Χαρακτηρισμός Νέων Κλάσεων Τέλειων Γραφημάτων και Αλγόριθμοι Χρωματισμού και Μέγιστων Διαδρομών

Η ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης

του Τμήματος Πληροφορικής Εξεταστική Επιτροπή

από την

Κυριακή Ιωαννίδου

ως μέρος των Υποχρεώσεων για τη λήψη του

ΔΙΔΑΚΤΟΡΙΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ

ΠΛΗΡΟΦΟΡΙΚΗ

Δεκέμβριος 2009

Τριμελής Συμβουλευτική Επιτροπή

- Σταύρος Δ. Νικολόπουλος, Καθηγητής του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων (επιβλέπων)
- Χρήστος Κακλαμάνης, Καθηγητής του Τμήματος Μηχανικών Η/Υ & Πληροφορικής του Πανεπιστημίου Πατρών
- Λεωνίδας Παληός, Αναπληρωτής Καθηγητής του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων

Επταμελής Εξεταστική Επιτροπή

- Σταύρος Δ. Νικολόπουλος, Καθηγητής του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων (επιβλέπων)
- Χρήστος Κακλαμάνης, Καθηγητής του Τμήματος Μηχανικών Η/Υ & Πληροφορικής του Πανεπιστημίου Πατρών
- Λεωνίδας Παληός, Αναπληρωτής Καθηγητής του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων
- Ευστάθιος Ζάχος, Καθηγητής της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου
- Βασίλειος Ζησιμόπουλος, Καθηγητής του Τμήματος Πληροφορικής & Τηλεπικοινωνιών του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών
- Σπυρίδωνας Κοντογιάννης, Λέκτορας του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων
- Χρήστος Νομικός, Επίκουρος Καθηγητής του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων

Ευχαριστιες

Θα ήθελα να απευθύνω τις θερμότερες ευχαριστίες μου στον επιβλέποντα καθηγητή μου κ. Σταύρο Δ. Νικολόπουλο για την πολύτιμη βοήθεια και καθοριστική συμβολή του στην ολοκλήρωση της διδακτορικής μου διατριβής. Θα ήθελα να του εκφράσω την ευγνωμοσύνη μου, αφού χάρη στην επιστημονική του γνώση, την συνεχή καθοδήγηση, και την αμέριστη υποστήριξή του, αυτή η εργασία ολοκληρώθηκε επιτυχώς. Επιπλέον, τον ευχαριστώ θερμά γιατί με την αυστηρή, και ταυτόχρονα ενθουσιώδη και συνεχή προτροπή του μου δίδαξε να βαδίζω "σωστά" στα μονοπάτια του "όμορφου" κόσμου της αλγοριθμικής θεωρίας γραφημάτων.

Θα ήθελα να ευχαριστήσω τον Καθηγητή κ. Χρήστο Κακλαμάνη και τον Αναπληρωτή Καθηγητή κ. Λεωνίδα Παληό (μέλη της τριμελούς συμβουλευτικής επιτροπής) για την βοήθεια, τις συμβουλές και τον χρόνο που μου αφιέρωσαν. Οι συμβουλές τους ήταν καθοριστικές σε όλα τα ερευνητικά θέματα που μελετήσαμε. Επίσης, θα ήθελα να ευχαριστήσω τους κκ. Ευστάθιο Ζάχο, Βασίλειο Ζησιμόπουλο, Χρήστο Νομικό, και Σπυρίδωνα Κοντογιάννη (μέλη της επταμελούς εξεταστικής επιτροπής) για την επιστημονική συμβολή τους στην εκπόνηση της διατριβής μου.

Επιπλέον, θα ήταν παράλειψη να μην αναφερθώ στην υλικοτεχνική υποδομή, καθώς και στο φιλικό και ευχάριστο περιβάλλον, που μου παρείχε το Τμήμα Πληροφορικής του Πανεπιστημίου Ιωαννίνων κατά τη διάρκεια των μεταπτυχιακών μου σπουδών.

Τέλος, θα ήθελα να ευχαριστήσω θερμά τους γονείς μου, οι οποίοι με την αμέριστη συμπαράσταση και ηθική υποστήριξή τους έκαναν εφικτή την ολοκλήρωση των σπουδών μου.

Κυριακή Ιωαννίδου

Ιωάννινα, Δεκέμβριος 2009

Το έργο συγχρηματοδοτείται

- 80 % της Δημόσιας Δαπάνης από την Ευρωπαϊκή Ένωση - Ευρωπαϊκό Κοινωτικό Ταμείο
- 20 % της Δημόσιας Δαπάνης από το Ελληνικό Δημόσιο - Υπουργείο Ανάπτυξης - Γενική Γραμματεία Έρευνας και Τεχνολογίας
- και από τον Ιδιωτικό Τομέα

στο πλαίσιο του Μέτρου 8.3 του Ε.Π. Ανταγωνιστηκότητα - Γ Κοινοτικό Πλαίσιο Στήριξης.

Contents

List of Figures

List of Algorithms

Abstract

In this work we provide characterizations of two new classes of perfect graphs, namely colinear and linear graphs. Moreover, we present polynomial-time algorithms or NP-completeness results for various types of the coloring problem on graphs and polynomial-time algorithms for the longest path problem on perfect graphs.

In particular, motivated by the definition of linear coloring on simplicial complexes, recently introduced in the context of algebraic topology, we introduce the colinear coloring on graphs, and propose a polynomial algorithm for colinearly coloring any graph G. Based on the colinear coloring, we define the χ-colinear and α-colinear properties and characterize known graph classes in terms of these properties. In the sequence, we study those graphs which are characterized completely by the χ-colinear or α-colinear property, and conclude that these graphs form two new classes of perfect graphs, which we call colinear and linear graphs. We provide characterizations for colinear and linear graphs and prove structural properties. Moreover, we study the harmonious coloring problem on connected permutation graphs and subclasses of co-chordal and chordal graphs, including colinear and interval graphs, and either prove NP-completeness results or provide polynomial algorithms for solving the problem.

Furthermore, we study the longest path problem and we first show that it has a polynomial solution on interval graphs, answering thus a question left open in [63]. Then we extend our results by proposing a polynomial time algorithm for solving the longest path problem on cocomparability graphs, resolving the open question for the status of the longest path problem on cocomparability graphs, and also on permutation graphs. We next describe the problems that we study in this work.

Colinear Coloring and Colinear Graphs

Motivated by the definition of linear coloring on simplicial complexes, recently introduced in the context of algebraic topology, and the framework through which it was studied, we introduce the colinear coloring on graphs. The colinear coloring of a graph G is a vertex coloring such that two vertices can be assigned the same color, if their corresponding clique sets are associated by the set inclusion relation (a clique set of a vertex u is the set of all maximal cliques containing u); the colinear chromatic number $\lambda(G)$ of G is the least integer k for which G admits a colinear coloring with k colors. We prove that for any graph G, $\lambda(\overline{G}) \geq \chi(G)$, providing thus an upper bound for the chromatic number $\chi(G)$ of G, and show that any graph G can be colinearly colored in polynomial time by proposing a simple algorithm. Based on the colinear coloring, we define the χ-colinear and α-colinear properties and characterize known graph classes in terms of these properties.

Based on these results and the definition of perfect graphs (a graph G is perfect if and only if $\chi(G_A) = \omega(G_A), \forall A \subseteq V(G)$, where $\omega(G)$ is the clique number of G [37]), we study those graphs which are characterized completely by the χ-colinear or the α-colinear property, and conclude that these graphs form two new classes of perfect graphs, which we call colinear and linear graphs. A graph G is called colinear if and only if $\chi(G_A) = \lambda(\overline{G}_A), \forall A \subseteq V(G)$. A graph G is called linear if and only if $\alpha(G_A) = \lambda(G_A), \forall A \subseteq V(G)$; note that $\alpha(G)$ is the stability number of G. We provide characterizations for colinear and linear graphs and prove structural properties in terms of forbidden induced subgraphs. An interesting question for future work would be to study structural and recognition properties of colinear and linear graphs and see whether they can be characterized by a finite set of forbidden induced subgraphs. Moreover, an

obvious though interesting open question would be whether combinatorial and/or optimization problems can be efficiently solved on the classes of colinear and linear graphs.

This study lead to the following publications:

- K. Ioannidou and S.D. Nikolopoulos, Colinear coloring on graphs, *3rd Annual Workshop on Algorithms and Computation (WALCOM'09)*, LNCS **5431** (2009) 117–128.
- K. Ioannidou and S.D. Nikolopoulos, Colinear coloring and colinear graphs, *Technical Report TR-2007-06*, Department of Computer Science, University of Ioannina, 2007 (submitted to journal).

The Harmonious Coloring Problem

A harmonious coloring of a simple graph G is a proper vertex coloring such that each pair of colors appears together on at most one edge. The harmonious chromatic number is the least integer k for which G admits a harmonious coloring with k colors. Extending previous work on the NP-completeness of the harmonious coloring problem when restricted to the class of disconnected graphs which are simultaneously cographs and interval graphs [8], we prove that the problem is also NP-complete for connected interval and permutation graphs. We also show that the harmonious coloring problem is NP-complete on split graphs.

In the sequence we extend our results for the harmonious coloring problem on subclasses of chordal and co-chordal graphs, by proving that the problem remains NP-complete for split undirected path graphs; we also prove that the problem is NP-complete for colinear graphs by showing graph class inclusion relations. Moreover, we provide a polynomial solution for the harmonious coloring problem for the class of split strongly chordal graphs, the interest of which lies on the fact that the problem is NP-complete on both split and strongly chordal graphs. In addition, polynomial solutions for the problem are only known for the classes of threshold graphs and connected quasi-threshold graphs; note that, the harmonious coloring problem is NP-complete on disconnected quasi-threshold graphs. Since linear graphs form a superclass of both quasi-threshold graphs and split strongly chordal graphs, the harmonious coloring problem is NP-complete on disconnected linear graphs, while it still remains open on connected linear graphs.

This work lead to the following publications:

- K. Asdre, K. Ioannidou, S.D. Nikolopoulos, The harmonious coloring problem is NP-complete for interval and permutation graphs, *Discrete Applied Math.* **155** (2007) 2377–2382.
- K. Ioannidou and S.D. Nikolopoulos, Harmonious coloring on subclasses of colinear graphs, *4rd Annual Workshop on Algorithms and Computation (WALCOM'10)*, accepted.

The Longest Path Problem

The longest path problem is the problem of finding a path of maximum length in a graph. A well studied problem in graph theory with numerous applications is the Hamiltonian path problem, i.e., the problem of determining whether a graph is Hamiltonian; a graph is said to be Hamiltonian if it contains a Hamiltonian path, that is, a simple path in which every vertex of the graph appears exactly once. Even if a graph is not Hamiltonian, it makes sense in several applications to search for a longest path, or equivalently, to find a maximum induced subgraph of the graph which is Hamiltonian. However, finding a longest path seems to be more difficult than deciding whether or not a graph admits a Hamiltonian path. The longest path problem is NP-hard on every class of graphs on which the Hamiltonian path problem is NP-complete. In contrast to the Hamiltonian path problem, there are few known polynomial time solutions for the longest path problem, and these restrict to trees and some small graph classes. We show that the longest path problem on interval graphs has a polynomial solution, thus, answering the question left open by Uehara and Uno in [63]. In particular, the proposed algorithm runs in $O(n^4)$ time, where n is the number of vertices of the input graph, and bases on a dynamic programming approach.

Moreover, we study the longest path problem on the class of cocomparability graphs, a well-known class of perfect graphs which includes both interval and permutation graphs. Although the Hamiltonian path problem on cocomparability graphs was proved to be polynomial almost two decades ago [23], the complexity status of the longest path problem on cocomparability graphs has remained open until now; actually, the complexity status of the longest path problem has been open even on the more special class of permutation graphs. In this work, we present a polynomial-time algorithm for solving the longest path problem on the class of cocomparability graphs. This result extends our polynomial solution of the longest path problem on interval graphs, and resolves the open question for the complexity of the problem on cocomparability graphs, and thus on permutation graphs.

This work lead to the following publications:

- K. Ioannidou, G.B. Mertzios, and S.D. Nikolopoulos, The longest path problem is polynomial on interval graphs, *34th International Symposium on Mathematical Foundations of Computer Science (MFCS'09)*, LNCS **5734** (2009) 403–414.
- K. Ioannidou and S.D. Nikolopoulos, The longest path problem is polynomial on cocomparability graphs, *Technical Report TR-2009-28*, Department of Computer Science, University of Ioannina, 2009 (submitted to journal).

Εκτεταμενη Περιληψη στα Ελληνικα

Σε αυτή την εργασία δίνουμε χαρακτηρισμούς για δύο νέες κλάσεις τέλειων γραφημάτων, τις οποίες ονομάζουμε colinear και linear γραφήματα. Επιπλέον, παρουσιάζουμε πολυωνυμικούς αλγορίθμους ή αποτελέσματα NP-πληρότητας για διάφορα προβλήματα χρωματισμού και πολυωνυμικούς αλγορίθμους για το πρόβλημα μέγιστων διαδρομών σε τέλεια γραφήματα.

Συγκεκριμένα, παρακινούμενοι από την έννοια του linear χρωματισμού σε simplicial complexes, που ορίστηκε πρόσφατα στα πλαίσια της αλγεβρικής τοπολογίας, ορίζουμε τον colinear χρωματισμό πάνω σε γραφήματα, και προτείνουμε ένα πολυωνυμικό αλγόριθμο που χρωματίζει colinearly κάθε γράφημα G. Βασιζόμενοι στον colinear χρωματισμό, ορίζουμε τις χ-colinear και α-colinear ιδιότητες και χαρακτηρίζουμε γνωστές κλάσεις γραφημάτων ως προς αυτές τις ιδιότητες. Στη συνέχεια, μελετούμε εκείνα τα γραφήματα τα οποία χαρακτηρίζονται πλήρως απο τη χ-colinear ή την α-colinear ιδιότητα, και συμπεραίνουμε ότι αυτά τα γραφήματα συνιστούν δύο νέες κλάσεις τέλειων γραφημάτων, τις οποίες ονομάζουμε colinear και linear γραφήματα. Προτείνουμε χαρακτηρισμούς για τα colinear και linear γραφήματα και αποδεικνύουμε δομικές ιδιότητες. Πέραν τούτου, μελετούμε το πρόβλημα αρμονικού χρωματισμού (harmonious coloring problem) πάνω σε συνεκτικά μεταθετικά γραφήματα (permutation graphs) και σε υποκλάσεις των τριγωνικών (chordal) και συμπληρωματικών τριγωνικών (co-chordal) γραφημάτων, συμπεριλαμβανομένων των colinear γραφημάτων και γραφημάτων διαστημάτων (interval graphs), και είτε αποδεικνύουμε αποτελέσματα NP-πληρότητας είτε προτείνουμε πολυωνυμικούς αλγορίθμους για επίλυση του προβλήματος.

Επιπλέον, μελετούμε το πρόβλημα μέγιστων διαδρομών (longest path problem) και αρχικά αποδεικνύουμε ότι επιδέχεται πολυωνυμική λύση στην κλάση των γραφημάτων διαστημάτων, απαντώντας έτσι ένα ερώτημα που αφέθηκε ανοικτό στο [63]. Έπειτα επεκτείνουμε τα αποτελέσματά μας προτείνοντας πολυωνυμικό αλγόριθμο επίλυσης του προβλήματος μέγιστων διαδρομών στα συμπληρωματικά μεταβατικά (cocomparability) γραφήματα, επιλύωντας έτσι το ανοικτό ερώτημα για την πολυπλοκότητα του προβλήματος μέγιστων διαδρομών στα συμπληρωματικά μεταβατικά γραφήματα, και επίσης στα μεταθετικά γραφήματα. Στη συνέχεια περιγράφουμε τα προβλήματα που μελετούμε σε αυτή την εργασία.

Colinear Χρωματισμός και Colinear Γραφήματα

Παρακινούμενοι από την έννοια του linear χρωματισμού σε simplicial complexes, που ορίστηκε πρόσφατα στα πλαίσια της αλγεβρικής τοπολογίας, καθώς και το θεωρητικό πλαίσιο στο οποίο μελετήθηκε, ορίζουμε τον colinear χρωματισμό πάνω σε γραφήματα. Ο colinear χρωματισμός ενός γραφήματος G είναι ένας χρωματισμός των κόμβων του G τέτοιος ώστε σε δύο κόμβους μπορεί να ανατεθεί το ίδιο χρώμα, εάν τα αντίστοιχα σύνολα κλικών τους σχετίζονται με τη σχέση του υποσυνόλου (το σύνολο κλικών ενός κόμβου u είναι το σύνολο όλων των μείζονων κλικών που περιέχουν τον u). Ο colinear χρωματικός αριθμός $\lambda(G)$ του G είναι ο ελάχιστος ακέραιος αριθμός k για τον οποίο το G επιδέχεται ένα colinear χρωματισμό με k χρώματα. Αποδεικνύουμε ότι για κάθε γράφημα G, $\lambda(\overline{G}) \geq \chi(G)$, δίνοντας έτσι ένα πάνω φράγμα για το χρωματικό αριθμό $\chi(G)$ του G, και δείχνουμε ότι κάθε γράφημα G μπορεί να χρωματιστεί colinearly σε πολυωνυμικό χρόνο προτείνοντας ένα απλό αλγόριθμο. Βασιζόμενοι στον colinear χρωματισμό, ορίζουμε τις χ-colinear και α-colinear ιδιότητες και χαρακτηρίζουμε γνωστές κλάσεις γραφημάτων ως προς αυτές τις ιδιότητες.

Παρακινούμενοι από αυτά τα αποτελέσματα και από τον ορισμό των τέλειων γραφημάτων (ένα γράφημα

G είναι τέλειο εάν και μόνο εάν $\chi(G_A) = \omega(G_A)$, $\forall A \subseteq V(G)$, όπου $\omega(G)$ είναι ο αριθμός κλίκας (clique number) του G [37]), μελετούμε εκείνα τα γραφήματα τα οποία χαρακτηρίζονται πλήρως απο τη χ-colinear ή την α-colinear ιδιότητα, και συμπεραίνουμε ότι αυτά τα γραφήματα συνιστούν δύο καινούριες κλάσεις τέλειων γραφημάτων, τις οποίες ονομάζουμε colinear και linear γραφήματα. Ένα γράφημα ονομάζεται colinear εάν και μόνο εάν $\chi(G_A) = \lambda(\overline{G}_A)$, $\forall A \subseteq V(G)$. Ένα γράφημα G ονομάζεται linear εάν και μόνο εάν $\alpha(G_A) = \lambda(G_A)$, $\forall A \subseteq V(G)$. Σημειώνουμε ότι $\alpha(G)$ είναι ο ευσταθής αριθμός (stability number) toy G. Προτείνουμε χαρακτηρισμούς για τα colinear και linear γραφήματα και αποδεικνύουμε δομικές ιδιότητες ως προς απαγορευμένα επαγόμενα γραφήματα. Ένα ενδιαφέρον ερώτημα για μελλοντική έρευνα θα ήταν η μελέτη δομικών ιδιοτήτων και ιδιοτήτων αναγνώρισης των colinear και linear γραφημάτων, καθώς και το ερώτημα εάν μπορούν να χαρακτηριστούν από ένα πεπερασμένο σύνολο απαγορευμένων επαγόμενων γραφημάτων. Επιπρόσθετα, ένα προφανές αλλά ενδιαφέρον ανοικτό ερώτημα θα ήταν εάν συνδυαστικά προβλήματα ή/και προβλήματα βελτιστοποίησης μπορούν να επιλυθούν αποτελεσματικά πάνω στις κλάσεις των colinear και linear γραφημάτων.

Αυτή η μελέτη οδήγησε στις παρακάτω εργασίες:

- K. Ioannidou and S.D. Nikolopoulos, Colinear coloring on graphs, *3rd Annual Workshop on Algorithms and Computation (WALCOM'09)*, LNCS **5431** (2009) 117–128.
- K. Ioannidou and S.D. Nikolopoulos, Colinear coloring and colinear graphs, *Technical Report TR-2007-06*, Department of Computer Science, University of Ioannina, 2007 (submitted to journal).

Το πρόβλημα Αρμονικού Χρωματισμού

Ο αρμονικός χρωματισμός ενός απλού γραφήματος G είναι ένας κανονικός χρωματισμός των κόμβων του τέτοιος ώστε κάθε ζεύγος χρωμάτων εμφανίζεται σε το πολύ μια ακμή. Ο αρμονικός χρωματικός αριθμός είναι ο ελάχιστος ακέραιος αριθμός k για τον οποίο το γράφημα G επιδέχεται ένα αρμονικό χρωματισμό με k χρώματα. Επεκτείνοντας προηγούμενα αποτελέσματα NP-πληρότητας του προβλήματος αρμονικού χρωματισμού πάνω στις κλάσεις των μη-συνεκτικών γραφημάτων που είναι ταυτόχρονα συμπληρωματικά παραγόμενα γραφήματα (cographs) και γραφήματα διαστημάτων [8], αποδεικνύουμε ότι το πρόβλημα παραμένει NP-πλήρες για συνεκτικά γραφήματα διαστημάτων και συνεκτικά μεταθετικά γραφήματα. Επιπλέον, δείχνουμε ότι το πρόβλημα αρμονικού χρωματισμού είναι NP-πλήρες και για split γραφήματα.

Στη συνέχεια επεκτείνουμε τα αποτελέσματα μας για το πρόβλημα αρμονικού χρωματισμού πάνω σε υποκλάσεις των τριγωνικών και συμπληρωματικών τριγωνικών γραφημάτων, αποδεικνύοντας ότι το πρόβλημα παραμένει NP-πλήρες για split undirected path γραφήματα. Δείχνουμε επίσης ότι το πρόβλημα είναι NP-πλήρες για colinear γραφήματα αποδεικνύοντας ότι τα split undirected path γραφήματα είναι υποκλάση των colinear γραφημάτων. Πέραν τούτου, δίνουμε μια πολυωνυμική λύση για το πρόβλημα αρμονικού χρωματισμού για την κλάση των split strongly chordal γραφημάτων, το ενδιαφέρον της οποίας έγκειται στο γεγονός ότι το πρόβλημα είναι NP-πλήρες για split γραφήματα, καθώς και για strongly chordal γραφήματα. Επιπρόσθετα, πολυωνυμικές λύσεις για το πρόβλημα είναι γνωστές μόνο για τις κλάσεις των threshold γραφημάτων και των συνεκτικών quasi-threshold γραφημάτων. Σημειώνουμε ότι το πρόβλημα αρμονικού χρωματισμού είναι NP-πλήρες για μη-συνεκτικά quasi-threshold γραφήματα. Εφόσον τα linear γραφήματα αποτελούν υπερκλάση των quasi-threshold γραφημάτων καθώς και των split strongly chordal γραφημάτων, το πρόβλημα αρμονικού χρωματισμού είναι NP-πλήρες για μη-συνεκτικά linear γραφήματα, ενώ παραμένει ανοικτό για συνεκτικά linear γραφήματα.

Αυτή η δουλειά οδήγησε στις παρακάτω δημοσιεύσεις:

- K. Asdre, K. Ioannidou, S.D. Nikolopoulos, The harmonious coloring problem is NP-complete for interval and permutation graphs, *Discrete Applied Math.* **155** (2007) 2377–2382.
- K. Ioannidou and S.D. Nikolopoulos, Harmonious coloring on subclasses of colinear graphs, *4rd Annual Workshop on Algorithms and Computation (WALCOM'10)*, accepted.

Το πρόβλημα Μέγιστων Διαδρομών

Το πρόβλημα μέγιστων διαδρομών είναι το πρόβλημα εύρεσης ενός μονοπατιού μέγιστου μήκους σε ένα γράφημα. Ένα ιδιαίτερα μελετημένο πρόβλημα στη θεωρία γραφημάτων με πολλές εφαρμογές είναι το πρόβλημα εύρεσης Hamiltonian μονοπατιού, δηλαδή το πρόβλημα απόφασης εάν ένα γράφημα είναι Hamiltonian ή όχι. Ένα γράφημα είναι Hamiltonian εάν περιέχει ένα Hamiltonian μονοπάτι, δηλαδή ένα απλό μονοπάτι στο οποίο κάθε κόμβος του γραφήματος εμφανίζεται ακριβώς μια φορά. Ακόμα και αν ένα γράφημα δεν είναι Hamiltonian, έχει νόημα για πολλές εφαρμογές να ψάξουμε για ένα μέγιστο μονοπάτι, ή ισοδύναμα, να βρούμε ένα μέγιστο επαγόμενο υπογράφημα του γραφήματος που είναι Hamiltonian. Εντούτοις, το πρόβλημα εύρεσης ενός μέγιστου μονοπατιού φαίνεται πιο δύσκολο από το πρόβλημα απόφασης εάν ένα γράφημα έχει Hamiltonian μονοπάτι ή όχι. Το πρόβλημα μεγίστων διαδρομών είναι NP-δύσκολο σε κάθε κλάση γραφημάτων για την οποία το πρόβλημα έυρεσης Hamiltonian μονοπατιού είναι NP-πλήρες. Σε αντίθεση με το πρόβλημα έυρεσης Hamiltonian μονοπατιού, υπάρχουν λίγοι γνωστοί πολυωνυμικοί αλγόριθμοι για το πρόβλημα μέγιστων διαδρομών, και αυτοί περιορίζονται στα δένδρα (trees) και σε μικρές κλάσεις γραφημάτων. Αποδεικνύουμε ότι το πρόβλημα μεγίστων διαδρομών επιδέχεται πολυωνυμική λύση στα γραφήματα διαστημάτων, απαντώντας έτσι στο ανοικτό ερώτημα που τέθηκε από τους Uehara and Uno στο [63]. Συγκεκριμένα, ο προτεινόμενος αλγόριθμος έχει πολυπλοκότητα χρόνου $O(n^4)$, όπου n είναι ο αριθμός των κόμβων στο δοθέν γράφημα, και βασίζεται στην τεχνική του δυναμικού προγραμματισμού.

