 – 50.

[23] Kratsch, D., Müller, H., and Todinca, I. Feedback vertex set on at-free graphs. *Discrete Applied Mathematics 156*, 10 (2008), 1936 – 1947.

[24] Liang, Y. D. On the feedback vertex set problem in permutation graphs. *Information Processing Letters 52*, 3 (1994), 123 – 129.

[25] 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.

[26] Liang, Y. D., and Rhee, C. Linear algorithms for two independent set problems in permutation graphs. In *Proceedings of the 22Nd Annual ACM Computer Science Conference on Scaling Up : Meeting the Challenge of Complexity in Real-world Computing Applications* (New York, NY, USA, 1994), CSC '94, ACM, pp. 90–93.

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

[28] Oum, S., and Seymour, P. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B 96*, 4 (2006), 514 – 528.

[29] Takaoka, A., Tayu, S., and Ueno, S. On minimum feedback vertex sets in graphs. In *Networking and Computing (ICNC), 2012 Third International Conference on* (Dec 2012), pp. 429–434.

[30] Yannakakis, M. Node-deletion problems on bipartite graphs. *SIAM Journal on Computing 10*, 2 (1981), 310–327.