Επιπλέον, μελετούμε το πρόβλημα μέγιστων διαδρομών στη κλάση των συμπληρωματικών μεταβατικών γραφημάτων, μια γνώστη κλάση τέλειων γραφημάτων που περιέχει τα γραφήματα διαστημάτων καθώς και τα μεταθετικά γραφήματα. Αν και το πρόβλημα εύρεσης Hamiltonian μονοπατιού σε συμπληρωματικά μεταβατικά γραφήματα αποδείχθηκε ότι είναι πολυωνυμικής πολυπλοκότητας σχεδόν πριν δύο δεκαετίες [23], η πολυπλοκότητα του προβλήματος μέγιστων διαδρομών στα συμπληρωματικά μεταβατικά γραφήματα παρέμενε άγνωστη μέχρι σήμερα. Κατακρίβειαν, η πολυπλοκότητα του προβλήματος μέγιστων διαδρομών παρέμενε άγνωστη ακόμα και στην πιο μικρή κλάση των μεταθετικών γραφημάτων. Σε αυτή τη δουλειά, παρουσιάζουμε ένα πολυωνυμικό αλγόριθμο επίλυσης του προβλήματος μέγιστων διαδρομών στη κλάση των συμπληρωματικών μεταβατικών γραφημάτων. Αυτό το αποτέλεσμα επεκτείνει την πολυωνυμική λύση που προτείναμε για το πρόβλημα μέγιστων διαδρομών στα γραφήματα διαστημάτων, και απαντά το ανοικτό ερώτημα για την πολυπλοκότητα του προβλήματος μέγιστων διαδρομών στα συμπληρωματικά μεταβατικά γραφήματα, και συνεπώς στα μεταθετικά γραφήματα.

Αυτή η δουλειά οδήγησε στην παρακάτω δημοσίευση:

- K. Ioannidou, G.B. Mertzios, and S.D. Nikolopoulos, The longest path problem is polynomial on interval graphs, *34th International Symposium on Mathematical Foundations of Computer Science (MFCS'09)*, LNCS **5734** (2009) 403–414.

- K. Ioannidou and S.D. Nikolopoulos, The longest path problem is polynomial on cocomparability graphs, *Technical Report TR-2009-28*, Department of Computer Science, University of Ioannina, 2009 (submitted to journal).

Chapter 1

Introduction

1.1 Preliminaries

We consider finite undirected graphs with no loops or multiple edges. For a graph G, we denote its vertex and edge set by $V(G)$ and $E(G)$, respectively. An edge is a pair of distinct vertices $u, v \in V(G)$, and is denoted by uv if G is an undirected graph and by $\overrightarrow{uv}$ if G is a directed graph. We say that the vertex u is adjacent to the vertex v or, equivalently, the vertex u sees the vertex v, if there is an edge uv in G. If $uv \notin E(G)$ then we say that the vertex u misses the vertex v or, equivalently, that vertices u and v are *antineighbors* in G. For a set $A \subseteq V(G)$ of vertices of the graph G, the subgraph of G *induced* by A is denoted by G_A or $G[A]$. Additionally, the cardinality of a set A is denoted by $|A|$. For a given vertex ordering $(v_1, v_2, \ldots, v_n)$ of a graph G, the subgraph of G induced by the set of vertices $\{v_i, v_{i+1}, \ldots, v_n\}$ is denoted by G_i.

The set $N(v) = \{u \in V(G) : uv \in E(G)\}$ is called the *neighborhood* of the vertex $v \in V(G)$ in G, sometimes denoted by $N_G(v)$ for clarity reasons. The set $N[v] = N(v) \cup \{v\}$ is called the *closed neighborhood* of the vertex $v \in V(G)$. The complement $\overline{G}$ of a graph G has the same vertex set as G, and distinct vertices u, v are adjacent in $\overline{G}$ if and only if they are not adjacent in G. Thus, by $N_{\overline{G}}(v)$ we denote the set of the antineighbors of the vertex v in the graph G. The *degree* of a vertex x in a graph G is the number of edges incident on x, and is denoted by $deg(x)$. A *clique* is a set of pairwise adjacent vertices while a *stable* (or *independent*) *set* is a set of pairwise non-adjacent vertices.

A *simple path* (resp. *simple antipath*) of a graph G is a sequence of distinct vertices $v_1, v_2, \ldots, v_k$ such that $v_i v_{i+1} \in E(G)$ (resp. $v_i v_{i+1} \notin E(G)$), for each i, $1 \leq i \leq k-1$, and is denoted by $(v_1, v_2, \ldots, v_k)$; throughout the paper all paths (resp. antipaths) considered are simple. We denote by $V(P)$ the set of vertices in the path (resp. antipath) P. We define the *length* of the path (resp. antipath) P to be the number of vertices in P, i.e., $|P| = |V(P)|$; there are case where we consider the length of a path P to be equal to the number of edges in P, and we explicitly clarify this in the relevant chapter. We call *right endpoint* of a path (resp. antipath) $P = (v_1, v_2, \ldots, v_k)$ the last vertex v_k of P. Moreover, let $P = (v_1, v_2, \ldots, v_{i-1}, v_i, v_{i+1}, \ldots, v_j, v_{j+1}, v_{j+2}, \ldots, v_k)$ and $P_0 = (v_i, v_{i+1}, \ldots, v_j)$ be

two paths (resp. antipaths) of a graph. Sometimes, we shall denote the path (resp. antipath) P by $P = (v_1, v_2, \ldots, v_{i-1}, P_0, v_{j+1}, v_{j+2}, \ldots, v_k)$. The *distance* $d(v, u)$ from vertex v to vertex u is the minimum length of a path from v to u; $d(v, u) = \infty$ if there is no path from v to u.

A sequence of vertices $[v_0, v_1, \ldots, v_k, v_0]$ is called a *cycle* of length $k+1$ if $v_{i-1}v_i \in E(G)$ for $i = 1, 2, \ldots, k$ and $v_k v_0 \in E(G)$. A cycle $[v_0, v_1, \ldots, v_k, v_0]$ is a *simple cycle* if $v_i \neq v_j$ for $i \neq j$. A simple cycle $[v_0, v_1, \ldots, v_k, v_0]$ is *chordless* if $v_i v_j \notin E(G)$ for every two non-successive vertices v_i, v_j in the cycle. A *hole* of G is an induced subgraph of G which is a chordless cycle C_n if $n \geq 5$; the complement of a hole is an antihole.

A *partial order* will be denoted by $P = (V, <_P)$, where V is the finite ground set of elements or vertices and $<_P$ is an irreflexive, antisymmetric, and transitive binary relation on V. Two elements $a, b \in V$ are comparable in P (denoted by $a \sim_P b$) if $a <_P b$ or $b <_P a$. Otherwise, they are said to be incomparable (denoted by $a \parallel b$). An *extension* of a partial order $P = (V, <_P)$ is a partial order $L = (V, <_L)$ on the same ground set that *extends* P, i.e., $a <_P b \Rightarrow a <_L b$, for all $a, b \in V$. The *dual partial order* P^d of $P = (V, <_P)$ is a partial order $P^d = (V, <_{P^d})$ such that for any two elements $a, b \in V$, $a <_{P^d} b$ if and only if $b <_P a$. A *linear order* is a partial order without incomparable elements. A *linear extension* of a partial order $P = (V, <_P)$ is a linear order $L = (V, <_L)$ on the same ground set that extends P.

For more basic definitions in graph theory refer to [10, 37, 56].

1.2 Perfect Graphs

The greatest integer r for which a graph G contains an independent set of size r is called the *independence number* or otherwise the *stability number* of G and is denoted by $\alpha(G)$. The *clique cover number* $\kappa(G)$ of a graph G is the smallest number of complete subgraphs needed to cover the vertices of G. A *proper vertex coloring* of a graph G is a coloring of its vertices such that no two adjacent vertices are assigned the same color. The *chromatic number* $\chi(G)$ of G is the smallest integer k for which G admits a proper vertex coloring with k colors. The cardinality of the vertex set of the maximum clique in G is called the *clique number* of G and is denoted by $\omega(G)$.

Clearly, for the numbers $\omega(G)$ and $\chi(G)$ of an arbitrary graph G the inequality $\omega(G) \leq \chi(G)$ holds. Also, since the intersection of a clique and a stable set of a graph G can be at most one vertex, it follows that $\alpha(G) \leq \kappa(G)$, for any graph G. These two equalities are dual to one another since $\alpha(G) = \omega(\overline{G})$ and $\kappa(G) = \chi(\overline{G})$.

A graph G is *perfect* if for every induced subgraph G_A of G, the chromatic number of G_A equals the size of the largest clique of G_A, i.e., $\chi(G_A) = \omega(G_A)$, $\forall A \subseteq V(G)$. The study of perfect graphs was initiated by Claude Berge in 1961 [4]. The following three conditions are the *perfection properties* of a graph G.

(P_1) $\omega(G_A) = \chi(G_A), \quad \forall A \subseteq V(G)$

(P_2) $\alpha(G_A) = \kappa(G_A), \quad \forall A \subseteq V(G)$

(P_3) $\omega(G_A) \cdot \alpha(G_A) \geq |A|, \quad \forall A \subseteq V(G)$

A graph is called χ-perfect if it satisfies (P_1) and α-perfect if it satisfies (P_2). Actually, it was conjectured by Berge [4], and proven by Lovász [53] that for an undirected graph G the perfection properties P_1, P_2 and P_3 are equivalent. This has become known as the *Perfect Graph Theorem* [53]. Therefore, it is sufficient to show that a graph satisfies any (P_i) in order to conclude that it is perfect, and a perfect graph will satisfy all properties (P_i).

A graph G is *Berge* if every hole and antihole of G has even length. In 1961 Berge [4] proposed two celebrated conjectures about perfect graphs. Since the second implies the first, they were known as the "weak" and "strong" perfect graph conjectures, respectively, although both are now theorems:

Theorem 1.1. *The complement of every perfect graph is perfect.*

Theorem 1.2. *A graph is perfect if and only if it is Berge.*

The first theorem was proved by Lovász [53] in 1972. The second theorem, which is known as the *Strong Perfect Graph Conjecture*, received a great deal of attention over the past 40 years. In 2002, Maria Chudnovsky and Paul Seymour, extending an earlier joint work with Neil Robertson and Robin Thomas, announced that they had completed the proof of the Strong Perfect Graph Conjecture which became the *Strong Perfect Graph Theorem*. The four joint authors published the 178-page paper in 2006 [16].

Therefore, holes and antiholes have been extensively studied in many different contexts in algorithmic graph theory and, thus, finding a hole or an antihole in a graph efficiently is an important graph-theoretic problem, both on its own and as a step in many recognition algorithms. In 2004, Nikolopoulos and Palios [60] proposed the fastest until today algorithm for the problem of finding a hole or an antihole in a graph, which runs in $O(n+m^2)$ time and requires $O(nm)$ space, where n is the number of vertices and m is the number of edges in the graph.

Classes of Perfect Graphs

In the case where an optimization problem is NP-complete on general graphs, it makes sense to look for polynomial solutions of the problem in special graph classes. The subclasses of perfect graphs have structural properties which allow us to find polynomial solutions for problems which are NP-complete on arbitrary graphs, such as coloring and path problems. These problems find applications in many fields of different sciences, from mathematics to philosophy [5, 40]. Throughout this work, several classes of perfect graph are mentioned, which are studied either in order to derive properties for the two new classes of graphs we define, namely colinear and linear graphs, or to provide polynomial algorithms or NP-completeness results for the coloring and longest path problems we study. Next, we give some definitions for these graph classes.

A graph is called a *chordal graph* if it does not contain an induced subgraph isomorphic to a chordless cycle of four or more vertices. Additionally, a graph G is chordal if and only if it admits a perfect elimination ordering; a perfect elimination ordering of a graph G is an ordering $(v_1, v_2, \ldots, v_n)$ of its vertices such that for every i, $1 \leq i \leq n$, v_i is a simplicial vertex in G_i, i.e., $N_{G_i}[v_i]$ is a clique in G_i. A graph is called a *co-chordal graph* if it is the complement of a chordal graph [37].

A graph G is a *split graph* if there is a partition of the vertex set $V(G) = K + I$, where K induces a clique in G and I induces an independent set. Split graphs are characterized as those graphs which do not contain a graph which is isomorphic to a $2K_2$, a C_4 or a C_5 graph as an induced subgraph, i.e., split graphs are characterized as $(2K_2, C_4, C_5)$-free; note that, a $2K_2$ graph is a graph such that $V(2K_2) = \{v_1, v_2, v_3, v_4\}$ and $E(2K_2) = \{v_1v_2, v_3v_4\}$. *Threshold graphs* were introduced by Chvátal and Hammer [17] and characterized as $(2K_2, P_4, C_4)$-free. *Quasi-threshold* graphs are characterized as the (P_4, C_4)-free graphs and are also known in the literature as trivially perfect graphs [37, 59].

We next give definitions and characterizations of some graph classes, on which we study the coloring and longest path problems and either provide polynomial algorithms or NP-completeness results; the characterizations mentioned here are used for obtaining some of our results.

Comparability and Cocomparability graphs. The graph G, edges of which are exactly the comparable pairs of a partial order P on $V(G)$, is called the *comparability graph* of P, and is denoted by $G(P)$. The complement graph $\overline{G}$, whose edges are the incomparable pairs of P, is called the *cocomparability graph* of P, and is denoted by $\overline{G}(P)$. Alternatively, a graph G is a cocomparability graph if its complement graph $\overline{G}$ has a transitive orientation, corresponding to the comparability relations of a partial order $P_{\overline{G}}$. Note that a partial order P uniquely determines its comparability graph $G(P)$ and its cocomparability graph $\overline{G}(P)$, but the reverse is not true, i.e., a cocomparability graph G has as many partial orders $P_{\overline{G}}$ as is the number of the transitive orientations of $\overline{G}$. Furthermore, the class of cocomparability graphs is *hereditary*, that is, every induced subgraph of a cocomparability graph G is also a cocomparability graph.

The following representation of comparability graphs is used for obtaining some of our results. Let G be a comparability graph, and let P_G be a partial order which corresponds to G. The graph G can

be represented by a directed covering graph with layers $H_1, H_2, \ldots, H_h$, in which each vertex is on the highest possible layer. That is, the maximal vertices of the partial order P_G are on the highest layer H_h, and for every vertex v on layer H_{i-1} there exists a vertex u on layer H_i such that $v <_{P_G} u$; such a layered representation of G (respectively P_G) is a called the *Hasse diagram* of G (respectively P_G).

Interval graphs. Interval graphs form an important and well-known class of perfect graphs [37], which form a subclass of chordal graphs. A graph G is an *interval graph* if its vertices can be put in a one-to-one correspondence with a family F of intervals on the real line such that two vertices are adjacent in G if and only if the corresponding intervals intersect; F is called an *intersection model* for G [1].

Additionally, the class of interval graphs is hereditary, and also a graph G is an interval graph if and only if it is a chordal graph and the graph $\overline{G}$ is a comparability graph [37]. Ramalingam and Rangan [61] proposed the following numbering of the vertices of an interval graph, which we use for obtaining some of our results: the vertices of any interval graph G can be numbered with integers $1, 2, \ldots, |V(G)|$ such that if $i < j < k$ and $ik \in E(G)$, then $jk \in E(G)$.

Permutation graphs. A graph G is a *permutation graph* if its vertices can be put in one to one correspondence with a set of line segments between two parallel lines, such that two vertices are adjacent if and only if their corresponding line segments intersect. Also, if G and $\overline{G}$ are comparability graphs, then G is a permutation graph [37].

Strongly chordal graphs. Strongly chordal graphs form a known subclass of chordal graphs [10, 27] and were first introduced by Farber [27]. A graph is *strongly chordal* iff it admits a strong elimination ordering; a vertex ordering $\sigma = (v_1, v_2, \ldots, v_n)$ is a strong elimination ordering of a graph G iff σ is a perfect elimination ordering and also has the property that for each i, j, k and ℓ, if $i < j$, $k < \ell$, $v_k, v_\ell \in N[v_i]$, and $v_k \in N[v_j]$, then $v_\ell \in N[v_j]$ [14, 27].

Also, the following characterization of strongly chordal graphs, due to Farber [27], appears useful for obtaining some results in this work: strongly chordal graphs are characterized completely as those chordal graphs which contain no k-sun as an induced subgraph. An *incomplete k-sun* S_k ($k \geq 3$) is a chordal graph on $2k$ vertices whose vertex set can be partitioned into two sets, $U = \{u_1, u_2, \ldots, u_k\}$ and $W = \{w_1, w_2, \ldots, w_k\}$, so that W is an independent set, and w_i is adjacent to u_j if and only if $i = j$ or $i = j + 1 \pmod{k}$; the graph S_k ($k \geq 3$) is a *k-sun* if U is a complete graph.

Undirected path graphs. Undirected path graphs form a subclass of chordal graphs. A chordal graph is an *undirected path graph* if it is the vertex intersection graph of undirected paths in a tree [35, 57, 62]. In particular, the following characterization of undirected path graphs given in [35, 57] is used for obtaining our results; note that, $\mathcal{C}$ denotes the set of all maximal cliques of a graph G, and $C(v)$ denotes the set of all maximal cliques containing v. A graph G is an undirected path graph if and only if there exists a tree T whose set of vertices is $\mathcal{C}$, so that for every vertex $v \in V(G)$, the subgraph $T[C(v)]$ of T induced by the vertex set $C(v)$, is a path in T. Such a tree will be called characteristic tree of G.

1.3 Computability and Complexity

Let us first underline the differences between computability and computational complexity [37]. *Computability* addresses itself mostly to questions of existence: Is there an algorithm which solves problem Π? Proving that a problem is computable usually consists of demonstrating an actual algorithm which will terminate with a correct answer for every input. The amount of resources (time and space) used in the calculation, although finite, is unlimited. On the contrary, *computational complexity* deals precisely with the quantitative aspects of problem solving. It addresses the issue of what can be computed within a practical or reasonable amount of time and space by measuring the resource requirements exactly or by obtaining upper and lower bounds.

Algorithms are step-by-step procedures for solving problems. An algorithm is said to solve a problem Π if it can be applied to any instance I of Π and is guaranteed always to produce a solution for that

instance I. The *time complexity function* for an algorithm expresses its time requirements by giving, for each possible input length, the largest amount of time needed by the algorithm to solve a problem instance of that size. A *polynomial time algorithm* is defined to be one whose time complexity function is $O(p(n))$ for some polynomial function p, where n is the input length; an *exponential time algorithm* is one whose time complexity function cannot be so bounded. The distinction between the two types of algorithms is central to the notion of inherent intractability and to the theory of NP-completeness.

1.3.1 Polynomial Transformations and NP-completeness

A problem is called *intractable* if it is so hard that no polynomial time algorithm can possibly solve it. As much work has been done for proving problems intractable, so much efforts focus on learning more about the ways in which various problems are interrelated with respect to their difficulty. The main technique used for showing that two problems are related with respect to their difficulty is that of "reducing" one to the other, by giving a constructive transformation that maps any instance of the first problem into an equivalent instance of the second. Such a transformation provides the means for converting any algorithm that solves the second problem into a corresponding algorithm for solving the first problem [33].

A reduction captures the informal notion of a problem being at least as difficult as another problem. For instance, if a problem Π_1 can be solved using an algorithm for Π_2, Π_1 is no more difficult than Π_2, and we say that Π_1 reduces to Π_2. The most commonly used reduction is a polynomial-time reduction; this means that the reduction process takes polynomial time.

We say that a problem Π_1 is *polynomially transformable* to another problem Π_2 denoted $\Pi_1 \preccurlyeq \Pi_2$, if there exists a polynomially computable function f mapping the instances of Π_1 into the instances of Π_2 such that a solution to the instance $f(I)$ of Π_2, gives a solution to the instance I of Π_1, for all I. Intuitively this means that Π_1 is no harder to solve than Π_2 up to added polynomial term, for we could solve Π_1 by combining the transformation f with the best algorithm for solving Π_2. Thus, if $\Pi_1 \preccurlyeq \Pi_2$, then COMPLEXITY(Π_1) $\leq$ COMPLEXITY(Π_2) + POLYNOMIAL. If Π_2 has a polynomial time algorithm, then so does Π_1; if every algorithm solving Π_1 requires at least an exponential amount of time, then the same is true for Π_2.

The *state* of an algorithm consists of the current values of all variables and the location of the current instruction to be executed. A *deterministic algorithm* is one for which each state upon execution of the instruction uniquely determines at most one next state. Virtually all computers run deterministically. A *nondeterministic algorithm* is one for which a state may determine many next states and which follows up on each of the next states simultaneously. We may regard a nondeterministic algorithm as having the capability of branching off into many copies of itself, one for each next state. Thus, while a deterministic algorithm must explore a set of alternatives one at a time, a nondeterministic algorithm examines all alternatives at the same time.

The complexity class P is often seen as a mathematical abstraction modelling those computational tasks that admit an efficient algorithm. The complexity class NP, on the other hand, contains many problems that people would like to solve efficiently, but for which no efficient algorithm is known. Strictly speaking, a problem Π is in the class P if there exists a deterministic polynomial time algorithm which solves Π. A problem Π is in the class NP if there exists a nondeterministic polynomial time algorithm which solves Π; all the problems in this class have the property that their solutions can be checked efficiently. Since deterministic Turing machines are special nondeterministic Turing machines, it is easily observed that each problem in P is also member of the class NP, i.e., P $\subseteq$ NP.

A problem Π is *NP-hard* if any one of the following equivalent conditions holds:

(1) $\Pi' \preccurlyeq \Pi$ for all $\Pi' \in$ NP;

(2) $\Pi \in$ P $\Rightarrow$ P=NP;

(3) the existence of a deterministic polynomial time algorithm for Π would imply the existence of a polynomial time algorithm for every problem in NP.

A problem Π is *NP-complete* if it is both a member of NP and it is NP-hard. The NP-complete problems are the most difficult of those in NP. To prove that Π is NP-complete we show that $\Pi \in$ NP and some known NP-complete problem Π' transforms to Π.

The foundations for the theory of NP-completeness were laid in a paper of Stephen Cook [19], presented in 1971. He emphasized the significance of "polynomial time reducibility", and he focused on the class of NP of decision problems that can be solved in polynomial time by a nondeterministic computer. He proved that one particular problem in NP, called the "satisfiability" problem, has the property that every other problem in NP can be polynomially reduced to it. If the satisfiability can be solved with a polynomial time algorithm, then so can every problem in NP, and if any problem in NP is intractable, then the satisfiability problem also must be intractable. Subsequently, Karp [48] presented a collection of results proving that indeed the decision versions of many well known combinatorial problems, including the travelling salesman problem, are just as "hard" as the satisfiability problem. Since then a wide variety of other problems have been proved equivalent in difficulty to these problems, and this equivalence class, consisting of the "hardest" problems in NP, has been called the class of NP-complete problems.

Cook's original ideas have provided the means for combining many individual complexity questions into the single question: Are the NP-complete problems intractable? The question of whether P equals NP is one of the most important open questions in theoretical computer science because of the wide implications of a solution.

1.3.2 Polynomial Algorithms

A way of classifying algorithms is by their design methodology or paradigm. There is a certain number of techniques for the design and analysis of algorithms, such as brute-force or exhaustive search, divide and conquer, dynamic programming, the greedy method, linear programming, reduction, search and enumeration, and the probabilistic and heuristic paradigm etc. The algorithms presented in this work for solving the optimization problems studied base mainly on dynamic programming, the greedy method, or reduction.

Dynamic programming typically applies to optimization problems in which a set of choices must be made in order to arrive at an optimal solution. When a problem shows optimal substructure, meaning the optimal solution to a problem can be constructed from optimal solutions to subproblems, and overlapping subproblems, meaning the same subproblems are used to solve many different problem instances, a quicker approach called dynamic programming avoids recomputing solutions that have already been computed. The difference between dynamic programming and straightforward recursion is in caching or memoization of recursive calls. When subproblems are independent and there is no repetition, memoization does not help; hence dynamic programming is not a solution for all complex problems. By using memoization or maintaining a table of subproblems already solved, dynamic programming transforms exponential time algorithms into polynomial time algorithms.

Greedy algorithms are similar to a dynamic programming algorithms, since they apply to optimization problems in which a set of choices must be made in order to arrive at an optimal solution. The difference is that in a greedy algorithm solutions to the subproblems do not have to be known at each stage; instead the idea of a greedy algorithm is to make each choice in a locally optimal manner. The greedy method extends the solution with the best possible decision (not all feasible decisions) at an algorithmic stage based on the current local optimum and the best decision (not all possible decisions) made in a previous stage. It is not exhaustive, and and when it works, it will be the fastest method.

The technique of *reduction* involves solving a difficult problem by transforming it into a better known problem for which we have (hopefully) asymptotically optimal algorithms. The goal is to find a reducing algorithm whose complexity is not dominated by the resulting reduced algorithm's.

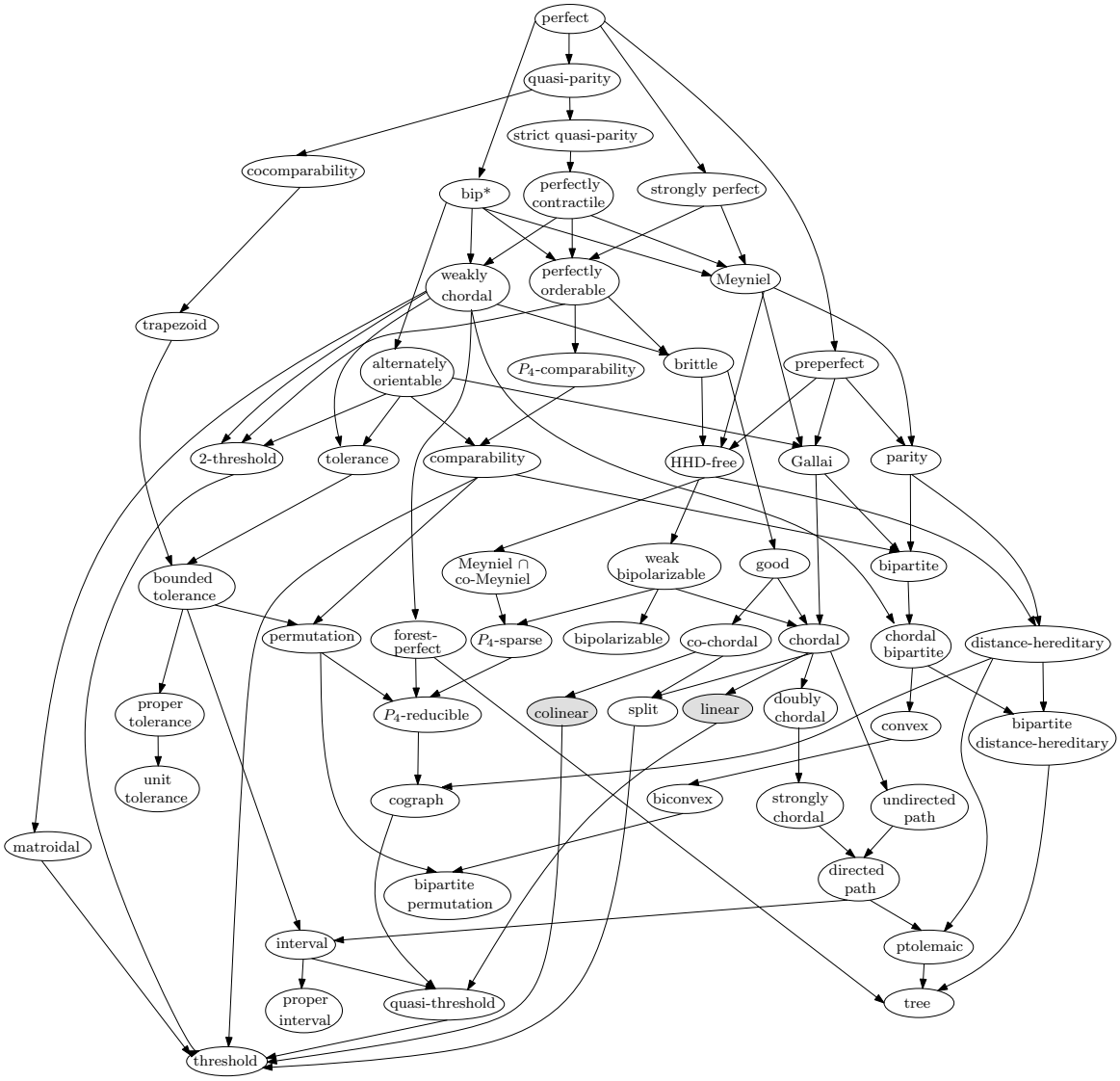

Figure 1.1: Illustrating a map of some classes of perfect graphs, including the classes of colinear and linear graphs.

1.4 Coloring and Longest Path Problems

We next describe the problems this work is concerned with. These include providing characterizations of two new classes of perfect graphs, namely colinear and linear graphs, and presenting polynomial-time algorithms or NP-completeness results for coloring problems on graphs and polynomial-time algorithms for the longest path problem on perfect graphs.

1.4.1 Colinear Coloring and Colinear Graphs

A *colinear coloring* of a graph G is a coloring of its vertices such that two vertices are assigned different colors, if their corresponding clique sets are not associated by the set inclusion relation; a *clique set* of a vertex u is the set of all maximal cliques in G containing u. The colinear chromatic number $\lambda(G)$ of G is the least integer k for which G admits a colinear coloring with k colors.

Motivated by the definition of linear coloring on simplicial complexes associated to graphs, first introduced by Civan and Yalçin [18] in the context of algebraic topology, we studied linear colorings on

simplicial complexes which can be represented by a graph. The outcome of this study was the definition of the colinear coloring of a graph G; the colinear coloring of a graph G is a coloring of G such that for any set of vertices taking the same color, the collection of their clique sets can be linearly ordered by inclusion. Recently, Civan and Yalçin [18] studied the linear coloring of the neighborhood complex $\mathcal{N}(G)$ of a graph G and proved that the linear chromatic number of $\mathcal{N}(G)$ gives an upper bound for the chromatic number $\chi(G)$ of the graph G. This approach lies in a general framework met in algebraic topology.

The interest to provide boundaries for the chromatic number $\chi(G)$ of an arbitrary graph G through the study of different simplicial complexes associated to G, which is found in algebraic topology bibliography, drove the motivation for defining the colinear coloring on the graph G and studying the relation between the chromatic number $\chi(G)$ and the colinear chromatic number $\lambda(\overline{G})$. We show that for any graph G, $\lambda(\overline{G})$ is an upper bound for $\chi(G)$. The interest of this result lies on the fact that we present a colinear coloring algorithm that can be applied to any graph G and provides an upper bound $\lambda(\overline{G})$ for the chromatic number of the graph G, i.e., $\chi(G) \leq \lambda(\overline{G})$; in particular, it provides a proper vertex coloring of G using $\lambda(\overline{G})$ colors. Additionally, recall that a known lower bound for the chromatic number of any graph G is the clique number $\omega(G)$ of G, i.e., $\chi(G) \geq \omega(G)$.

Motivated by these results and the definition of perfect graphs, for which $\chi(G_A) = \omega(G_A)$ holds $\forall A \subseteq V(G)$ [37], we study those graphs for which the equality $\chi(G) = \lambda(\overline{G})$ holds for every induced subgraph and characterize known graph classes in terms of the χ-colinear and the α-colinear properties. A graph G has the *χ-colinear property* if its chromatic number $\chi(G)$ equals to the colinear chromatic number $\lambda(\overline{G})$ of its complement graph $\overline{G}$, and the equality holds for every induced subgraph of G, i.e., $\chi(G_A) = \lambda(\overline{G}_A)$, $\forall A \subseteq V(G)$; a graph G has the *α-colinear property* if its stability number $\alpha(G)$ equals to its colinear chromatic number $\lambda(G)$, and the equality holds for every induced subgraph of G, i.e., $\alpha(G_A) = \lambda(G_A)$, $\forall A \subseteq V(G)$. We show that the class of threshold graphs is characterized by the χ-colinear property and the class of quasi-threshold graphs is characterized by the α-colinear property.

Moreover, it was interesting to study those graphs which are characterized completely by the χ-colinear or the α-colinear property. The outcome of this study was to conclude that these graphs form two new classes of perfect graphs, which we call colinear and linear graphs, respectively. We also provide characterizations for colinear and linear graphs and prove structural properties. More specifically, we show that the class of colinear graphs is a subclass of co-chordal graphs, a superclass of threshold graphs, and is distinguished from the class of split graphs. Additionally, we infer that linear graphs form a subclass of chordal graphs and a superclass of quasi-threshold graphs. We also prove that any P_6-free chordal graph, which is not a linear graph, properly contains a k-sun as an induced subgraph. However, the k-sun is not a forbidden induced subgraph for the class of linear graphs and, thus, linear graphs form a superclass of the class of P_6-free strongly chordal graphs. Figure 1.1 depicts a map of some classes of perfect graphs, including the classes of colinear and linear graphs which we define within this work.

1.4.2 The Harmonious Coloring Problem

A *harmonious coloring* of a simple graph G is a proper vertex coloring such that each pair of colors appears together on at most one edge, while the *harmonious chromatic number* $h(G)$ is the least integer k for which G admits a harmonious coloring with k colors [13].

Harmonious coloring developed from the closely related concept of line-distinguishing coloring which was introduced independently by Frank et al. [30] and by Hopcroft and Krishnamoorthy [45] who showed that the harmonious coloring problem is NP-complete on general graphs. The complexity of the harmonious coloring problem has been extensively studied on various classes of perfect graphs such as cographs, interval graphs, bipartite graphs and trees [10, 37]. Bodlaender [8] provides a proof for the NP-completeness of the harmonious coloring problem for disconnected cographs and disconnected interval graphs. It is worth noting that the problem of determining the harmonious chromatic number of a connected cograph is trivial, since in such a graph each vertex must receive a distinct color as it is at distance at most 2 from all other vertices [13]. Bodlaender's results establish the NP-hardness of the

harmonious coloring problem when restricted to disconnected permutation graphs. Extending the above results, in this work we show that the harmonious coloring problem remains NP-complete on connected interval and permutation graphs. In addition, we show that the problem remains NP-complete for the class of split graphs.

Additionally, the NP-completeness of the problem has been also proved for the classes of trees and disconnected bipartite permutation graphs [25, 26], connected bipartite permutation graphs [2], and disconnected quasi-threshold graphs [2]. Since the problem of determining the harmonious chromatic number of a connected cograph is trivial, the harmonious coloring problem is polynomially solvable on connected quasi-threshold graphs and threshold graphs.

Since we prove that the harmonious coloring problem is NP-complete on interval , we obtain that the problem is also NP-complete on the classes of strongly chordal and undirected path graphs. Extending our results for the harmonious coloring problem on interval graphs and split graphs, in this work we also study the complexity status of the harmonious coloring problem on two subclasses of colinear graphs. We first show that the harmonious coloring problem is NP-complete on split undirected path graphs and, then, we show that the class of split undirected path graphs forms a subclass of colinear graphs; thus, we obtain the NP-completeness of the harmonious coloring problem on colinear graphs as well.

Moreover, we provide a polynomial solution for the harmonious coloring problem on split strongly chordal graphs, the interest of which lies on the fact that the problem is NP-complete on both split graphs and strongly chordal graphs. However, the complexity status of the problem for the class of connected linear graphs still remains an open question; note that the harmonious coloring problem is NP-complete on disconnected linear graphs, since it is NP-complete on disconnected quasi-threshold graphs [2] and quasi-threshold graphs form a subclass of linear graphs.

1.4.3 The Longest Path Problem

The longest path problem, i.e., the problem of finding a path of maximum length in a graph, is a generalization of the Hamiltonian path problem. The Hamiltonian path problem is the problem of determining whether a graph is Hamiltonian; a graph is said to be Hamiltonian if it contains a Hamiltonian path, that is, a simple path in which every vertex of the graph appears exactly once. The longest path problem or, equivalently, the problem of finding a maximum Hamiltonian induced subgraph of a graph, is NP-complete on general graphs and, in fact, on every class of graphs that the Hamiltonian path problem is NP-complete. However, it is interesting to study the longest path problem on classes of graphs where the Hamiltonian path problem is polynomial, since even if a graph is not Hamiltonian, it makes sense in several applications to search for a longest path of the graph. Although the Hamiltonian path problem has received a great deal of attention the past two decades in looking for polynomial solutions for the problem on special graph classes, only recently did the longest path problem start receiving attention in this direction.

As we have mentioned, the longest path problem is NP-hard on every class of graphs on which the Hamiltonian path problem is NP-complete. The Hamiltonian path problem is known to be NP-complete in general graphs [33, 34], and remains NP-complete even when restricted to some small classes of graphs such as split graphs [37], chordal bipartite graphs, split strongly chordal graphs [58], circle graphs [22], planar graphs [34], and grid graphs [46]. However, it makes sense to investigate the tractability of the longest path problem on the classes of graphs for which the Hamiltonian path problem admits polynomial time solutions. Such classes include interval graphs [1], circular-arc graphs [24], convex bipartite graphs [58], and co-comparability graphs [23]. Note that the problem of finding a longest path on proper interval graphs is easy, since all connected proper interval graphs have a Hamiltonian path which can be computed in linear time [6]. On the contrary, not all interval graphs are Hamiltonian; in the case where an interval graph has a Hamiltonian path, it can be computed in linear time [1, 15]. However, in the case where an interval graph is not Hamiltonian, there is no known algorithm for finding a longest path on it.

In contrast to the Hamiltonian path problem, the known polynomial time solutions for the longest path problem are rather recent, and restrict to smaller graph classes. Specifically, a linear time algorithm for finding a longest path in a tree was proposed by Dijkstra around 1960, a formal proof of which can be found in [12]. Later, through a generalization of Dijkstra's algorithm for trees, Uehara and Uno [63] solved the longest path problem for weighted trees and block graphs in linear time and space, and for cacti in $O(n^2)$ time and space, where n and m denote the number of vertices and edges of the input graph, respectively. More recently, polynomial algorithms have been proposed that solve the longest path problem on bipartite permutation graphs in $O(n)$ time and space [64], and on ptolemaic graphs in $O(n^5)$ time and $O(n^2)$ space [65]. Furthermore, Uehara and Uno in [63] solved the longest path problem on a subclass of interval graphs, namely interval biconvex graphs, in $O(n^3(m + n \log n))$ time, and as a corollary they showed that a longest path on threshold graphs can be found in $O(n+m)$ time and space. They left open the complexity of the longest path problem on interval graphs.

In this work, we resolve the open problem posed in [63] by showing that the longest path problem admits a polynomial time solution on interval graphs. In particular, we propose an algorithm for solving the longest path problem on interval graphs which runs in $O(n^4)$ time using a dynamic programming approach. Thus, not only we answer the question left open by Uehara and Uno in [63], but also improve the known time complexity of the problem on interval biconvex graphs, a subclass of interval graphs [63].

Moreover, we study the longest path problem on the class of cocomparability graphs, a well-known class of perfect graphs which includes both interval and permutation graphs. Although the Hamiltonian path problem on cocomparability graphs has been proved to be polynomial [23], the status of the longest path problem on cocomparability graphs is unknown, since no polynomial-time algorithm or NP-completeness result exists; actually, the status of the longest path problem is unknown even on the more special class of permutation graphs. In this work, we propose a polynomial-time algorithm for solving the longest path problem on cocomparability graphs, which extends our polynomial solution of the longest path problem on interval graphs, and resolves the open question for the status of the problem on cocomparability graphs, and thus on permutation graphs.

Chapter 2

Colinear Coloring and Colinear Graphs

2.1 Introduction

A *colinear coloring* of a graph G is a coloring of its vertices such that two vertices are assigned different colors, if their corresponding clique sets are not associated by the set inclusion relation; a *clique set* of a vertex u is the set of all maximal cliques in G containing u. The colinear chromatic number $\lambda(G)$ of G is the least integer k for which G admits a colinear coloring with k colors.

Motivated by the definition of linear coloring on simplicial complexes associated to graphs, first introduced by Civan and Yalçin [18] in the context of algebraic topology, we studied linear colorings on simplicial complexes which can be represented by a graph. In particular, we studied the linear coloring problem on a simplicial complex, namely independence complex $\mathcal{I}(G)$ of a graph G. The independence complex $\mathcal{I}(G)$ of a graph G can always be represented by a graph and, more specifically, is identical to the complement graph $\overline{G}$ of the graph G; indeed, the facets of $\mathcal{I}(G)$ are exactly the maximal cliques of $\overline{G}$. The outcome of this study was the definition of the colinear coloring of a graph G; the colinear coloring of a graph G is a coloring of G such that for any set of vertices taking the same color, the collection of their clique sets can be linearly ordered by inclusion. Note that, the two definitions cannot always be considered as identical since not in all cases a simplicial complex can be represented by a graph; such an example is the neighborhood complex $\mathcal{N}(G)$ of a graph G. Recently, Civan and Yalçin [18] studied the linear coloring of the neighborhood complex $\mathcal{N}(G)$ of a graph G and proved that the linear chromatic

number of $\mathcal{N}(G)$ gives an upper bound for the chromatic number $\chi(G)$ of the graph G. This approach lies in a general framework met in algebraic topology.

In the context of algebraic topology, one can find much work done on providing boundaries for the chromatic number of an arbitrary graph G, by examining the topology of the graph through different simplicial complexes associated to the graph. This domain was motivated by Kneser's conjecture, which was posed in 1955, claiming that "if we split the n-subsets of a $(2n+k)$-element set into $k+1$ classes, one of the classes will contain two disjoint n-subsets" [50]. Kneser's conjecture was first proved by Lovász in 1978, with a proof based on graph theory, by rephrasing the conjecture into "the chromatic number of Kneser's graph $KG_{n,k}$ is k+2" [54]. Many more topological and combinatorial proofs followed, the interest of which extends beyond the original conjecture [69]. Although Kneser's conjecture is concerned with the chromatic numbers of certain graphs (Kneser graphs), the proof methods that are known provide lower bounds for the chromatic number of any graph [55]. Thus, this initiated the application of topological tools in studying graph theory problems and more particularly in graph coloring problems [21].

The interest to provide boundaries for the chromatic number $\chi(G)$ of an arbitrary graph G through the study of different simplicial complexes associated to G, which is found in algebraic topology bibliography, drove the motivation for defining the colinear coloring on the graph G and studying the relation between the chromatic number $\chi(G)$ and the colinear chromatic number $\lambda(\overline{G})$. We show that for any graph G, $\lambda(\overline{G})$ is an upper bound for $\chi(G)$. The interest of this result lies on the fact that we present a colinear coloring algorithm that can be applied to any graph G and provides an upper bound $\lambda(\overline{G})$ for the chromatic number of the graph G, i.e., $\chi(G) \leq \lambda(\overline{G})$; in particular, it provides a proper vertex coloring of G using $\lambda(\overline{G})$ colors. Additionally, recall that a known lower bound for the chromatic number of any graph G is the clique number $\omega(G)$ of G, i.e., $\chi(G) \geq \omega(G)$. Motivated by the definition of perfect graphs, for which $\chi(G_A) = \omega(G_A)$ holds $\forall A \subseteq V(G)$, it was interesting to study those graphs for which the equality $\chi(G) = \lambda(\overline{G})$ holds, and even more those graphs for which this equality holds for every induced subgraph.

In this work, we first introduce the colinear coloring of a graph G and study the relation between the colinear coloring of $\overline{G}$ and the proper vertex coloring of G. We prove that, for any graph G, a colinear coloring of $\overline{G}$ is a proper vertex coloring of G and, thus, $\lambda(\overline{G})$ is an upper bound for $\chi(G)$, i.e., $\chi(G) \leq \lambda(\overline{G})$. We present a colinear coloring algorithm that can be applied to any graph G. Motivated by these results and the Perfect Graph Theorem [37], we study those graphs for which the equality $\chi(G) = \lambda(\overline{G})$ holds for every induced subgraph and characterize known graph classes in terms of the χ-colinear and the α-colinear properties. A graph G has the *χ-colinear property* if its chromatic number $\chi(G)$ equals to the colinear chromatic number $\lambda(\overline{G})$ of its complement graph $\overline{G}$, and the equality holds for every induced subgraph of G, i.e., $\chi(G_A) = \lambda(\overline{G}_A)$, $\forall A \subseteq V(G)$; a graph G has the *α-colinear property* if its stability number $\alpha(G)$ equals to its colinear chromatic number $\lambda(G)$, and the equality holds for every induced subgraph of G, i.e., $\alpha(G_A) = \lambda(G_A)$, $\forall A \subseteq V(G)$. Note that the *stability number* $\alpha(G)$ of a graph G is the greatest integer r for which G contains an independent set of size r. We show that the class of threshold graphs is characterized by the χ-colinear property and the class of quasi-threshold graphs is characterized by the α-colinear property.

Moreover, it was interesting to study those graphs which are characterized completely by the χ-colinear or the α-colinear property. The outcome of this study was to conclude that these graphs form two new classes of perfect graphs, which we call colinear and linear graphs, respectively. We also provide characterizations for colinear and linear graphs and prove structural properties. More specifically, we show that the class of colinear graphs is a subclass of co-chordal graphs, a superclass of threshold graphs, and is distinguished from the class of split graphs. Additionally, we infer that linear graphs form a subclass of chordal graphs and a superclass of quasi-threshold graphs. We also prove that any P_6-free chordal graph, which is not a linear graph, properly contains a k-sun as an induced subgraph. However, the k-sun is not a forbidden induced subgraph for the class of linear graphs and, thus, linear graphs form a superclass of the class of P_6-free strongly chordal graphs.

The rest of this chapter is organized as follows. In Section 2.2 we define the colinear coloring on graphs,

while in Section 2.3 we present a polynomial time algorithm for colinear coloring which can be applied to any graph G and provides an upper bound for the chromatic number $\chi(G)$ of the graph G. In Section 2.4 we define the χ-colinear and α-colinear properties and characterize known graph classes in terms of these properties. Based on these results, in Section 2.5 we study the graphs which are characterized completely by the χ-colinear or α-colinear property and, thus, define two new classes of perfect graphs, which we call colinear and linear graphs. Characterizations and structural properties of linear graphs are proved in Section 2.6. Some concluding remarks follow.

2.2 Colinear Coloring on Graphs

In this section we define the colinear coloring of a graph G, and we prove some properties of such a coloring. It is worth noting that these properties have been also proved for the linear coloring of the neighborhood complex $\mathcal{N}(G)$ in [18].

Definition 2.1. Let G be a graph and let $v \in V(G)$. The *clique set* of a vertex v is the set of all maximal cliques of G containing v and is denoted by $\mathcal{C}_G(v)$.

Definition 2.2. Let G be a graph and let k be an integer. A surjective map $\kappa : V(G) \to \{1, 2, \ldots, k\}$ is called a *k-colinear coloring* of G if the collection $\{\mathcal{C}_G(v) : \kappa(v) = i\}$ is linearly ordered by inclusion for all $i \in \{1, 2, \ldots, k\}$, where $\mathcal{C}_G(v)$ is the clique set of v, or, equivalently, for two vertices $v, u \in V(G)$, if $\kappa(v) = \kappa(u)$ then either $\mathcal{C}_G(v) \subseteq \mathcal{C}_G(u)$ or $\mathcal{C}_G(v) \supseteq \mathcal{C}_G(u)$. The least integer k for which G is k-colinear colorable is called the *colinear chromatic number* of G and is denoted by $\lambda(G)$.

Next, we study the colinear coloring on graphs and its association to the proper vertex coloring. In particular, we show that for any graph G the colinear chromatic number of $\overline{G}$ is an upper bound for $\chi(G)$.

Proposition 2.1. *Let G be a graph. If $\kappa : V(G) \to \{1, 2, \ldots, k\}$ is a k-colinear coloring of $\overline{G}$, then κ is a coloring of the graph G.*

Proof. Let G be a graph and let $\kappa : V(G) \to \{1, 2, \ldots, k\}$ be a k-colinear coloring of $\overline{G}$. From Definition 4.2, we have that for any two vertices $v, u \in V(G)$, if $\kappa(v) = \kappa(u)$ then either $\mathcal{C}_{\overline{G}}(v) \subseteq \mathcal{C}_{\overline{G}}(u)$ or $\mathcal{C}_{\overline{G}}(v) \supseteq \mathcal{C}_{\overline{G}}(u)$ holds. Without loss of generality, assume that $\mathcal{C}_{\overline{G}}(v) \subseteq \mathcal{C}_{\overline{G}}(u)$ holds. Consider a maximal clique $C \in \mathcal{C}_{\overline{G}}(v)$. Since $\mathcal{C}_{\overline{G}}(v) \subseteq \mathcal{C}_{\overline{G}}(u)$, we have $C \in \mathcal{C}_{\overline{G}}(u)$. Thus, both $u, v \in C$ and therefore $uv \in E(\overline{G})$ and $uv \notin E(G)$. Hence, any two vertices assigned the same color in a k-colinear coloring of $\overline{G}$ are not neighbors in G. Concluding, any k-colinear coloring of $\overline{G}$ is a coloring of G. ∎

It is therefore straightforward to conclude the following.

Corollary 2.1. *For any graph G, $\lambda(\overline{G}) \geq \chi(G)$.*

In Figure 2.1 we depict a colinear coloring of the well known graphs $2K_2$, C_4 and P_4, using the least possible colors, and show the relation between the chromatic number $\chi(G)$ of each graph $G \in \{2K_2, C_4, P_4\}$ and the colinear chromatic number $\lambda(\overline{G})$.

Proposition 2.2. *Let G be a graph. A coloring $\kappa : V(G) \to \{1, 2, \ldots, k\}$ of G is a k-colinear coloring of G if and only if either $N_G[u] \subseteq N_G[v]$ or $N_G[u] \supseteq N_G[v]$ holds in G, for every $u, v \in V(G)$ with $\kappa(u) = \kappa(v)$.*

Proof. Let G be a graph and let $\kappa : V(G) \to \{1, 2, \ldots, k\}$ be a k-colinear coloring of G. We will show that either $N_G[u] \subseteq N_G[v]$ or $N_G[u] \supseteq N_G[v]$ holds in G for every $u, v \in V(G)$ with $\kappa(u) = \kappa(v)$. Consider two vertices $v, u \in V(G)$, such that $\kappa(u) = \kappa(v)$. Since κ is a colinear coloring of G, we have either $\mathcal{C}_G(u) \subseteq \mathcal{C}_G(v)$ or $\mathcal{C}_G(u) \supseteq \mathcal{C}_G(v)$ holds. Without loss of generality, assume that $\mathcal{C}_G(u) \subseteq \mathcal{C}_G(v)$. We will

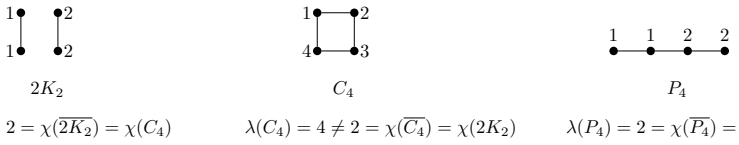

Figure 2.1: Illustrating a colinear coloring of the graphs $2K_2$, C_4 and P_4 with the least possible colors.

show that $N_G[u] \subseteq N_G[v]$ holds in G. Assume the opposite. Thus, a vertex $z \in V(G)$ exists, such that $z \in N_G[u]$ and $z \notin N_G[v]$ and, thus, $zu \in E(G)$ and $zv \notin E(G)$. Now consider a maximal clique C in G which contains z and u. Since $zv \notin E(G)$, it follows that $v \notin C$. Thus, there exists a maximal clique C in G such that $C \in \mathcal{C}_G(u)$ and $C \notin \mathcal{C}_G(v)$, which is a contradiction to our assumption that $\mathcal{C}_G(u) \subseteq \mathcal{C}_G(v)$. Therefore, $N_G[u] \subseteq N_G[v]$ holds in G.

Let G be a graph and let $\kappa : V(G) \to \{1, 2, \ldots, k\}$ be a coloring of G. Assume now that either $N_G[u] \subseteq N_G[v]$ or $N_G[u] \supseteq N_G[v]$ holds in G, for every $u, v \in V(G)$ with $\kappa(u) = \kappa(v)$. We will show that the coloring κ of G is a k-colinear coloring of G. Without loss of generality, assume that $N_G[u] \subseteq N_G[v]$ holds in G, and we will show that $\mathcal{C}_G(u) \subseteq \mathcal{C}_G(v)$. Assume the opposite. Thus, a maximal clique C exists in G, such that $C \in \mathcal{C}_G(u)$ and $C \notin \mathcal{C}_G(v)$. Consider now a vertex $z \in V(G)$ $(z \neq v)$, such that $z \in C$ and $zv \notin E(G)$. Such a vertex exists since C is maximal in G and $C \notin \mathcal{C}_G(v)$. Thus, $zv \notin E(G)$ and either $zu \in E(G)$ or $z = u$, which is a contradiction to our assumption that $N_G[u] \subseteq N_G[v]$. ∎

2.3 An Algorithm for Colinear Coloring

In this section we present a polynomial time algorithm for colinear coloring which can be applied to any graph G, and provides an upper bound for $\chi(G)$. Although we have introduced colinear coloring through Definition 4.2, in our algorithm we exploit the property proved in Proposition 4.11, since the problem of finding all maximal cliques of a graph G is not polynomially solvable on general graphs. Before describing our algorithm, we first construct a directed acyclic graph (DAG) D_G of a graph G, which we call *DAG associated to the graph* G, and we use it in the proposed algorithm.

The DAG D_G associated to the graph G. Let G be a graph. We first compute the closed neighborhood $N_G[v]$ of each vertex v of G and, then, we construct the following directed acyclic graph D, which depicts all inclusion relations among the vertices' closed neighborhoods: $V(D) = V(G)$ and $E(D) = \{\overrightarrow{xy} \ : \ x, y \in V(D) \text{ and } N_G[x] \subseteq N_G[y]\}$, where $\overrightarrow{xy}$ is a directed edge from x to y. In the case where the equality $N_G[x] = N_G[y]$ holds, we choose to add one of the two edges so that the resulting graph D is acyclic. To achieve this, we consider a partition of the vertex set $V(G)$ into the sets $S_1, S_2, \ldots, S_\ell$, such that for any $i \in \{1, 2, \ldots, \ell\}$ vertices x and y belong to a set S_i if and only if $N_G[x] = N_G[y]$. For vertices x and y belonging to the same set S_i we add the edge $\overrightarrow{xy}$ if and only if $x < y$. For vertices x and y belonging to different sets S_i and S_j respectively, we add the edge $\overrightarrow{xy}$ if and only if $N_G[x] \subset N_G[y]$. It is easy to see that the resulting graph D is unique up to isomorphism.

Additionally, it is easy to see that D is a transitive directed acyclic graph. Indeed, by definition D is constructed on a partially ordered set of elements $(V(D), \leq)$, such that for some $x, y \in V(D)$, $x \leq y \Leftrightarrow N_G[x] \subseteq N_G[y]$. Throughout this work we refer to the constructed directed acyclic graph as the DAG associated to the graph G and denote it by D_G.

The proposed algorithm, namely Algorithm 1, computes a colinear coloring and the colinear chromatic number of a graph G.

Correctness of the algorithm. Let G be a graph and let D_G be the DAG associated to the graph G, which is unique up to isomorphism. Consider the value $\kappa(v)$ for each vertex $v \in V(D_G)$ returned by the algorithm and the size $\rho(D_G)$ of a minimum path cover of D_G. We show that the surjective map

Algorithm Colinear_Coloring

Input: a graph G.

Output: a colinear coloring and the colinear chromatic number of G.

(i) `compute` the closed neighborhood set of every vertex of G and, then, find the inclusion relations among the neighborhood sets and construct the DAG D_G associated to the graph G.

(ii) `find` a minimum path cover $\mathcal{P}(D_G)$, and its size $\rho(D_G)$, of the transitive DAG D_G (e.g. see [9, 44]).

(iii) `assign` a color $\kappa(v)$ to each vertex $v \in V(D_G)$, such that vertices belonging to the same path of $\mathcal{P}(D_G)$ are assigned the same color and vertices of different paths are assigned different colors; this is a surjective map $\kappa : V(D_G) \to [\rho(D_G)]$.

(iv) `return` the value $\kappa(v)$ for each vertex $v \in V(D_G)$ and the size $\rho(D_G)$ of the minimum path cover of D_G; κ is a colinear coloring of G and $\rho(D_G)$ equals the colinear chromatic number $\lambda(G)$ of G.

Algorithm 1: Algorithm Colinear_Coloring

$\kappa : V(D_G) \to [\rho(D_G)]$ is a colinear coloring of the vertices of G, and prove that the size $\rho(D_G)$ of a minimum path cover $\mathcal{P}(D_G)$ of the DAG D_G is equal to the colinear chromatic number $\lambda(G)$ of the graph G.

Proposition 2.3. *Let G be a graph and let D_G be the DAG associated to the graph G. A colinear coloring of the graph G can be obtained by assigning a particular color to all vertices of each path of a path cover of the DAG D_G. Moreover, the size $\rho(D_G)$ of a minimum path cover $\mathcal{P}(D_G)$ of the DAG D_G equals to the colinear chromatic number $\lambda(G)$ of the graph G.*

Proof. Let G be a graph, D_G be the DAG associated to G, and let $\mathcal{P}(D_G)$ be a minimum path cover of D_G. The size $\rho(D_G)$ of the DAG D_G, equals to the minimum number of directed paths in D_G needed to cover the vertices of D_G and, thus, the vertices of G. Now, consider a coloring $\kappa : V(D_G) \to \{1, 2, \ldots, k\}$ of the vertices of D_G, such that vertices belonging to the same path are assigned the same color and vertices of different paths are assigned different colors. Therefore, we have $\rho(D_G)$ colors and $\rho(D_G)$ sets of vertices, one for each color. For every set of vertices belonging to the same path, their corresponding closed neighborhood sets can be linearly ordered by inclusion. Indeed, consider a path in D_G with vertices $\{v_1, v_2, \ldots, v_m\}$ and edges $\overrightarrow{v_i v_{i+1}}$ for $i \in \{1, 2, \ldots, m\}$. From the construction of D_G, it holds that $\forall i, j \in \{1, 2, \ldots, m\}$, $\overrightarrow{v_i v_j} \in E(D_G) \Leftrightarrow N_G[v_i] \subseteq N_G[v_j]$. In other words, the corresponding neighborhood sets of the vertices belonging to a path in D_G are linearly ordered by inclusion. Thus, the coloring κ of the vertices of D_G gives a colinear coloring of G.

This colinear coloring κ is optimal, uses $k = \rho(D_G)$ colors, and gives the colinear chromatic number $\lambda(G)$ of the graph G. Indeed, suppose that there exists a different colinear coloring $\kappa' : V(D_G) \to [k']$ of G using k' colors, such that $k' < k$. For every color given in κ', consider a set consisted of the vertices assigned that color. It is true that for the vertices belonging to the same set, their neighborhood sets are linearly ordered by inclusion. Therefore, these vertices can belong to the same path in D_G. Thus, each set of vertices in G corresponds to a path in D_G and, additionally, all vertices of G (and therefore of D_G) are covered. This is a path cover of D_G of size $\rho'(D_G) = k' < k = \rho(D_G)$, which is a contradiction since $\mathcal{P}(D_G)$ is a minimum path cover of D_G. Therefore, we conclude that the colinear coloring $\kappa : V(D_G) \to [\rho(D_G)]$ is optimal and, hence, $\rho(D_G) = \lambda(G)$. ∎

Complexity of the algorithm. Let G be a graph, $V(G) = n$, $E(G) = m$, and let D_G be the DAG associated to the graph G. Step (i) of the algorithm, which includes the construction of the DAG D_G,

takes $O(nm)$ time. In particular, it takes $O(nm)$ time to compute the closed neighborhood set of every vertex of G, $O(nm)$ time to find the inclusion relations among the neighborhood sets, and $O(n+m)$ time to construct the DAG D_G. Note that, we only need to check pairs of vertices that are connected by an edge in G. Step (ii) computes a minimum path cover in the transitive DAG D_G; the problem is known to be polynomially solvable, since it can be reduced to the maximum matching problem in a bipartite graph formed from the transitive DAG [9]. The maximum matching problem in a bipartite graph takes $O((m+n)\sqrt{n})$ time, due to an algorithm by Hopcroft and Karp [44]. Finally, both Steps (iii) and (iv) can be executed in $O(n)$ time. Therefore, the complexity of the algorithm is $O(nm + n\sqrt{n})$.

2.4 Graphs having the χ-colinear and α-colinear Properties

In Section 2.2 we showed that for any graph G, the colinear chromatic number $\lambda(\overline{G})$ of the graph $\overline{G}$ is an upper bound for the chromatic number $\chi(G)$ of G, i.e., $\chi(G) \leq \lambda(\overline{G})$. Recall that a known lower bound for the chromatic number of G is the clique number $\omega(G)$ of G, i.e., $\chi(G) \geq \omega(G)$. Motivated by the Perfect Graph Theorem [37], in this section we exploit our results on colinear coloring and we study those graphs for which the equality $\chi(G) = \lambda(\overline{G})$ holds for every induced subgraph. The outcome of this study was the definition of the following two graph properties and the characterization of known graph classes in terms of these properties.

- **χ-colinear property.** A graph G has the χ-colinear property if for every induced subgraph G_A of the graph G, $\chi(G_A) = \lambda(\overline{G}_A)$, $A \subseteq V(G)$.
- **α-colinear property.** A graph G has the α-colinear property if for every induced subgraph G_A of a graph G, $\alpha(G_A) = \lambda(G_A)$, $A \subseteq V(G)$.

Next, we show that the class of threshold graphs is characterized by the χ-colinear property and the class of quasi-threshold graphs is characterized by the α-colinear property. We also show that any graph that has the χ-colinear property is perfect; actually, we show that any graph that has the χ-colinear property is a co-chordal graph. We first give some definitions and show some interesting results.

Definition 2.3. An edge uv of a graph G is called *actual* if neither $N_G[u] \subseteq N_G[v]$ nor $N_G[u] \supseteq N_G[v]$. The set of all actual edges of G will be denoted by $E_\alpha(G)$.

Definition 2.4. A graph G is called *quasi-threshold* if it has no induced subgraph isomorphic to a C_4 or a P_4 or, equivalently, if it contains no actual edges.

More details on actual edges and characterizations of quasi-threshold graphs through a classification of their edges can be found in [59]. The following result directly follows from Definition 2.3 and Proposition 4.11.

Proposition 2.4. *Let $\kappa : V(G) \to \{1, 2, \ldots, k\}$ be a k-colinear coloring of the graph G. If the edge $uv \in E(G)$ is an actual edge of G, then $\kappa(u) \neq \kappa(v)$.*

Based on Definition 2.3, the χ-colinear property, and Proposition 5.1, we prove the following result.

Proposition 2.5. *Let G be a graph and let F be the graph such that $V(F) = V(G)$ and $E(F) = E(G) \cup E_\alpha(\overline{G})$. The graph G has the χ-colinear property if $\chi(G_A) = \omega(F_A)$, $\forall A \subseteq V(G)$.*

Proof. Let G be a graph and let F be a graph such that $V(F) = V(G)$ and $E(F) = E(G) \cup E_\alpha(\overline{G})$, where $E_\alpha(\overline{G})$ is the set of all actual edges of $\overline{G}$. By definition, G has the χ-colinear property if $\chi(G_A) = \lambda(\overline{G}_A)$, $\forall A \subseteq V(G)$. It suffices to show that $\lambda(\overline{G}_A) = \omega(F_A)$, $\forall A \subseteq V(G)$. From Definition 4.2, it is easy to see that two vertices which are not connected by an edge in $\overline{G}_A$ belong necessarily to different cliques and, thus, they cannot receive the same color in a colinear coloring of $\overline{G}_A$. In other words, the vertices which are connected by an edge in G_A cannot take the same color in a colinear coloring of $\overline{G}_A$. Moreover,

from Proposition 2.4 vertices which are endpoints of actual edges in $\overline{G}_A$ cannot take the same color in a colinear coloring of $\overline{G}_A$.

Next, we construct the graph F_A with vertex set $V(F_A) = V(G_A)$ and edge set $E(F_A) = E(G_A) \cup E_\alpha(\overline{G}_A)$, where $E_\alpha(\overline{G}_A)$ is the set of all actual edges of $\overline{G}_A$. Every two vertices in F_A, which have to take a different color in a colinear coloring of $\overline{G}_A$ are connected by an edge. Thus, the size of the maximum clique in F_A equals to the size of the maximum set of vertices which pairwise must take a different color in $\overline{G}_A$, i.e., $\omega(F_A) = \lambda(\overline{G}_A)$ holds for all $A \subseteq V(G)$. Concluding, G has the χ-colinear property if $\chi(G_A) = \omega(F_A)$, $\forall A \subseteq V(G)$. ∎

Taking into consideration Proposition 2.5 and the structure of the edge set $E(F) = E(G) \cup E_\alpha(\overline{G})$ of the graph F, it is easy to see that $E(F) = E(G)$ if $\overline{G}$ has no actual edges. Actually, this will be true for all induced subgraphs, since if G is a quasi-threshold graph then G_A is also a quasi-threshold graph for all $A \subseteq V(G)$. Thus, $\chi(G_A) = \omega(F_A)$, $\forall A \subseteq V(G)$. Therefore, the following result holds.

Corollary 2.2. *Let G be a graph. If $\overline{G}$ is quasi-threshold, then G has the χ-colinear property.*

Using Corollary 2.2 we can prove a more interesting result.

Proposition 2.6. *Any threshold graph has the χ-colinear property.*

Proof. Let G be a threshold graph. It has been proved that an undirected graph G is a threshold graph if and only if G and its complement $\overline{G}$ are quasi-threshold graphs [59]. From Corollary 2.2, if $\overline{G}$ is quasi-threshold then G has the χ-colinear property. Concluding, if G is threshold, then $\overline{G}$ is quasi-threshold and thus G has the χ-colinear property. ∎

We note that the proof that any threshold graph G has the χ-colinear property can be also obtained by showing that any coloring of a threshold graph G is a colinear coloring of $\overline{G}$ by using Proposition 4.11, Corollary 3.1, the fact that $N_G(u) = V(G) \setminus N_{\overline{G}}[u]$, and the property that $N(u) \subseteq N[v]$ or $N(v) \subseteq N[u]$ for any two vertices u, v of G. However, Proposition 2.5 and Corollary 2.2 actually give us a stronger result, since the class of quasi-threshold graphs is a superclass of the class of threshold graphs.

The following result is even more interesting, since it shows that any graph that has the χ-colinear property is a perfect graph.

Proposition 2.7. *Any graph that has the χ-colinear property is a co-chordal graph.*

Proof. Let G be a graph that has the χ-colinear property. It has been shown that a co-chordal graph is $(2K_2, \text{antihole})$-free [37]. To show that any graph G that has the χ-colinear property is a co-chordal graph we will show that if G has a $2K_2$ or an antihole as induced subgraph, then G is does not have the χ-colinear property. Since by definition a graph G has the χ-colinear property if the equality $\chi(G_A) = \lambda(\overline{G}_A)$ holds for every induced subgraph G_A of G, it suffices to show that the graphs $2K_2$ and antihole do not have the χ-colinear property.

The graph $2K_2$ does not have the χ-colinear property, since $\chi(2K_2) = 2 \neq 4 = \lambda(C_4)$; see Figure 2.1. Now, consider the graph $G = \overline{C}_n$ which is an antihole of size $n \geq 5$. We will show that $\chi(G) \neq \lambda(\overline{G})$. It follows that $\lambda(\overline{G}) = \lambda(C_n) = n \geq 5$, i.e., if the graph $\overline{G} = C_n$ is to be colored colinearly, every vertex has to take a different color. Indeed, assume that a colinear coloring $\kappa : V(G) \to \{1, 2, \ldots, k\}$ of $\overline{G} = C_n$ exists such that for some $u_i, u_j \in V(G)$, $i \neq j$, $1 \leq i, j \leq n$, $\kappa(u_i) = \kappa(u_j)$. Since u_i, u_j are vertices of a hole, their neighborhoods in $\overline{G}$ are $N[u_i] = \{u_{i-1}, u_i, u_{i+1}\}$ and $N[u_j] = \{u_{j-1}, u_j, u_{j+1}\}$, $2 \leq i, j \leq n-1$. For $i = 1$ or $i = n$, $N[u_1] = \{u_n, u_2\}$ and $N[u_n] = \{u_{n-1}, u_1\}$. Since $\kappa(u_i) = \kappa(u_j)$, from Proposition 4.11 we obtain that one of the inclusion relations $N[u_i] \subseteq N[u_j]$ or $N[u_i] \supseteq N[u_j]$ must hold in $\overline{G}$. Obviously this is possible if and only if $i = j$, for $n \geq 5$; this is a contradiction to the assumption that $i \neq j$. Thus, no two vertices in a hole take the same color in a colinear coloring. Therefore, $\lambda(\overline{G}) = n$. It suffices to show that $\chi(G) < n$. It is easy to see that for the antihole $\overline{C}_n$, $deg(u) = n - 3$, for every vertex $u \in V(G)$. Brook's theorem [11] states that for an arbitrary graph G and for all $u \in V(G)$,

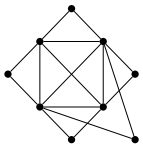

Figure 2.2: A graph G which is a split graph but it does not have the χ-colinear property, since $\chi(G) = 4$ and $\lambda(\overline{G}) = 5$.

$\chi(G) \leq max\{d(u) + 1\} = (n-3) + 1 = n - 2$. Therefore, $\chi(G) \leq n - 2 < n = \lambda(\overline{G})$. Thus the antihole $\overline{C}_n$ does not have the χ-colinear property.

We have showed that the graphs $2K_2$ and antihole do not have the χ-colinear property. It follows that any graph that has the χ-colinear property is $(2K_2, \text{antihole})$-free and, thus, any graph that has the χ-colinear property is a co-chordal graph. ∎

Since graphs having the χ-colinear property are perfect, it follows that any graph G having the χ-colinear property satisfies $\chi(G_A) = \omega(G_A) = \alpha(\overline{G_A})$, $\forall A \subseteq V(G)$. Therefore, the following result holds.

Proposition 2.8. *A graph G has the α-colinear property if and only if the graph $\overline{G}$ has the χ-colinear property.*

From Corollary 2.2 and Proposition 2.8 we can obtain the following result.

Proposition 2.9. *Any quasi-threshold graph has the α-colinear property.*

In this section we defined the χ-colinear and α-colinear properties and characterized known graph classes in terms of these properties. Based on these results, we next study the graphs which are characterized completely by the χ-colinear or α-colinear property.

2.5 Colinear and Linear Graphs

In Section 2.4 we showed that any threshold graph has the χ-colinear property and any quasi-threshold graph has the α-colinear property. In this section we study the graphs that are characterized completely by the χ-colinear property or the α-colinear property. We call these graphs colinear and linear graphs and as we next show they constitute two new classes of perfect graphs.

Definition 2.5. A graph G is called colinear if and only if G has the χ-colinear property, i.e., $\chi(G_A) = \lambda(\overline{G}_A)$, $\forall A \subseteq V(G)$. A graph G is called linear if and only if G has the α-colinear property, i.e., $\alpha(G_A) = \lambda(G_A)$, $\forall A \subseteq V(G)$.

From Proposition 2.6 we know that any threshold graph is a colinear graph. However, not any colinear graph is a threshold graph. Indeed, Chvátal and Hammer [17] showed that threshold graphs are $(2K_2, P_4, C_4)$-free and, thus, the graphs P_4 and C_4 are colinear graphs but they are not threshold graphs (see Figure 2.1). Therefore, we directly obtain the following result concerning the class of colinear graphs.

Proposition 2.10. *Colinear graphs form a superclass of threshold graphs.*

Moreover, from Proposition 2.7 we have that any colinear graph is a co-chordal graph. However, the reverse is not always true. For example, the graph G in Figure 3.4 is a co-chordal graph but it is not a colinear graph. Indeed, $\chi(G) = 4$ and $\lambda(\overline{G}) = 5$. It is easy to see that this graph is also a split graph. Moreover, not any colinear graph is a split graph, since the graph C_4 is colinear but it is not a split graph. However, there exist split graphs which are also colinear graphs; an example is the graph C_3. Recall that a graph G is a split graph if there is a partition of the vertex set $V(G) = K + I$, where K induces a

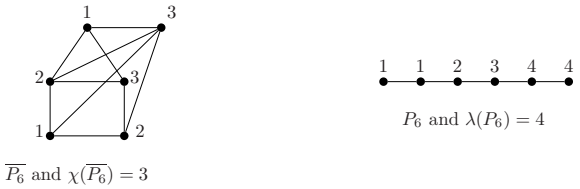

Figure 2.3: Illustrating the graph $\overline{P}_6$ which is not a colinear graph, since $\chi(\overline{P}_6) \neq \lambda(P_6)$.

clique in G and I induces an independent set; split graphs are characterized as $(2K_2, C_4, C_5)$-free graphs. Thus, the following result holds.

Proposition 2.11. *Colinear graphs form a subclass of co-chordal graphs.*

We have proved that colinear graphs do not contain a $2K_2$ or an antihole. Note that, since $\overline{C_5} = C_5$ and also the chordless cycle C_n is not $2K_2$-free for $n \geq 6$, it is easy to see that colinear graphs are hole-free. In addition, the graph $\overline{P}_6$ is not a colinear graph (see Figure 2.3). Thus, we obtain the following result.

Proposition 2.12. *If a graph G is colinear, then G is a $(2K_2, \text{antihole}, \overline{P}_6)$-free graph.*

From Proposition 2.9 we obtain that any quasi-threshold graph is a linear graph. Again, the reverse is not always true; an example is the graph P_4, which is a linear graph but not a quasi-threshold graph. Therefore, the following result holds.

Proposition 2.13. *Linear graphs form a superclass of quasi-threshold graphs.*

From Propositions 2.12 and 2.8 we obtain that linear graphs are (C_4, hole, P_6)-free graphs. Therefore, it follows that any linear graph is chordal. However, the reverse is not always true, i.e., not any chordal graph is linear; an obvious example is the graph P_6. Another interesting example is the complement $\overline{G}$ of the graph illustrated in Figure 3.4, which is a chordal graph but not a linear graph. Indeed, $\alpha(\overline{G}) = 4$ and $\lambda(\overline{G}) = 5$. It is easy to see that this graph is also a split graph. Moreover, not any linear graph is a split graph, since the graph $2K_2$ is linear but it is not a split graph. However, there exist split graphs that are linear graphs; an example is the graph C_3. Therefore, the following result holds.

Proposition 2.14. *Linear graphs form a subclass of chordal graphs.*

Proposition 2.14 implies that linear graphs are perfect graphs and, thus, it follows that any linear graph satisfies $\alpha(G_A) = \omega(\overline{G_A}) = \chi(\overline{G_A})$, $\forall A \subseteq V(G)$. Therefore, from Corollary 3.1 we obtain the following characterization.

Proposition 2.15. *Linear graphs are those graphs G for which the colinear chromatic number achieves its theoretical lower bound in every induced subgraph of G.*

From the results proved in this section, we conclude that colinear and linear graphs form two new classes of perfect graphs. The inclusion relations among the classes of colinear graphs, linear graphs, and other subclasses of co-chordal and chordal graphs are depicted in Figure 3.3.

In the next section we prove structural properties of linear graphs by studying the relation between the class of strongly chordal graphs, which is a known subclass of chordal graphs [10, 27], and the class of linear graphs.

2.6 Structural Properties

In this section we prove structural properties of linear graphs, by investigating the structure of their forbidden induced subgraphs. In particular, we prove that any P_6-free chordal graph which is not a

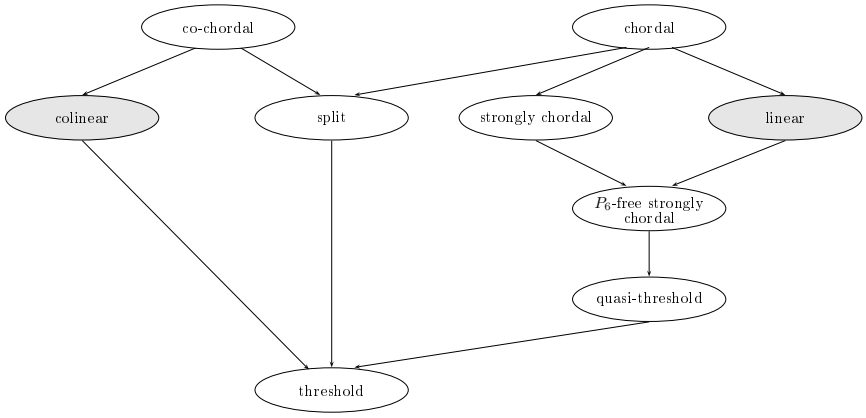

Figure 2.4: Illustrating the inclusion relations among the classes of colinear graphs, linear graphs, and other classes of perfect graphs.

linear graph properly contains a k-sun as an induced subgraph. Let us give the definitions of a k-sun and an incomplete k-sun. An incomplete k-sun S_k ($k \geq 3$) is a chordal graph on $2k$ vertices whose vertex set can be partitioned into two sets, $U = \{u_1, u_2, \ldots, u_k\}$ and $W = \{w_1, w_2, \ldots, w_k\}$, so that W is an independent set, and w_i is adjacent to u_j if and only if $i = j$ or $i = j + 1 \pmod{k}$. A k-sun is an incomplete k-sun S_k in which U is a complete graph.

The following definitions and results on strongly chordal graphs given in [14, 27], turn up to be useful in proving structural properties of linear graphs.

Definition 2.6. (Farber [27]) A vertex ordering $\sigma = (v_1, v_2, \ldots, v_n)$ is a strong elimination ordering of a graph G iff σ is a perfect elimination ordering and also has the property that for each i, j, k and ℓ, if $i < j$, $k < \ell$, $v_k, v_\ell \in N[v_i]$, and $v_k \in N[v_j]$, then $v_\ell \in N[v_j]$. A graph is strongly chordal iff it admits a strong elimination ordering.

A vertex v of a graph G is called simple if $\{N[x] : x \in N[v]\}$ is linearly ordered by inclusion. It has been proved that a strong elimination ordering of a graph G is a vertex ordering $(v_1, v_2, \ldots, v_n)$ such that for every $i \in \{1, 2, \ldots, n\}$ the vertex v_i is simple in G_i and also $N_{G_i}[v_\ell] \subseteq N_{G_i}[v_k]$ whenever $i \leq \ell \leq k$ and $v_\ell, v_k \in N_{G_i}[v_i]$ [14]; recall that for a given vertex ordering $(v_1, v_2, \ldots, v_n)$ of a graph G, we denote by G_i the subgraph of G induced by the set of vertices $\{v_i, v_{i+1}, \ldots, v_n\}$. Additionally, a graph G is strongly chordal if and only if every induced subgraph of G has a simple vertex. Actually, if G is a non-trivial strongly chordal graph, then G has at least two simple vertices [27].

The following characterization of strongly chordal graphs was proved by Farber [27].

Proposition 2.16. *(Farber [27]) A chordal graph G is strongly chordal if and only if it contains no induced k-sun.*

We next prove the main result of this section. Let $\mathcal{F}$ be the family of all the minimal forbidden induced subgraphs of the class of linear graphs, and let F_i be a member of $\mathcal{F}$ which is a P_6-free chordal graph. We show that F_i properly contains a k-sun ($k \geq 3$) as an induced subgraph. It is easy to see that, due to Proposition 3.2, it suffices to show both that any P_6-free strongly chordal graph is a linear graph, and that the k-sun ($k \geq 3$) is a linear graph.

The proof that a k-sun ($k \geq 3$) is a linear graph is given in Lemma 2.3. In order to show that a P_6-free strongly chordal graph G is a linear graph, we will prove that $\alpha(G_A) = \lambda(G_A)$, $\forall A \subseteq V(G)$. The proof is completed in the following four parts:

(I) we construct a strong elimination ordering σ and a maximum independent set I of G with special properties,

(II) we compute a vertex coloring κ for the graph G using $\alpha(G) = |I|$ colors,

(III) we show that κ is an optimal colinear coloring of G, and

(IV) we show that the equality $\lambda(G_A) = \alpha(G_A)$ holds for every induced subgraph G_A of G.

Next, we present our proof in detail. Throughout this section we denote by L the set of all the simple vertices of G and by S the set of all the simplicial vertices of G; note that $L \subseteq S$ since a simple vertex is also a simplicial vertex.

Part (I): Construction of I and σ. Let G be a P_6-free strongly chordal graph, and let L be the set of all the simple vertices in G. From Definition 2.6, G admits a strong elimination ordering. Using a modified version of the algorithm given by Farber in [27] we construct a strong elimination ordering $\sigma = (v_1, v_2, \ldots, v_n)$ of the graph G having specific properties. Our algorithm also constructs the maximum independent set I of G; since G is a chordal graph and σ is a perfect elimination ordering, we can use a known algorithm (e.g. see [37]) to compute a maximum independent set of the graph G. Throughout the algorithm, we denote by G_i the subgraph of G induced by the set of vertices $V(G)\backslash\{v_1, v_2, \ldots, v_{i-1}\}$, where $v_1, v_2, \ldots, v_{i-1}$ are the vertices which have already been added to the ordering σ during the construction. Moreover, we denote by I^* the set of vertices which have not been added to σ yet and additionally do not have a neighbor already added to σ which belongs to I.

Algorithm 2 is a modified version of the algorithm given by Farber [27] for constructing a strong elimination ordering σ and a maximum independent set I of G. Our algorithm in each iteration of Steps 3–5 adds to the ordering σ all the vertices which are simple in G_i, while Farber's algorithm selects only one simple vertex of G_i and adds it to σ. We note that L_i is the set of all the simple vertices of G_i, and also v_k for $k = i$ is that vertex of L_i which is added first to the ordering σ. It is easy to see that the constructed ordering σ is a strong elimination ordering of G, since every vertex which is simple in G is also simple in every induced subgraph of G. Clearly, the constructed set I is a maximum independent set of G.

From the fact that G is a P_6-free strongly chordal graph and from the construction of I and σ we obtain the following properties. Recall that the *distance* $d(v, u)$ from vertex v to vertex u is the minimum length of a path from v to u (note that within this chapter we consider length of a path the number of edges in the path); $d(v, u) = \infty$ if there is no path from v to u.

Property 2.1. *Let G be a P_6-free strongly chordal graph and let L be the set of all the simple vertices of G. For each vertex $v_x \notin L$, there exists a chordless path of length at most 4 connecting v_x to any vertex $v \in L$.*

Property 2.2. *Let G be a P_6-free strongly chordal graph, let L be the set of all the simple vertices of G, and let I and $\sigma = (v_1, v_2, \ldots, v_n)$ be the maximum independent set and the ordering, respectively, constructed by our algorithm. Then,*

(i) if $v_i \notin L$ and $i < j$, then $v_j \notin L$;

(ii) for each vertex $v_x \notin I$, there exists a vertex $v_i \in I$, $i < x$, such that $v_x \in N_{G_i}[v_i]$.

Next, in Part (II) we describe an algorithm for computing a vertex coloring κ of G using exactly $\alpha(G)$ colors and, then, in Part (III) we show that κ is a colinear coloring of G.

Part (II): The coloring κ of G. Let G be a P_6-free strongly chordal graph, and let L (resp. S) be the set of all the simple (resp. simplicial) vertices in G. We consider a maximum independent set I and a strong elimination ordering σ of G, as constructed in Part (I). Now, in order to compute the colinear chromatic number $\lambda(G)$ of G, we compute a vertex coloring κ of G using $\alpha(G)$ colors and, then, we show that κ is a colinear coloring of G. Actually, in Parts (II)-(III) we show that we can compute a colinear coloring of any P_6-free strongly chordal graph with $\lambda(G) = \alpha(G)$ colors, by using the constructed strong elimination ordering σ of G.

Algorithm Strong_Elimination_Ordering

This algorithm is a modified version of Farber's algorithm for constructing a strong elimination ordering σ and a maximum independent set I of a strongly chordal graph G.

Input: a strongly chordal graph G;

Output: a strong elimination ordering σ and a maximum independent set I of G;

1. `set` $I = \emptyset$, $I^* = V(G)$, $\sigma = \emptyset$, $n = |V(G)|$, and $V_0 = V(G)$;

2. Let $(V_0, <_0)$ be the partial ordering on V_0 in which $v <_0 u$ if and only if $v = u$.
   `set` $V_1 = V(G)$ and $i = 1$;

3. Let G_i be the subgraph of G induced by V_i, that is, $V_i = V(G_i)$.
   `construct` an ordering on V_i by $v <_i u$ if $v <_{i-1} u$ or $N_{G_i}[v] \subset N_{G_i}[u]$;

4. Let L_i be the set of all the simple vertices in G_i.
   $k = i$;

   `while` $L_i \neq \emptyset$ `do`
   - `construct` an ordering on V_k by $v <_k u$ if $v <_{k-1} u$ or $N_{G_k}[v] \subset N_{G_k}[u]$;
     `choose` a vertex v_k which belongs to L_i and is minimal in $(V_k, <_k)$, and add it to σ;
     `set` $V_{k+1} = V_k \backslash \{v_k\}$ and $L_i = L_i \backslash \{v_k\}$;
   - `if` $v_k \in I^*$ `then`
     `set` $I = I \cup \{v_k\}$ and $I^* = I^* \backslash \{v_k\}$;
     `delete` all neighbors of v_k from I^*;
   - `set` $k = k + 1$;

   `end-while;`

   $i = k$;

5. `if` $i = n + 1$ `then` output the ordering $\sigma = (v_1, v_2, \ldots, v_n)$ of $V(G)$ and `stop`;
   `else` go to step 3;

Algorithm 2: Algorithm Strong_Elimination_Ordering

First, we compute a vertex coloring κ of G using $\alpha(G)$ colors as follows:

1. Successively visit the vertices in the ordering σ from left to right, and assign the color $\kappa(v_i)$ to the first vertex $v_i \in I$ which has not been assigned a color yet.

2. For every uncolored vertex $v_k \in N_{G_i}(v_i)$, if the collection $\{N_G[v_j] : v_j \in N_{G_i}[v_i] \text{ and } \kappa(v_j) = \kappa(v_i)\} \cup \{N_G[v_k]\}$ is linearly ordered by inclusion, then assign the color $\kappa(v_k) = \kappa(v_i)$ to the vertex v_k.

3. Repeat steps 1 and 2 until there are no uncolored vertices $v_i \in I$ in G.

Based on this process, we obtain that every vertex v_i belonging to the maximum independent set I of G is assigned a different color in step 1, and for each such vertex v_i the collection $\{N_G[v_j] : v_j \in N_{G_i}[v_i] \text{ and } \kappa(v_j) = \kappa(v_i)\}$ is linearly ordered by inclusion. Therefore, we have assigned $\alpha(G)$ colors to the vertices of G. Now, if we show that there is no vertex in σ which has not been assigned a color, then it follows that κ is a colinear coloring of G with $\alpha(G)$ colors, since by the computation of κ the collection $\{N_G[v_\ell] : \kappa(v_\ell) = j\}$ is linearly ordered by inclusion for all $j \in \{1, 2, \ldots, \alpha(G)\}$.

The following property will be used for proving Lemma 2.1. The property holds, since simple vertices

are simplicial vertices, and for every simple vertex $v \in L$ the set $\{N_G[v_x] : v_x \in N_{G_i}[v]\}$ is linearly ordered by inclusion.

Property 2.3. *For every simple vertex $v_i \in L \cap I$ of G, every uncolored vertex $v_x \in N_{G_i}[v_i]$ is assigned the color $\kappa(v_x) = \kappa(v_i)$ during the coloring κ of G. Additionally, for each vertex $v_x \notin L$, if there exists a vertex $v \in L$ such that $vv_x \in E(G)$, then v_x is assigned the color $\kappa(v_x) = \kappa(v')$ from a simple vertex $v' \in L$, $v' \leq v$.*

Note that κ is not a proper vertex coloring of G. Actually, since Lemma 2.1 holds, from Proposition 4.10 it follows that κ is a proper vertex coloring of $\overline{G}$.

Part (III): The coloring κ is a colinear coloring of G. In this part we prove the following result, by showing that there is no vertex in σ which has not been assigned a color during the coloring κ.

Lemma 2.1. *The coloring κ is a colinear coloring of G.*

Proof. Let G be a P_6-free strongly chordal graph, and let L (resp. S) be the set of all the simple (resp. simplicial) vertices in G. We consider a maximum independent set I, a strong elimination ordering σ, and a coloring κ of G, as computed above. Hereafter, for two vertices v_i and v_j in the ordering σ, we say that $v_i < v_j$ if the vertex v_i appears before the vertex v_j in σ.

Next, we show that there is no vertex in σ which has not been assigned a color during the coloring κ, and since the collection $\{N_G[v_\ell] : \kappa(v_\ell) = j\}$ is linearly ordered by inclusion for all $j \in \{1, 2, \ldots, \alpha(G)\}$, from Proposition 4.11 it follows that κ is a colinear coloring of G.

Assume that there exists at least one uncolored vertex v_j in G. It follows that $v_j \notin I$, since otherwise v_j would have been assigned a color in step 1 of the coloring κ. Therefore, from Property 2.2(ii) v_j has a neighbor to its left in σ which belongs to the independent set I. Let v_i be the leftmost vertex in σ which belongs to the independent set I and did not color all its neighbors to its right in σ, and let v_j be the leftmost such uncolored neighbor of v_i in σ. Next, we distinguish two cases regarding the vertex $v_i \in I$; in the first case we consider v_i to be a simplicial vertex, i.e., $v_i \in S$, and in the second case we consider $v_i \notin S$. In both cases we show that our assumptions come to a contradiction.

Case 1: The vertex $v_i \in I$ and $v_i \in S$. Since σ is a strong elimination ordering, each vertex $v_i \in I$ is simple in G_i and, thus, $\{N_{G_i}[v_k] : v_k \in N_{G_i}[v_i]\}$ is linearly ordered by inclusion. Also, by definition, if $v_i \in L$ then the collection $\{N_G[v_k] : v_k \in N_{G_i}[v_i]\}$ is linearly ordered by inclusion. Thus, $v_i \in I \cap S$ and $v_i \notin L$, since otherwise v_j could have been assigned the color $\kappa(v_j) = \kappa(v_i)$.

Therefore, there exists a neighbor v_k of v_i such that $v_i < v_k < v_j$, $\kappa(v_k) = \kappa(v_i)$, and neither $N_G[v_k] \subseteq N_G[v_j]$ nor $N_G[v_k] \supseteq N_G[v_j]$; recall that $N_{G_i}[v_k] \subseteq N_{G_i}[v_j]$. In the case where the equality $N_{G_i}[v_k] = N_{G_i}[v_j]$ holds, without loss of generality, we may assume that the degree of v_k in G is less than or equal to the degree of v_j in G (note that in this case σ is still a strong elimination ordering).

Since neither $N_G[v_k] \subseteq N_G[v_j]$ nor $N_G[v_k] \supseteq N_G[v_j]$, there exist vertices v_2 and v_3 in G such that $v_2 \in N_G[v_k]$, $v_2 \notin N_G[v_j]$, $v_3 \in N_G[v_j]$, and $v_3 \notin N_G[v_k]$. Since $N_{G_i}[v_k] \subseteq N_{G_i}[v_j]$, it is easy to see that $v_2 < v_i$ in σ. By the assumption that v_i is the leftmost vertex in σ which belongs to the independent set I and has not colored all its neighbors to the right, and since $\kappa(v_k) = \kappa(v_i)$ it follows that $v_2 \notin I$. Thus, from Property 2.2(ii) there exists a vertex $v_4 \in I$, such that $v_4 < v_2$ and $v_2 \in N_G[v_4]$. Additionally, since $\kappa(v_k) = \kappa(v_i)$ and G is chordal it holds that $v_k, v_j \notin N_G[v_4]$. Hence, the subgraph of G induced by the vertices $\{v_4, v_2, v_k, v_j, v_3\}$ is a P_5. Concerning now the position of the vertex v_3 in the ordering σ, we can have either $v_3 < v_i$ or $v_3 > v_i$. We will show that in both cases we come to a contradiction to our initial assumptions; that is, either it results that G has a P_6 as an induced subgraph or that the vertices should be added to σ in an order different than the one originally assumed.

Case 1.1. $v_3 < v_i$. Assume that v_j has a neighbor $v_3 < v_i$. Since v_i is the leftmost vertex in σ which belongs to the independent set I and has not colored all its neighbors to the right, it follows that $v_3 \notin I$, since otherwise v_j would have taken the color $\kappa(v_j) = \kappa(v_3)$ during the coloring κ of G. Thus, similarly

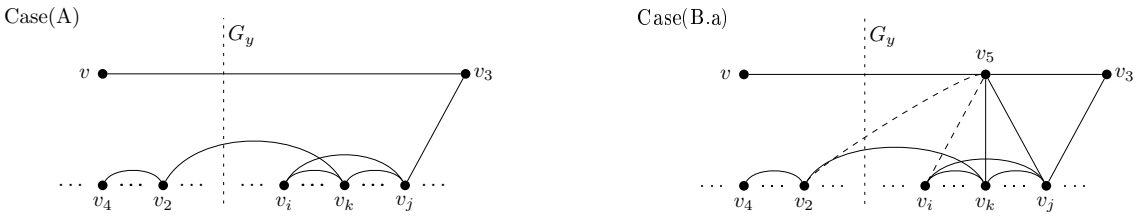

Figure 2.5: Illustrating Case (A) and Case (B.a)

to the above, from Property 2.2(ii) there exists a vertex $v_5 \in I$, such that $v_5 < v_3$ and $v_3 \in N_G[v_5]$. Therefore, the vertices $\{v_4, v_2, v_k, v_j, v_3, v_5\}$ induce a P_6 in G, which is also chordless since G is chordal.

Case 1.2. $v_3 > v_i$. Assume that v_j does not have a neighbor $v_3 < v_i$, i.e., it has a neighbor $v_3 > v_i$. Since $v_i \notin L$, from Property 2.2(i) it follows that $v_3 \notin L$. Thus, from Property 2.1 we obtain that there exists a chordless path of length at most 4 connecting $v_3 \notin L$ to any vertex $v \in L$. The vertex v_4 may be a simple vertex or not. However, we know that in a non-trivial strongly chordal graph there exist at least two non adjacent simple vertices [27]. Thus, there exists a vertex $v \in L$, $v \neq v_4$, such that the distance $d(v, v_3)$ of v_3 from v is at most 4, due to Property 2.1. Let $d_m(v_3, v) = max\{d(v_3, v) : \forall v \in L,\ v \neq v_4\}$. Since $v_3 \notin L$ and G is a P_6-free graph, it follows that $1 \leq d_m(v, v_3) \leq 4$.

Next, we distinguish four cases regarding the maximum distance $d_m(v_3, v)$ and show that each one comes to a contradiction. In each case we have that $\{v_4, v_2, v_k, v_j, v_3\}$ is a chordless path on five vertices. We first explain what is illustrated in Figures 2.5 and 2.6. Let G_y be the induced subgraph of G, such that during the construction of σ the vertex v_i becomes simple in G_y, i.e., $v_i \in L_y$ and $v_y \leq v_i$. In the two figures, the vertices are placed on the horizontal dotted line in the order that appear in the ordering σ. For the vertices which are not placed on the dotted line, we are only interested about illustrating the edges among them. The vertices which are to the right of the vertical dashed line belong to the induced subgraph G_y of G. The dashed edges illustrate edges that may or may not exist in the specific case. Next, we distinguish the four cases, and show that each one of them comes to a contradiction:

Case (A): $d_m(v_3, v) = 1$.

It is easy to see that $v_j v \notin E(G)$, since otherwise v_j would have been assigned the color $\kappa(v)$ due to Property 2.3, and would not be an uncolored neighbor of v_i as assumed. Thus, in this case there exists a P_6 in G induced by the vertices $\{v_4, v_2, v_k, v_j, v_3, v\}$; since G is a chordal graph, other edges among the vertices of this path do not exist. This is a contradiction to our assumption that G is a P_6-free graph.

Case (B): $d_m(v_3, v) = 2$.

In this case there exists a vertex v_5 such that $\{v_3, v_5, v\}$ is a chordless path from v_3 to v. It follows that there exists a P_7 induced by the vertices $\{v_4, v_2, v_k, v_j, v_3, v_5, v\}$. Having assumed that G is a P_6-free graph, the path $\{v_4, v_2, v_k, v_j, v_3\}$ is chordless and $v_j, v_k \notin N_G[v]$ due to Property 2.3, we obtain that $v_j v_5 \in E(G)$ and $v_k v_5 \in E(G)$. Next, we distinguish three cases regarding the neighborhood of the vertex v_3 in G and show that each one comes to a contradiction.

(B.a) The vertex v_3 does not have neighbors in G other than v_5 and v_j. We will show that v_3 becomes simple before v_j becomes simple. Assume otherwise that v_j becomes simple not after v_3 becomes simple. Since by assumption $v_k < v_j$ we know that v_k becomes simple not after v_j becomes simple. Therefore, v_k becomes simple not after v_3 becomes simple. Assume that v_k becomes simple in a subgraph G' of G. We have assumed that $v_2 < v_i < v_k$ and, thus, v_2 and v_i have been already added to σ. It follows that v_5 has not been already added to σ,

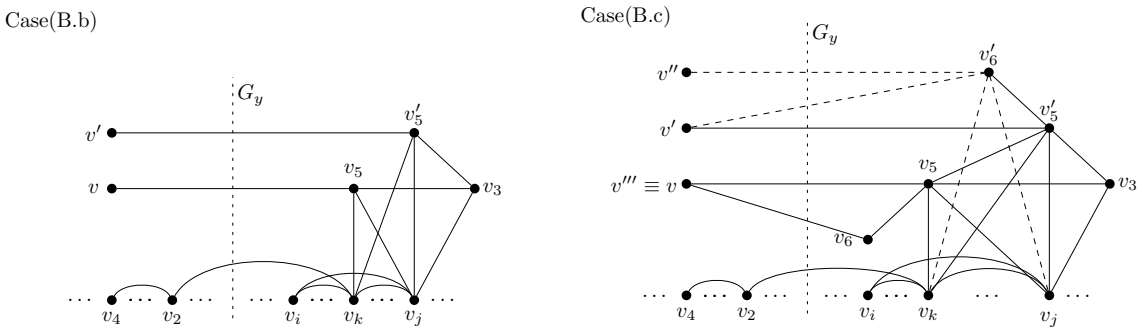

Figure 2.6: Illustrating Cases (B.b) and (B.c) of the proof.

since it cannot become simple before at least one between v_k and v_3 is added to σ. Therefore, when v_k becomes simple in G', it follows that either $N_{G_i}[v_5] \subseteq N_{G_i}[v_j]$ or $N_{G_i}[v_5] \supseteq N_{G_i}[v_j]$. Therefore, since we have assumed that v_3 does not have neighbors in G other than v_5 and v_j, it follows that v_3 becomes also simple in G', along with v_k. However, v_j is not simplicial in G' since $v_k v_3 \notin E(G)$ and, thus, v_j is not simple in G'. Therefore, v_3 will be added to σ before v_j will be added to σ.

Additionally, since v_5 sees the simple vertex v, from Property 2.3 it follows that v_5 will be assigned a color from a simple vertex and $v_5 \notin I$. Moreover, by assumption $v_j \notin I$. Therefore, $v_3 \in I$ and since we have showed that $v_3 < v_j$, it follows that v_j is the only uncolored neighbor of v_3 to its right in σ and, thus, v_j will be assigned the color $\kappa(v_j) = \kappa(v_3)$. This is a contradiction to our assumption that v_j has not been assigned a color.

So far, we have shown that if the vertex v_3 does not have neighbors in G other than v_5 and v_j, then we come to a contradiction to our assumptions. Since we initially assumed that $v_3 > v_i$ in σ, i.e., that v_3 does not become simple before v_i becomes simple, we continue by examining the cases where v_3 has neighbors in G_y other than v_5 and v_j.

(B.b) The vertex v_3 has two neighbors v_5 and v_5' in G_y, such that $v_5 v_5' \notin E(G)$. Since we have assumed that the maximum distance of the vertex v_3 from v in G, for any vertex $v \in L$, $v \neq v_4$, is $d_m(v_3, v) = 2$, and v_3 has no neighbor belonging to L since G is a P_6-free graph, it follows that $v_5, v_5' \notin L$ and there exist vertices $v, v' \in L$ such that the vertices $\{v_3, v_5, v\}$ induce a chordless path from v_3 to v and $\{v_3, v_5', v'\}$ induce a chordless path from v_3 to v'. It is easy to see that $v \neq v'$ and $vv' \notin E(G)$ since G is a chordal graph. Therefore, from Case (B.a) we have $v_k, v_j \in N_G[v_5]$ and $v_k, v_j \in N_G[v_5']$. However, in this case there exists a C_4 in G induced by the vertices $\{v_5, v_3, v_5', v_k\}$, since by assumption $v_5 v_5' \notin E(G)$ and $v_3 v_k \notin E(G)$. Concluding, the vertex v_3 cannot have two neighbors v_5 and v_5' in G, such that $v_5 v_5' \notin E(G)$. Thus, $v_3 \in S$.

(B.c) The vertex v_3 has two neighbors v_5 and v_5' (where $v_5 \neq v_j$ and $v_5' \neq v_j$) in G_y, such that $v_5 v_5' \in E(G)$, but neither $N_{G_y}[v_5] \subseteq N_{G_y}[v_5']$ nor $N_{G_y}[v_5'] \subseteq N_{G_y}[v_5]$; thus, there exist vertices v_6 and v_6' in G_y such that $v_5 v_6 \in E(G)$ and $v_5 v_6' \notin E(G)$ and, also, $v_5' v_6' \in E(G)$ and $v_5' v_6 \notin E(G)$. Since $v_3 \in S$, it follows that $v_6, v_6' \notin N_G[v_3]$. Since $d_m(v_3, v) = 2$, there exists a vertex $v \in L$ such that $\{v_3, v_5, v\}$ is a chordless path from v_3 to v. Similarly, there exists a vertex $v' \in L$ such that $\{v_3, v_5, v'\}$ is a chordless path from v_3 to v'. We have that $v \neq v'$, $vv_5' \notin E(G)$ and $v'v_5 \notin E(G)$, since otherwise v and v' would not be simple in G. Additionally, $vv' \notin E(G)$, $vv_6' \notin E(G)$, and $v'v_6 \notin E(G)$, since G is a chordal graph. Therefore, from Case (B.a) we have $v_k, v_j \in N_G[v_5]$ and $v_k, v_j \in N_G[v_5']$. Assume that there exist vertices $v'', v''' \in L$, such that $v_6 v''' \in E(G)$ and $v_6' v'' \in E(G)$. It is easy to see that at least one of the equivalences

$v \equiv v'''$ and $v' \equiv v''$ holds, otherwise G has a P_6 induced by the vertices $\{v''', v_6, v_5, v'_5, v'_6, v''\}$. Without loss of generality, assume that $v \equiv v'''$ holds.

Since $v \in L$, $v_5, v_6 \in N_G[v]$, $v'_5 \in N_G[v_5]$, and $v'_5 \notin N_G[v_6]$, it follows that $N_G[v_6] \subset N_G[v_5]$. In the case where $v_k, v_j \notin N_G[v_6]$ we have $v_6 \in L$ and, thus, v_6 would be added to σ in the first iteration which is a contradiction to our assumption that $v_6 \in G_y$. Assume that $v_j v_6 \in E(G)$; it follows that $v_k v_6 \in E(G)$, since otherwise G has a P_6 induced by the vertices $\{v_4, v_2, v_k, v_j, v_6, v\}$. If $v' \equiv v''$, the same arguments hold for v'_6 too and, thus, if $v_j v'_6 \in E(G)$ then $v_k v'_6 \in E(G)$. In the case where $v' \neq v''$ we have $v'_6 v_k \in E(G)$, since otherwise G has a P_6 induced by the vertices $\{v_4, v_2, v_k, v'_5, v'_6, v''\}$. Thus, in any case $v_6, v'_6 \in N_G[v_k]$, and G has a 3-sun induced by the vertices $\{v_k, v_5, v'_5, v'_6, v_6, v_3\}$. Since other edges between the vertices of the 3-sun do not exist, it follows that at least one of the vertices v_6 and v'_6 does not belong to the neighborhood of v_k and, thus, of v_j in G. Without loss of generality, let v_6 be that vertex. Thus, $v_6 \in L$ and, subsequently, v_6 will be added to σ during the first iteration. Thus, v_3 is simple and will be added to σ during the second iteration, along with v_2, while v_i will be added to σ after the second iteration (i.e., $v_3 < v_y \leq v_i$). This is a contradiction to our assumption that $v_3 > v_i$.

Using similar arguments, we can prove that v_3 will be added to σ before v_i, even if there exist edges between v_2 and the vertices v_5, v'_5, v_6, and v'_6. Actually, it easily follows that $v_2 v_6 \notin E(G)$, since $v_6 v_k \notin E(G)$ and G is a chordal graph. Additionally, $v_2 v_5 \notin E(G)$, since we know that $v_5 v'_6 \notin E(G)$, $v_k v_3 \notin E(G)$ and v_2 is simple in G_2. Therefore, whether $v_2 v'_5, v_2 v'_6 \in E(G)$ or not, it does not change the fact that v_3 becomes simple after the first iteration and, thus, v_3 is added to σ before v_i. Note, that even in the case where $v \equiv v_4$ or $v' \equiv v_4$ (in the case where $v_4 \in L$), it similarly follows that $v'_6 \in L$ or $v_6 \in L$ respectively and, thus, v_3 becomes simple after the first iteration and is added to σ before v_i.

Case (C): $d_m(v_3, v) = 3$.

In this case there exist vertices v_5 and v_6 such that $\{v_3, v_5, v_6, v\}$ is a chordless path from v_3 to v. Since now G has a P_8, it follows that $v_5 v_j \in E(G)$ and, additionally, some other edges must exist among the vertices v_2, v_k, v_j, v_5, and v_6. In any case, we will prove that either $N_G[v_5] \subseteq N_G[v_j]$ or $N_G[v_j] \subseteq N_G[v_5]$ and, thus, $v_3 \in L$. Similarly to Case (B), we distinguish three cases regarding the neighborhood of the vertex v_3 in G and show that if $v_3 \notin L$ then each one comes to a contradiction.

(C.a) The vertex v_3 does not have neighbors in G other than v_5 and v_j. Since $v_5 v_j \in E(G)$ then some other edges must exist, since otherwise G has a P_7 induced by the vertices $\{v_4, v_2, v_k, v_j, v_5, v_6, v\}$.

- Consider the case where $v_k v_5 \in E(G)$. Then either $v_2 v_5 \in E(G)$ or $v_k v_6 \in E(G)$, since G has a P_6. In the case where $v_2 v_5 \in E(G)$ then $v_i v_5 \in E(G)$. In the case where $v_k v_6 \in E(G)$ then either $v_i v_5 \in E(G)$ or $v_i v_6 \in E(G)$. For both cases, assume that $v_i v_5 \in E(G)$. Since v_5 and v_j are adjacent to v_k and v_3, and $v_k v_3 \notin E(G)$, it follows that v_5 and v_j cannot be added to σ unless at least one of v_k and v_3 is added to σ. Additionally, by assumption, $v_i < v_k < v_j$ and $v_i < v_3$. Thus, when v_i is added to σ it follows that v_k and v_3 have not been added to σ yet and, thus, v_5 and v_j have not been added to σ yet neither, i.e., v_k, v_3, v_5, and v_j belong to G_i. Thus, $N_{G_i}[v_5] \supseteq N_{G_i}[v_j]$.

  If $v_2 v_5 \in E(G)$ then $N_{G_2}[v_5] \supseteq N_{G_2}[v_k]$. Then, since in Case 1.2 we have assumed that there exists no vertex v_x such that $v_x < v_i$, $v_x v_j \in E(G)$, and $v_x v_k \notin E(G)$; thus, for every neighbor v_x of v_j such that $v_2 < v_x < v_i$, it follows that $v_x v_5 \in E(G)$. Also, there exists no vertex v_x such that $v_x < v_2$ and $v_x v_j \in E(G)$. Indeed, if we assume otherwise then $v_x v_k \in E(G)$, and since $v_2 v_k \in E(G)$, it follows from Definition 2.6 that $v_2 v_j \in E(G)$. This is a contradiction on the choice of v_2. Summarizing, there exists no vertex v_x such

that $v_x < v_i$, $v_x v_j \in E(G)$, and $v_x v_5 \notin E(G)$. Therefore, since $N_{G_i}[v_5] \supseteq N_{G_i}[v_j]$, it follows that $N_G[v_5] \supseteq N_G[v_j]$. Thus, $v_3 \in L$ which is a contradiction.

Consider now the case where $v_2 v_5 \notin E(G)$. In other words, v_5 does not have a neighbor v_x in σ such that $v_x < v_i$, $v_x v_k \in E(G)$ and $v_x v_j \notin E(G)$. Then $v_k v_6 \in E(G)$ and either $v_i v_5 \in E(G)$ or $v_i v_6 \in E(G)$. We now show that in both cases $v_j v_6 \in E(G)$. If $v_i v_5 \in E(G)$ then either $v_6 < v_i$ or $v_6 v_j \in E(G)$, since $N_{G_i}[v_k] \subseteq N_{G_i}[v_j]$. However, even if $v_6 < v_i$ then again $v_6 v_j \in E(G)$, since we have proved that v_5 does not have a neighbor v_x in σ such that $v_x < v_i$, $v_x v_k \in E(G)$ and $v_x v_j \notin E(G)$. Also, if $v_i v_6 \in E(G)$ then again $v_6 v_j \in E(G)$, since $v_i \in S$. Therefore, in both cases $v_6 v_j \in E(G)$. We now show that v_5 does not have a neighbor v_x such that $v_x < v_i$ and $v_x v_j \notin E(G)$. Indeed, if such a vertex v_x exists then also $v_x v_k \notin E(G)$. If $v_x \in L$ then $\kappa(v_5) = \kappa(v)$; thus, we can prove similarly to Case (B.a) that $v_3 < v_j$ and $\kappa(v_j) = \kappa(v_3)$. In the case where $v_x \notin L$, then $d(v_x, v_\ell) \geq 1$ for any vertex $v_\ell \in L$; in this case it follows that G has a P_6 induced by the vertices $\{v_4, v_2, v_k, v_5, v_x, v_\ell\}$, since $v_x v_k \notin E(G)$. Therefore, we have showed that v_5 does not have a neighbor v_x such that $v_x < v_i$ and $v_x v_j \notin E(G)$. Assume that v_5 has a neighbor v_x such that $v_x > v_i$ and $v_x v_j \notin E(G)$. Then $v_x v_k \notin E(G)$ since $N_{G_i}[v_k] \subset N_{G_i}[v_j]$. Similarly to the above it follows that if v_5 has such a neighbor v_x then either $v_x \in L$ or G has a P_6. Therefore, we have showed that $N_G[v_5] \subseteq N_G[v_j]$ and, thus, $v_3 \in L$ which is a contradiction.

- Consider now the case where $v_k v_5 \notin E(G)$. Then $v_k, v_j \in N_G[v_6]$, since otherwise G has a P_6. Assume that v_5 has a neighbor v_x, such that $v_x v_j \notin E(G)$. Since $v_k v_5 \notin E(G)$, it follows similarly to the above that in this case G has a P_6. Therefore, we have showed that $N_G[v_5] \subseteq N_G[v_j]$ and, thus, $v_3 \in L$ which is a contradiction.

(C.b) The vertex v_3 has two neighbors v_5 and v_5' in G_y, such that $v_5 v_5' \notin E(G)$. Using the same arguments as in Case (B.b), we obtain that in this case G has a C_4 which is a contradiction to our assumptions.

(C.c) The vertex v_3 has two neighbors v_5 and v_5' (where $v_5 \neq v_j$ and $v_5' \neq v_j$) in G_y, such that $v_5 v_5' \in E(G)$, and neither $N_{G_y}[v_5] \subseteq N_{G_y}[v_5']$ nor $N_{G_y}[v_5'] \subseteq N_{G_y}[v_5]$; that is, there exist vertices v_6 and v_6' in G_y such that $v_5 v_6 \in E(G)$ and $v_5 v_6' \notin E(G)$ and, also, $v_5' v_6' \in E(G)$ and $v_5' v_6 \notin E(G)$. Similarly to Case (B.c), we can prove that this case comes to a contradiction as well. Note that, in this case $d_m(v_3, v) = 3$ and, thus, there exists a chordless path $\{v_3, v_5, v_7, v\}$ from v_3 to v. Again, at least one of $v \equiv v'''$ and $v' \equiv v''$ must hold, since otherwise G has a P_6 induced by the vertices $\{v''', v_6, v_5, v_5', v_6', v''\}$. Using the same arguments as in Case (B.c), we obtain that if $v \equiv v'''$ then $v_k, v_j \notin N_G[v_6]$. However, now, we must additionally have $v_6 v_7 \in E(G)$, since otherwise G has a C_4 induced by the vertices $\{v, v_7, v_5, v_6\}$. Therefore, as in Case (B.c) we obtain $v_6 \in L$, which is a contradiction to our assumption that the vertex v_i appears in the ordering before the vertices v_6, v_6', v_5, and v_5'.

Case (D): $d_m(v_3, v) = 4$.

In this case there exist vertices v_5, v_6 and v_7 such that $\{v_3, v_5, v_6, v_7, v\}$ is a chordless path from v_3 to v. Since now G has a P_9, it follows that $v_5 v_j \in E(G)$ and, additionally, some other edges must exist. Similarly to Cases (A) and (B), we distinguish three cases regarding the neighborhood of the vertex v_3 in G and show that if $v_3 \notin L$ then each one comes to a contradiction.

(D.a) The vertex v_3 does not have neighbors in G other than v_5 and v_j. If we assume that $v_3 \notin L$, then v_5 has a neighbor in G which is not a neighbor of v_j and, additionally, v_j has a neighbor in G which is not a neighbor of v_5. Thus, we can have one of the following three cases, each of which comes to a contradiction:

- $v_2 \in N_G[v_5]$ and $v_7 \in N_G[v_j]$. Now, we have that $v_2 v_6 \in E(G)$, since otherwise G has a P_6 induced by the vertices $\{v_4, v_2, v_5, v_6, v_7, v\}$. However, in this case v_2 would not be simple

in G_2, where G_2 is the subgraph of G induced by the vertices to the right of v_2 in σ, since $v_7 \in N_G[v_6]$ and $v_7 \notin N_G[v_5]$ and, also, $v_3 \in N_G[v_5]$ and $v_3 \notin N_G[v_6]$. Indeed, it suffices to show that the vertices v_5, v_6, v_7, and v_3 belong to the induced subgraph G_2 of G.

We know that $v_5, v_3 \in N_G[v_j]$ and, thus, $v_5 > v_i$ and $v_3 > v_i$ since we have assumed that v_j does not have a neighbor v_x, such that $v_x < v_i$. Additionally, from $v_7 \in N_G[v_j]$ it follows that $v_6 \in N_G[v_j]$, since otherwise G has a C_4 induced by the vertices $\{v_j, v_5, v_6, v_7\}$. Therefore, $v_6, v_7 \in N_G[v_j]$ and, thus, $v_i < v_6$ and $v_i < v_7$. Therefore, the vertices v_5, v_6, v_7, and v_3 belong to the induced subgraph G_2 of G, and thus, the vertex v_2 is not simple in G_2, which is a contradiction to our assumption that σ is a strong elimination ordering.

- $v_k \notin N_G[v_5]$ and $v_6 \notin N_G[v_j]$. From $v_k \notin N_G[v_5]$ we obtain that $v_2, v_i \notin N_G[v_5]$. In this case G has a P_8 induced by the vertices $\{v_4, v_2, v_k, v_j, v_5, v_6, v_7, v\}$. This path is chordless since G is a chordal graph.
- $v_i \notin N_G[v_5]$ and $v_6 \notin N_G[v_j]$. In this case, we have a P_8 in G induced by the vertices $\{v_4, v_2, v_k, v_j, v_5, v_6, v_7, v\}$; thus, $v_k v_5 \in E(G)$. From $v_i \notin N_G[v_5]$ we obtain that $v_2 \notin N_G[v_5]$ and, thus, $v_6 v_k \in E(G)$. Now, G has a 3-sun induced by the vertices $\{v_5, v_k, v_j, v_6, v_i, v_3\}$, since we have assumed that $v_i v_5 \notin E(G)$, $v_6 v_j \notin E(G)$, and other edges do not exist by assumption. This is a contradiction to our assumption that G is a strongly chordal graph.

Using similar arguments as in Case (B.a) and Case (C.a), we can prove that either $N_G[v_5] \subseteq N_G[v_j]$ or $N_G[v_j] \subseteq N_G[v_5]$ and, thus, $v_3 \in L$. Similarly to Cases (B) and (C), we distinguish three cases regarding the neighborhood of the vertex v_3 in G and can show that each one comes to a contradiction.

(D.b) The vertex v_3 has two neighbors v_5 and v_5' in G_y, such that $v_5 v_5' \notin E(G)$. Using the same arguments as in Case (B.b), we obtain that in this case G has a C_4 which is a contradiction to our assumptions.

(D.c) The vertex v_3 has two neighbors v_5 and v_5' (where $v_5 \neq v_j$ and $v_5' \neq v_j$) in G_y, such that $v_5 v_5' \in E(G)$, and neither $N_{G_y}[v_5] \subseteq N_{G_y}[v_5']$ nor $N_{G_y}[v_5'] \subseteq N_{G_y}[v_5]$. Using the same arguments as in Cases (B.c) and (C.c), we can prove that this case comes to a contradiction.

Case 2: The vertex $v_i \in I$ and $v_i \notin S$. Since σ is a strong elimination ordering, each vertex $v_i \in I$ is simple in G_i and, thus, $\{N_{G_i}[v_k] : v_k \in N_{G_i}[v_i]\}$ is linearly ordered by inclusion. Since v_i is not a simplicial vertex in G, there exist at least two vertices $v_1, v_2 \in N_G(v_i)$ such that $v_1 v_2 \notin E(G)$ and $v_1 < v_i < v_2$. If there exists a neighbor v_k of v_i such that $v_i < v_k < v_j$ and neither $N_G[v_k] \subseteq N_G[v_j]$ nor $N_G[v_k] \supseteq N_G[v_j]$, then as we showed in Case 1, we come to a contradiction; recall that we have assumed that v_j is the uncolored vertex.

Assume that such a vertex v_k does not exist. Therefore, since $\kappa(v_j) \neq \kappa(v_i)$ it follows that neither $N_G[v_i] \subseteq N_G[v_j]$ nor $N_G[v_i] \supseteq N_G[v_j]$ and, thus, there exists a vertex v_2 such that $v_2 < v_i < v_j$, $v_2 v_i \in E(G)$, and $v_2 v_j \notin E(G)$. Additionally, there exists a vertex v_3 in σ such that $v_3 v_i \notin E(G)$ and $v_3 v_j \in E(G)$. Thus, $\{v_2, v_i, v_j, v_3\}$ is a chordless path on 4 vertices. Additionally, since v_2 is a neighbor of $v_i \in I$ it follows that $v_2 \notin I$, and from Property 2.2(i) it follows that there exists a vertex $v_4 \in I$ in σ such that $v_4 < v_2$ and $v_4 v_2 \in E(G)$. Therefore, $\{v_4, v_2, v_i, v_j, v_3\}$ is a chordless path on 5 vertices. Using the same arguments as in Case 1, we can come to a contradiction by substituting v_k by v_i in the proof of Case 1.

From Cases 1 and 2 we conclude that using the constructed strong elimination ordering σ of G, we have proved that there is no uncolored vertex in σ, and since the set $\{N_G[v_k] : \kappa(v_k) = j\}$ is linearly ordered by inclusion for every $j \in \{1, 2, \ldots, \alpha(G)\}$, it follows that κ is a colinear coloring of G. Thus, the lemma holds.

∎

Part (IV): The equality $\lambda(G_A) = \alpha(G_A)$ holds for every $A \subseteq V(G)$. It is easy to see that, in Parts (I)–(III), we have showed that we can assign a colinear coloring with $\lambda(G) = \alpha(G)$ colors to any P_6-free strongly chordal graph, by using the constructed strong elimination ordering σ of G.

From Corollary 3.1, we have that $\lambda(G) \geq \alpha(G)$ holds for any graph G. Since κ is a colinear coloring of G using $\alpha(G)$ colors, it follows that the equality $\lambda(G) = \alpha(G)$ holds for G. Since every induced subgraph of a strongly chordal graph is strongly chordal [27], we can construct a strong elimination ordering σ as described above for every induced subgraph G_A of G, $\forall A \subseteq V(G)$; thus, we can assign a coloring κ to G_A with $\alpha(G_A)$ colors. Concluding, the equality $\lambda(G_A) = \alpha(G_A)$ holds for every induced subgraph G_A of a strongly chordal graph G and, therefore, any strongly chordal graph G is a linear graph.

Therefore, in Parts (I)–(IV) we have proved the following result.

Lemma 2.2. *Any P_6-free strongly chordal graph is a linear graph.*

From Lemma 2.2, we obtain the following result.

Lemma 2.3. *If G is a k-sun graph ($k \geq 3$), then G is a linear graph.*

Proof. Let G be a k-sun graph. It is easy to see that the equality $\alpha(G) = \lambda(G)$ holds for the k-sun G. Since a k-sun constitutes a minimal forbidden subgraph for the class of strongly chordal graphs, it follows that every induced subgraph of a k-sun is a strongly chordal graph and, thus, from Lemma 2.2 we obtain that G is a linear graph. ∎

From Lemmas 2.2 and 2.3, we also derive the following results.

Proposition 2.17. *Linear graphs form a superclass of the class of P_6-free strongly chordal graphs.*

We have proved that any P_6-free chordal graph which is not a linear graph has a k-sun as an induced subgraph; however, the k-sun itself is a linear graph. The interest of these results lies on the following characterization that we obtain for the class of linear graphs in terms of forbidden induced subgraphs.

Theorem 2.1. *Let $\mathcal{F}$ be the family of all the minimal forbidden induced subgraphs of the class of linear graphs, and let F_i be a member of $\mathcal{F}$. The graph F_i is either a C_n ($n \geq 4$), or a P_6, or it properly contains a k-sun ($k \geq 3$) as an induced subgraph.*

In light of the above result, it would be interesting to investigate whether or not linear graphs are characterized completely by a finite set of forbidden induced subgraphs. To this end, we need to investigate the P_6-free chordal graphs which are forbidden subgraphs for linear graphs; as we have shown these graphs properly contain a k-sun. An example of such a graph is the complement of the graph depicted in Figure 3.4; this graph is a P_6-free chordal graph on 9 vertices which properly contains a 4-sun, and is not a linear graph.

In general, an example of a forbidden induced subgraph of linear graphs is a graph H on $2k+1$ vertices which properly contains a k-sun S_k ($k \geq 4$) such that $H = \{v\} \cup \{u_1, u_2, \ldots, u_k\} \cup \{w_1, w_2, \ldots, w_k\}$ and v is adjacent to every vertex of the clique $W = \{w_1, w_2, \ldots, w_k\}$ and to exactly two vertices, say, u_j and u_i ($j < i$) of the independent set $U = \{u_1, u_2, \ldots, u_k\}$ such that $i \neq j + 1 \pmod{k}$; recall that U is the independent set and W is the clique of the sun S_k. We claim that the P_6-free chordal graphs which are forbidden subgraphs for linear graphs do not restrict to graphs with such a structure and, also, that linear graphs are characterized completely by a finite set of forbidden induced subgraphs.

A finite set of forbidden subgraphs could lead to a recognition algorithm for linear graphs. Such an algorithm would require the detection of graphs of a specific structure which properly contain a k-sun. It is worth noting that finding a k-sun in a general graph has been recently proved to be NP-complete [43]. However, one can answer the question whether or not a chordal graph G contains a k-sun by using Farber's algorithm [27]; if G contains a k-sun as an induced subgraph, Farber's algorithm reports that G is not a strongly chordal graph and, also, returns an induced subgraph of G which contains a k-sun. However, there is no known polynomial algorithm for detecting and reporting a k-sun in a chordal graph.

Investigating an algorithm for detecting and reporting a k-sun in a chordal graph is of great interest, since it could be a step toward the recognition of the class of linear graphs. Additionally, such an algorithm along with a minimal set of forbidden induced subgraphs could help us to characterize and provide properties of linear graphs which could be used for finding polynomial solutions for problems on linear graphs, which are NP-complete on chordal graphs.

2.7 Concluding Remarks

In this work we introduced the colinear coloring on graphs, and proposed a colinear coloring algorithm that can be applied to any graph G. Based on the colinear coloring we defined two graph properties, namely the χ-colinear and α-colinear properties, and characterized known graph classes in terms of these properties. We also studied the graphs that are characterized completely by the χ-colinear or the α-colinear property, which form two new classes of perfect graphs, namely colinear and linear graphs.

An interesting question would be to study structural and recognition properties of colinear and linear graphs and see whether they can be characterized by a finite set of forbidden induced subgraphs. Moreover, an obvious though interesting open question would be whether combinatorial and/or optimization problems can be efficiently solved on the classes of linear and colinear graphs. In addition, it would be interesting to study the relation between the colinear chromatic number and other coloring numbers such as the harmonious number and the achromatic number on classes of graphs.

Chapter 3

The Harmonious Coloring Problem

3.1 Introduction

A *harmonious coloring* of a simple graph G is a proper vertex coloring such that each pair of colors appears together on at most one edge, while the *harmonious chromatic number* $h(G)$ is the least integer k for which G admits a harmonious coloring with k colors [13].

Several NP-complete problems on arbitrary graphs admit polynomial solutions when restricted to the classes of strongly chordal graphs, undirected path graphs, and interval graphs (see e.g. [1, 7, 20, 28, 38, 49, 51, 52]). However, the pair-complete coloring problem, which is NP-hard on arbitrary graphs [67], remains NP-complete when restricted to graphs that are simultaneously interval and cographs [8]. A *pair-complete coloring* of a simple graph G is a proper vertex coloring such that each pair of colors appears together on at least one edge, while the *achromatic number* $\psi(G)$ is the largest integer k for which G admits a pair-complete coloring with k colors. The achromatic number was introduced in [41, 42].

Bodlaender [8] provides a proof for the NP-completeness of the pair-complete coloring problem for disconnected cographs and interval graphs and extends his results for connected such graphs. His proof also establishes the NP-hardness of the harmonious coloring problem for disconnected interval graphs and cographs. Note that the problem of determining the harmonious chromatic number of connected cographs is trivial, since in such a graph each vertex must receive a distinct color as it is at distance at most 2 from all other vertices [13]. On the contrary, although the harmonious coloring problem is NP-complete for disconnected interval graphs, the complexity of the problem for connected interval graphs is not straightforward. Moreover, the NP-hardness of the pair-complete coloring problem for cographs also establishes the NP-hardness of the pair-complete coloring problem for the class of permutation graphs, and, also, the NP-hardness of the harmonious coloring problem when restricted to disconnected permutation graphs. However, the complexity of the harmonious coloring problem for connected permutation

graphs has not been studied. Motivated by these issues we prove that the harmonious coloring problem is also NP-complete for connected interval and permutation graphs. In addition, we show that the problem remains NP-complete for the class of split graphs.

Additionally, the NP-completeness of the problem has been also proved for the classes of trees and disconnected bipartite permutation graphs [25, 26], connected bipartite permutation graphs [2], and disconnected quasi-threshold graphs [2]. Since the problem of determining the harmonious chromatic number of a connected cograph is trivial, the harmonious coloring problem is polynomially solvable on connected quasi-threshold graphs and threshold graphs.

Since we prove that the harmonious coloring problem is NP-complete on interval graphs, we obtain that the problem is also NP-complete on the classes of strongly chordal and undirected path graphs. Extending our results for the harmonious coloring problem on interval graphs and split graphs, in this work we also study the complexity status of the harmonious coloring problem on two subclasses of colinear graphs; for definitions and results on colinear and linear graphs see Chapter 2. We first show that the harmonious coloring problem is NP-complete on split undirected path graphs and, then, we show that the class of split undirected path graphs forms a subclass of colinear graphs; thus, we obtain the NP-completeness of the harmonious coloring problem on colinear graphs as well. Moreover, we provide a polynomial solution for the harmonious coloring problem on split strongly chordal graphs, the interest of which lies on the fact that the problem is NP-complete on both split graphs and strongly chordal graphs. However, the complexity status of the problem for the class of connected linear graphs still remains an open question; note that the harmonious coloring problem is NP-complete on disconnected linear graphs, since it is NP-complete on disconnected quasi-threshold graphs [2] and quasi-threshold graphs form a subclass of linear graphs.

The rest of this chapter is organized as follows. In Section 3.2 we we prove that the harmonious coloring problem is NP-complete on connected interval and permutation graphs, while in Section 3.3 we prove the NP-completeness of the problem on split graphs. In Section 3.4 we prove that the problem remains NP-complete for split undirected path graphs; we also prove that the problem is NP-complete for colinear graphs by showing that split undirected path graphs form a subclass of colinear graphs. Moreover, in Section 3.5 we provide a polynomial solution for the harmonious coloring problem for the class of split strongly chordal graphs. Some concluding remarks follow.

3.2 Connected Interval and Permutation Graphs

The formulation of the harmonious coloring problem in [13] is equivalent to the following formulation.

Harmonious Coloring Problem
Instance: Graph G, positive integer $K \leq |V(G)|$.
Question: Is there a positive integer $k \leq K$ and a proper coloring using k colors such that each pair of colors appears together on at most one edge?

We next prove our main result of this section, that is, the harmonious coloring problem is NP-complete for connected interval graphs.

Theorem 3.1. *Harmonious coloring is NP-complete when restricted to connected interval graphs.*

Proof. Harmonious coloring is obviously in NP. In order to prove NP-hardness, we use a transformation from a strongly NP-complete problem, that is, the 3-PARTITION problem. The formulation of the 3-PARTITION problem [33] is presented below.

3-PARTITION
Instance: Set A of $3m$ elements, a bound $B \in Z^+$, and a size $s(a) \in Z^+$ for each $a \in A$, such that $\frac{1}{4}B < s(a) < \frac{1}{2}B$, and such that $\sum_{a \in A} s(a) = mB$.
Question: Can A be partitioned into m disjoined sets $A_1, A_2, \ldots, A_m$ such that, for $1 \leq i \leq m$, $\sum_{a \in A_i} s(a) = B$ (note that each A_i must therefore contain exactly three elements from A)?

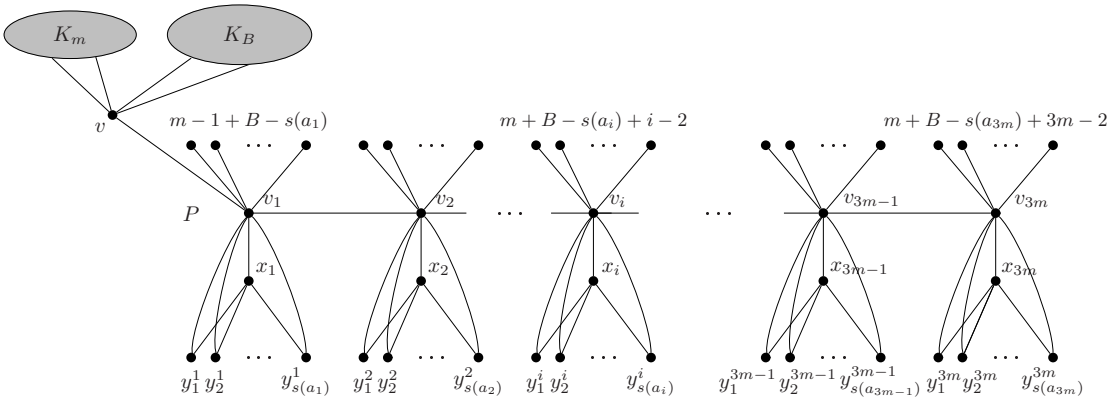

Figure 3.1: Illustrating the constructed connected interval and permutation graph G.

Let a set $A = \{a_1, \ldots, a_{3m}\}$ of $3m$ elements, a positive integer B and let positive integer sizes $s(a_i)$ for each $a_i \in A$ be given, such that $\frac{1}{4}B < s(a_i) < \frac{1}{2}B$, and such that $\sum_{a_i \in A} s(a_i) = mB$. We may suppose that, for each $a_i \in A$, $s(a_i) > m$ (if not, then we can multiply all $s(a_i)$ and B with $m+1$).

Extending the result of Bodlaender [8], we construct the following connected graph which is an interval and a permutation graph: Consider a clique with m vertices, a clique with B vertices, and add a vertex v that is connected to every vertex in the two cliques; let G_1 be the resulting graph. Next we construct for every $a_i \in A$ a tree T_i of depth one with $s(a_i)$ leaves and root x_i, that is, every leaf is adjacent to the root; note that there are $3m$ such trees $T_1, T_2, \ldots, T_{3m}$. Then we construct a path $P = [v_1, v_2, \ldots, v_{3m}]$ of $3m$ vertices, and we connect each vertex v_i of the path P to all the vertices of the tree T_i, $1 \leq i \leq 3m$. Additionally, for each vertex $v_i \in P$, we add $m-1+B-s(a_i)+i-1$ vertices and connect them to vertex v_i; let G_2 be the resulting graph. Note that the graph $G_1 \cup G_2$ is disconnected. Finally, we add an edge to the graph $G_1 \cup G_2$ connecting vertices v_1 and v and let G be the resulting graph. The graph G is a connected graph and it is illustrated in Fig. 3.1.

One can easily verify that G is an interval graph. A clique can be represented as a number of intervals that share at least one point in common. Two cliques sharing a vertex u can be represented as a number of intervals such that one of them, which corresponds to u, shares at least one point with the intervals corresponding to the vertices of each clique. Thus, the vertices of G can be put in one-to-one correspondence with a family of intervals on the real line such that two vertices are adjacent in G if and only if their corresponding intervals intersect.

It is easy to see that the total number of edges in G is

$$\binom{m}{2} + \binom{B}{2} + m + B + 3m + mB + 3m + mB + 3m(m-2) + 2mB + \sum_{i=1}^{3m} i = \binom{4m+B+1}{2}$$

For every harmonious coloring of G and every pair of distinct colors i, j, $i \neq j$, there must be at most one edge with its endpoints colored with i and j. Thus, it follows that the harmonious chromatic number cannot be less than $4m+B+1$, and if it is equal to $4m+B+1$ then we have, for every pair of distinct colors i, j, $1 \leq i, j \leq 4m+B+1$, a unique edge with its end-points colored with i and j. Thus, we have an exact coloring of G; an *exact coloring* of G with k colors is a harmonious coloring of G with k colors in which, for each pair of colors i, j, there is exactly one edge (a, b) such that a has color i and b has color j.

We now claim that the harmonious chromatic number of G is (less or equal to) $4m+B+1$ if and only if A can be partitioned in m sets $A_1, \ldots, A_m$ such that $\sum_{a \in A_j} s(a) = B$, for all j, $1 \leq j \leq m$.

($\Longleftarrow$) Suppose now a 3-partition of A in $A_1, \ldots, A_m$ such that $\forall j : \sum_{a \in A_j} s(a) = B$ exists. We show how to find a harmonious coloring of G using $4m + B + 1$ colors. We color the vertices of the first clique with colors $1, 2, \ldots, m$, the vertices of the second clique with $m+1, m+2, \ldots, m+B$, and vertex v with $m + B + 1$. For convenience and ease of presentation, let $\mathcal{M}$ be the set containing colors $1, 2, \ldots, m$, let $\mathcal{B}$ be the set containing colors $m+1, m+2, \ldots, m+B$, and let $\mathcal{K}$ be the set containing colors $m + B + 2, m + B + 3, \ldots, 4m + B + 1$. If $a_i \in A_j$ then we color the vertex x_i with color j. Each color $j \in \mathcal{M}$ is assigned to the three vertices corresponding to three a_i that have together exactly B neighbors of degree 2. We assign to each one of these B neighbors a different color from $\mathcal{B}$, and next we assign to each vertex v_i of the path P a distinct color from $\mathcal{K}$. Recall that each vertex v_i, $1 < i < 3m$, is connected to two other vertices of P, i.e., v_{i-1} and v_{i+1}, and $m + B + i - 1$ more vertices, vertex v_1 is connected to v_2, v and $m + B$ other vertices, while vertex v_{3m} is connected to v_{3m-1} and $m + B + 3m - 1$ more vertices (see Fig. 1).

Next, we color the rest $m - 1 + B - s(a_i) + i - 1$ neighbors of each v_i. We assign a distinct color from the set $\mathcal{M} \backslash c_i$ to $m - 1$ neighbors of v_i, where c_i is the color previously assigned to the vertex x_i. We next assign a distinct color from the set $\mathcal{B} \backslash C_i$ to $B - s(a_i)$ neighbors of v_i, where C_i is the set of the colors previously assigned to $s(a_i)$ neighbors of the vertex x_i. Finally, we assign a different color to the rest $i - 1$ neighbors of v_i, $3 \leq i \leq 3m$, using color $m + b + 1$ and the colors assigned to the vertices v_j, $1 \leq j \leq i - 2$. Note that, in order to color the $m + B - s(a_2)$ neighbors of v_2, we only need to use color $m + B + 1$ and colors from $\mathcal{M}$ and $\mathcal{B}$, while for the $m - 1 + B - s(a_1)$ neighbors of v_1 we only use colors from $\mathcal{M}$ and $\mathcal{B}$. A harmonious coloring of G using $4m + B + 1$ colors results, and thus, the harmonious chromatic number of G is $4m + B + 1$.

($\Longrightarrow$) We next suppose that the harmonious chromatic number of G is (less or equal to) $4m + B + 1$. Consider a harmonious coloring of G using $4m + B + 1$ colors. Without loss of generality we may suppose that the m vertices of the first clique have distinct colors from $\mathcal{M}$, while the B vertices of the second clique have distinct colors from $\mathcal{B}$. Also, without loss of generality, we color vertex v with color $m + B + 1$ since v is adjacent to all the vertices of the two cliques. Since v_{3m} is the vertex having the maximum degree, that is, $4m + B$, it has to take a color from $\mathcal{K}$. Indeed, if it takes a color from $\mathcal{M}$, then none of its neighbors can take a color from $\mathcal{M}$ and we cannot color $4m + B$ vertices using only $4m + B + 1 - m$ colors. Using similar arguments, we cannot color vertex v_{3m} using a color from $\mathcal{B}$ or the color $m + B + 1$. Thus, without loss of generality, we assign to v_{3m} the color $4m + B + 1$. We color all its neighbors with distinct colors from $\mathcal{M} \cup \mathcal{B} \cup \{m + B + 1\} \cup \mathcal{K} \backslash \{4m + B + 1\}$. Note that, vertex v_{3m-1} takes a color from $\mathcal{K} \backslash \{4m + B + 1\}$; let $4m + B$ be this color. Indeed, using similar arguments, it cannot take a color from $\mathcal{M} \cup \mathcal{B} \cup \{m + B + 1\} \cup \{4m + B + 1\}$. Note that, color $4m + B + 1$ cannot be assigned to any other vertex of G since any pair of colors $(4m + B + 1, j)$, $1 \leq j \leq 4m + B$, already appears in the harmonious coloring. Recall that, for every pair of distinct colors i, j, $1 \leq i, j \leq 4m + B + 1$, there is a unique edge with its end-points colored with i and j. Recursively, as can easily be proved by induction on i, the same holds for all $v_i \in P$, $1 \leq i \leq 3m - 2$, that is, v_i takes a color from $\mathcal{K} \backslash \mathcal{L}$, where $\mathcal{L}$ is the set containing colors $m + B + 1 + i + 1, m + B + 1 + i + 2, \ldots, 4m + B + 1$, which are the colors already assigned to vertices v_j, $i < j \leq 3m$.

Note that pairs (μ, ν), $\mu \in \mathcal{M}$, $\nu \in \mathcal{B}$, have not appeared yet. Since every pair of colors must appear, we assign these pairs to the mB edges that have both endpoints uncolored. Note that these edges are the edges (x_i, y_j^i), $1 \leq i \leq 3m$, $1 \leq j \leq s(a_i)$, where x_i corresponds to a_i and y_j^i corresponds to the j-th neighbor of x_i having degree 2. The vertices x_i cannot take a color from $\mathcal{B}$, otherwise its $s(a_i) > m$ uncolored neighbors y_j^i cannot be colored with m colors from $\mathcal{M}$. Thus, vertices x_i are assigned a color from $\mathcal{M}$ and vertices y_j^i are assigned a color from $\mathcal{B}$ (recall that $\frac{B}{4} < s(a_i) < \frac{B}{2}$). Note that the only uncolored vertices are $m - 1 + B - s(a_i) + i - 1$ neighbors of each v_i, $1 \leq i \leq 3m$. In order to color $m - 1 + B - s(a_i)$ of the uncolored neighbors of v_i, we use distinct colors from $(\mathcal{M} \cup \mathcal{B}) \backslash \mathcal{F}$, where $\mathcal{F}$ is the set containing all colors already assigned to the $s(a_i) + 1$ neighbors of v_i. In order to color the last $i - 1$ uncolored neighbors of v_i, $i > 1$, we can only use colors from $\mathcal{K} \backslash \mathcal{L} \backslash \{m + B + 1 + i, m + B + i\}$ because the only unused pairs are $(m + B + 1 + i, j)$, where $m + B + 1 \leq j \leq m + B + 1 + i - 2$.

Finally, let $a_i \in A_j$ if and only if the vertex x_i (with neighbors y_j^i) is colored with color $j \in \mathcal{M}$. We claim that for all j, $\sum_{a \in A_j} s(a) = B$. Indeed, each color j must be adjacent to some colors from $\mathcal{B}$, and each color from $\mathcal{B}$ is assigned to exactly one vertex which is adjacent to all x_i colored with j. Hence, a correct 3-partition exists.

The theorem follows from the strong NP-completeness of 3-PARTITION, since the transformation can be done easily in polynomial time. ∎

We can easily show that the interval graph G illustrated in Fig. 3.1 is also a permutation graph. The graph G is an interval graph if and only if it is a chordal graph and the graph $\overline{G}$ is a comparability graph [37]. Moreover, one can easily verify that G admits an acyclic transitive orientation and, thus, it is a comparability graph. Since G and $\overline{G}$ are comparability graphs, it follows that G is a permutation graph [37]. Consequently, we can state the following theorem.

Theorem 3.2. *Harmonious coloring is NP-complete when restricted to connected permutation graphs.*

We have shown that the connected interval graph G presented in this section, which is also a permutation graph, has $\binom{4m+B+1}{2}$ edges and $h(G) = 4m + B + 1$. In [25] it was shown that if G is a graph with exactly $\binom{k}{2}$ edges, then a proper vertex coloring of G with k colors is pair-complete if and only if it is a harmonious coloring. Thus, if G is a graph with $\binom{k}{2}$ edges, then $\psi(G) = k$ if and only if $h(G) = k$ [13]. Consequently, for the graph G, which is simultaneously an interval and a permutation graph, we have that $\psi(G) = 4m + B + 1$ and, thus, our results could be also used to prove that the achromatic number is NP-complete for connected interval and permutation graphs.

3.3 Split Graphs

We next show that the harmonious coloring problem is NP-complete for split graphs, by exhibiting a reduction from the chromatic number problem for general graphs, which is known to be NP-complete [33].

Let G be an arbitrary graph with n vertices $v_1, v_2, \ldots, v_n$ and m edges $e_1, e_2, \ldots, e_m$. We construct in polynomial time a split graph $\widehat{G}$, where $V(\widehat{G}) = K + I$, as follows: the independent set I consists of n vertices $\widehat{v}_1, \widehat{v}_2, \ldots, \widehat{v}_n$ which correspond to the vertices $v_1, v_2, \ldots, v_n$ of the graph G and the clique K consists of m vertices $\widehat{u}_1, \widehat{u}_2, \ldots, \widehat{u}_m$ which correspond to the edges $e_1, e_2, \ldots, e_m$ of G. A vertex $\widehat{u}_t \in K$, $1 \leq t \leq m$, is connected to two vertices $\widehat{v}_i, \widehat{v}_j \in I$, $1 \leq i, j \leq n$, if and only if the corresponding vertices v_i and v_j are adjacent in G. Note that, every $\widehat{u}_i \in K$ sees all the vertices of the clique K and two vertices of the independent set I; thus, $|E(\widehat{G})| = \frac{m(m-1)}{2} + 2m$.

We claim that the graph G has a chromatic number $\chi(G)$ if and only if the split graph $\widehat{G}$ has a harmonious chromatic number $h(\widehat{G}) = \chi(G) + m$.

Let $c_i \in \{1, \ldots, \chi(G)\}$ be the color assigned to the vertex $v_i \in G$, $1 \leq i \leq n$, in a minimum coloring of G. We assign the color c_i to the vertex $\widehat{v}_i$ of the set I and a distinct color from the set $\{\chi(G) + 1, \ldots, \chi(G) + m\}$ to each vertex of the clique K. Since two adjacent vertices of G receive a different color, the neighbors of each $\widehat{u}_i \in K$ belonging to the independent set have distinct colors. Moreover, every vertex $\widehat{v}_i \in I$ sees $|N_G(v_i)|$ vertices of the clique K, where $N_G(v_i)$ is the neighborhood of the vertex v_i in G. Thus, every pair of colors appears in at most one edge. In addition, the number of colors assigned to the set I is equal to $\chi(G)$ and the number of colors assigned to the clique is equal to m. This results to a harmonious coloring of $\widehat{G}$ using $\chi(G) + m$ colors, which is minimum since the vertices of the set I cannot receive a color assigned to a vertex of the clique K.

Conversely, a harmonious coloring of $\widehat{G}$ using $h(\widehat{G}) = \chi(G) + m$ colors assigns m colors to the vertices of the clique K and $\chi(G)$ colors to the vertices of the set I. Note that, $\chi(G)$ is the minimum number of

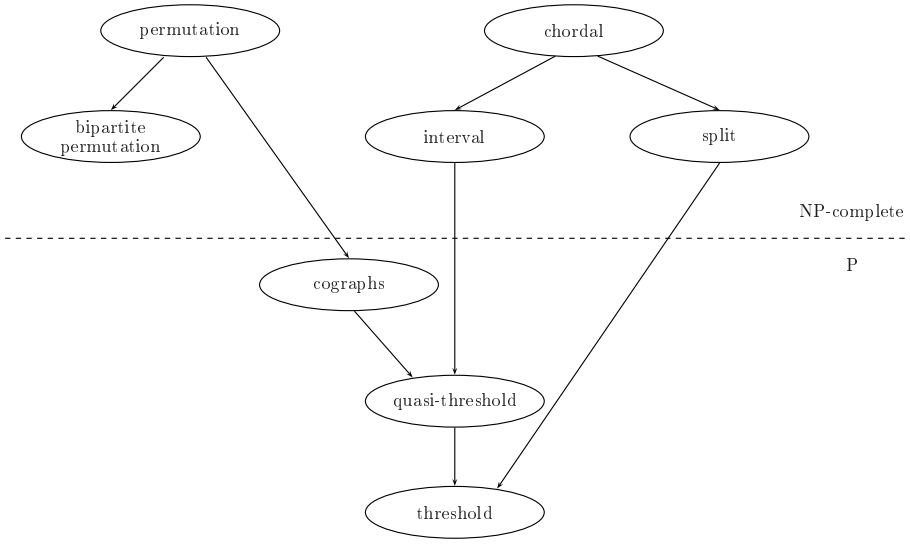

Figure 3.2: The complexity status of the harmonious coloring problem for some graph subclasses of permutation and chordal graphs. $A \rightarrow B$ indicates that class A contains class B.

colors so that vertices $\widehat{v}_i, \widehat{v}_j$ having a neighbor in common are assigned different colors. Since v_i, v_j are adjacent in G, it follows that we have a minimum coloring of G using $\chi(G)$ colors.

Thus, we have proved the following result.

Theorem 3.3. *Harmonious coloring is NP-complete for split graphs.*

Figure 3.2 shows a diagram of class inclusions for a number of graph classes, subclasses of permutation and chordal graphs, and the current complexity status of the harmonious coloring problem for connected graphs of these classes; for definitions of the classes shown, see [10, 37].

3.4 Harmonious Coloring on Colinear Graphs

In this section we show that the harmonious coloring problem remains NP-complete when restricted to the class of colinear graphs, which is a subclass of co-chordal graphs and a superclass of threshold graphs. The problem is NP-complete on co-chordal graphs, since in Section 3.3 we proved that it is NP-complete on split graphs, and also it has a polynomial solution on threshold graphs. Therefore, it is interesting to study the complexity of the problem on colinear graphs.

We first show that the problem remains NP-complete even when restricted to graphs which are simultaneously split graphs and undirected path graphs. Then, we show that every split undirected path graph is a colinear graph, thus, proving that the problem is NP-complete on colinear graphs.

We first give some definitions and results which were introduced or proven in Chapter 2 and will be used for obtaining some results in the rest of this chapter.

Definition 3.1. Let G be a graph and let $v \in V(G)$. The *clique set* of a vertex v is the set of all maximal cliques of G containing v and is denoted by $\mathcal{C}_G(v)$.

Definition 3.2. Let G be a graph and let k be an integer. A surjective map $\kappa : V(G) \rightarrow \{1, 2, \ldots, k\}$ is called a *k-colinear coloring* of G if the collection $\{\mathcal{C}_G(v) : \kappa(v) = i\}$ is linearly ordered by inclusion for all $i \in \{1, 2, \ldots, k\}$. Equivalently, for two vertices $v, u \in V(G)$, if $\kappa(v) = \kappa(u)$ then either $\mathcal{C}_G(v) \subseteq \mathcal{C}_G(u)$ or $\mathcal{C}_G(v) \supseteq \mathcal{C}_G(u)$. The least integer k for which G is k-colinear colorable is called the *colinear chromatic number* of G and is denoted by $\lambda(G)$.

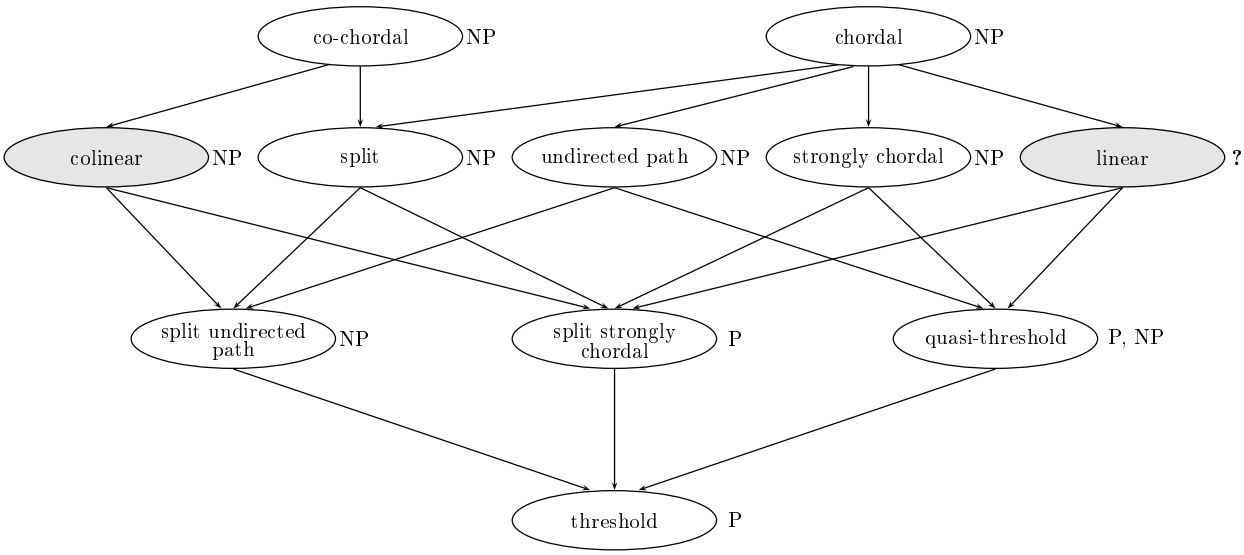

Figure 3.3: Illustrating the complexity status of the harmonious coloring problem, and the inclusion relations, for the classes of colinear graphs, linear graphs, and other subclasses of co-chordal and chordal graphs.

In Chapter 2 we presented a polynomial time algorithm for colinear coloring which can be applied to any graph G and, also, we proved the following results.

Proposition 3.1. *For any graph G, $\lambda(\overline{G}) \geq \chi(G)$.*

Definition 3.3. A graph G is called colinear if and only if $\chi(G_A) = \lambda(\overline{G}_A)$, $\forall A \subseteq V(G)$. A graph G is called linear if and only if $\alpha(G_A) = \lambda(G_A)$, $\forall A \subseteq V(G)$.

In Chapter 2 we also showed inclusion relations between the classes of colinear and linear graphs and other subclasses of co-chordal and chordal graphs. More specifically, the class of colinear graphs is a subclass of co-chordal graphs, a superclass of threshold graphs, and is distinguished from the class of split graphs. Additionally, linear graphs form a subclass of chordal graphs and a superclass of quasi-threshold graphs. We also proved that any P_6-free strongly chordal graph is a linear graph.

The inclusion relations among the classes of colinear graphs, linear graphs, and other subclasses of co-chordal and chordal graphs are depicted in Figure 3.3. Note that since any P_6-free strongly chordal graph is a linear graph, it follows that split strongly chordal graphs form a subclass of linear graphs. Then, we can easily obtain that any split strongly chordal graph is a colinear graph, since if a graph G is strongly chordal then $\overline{G}$ is also a strongly chordal graph.

The following characterization of undirected path graphs will be used for obtaining our results. Note that, $\mathcal{C}$ denotes the set of all maximal cliques of a graph G; recall that, $C(v)$ denotes the set of all maximal cliques containing v.

Theorem 3.4. *([35, 57]) A graph G is an undirected path graph if and only if there exists a tree T whose set of vertices is $\mathcal{C}$, so that for every vertex $v \in V(G)$, the subgraph $T[C(v)]$ of T induced by the vertex set $C(v)$, is a path in T. Such a tree will be called characteristic tree of G.*

We next show that the harmonious coloring problem is NP-complete for split undirected path graphs by exhibiting a reduction from the chromatic number problem for general graphs, which is known to be NP-complete [33].

Let G be an arbitrary graph with n vertices $v_1, v_2, \ldots, v_n$ and m edges $e_1, e_2, \ldots, e_m$. We construct in polynomial time a split graph $\widehat{G}$, where $V(\widehat{G}) = K + I$, as follows: the independent set I consists of

n vertices $\widehat{v}_1, \widehat{v}_2, \ldots, \widehat{v}_n$ which correspond to the vertices $v_1, v_2, \ldots, v_n$ of the graph G and the clique K consists of m vertices $\widehat{u}_1, \widehat{u}_2, \ldots, \widehat{u}_m$ which correspond to the edges $e_1, e_2, \ldots, e_m$ of G. A vertex $\widehat{u}_t \in K$, $1 \leq t \leq m$, is connected to two vertices $\widehat{v}_i, \widehat{v}_j \in I$, $1 \leq i, j \leq n$, if and only if the corresponding vertices v_i and v_j are adjacent in G. Note that, every $\widehat{u}_i \in K$ sees all the vertices of the clique K and two vertices of the independent set I; thus, $|E(\widehat{G})| = \frac{m(m-1)}{2} + 2m$.

Moreover, we claim that the constructed split graph $\widehat{G}$ is also an undirected path graph. Indeed, we prove this by showing that the graph $\widehat{G}$ has a characteristic tree. Let $\mathcal{C}$ be the set of all maximal cliques of $\widehat{G}$. Note that K is a maximal clique for $\widehat{G}$, thus, we have $|\mathcal{C}| = |I| + 1$. Every vertex $\widehat{v}_i \in I$ belongs to exactly one maximal clique, i.e., $|C(\widehat{v}_i)| = 1$. Additionally, every vertex $\widehat{u}_i \in K$ belongs to exactly three maximal cliques, one of which is maximal clique K, i.e., $|C(\widehat{u}_i)| = |N[\widehat{u}_i]| - |K| + 1 = 3$.

Consider now a tree T with vertex set $\mathcal{C}$, such that the maximal clique K is connected by an edge to every maximal clique $C(\widehat{v}_i)$ for every $\widehat{v}_i \in I$, i.e., T is a star. We now show that T is a characteristic tree for $\widehat{G}$. Indeed, for every vertex $\widehat{v}_i \in I$, the subgraph $T[C(\widehat{v}_i)]$ induced by $C(\widehat{v}_i)$ is a path on one vertex, and also for every vertex $\widehat{u}_i \in K$, the subgraph $T[C(\widehat{u}_i)]$ is a path on three vertices. Therefore, the constructed graph $\widehat{G}$ has a characteristic tree and, thus, from Theorem 3.4 it follows that $\widehat{G}$ is a split undirected path graph.

We claim that the graph G has a chromatic number $\chi(G)$ if and only if the split undirected path graph $\widehat{G}$ has a harmonious chromatic number $h(\widehat{G}) = \chi(G) + m$. Note that the same arguments are used in Section 3.3 for proving the NP-completeness of the problem for split graphs.

Let $c_i \in \{1, \ldots, \chi(G)\}$ be the color assigned to the vertex $v_i \in G$, $1 \leq i \leq n$, in a minimum coloring of G. We assign the color c_i to the vertex $\widehat{v}_i$ of the set I and a distinct color from the set $\{\chi(G) + 1, \ldots, \chi(G) + m\}$ to each vertex of the clique K. Since two adjacent vertices of G receive a different color, the neighbors of each $\widehat{u}_i \in K$ belonging to the independent set have distinct colors. Moreover, every vertex $\widehat{v}_i \in I$ sees $|N_G(v_i)|$ vertices of the clique K, where $N_G(v_i)$ is the neighborhood of the vertex v_i in G. Thus, every pair of colors appears in at most one edge. In addition, the number of colors assigned to the set I is equal to $\chi(G)$ and the number of colors assigned to the clique is equal to m. This results to a harmonious coloring of $\widehat{G}$ using $\chi(G) + m$ colors, which is minimum since the vertices of the set I cannot receive a color assigned to a vertex of the clique K.

Conversely, a harmonious coloring of $\widehat{G}$ using $h(\widehat{G}) = \chi(G) + m$ colors assigns m colors to the vertices of the clique K and $\chi(G)$ colors to the vertices of the set I. Note that, $\chi(G)$ is the minimum number of colors so that vertices $\widehat{v}_i, \widehat{v}_j$ having a neighbor in common are assigned different colors. Since v_i, v_j are adjacent in G, it follows that we have a minimum coloring of G using $\chi(G)$ colors.

Thus, we have proved the following result.

Theorem 3.5. *The harmonious coloring problem is NP-complete for split undirected path graphs.*

Next, we show the following result.

Theorem 3.6. *Any split undirected path graph is a colinear graph.*

Proof. Let G be a split undirected path graph. Assume that G is not a colinear graph. Then, from Definition 3.3 there exists an induced subgraph G_A of G such that $\lambda(\overline{G}_A) \neq \chi(G_A)$; thus, due to Proposition 3.1, $\lambda(\overline{G}_A) > \chi(G_A)$.

From Theorem 3.4, we obtain that split undirected path graphs are hereditary, that is, every induced subgraph G_A of G is a split undirected path graph. Let $V(G_A) = K + I$ be a partition of the vertex set of G_A into a maximal clique K and an independent set I. Also, from Theorem 3.4 we have that G_A has a characteristic tree T with vertex set $\mathcal{C}$, where $\mathcal{C}$ is the set of all maximal cliques of G_A, such that for every vertex $v \in V(G_A)$, the subgraph $T[C(v)]$ of T induced by the vertex set $C(v)$ is a path in T.

In particular, since G_A is a split graph, for every vertex $v \in I$, the subgraph $T[C(v)]$ of T induced by the vertex set $C(v)$ is a vertex in T that corresponds to the unique maximal clique of G_A that v belongs to; we will denote this clique by C_v, i.e., $C_v = N_{G_A}[v]$ and $C(v) = \{C_v\}$ for every vertex $v \in I$. Also, for every vertex $v \in K$, the path $(C_u, \ldots, C_x, K, C_y, \ldots, C_z)$ of T induced by the vertex set $C(v)$, always passes

from the vertex K; equivalently, for every vertex $v \in K$, the subgraph of T induced by the vertex set $C(v)$, corresponds to the vertex K and to at most two vertex disjoint paths $(C_y, \ldots, C_z)$ and $(C_x, \ldots, C_u)$ where C_y and C_x are adjacent to K in T. Moreover, observe that for any path $(K, C_{v_1}, C_{v_2}, \ldots, C_{v_k})$ of the characteristic tree T of G_A, we have $C_{v_1} \setminus \{v_1\} \supseteq C_{v_2} \setminus \{v_2\} \supseteq \ldots \supseteq C_{v_k} \setminus \{v_k\}$, since $C_{v_i} \setminus \{v_i\} = N_{G_A}(v_i) \subset K$, where $v_i \in I$ for every i, $1 \leq i \leq k$.

Let $\kappa : V(\overline{G}_A) \rightarrow \{1, 2, \ldots, \lambda(\overline{G}_A)\}$ be a colinear coloring of $\overline{G}_A$. In order to see how a colinear coloring can be assigned to the vertices of $\overline{G}_A$ we refer to the colinear coloring algorithm presented in Chapter 2. Recall that, the algorithm first constructs the directed acyclic graph (DAG) $D_{\overline{G}_A}$ associated to the graph $\overline{G}_A$ and, then, finds a minimum path cover of the transitive DAG $D_{\overline{G}_A}$. The size of the minimum path cover of $D_{\overline{G}_A}$ equals the colinear chromatic number $\lambda(\overline{G}_A)$. Also, the algorithm assigns a colinear coloring κ to the vertices of $\overline{G}_A$ such that a set of vertices are assigned the same color in κ if and only if they belong to the same path of the minimum path cover of $D_{\overline{G}_A}$. Moreover, the DAG $D_{\overline{G}_A}$ associated to the graph $\overline{G}_A$ is constructed as follows: $V(D_{\overline{G}_A}) = V(\overline{G}_A)$ and $E(D_{\overline{G}_A}) = \{\overrightarrow{xy} \ : \ x, y \in V(D_{\overline{G}_A})$ and $N_{\overline{G}_A}[x] \subseteq N_{\overline{G}_A}[y]\}$, where $\overrightarrow{xy}$ is a directed edge from x to y. Note that $D_{\overline{G}_A}$ is a transitive DAG. For simplicity, throughout the proof we will denote the DAG $D_{\overline{G}_A}$ associated to the graph $\overline{G}_A$ by D.

The following observations will be useful in the rest of this proof. Two vertices $u, v \in V(D)$ are not adjacent in D if and only if neither $N_{\overline{G}_A}[v] \subseteq N_{\overline{G}_A}[u]$ nor $N_{\overline{G}_A}[v] \supseteq N_{\overline{G}_A}[u]$; we call two sets with this property incompatible. In $\overline{G}_A$ the vertices of I form a clique, therefore, for two vertices $u, v \in I$, u and v are not adjacent in D if and only if the sets $N_{\overline{G}_A}[u] \cap K$ and $N_{\overline{G}_A}[v] \cap K$ are incompatible. Note that, for any two vertices u, v of G_A, $N_{\overline{G}_A}[u] \subseteq N_{\overline{G}_A}[v]$ if and only if $N_{G_A}(u) \supseteq N_{G_A}(v)$. Additionally, for every vertex $u \in I$, we have $N_{G_A}(u) \subset K$.

Having assumed that $\lambda(\overline{G}_A) > \chi(G_A) = |K|$, there exists a minimum path cover of D with size $\lambda(\overline{G}_A) \geq |K|+1$. The size of a minimum path cover of D equals the cardinality of a maximum independent set I_D of D [37]; thus, $|I_D| \geq |K|+1$. Moreover, the independent set I_D corresponds to a collection C of mutually incompatible sets $N_{\overline{G}_A}[v]$, for all $v \in I_D$, that is, $C = \{N_{\overline{G}_A}[v] : v \in I_D\}$. Thus, $|C| \geq |K|+1$ and the sets of C contain at most $|K|$ vertices of K. Also, recall that for any two vertices $u, v \in V(D)$ such that $u \in K$ and $v \in I$, if $uv \in E(\overline{G}_A)$ then $N_{\overline{G}_A}[u] \subset N_{\overline{G}_A}[v]$; thus, for any two vertices $u, v \in V(D)$ such that $u \in K$ and $v \in I$, u and v are adjacent in $\overline{G}_A$ if and only if u and v are adjacent in D.

Assume that $K \subset I_D$. Then, no vertex $v \in I$ can belong to I_D since every vertex of I is adjacent to at least one vertex of K in $\overline{G}_A$ and, thus, in D, due to our assumption that K is a maximal clique of G_A. Thus, not every vertex of K can belong to I_D, since $|I_D| \geq |K|+1$. Assume that a vertex $u \in K$ belongs to I_D. Then, no vertex $v \in I$ that is adjacent to u in D and, thus, in $\overline{G}_A$, belongs to I_D; equivalently, $u \notin N_{\overline{G}_A}[v]$, for every vertex $v \in I_D$. Therefore, if we delete the vertex $u \in K$ from the set I_D, we obtain an independent set $I'_D = I_D \setminus \{u\}$ and a collection $C' = C \setminus \{N_{\overline{G}_A}[u]\}$ of at least $|K|$ mutually incompatible sets, which contain at most $|K|-1$ vertices of K. Using the same arguments, if we delete every vertex of K from the independent set I_D, we obtain an independent set I''_D, such that $I''_D \subseteq I$ and $|I''_D| \geq k+1$ (where $k \leq |K|$), which corresponds to a collection C'' of at least $k+1$ mutually incompatible sets $N_{\overline{G}_A}[v]$, $v \in I$, which contain at most k vertices of K.

A collection C'' of at least $k+1$ mutually incompatible sets $N_{\overline{G}_A}[v]$, $v \in I$, corresponds to a collection F of at least $k+1$ mutually incompatible sets $N_{G_A}(v)$, $v \in I$. Since, for every vertex $v \in I$ we have $N_{G_A}(v) = C_v \setminus \{v\}$, it follows that a collection F of at least $k+1$ mutually incompatible sets $N_{G_A}(v)$, $v \in I$, corresponds to a collection of at least $k+1$ maximal cliques C_v of G_A, $v \in I$, each of which must belong to a different path $(K, C_{v_1}, C_{v_2}, \ldots, C_{v_k})$ of a characteristic tree T of G_A. However, every vertex $z \in K$ belongs to at most two such paths, therefore, every vertex $z \in K$ belongs to at most two sets of the collection F. Thus, every vertex $z \in K$ belongs to at least $|C''| - 2$ sets of the collection C''.

Summarizing, we have a collection C'' of at least $k+1$ mutually incompatible sets $N_{\overline{G}_A}[v]$, $v \in I$, which contain at most k vertices of K and, also, every vertex $z \in K$ belongs to at least $|C''| - 2$ sets of the collection C''. Recall that for two vertices $u, v \in I$, the sets $N_{\overline{G}_A}[u]$ and $N_{\overline{G}_A}[v]$ are incompatible if and only if the sets $N_{\overline{G}_A}[u] \cap K$ and $N_{\overline{G}_A}[v] \cap K$ are incompatible. Therefore, we have a collection of at least $k+1$ mutually incompatible vertex sets on k vertices. It is easy to see that it is impossible to find

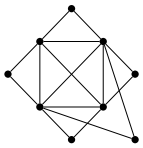

Figure 3.4: A split graph G which is not a colinear graph, since $\chi(G) = 4$ and $\lambda(\overline{G}) = 5$. Also, G is not an undirected path graph.

a collection of at least $k+1$ mutually incompatible sets on k vertices, if every vertex belongs to at least k sets of the collection. This is a contradiction to our assumptions. Therefore, G is a colinear graph. ∎

Note that, not any split graph is a colinear graph (for example see Fig. 3.4). From Theorems 3.5 and 3.6, we obtain the following result.

Corollary 3.1. *The harmonious coloring problem is NP-complete on the class of colinear graphs.*

3.5 Harmonious Coloring on Split Strongly Chordal Graphs

In this section we show that the harmonious coloring problem admits a polynomial solution on the class of split strongly chordal graphs. Strongly chordal graphs form a known subclass of chordal graphs [10, 27] and were first introduced by Farber [27]. A graph is strongly chordal iff it admits a strong elimination ordering; a vertex ordering $\sigma = (v_1, v_2, \ldots, v_n)$ is a strong elimination ordering of a graph G iff σ is a perfect elimination ordering and also has the property that for each i, j, k and ℓ, if $i < j$, $k < \ell$, $v_k, v_\ell \in N[v_i]$, and $v_k \in N[v_j]$, then $v_\ell \in N[v_j]$ [14, 27].

Let us now give the definitions of a k-sun and an incomplete k-sun. An *incomplete k-sun* S_k $(k \geq 3)$ is a chordal graph on $2k$ vertices whose vertex set can be partitioned into two sets, $U = \{u_1, u_2, \ldots, u_k\}$ and $W = \{w_1, w_2, \ldots, w_k\}$, so that W is an independent set, and w_i is adjacent to u_j if and only if $i = j$ or $i = j + 1 \pmod{k}$; the graph S_k $(k \geq 3)$ is a *k-sun* if U is a complete graph.

The following characterization of strongly chordal graphs was proved by Farber [27] and turns up to be useful in obtaining a polynomial solution for the harmonious coloring problem on split strongly chordal graphs.

Proposition 3.2. *(Farber [27]) A chordal graph G is strongly chordal if and only if it contains no induced k-sun.*

Note also that a bipartite graph G is chordal bipartite if and only if the split graph obtained from G by making one of its two color classes complete is strongly chordal [56].

Next, we present a polynomial solution for the harmonious coloring problem on split strongly chordal graphs. Before describing our algorithm, we first construct a graph H_G from a split graph G, which we call *neighborhood intersection graph of G*, and we use it in the proposed algorithm.

The neighborhood intersection graph H_G of a split graph G. Let G be a split graph, and let $V(G) = K + I$ be a partition of its vertex set, where K induces a clique in G and I induces an independent set. We first compute the open neighborhood $N_G(v)$ of each vertex $v \in I$ and, then, we construct the following graph H_G, which depicts all intersection relations among the vertices' open neighborhoods: $V(H_G) = I$ and $E(H_G) = \{xy \ : \ x, y \in I \text{ and } N_G(x) \cap N_G(y) \neq \emptyset\}$. It is easy to see that the resulting graph H_G is unique up to isomorphism.

The following result is important for proving the correctness of our algorithm.

Lemma 3.1. *The neighborhood intersection graph H_G of a split strongly chordal graph G is a chordal graph.*

Proof. Let G be a split strongly chordal graph and let H_G be the neighborhood intersection graph of G. We will show that H_G is a chordal graph, i.e., that H_G is a C_k-free graph, for every $k \geq 4$. Since G is a split graph, there exists a partition of its vertex set $V(G) = K + I$, where K induces a clique and I induces an independent set in G. By the construction of H_G, there is a one to one correspondence between the vertices of $V(H_G)$ and the vertices of $V(G) \cap I$.

Assume that H_G is not a chordal graph and let $C_k = (v_1, v_2, \ldots, v_k)$ be a chordless cycle of H_G on k vertices, $k \geq 4$; thus, $v_i v_j \in E(H_G)$ if and only if $j = i + 1 \pmod{k}$. Therefore, we have that $N_G(v_i) \cap N_G(v_j) \neq \emptyset$ if and only if $j = i + 1 \pmod{k}$ or, equivalently, there exists at least one vertex $w_i \in K$ in G such that $w_i \in N_G(v_i) \cap N_G(v_j)$ if and only if $j = i + 1 \pmod{k}$; note that, the set $W = \{w_1, w_2, \ldots, w_k\}$ consists of distinct vertices, since C_k is a chordless cycle. Thus, $U = \{v_1, v_2, \ldots, v_k\}$ induces an independent set in G, $W = \{w_1, w_2, \ldots, w_k\}$ induces a clique in G, and w_i is adjacent to v_j if and only if $j = i$ or $j = i + 1 \pmod{k}$. Therefore, the subgraph of G induced by the vertices $U \cup W$ is a k-sun, $k \geq 4$. It follows that G is a split graph and, thus, it is a chordal graph, which contains a k-sun as an induced subgraph. This is a contradiction to our assumption that G is a strongly chordal graph due to Proposition 3.2. Therefore, we conclude that H_G is a chordal graph. ∎

The algorithm for a harmonious coloring of a split strongly chordal graph. The proposed algorithm computes a harmonious coloring and the harmonious chromatic number $h(G)$ of a split strongly chordal graph G, and works as follows:

Algorithm Harmonious_Coloring

Input: a split strongly chordal graph G, and a partition of its vertex set $V(G) = K + I$, where I induces an independent set in G and K induces a clique.

Output: a harmonious coloring of G.

(i) `construct` the neighborhood intersection graph H_G of G.

(ii) `compute` a minimum proper vertex coloring $\kappa : V(H_G) \rightarrow \{1, 2, \ldots, \chi(H_G)\}$, and the chromatic number $\chi(H_G)$, of the chordal graph H_G (see e.g. [37]).

(iii) `compute` a coloring $\kappa' : V(G) \rightarrow \{1, 2, \ldots, h(G)\}$ of G, by assigning $\kappa'(v) = \kappa(v)$ to each vertex $v \in I$, and a distinct color $\kappa'(v)$ from the set $\{\chi(H_G) + 1, \chi(H_G) + 2, \ldots, \chi(H_G) + |K|\}$ to each vertex $v \in K$.

(iv) `return` the value $\kappa'(v)$ for each vertex $v \in V(G)$ and the size $\chi(H_G) + |K|$ of the number of different colors used in κ'; the coloring κ' is a harmonious coloring of G, and $\chi(H_G) + |K|$ equals the harmonious chromatic number $h(G)$ of G.

Algorithm 3: Algorithm Harmonious_Coloring

Correctness of the algorithm. Let G be a split strongly chordal graph, and let $V(G) = K + I$ be a partition of its vertex set, where I induces an independent set in G and K induces a clique. Let H_G be the neighborhood intersection graph of G.

We claim that the split strongly chordal graph G has a harmonious chromatic number $h(G) = |K| + r$, where r equals the chromatic number $\chi(H_G)$ of the graph H_G. Indeed, a harmonious coloring of G, using $h(G) = |K| + r$ colors, assigns a distinct color from the set $\{1, 2, \ldots, |K|\}$ to each vertex of the clique K, and also assigns r colors to the vertices of the set I. Note that, r is the minimum number of colors so that vertices $v_i, v_j \in I$ having a neighbor in common are assigned different colors. Since v_i, v_j are adjacent in H_G, it follows that r is the minimum number of colors for which a proper vertex coloring of H_G exists, i.e., $r = \chi(H_G)$.

Therefore, the split strongly chordal graph G has a harmonious chromatic number $h(G) = |K| + \chi(H_G)$, where $\chi(H_G)$ is the chromatic number of the neighborhood intersection graph H_G of G. Additionally, it is easy to see that the coloring κ' computed by the algorithm is a harmonious coloring of G using $h(G) = |K| + \chi(H_G)$ colors.

Complexity of the algorithm. Let G be a split strongly chordal graph on n vertices and m edges. Let $V(G) = K + I$ be a partition of its vertex set into a clique K and an independent set I, and let H_G be the neighborhood intersection graph of G. Step (i) of the algorithm, which includes the construction of the graph H_G, takes $O(n^3)$ time. Step (ii) computes a minimum proper vertex coloring of H_G; since from Lemma 3.1, H_G is a chordal graph, the problem is solvable in $O(n + m')$ time (see e.g. [37]), where $m' = |E(H_G)| = O(n^2)$. Finally, both Steps (iii) and (iv) can be executed in $O(n)$ time. Therefore, the complexity of the algorithm is $O(n^3)$ time.

Therefore, the following result holds.

Theorem 3.7. *The harmonious coloring problem has a polynomial solution on split strongly chordal graphs.*

3.6 Concluding Remarks

In this work we first show that the harmonious coloring problem is NP-complete on connected interval and permutation graphs. Also we prove the NP-completeness of the problem on the class of split graphs. Extending our results, we then prove that the harmonious coloring problem is NP-complete on the classes of split undirected path graphs and colinear graphs. We also present a polynomial solution for the same problem on the class of split strongly chordal graphs. The interest of this result lies on the fact that the harmonious coloring problem is NP-complete on split graphs and strongly chordal graphs. In addition, polynomial solutions for the problem are only known for the classes of threshold graphs and connected quasi-threshold graphs; note that, the harmonious coloring problem is NP-complete on disconnected quasi-threshold graphs. Since linear graphs form a superclass of both split strongly chordal graphs and quasi-threshold graphs, the harmonious coloring problem is NP-complete on disconnected linear graphs, while it still remains open on connected linear graphs.

Chapter 4

The Longest Path Problem on Interval Graphs

4.1 Introduction

A well studied problem in graph theory with numerous applications is the Hamiltonian path problem, i.e., the problem of determining whether a graph is Hamiltonian; a graph is said to be Hamiltonian if it contains a Hamiltonian path, that is, a simple path in which every vertex of the graph appears exactly once. Even if a graph is not Hamiltonian, it makes sense in several applications to search for a longest path, or equivalently, to find a maximum induced subgraph of the graph which is Hamiltonian. However, finding a longest path seems to be more difficult than deciding whether or not a graph admits a Hamiltonian path. Indeed, it has been proved that even if a graph has a Hamiltonian path, the problem of finding a path of length $n - n^{\varepsilon}$ for any $\varepsilon < 1$ is NP-hard, where n is the number of vertices of the graph [47]. Moreover, there is no polynomial-time constant-factor approximation algorithm for the longest path problem unless P=NP [47]. For related results see also [29, 31, 32, 66, 68].

It is clear that the longest path problem is NP-hard on every class of graphs on which the Hamiltonian path problem is NP-complete. The Hamiltonian path problem is known to be NP-complete in general graphs [33, 34], and remains NP-complete even when restricted to some small classes of graphs such as split graphs [37], chordal bipartite graphs, split strongly chordal graphs [58], circle graphs [22], planar graphs [34], and grid graphs [46]. However, it makes sense to investigate the tractability of the longest path problem on the classes of graphs for which the Hamiltonian path problem admits polynomial time solutions. Such classes include interval graphs [1], circular-arc graphs [24], biconvex graphs [3], and co-comparability graphs [23]. Note that the problem of finding a longest path on proper interval graphs is easy, since all connected proper interval graphs have a Hamiltonian path which can be computed in linear time [6]. On the contrary, not all interval graphs are Hamiltonian; in the case where an interval

graph has a Hamiltonian path, it can be computed in linear time [1, 15]. However, in the case where an interval graph is not Hamiltonian, there is no known algorithm for finding a longest path on it.

In contrast to the Hamiltonian path problem, there are few known polynomial time solutions for the longest path problem, and these restrict to trees and some small graph classes. Specifically, a linear time algorithm for finding a longest path in a tree was proposed by Dijkstra around 1960, a formal proof of which can be found in [12]. Later, through a generalization of Dijkstra's algorithm for trees, Uehara and Uno [63] solved the longest path problem for weighted trees and block graphs in linear time and space, and for cacti in $O(n^2)$ time and space, where n and m denote the number of vertices and edges of the input graph, respectively. More recently, polynomial algorithms have been proposed that solve the longest path problem on bipartite permutation graphs in $O(n)$ time and space [64], and on ptolemaic graphs in $O(n^5)$ time and $O(n^2)$ space [65].

Furthermore, Uehara and Uno in [63] introduced a subclass of interval graphs, namely interval biconvex graphs, which is a superclass of proper interval and threshold graphs, and solved the longest path problem on this class in $O(n^3(m + n \log n))$ time. As a corollary, they showed that a longest path of a threshold graph can be found in $O(n + m)$ time and space. They left open the complexity of the longest path problem on interval graphs.

In this work, we resolve the open problem posed in [63] by showing that the longest path problem admits a polynomial time solution on interval graphs. Interval graphs form an important and well-known class of perfect graphs [37]; a graph G is an interval graph if its vertices can be put in a one-to-one correspondence with a family of intervals on the real line, such that two vertices are adjacent in G if and only if their corresponding intervals intersect. In particular, we propose an algorithm for solving the longest path problem on interval graphs which runs in $O(n^4)$ time using a dynamic programming approach. Thus, not only we answer the question left open by Uehara and Uno in [63], but also improve the known time complexity of the problem on interval biconvex graphs, a subclass of interval graphs [63].

Interval graphs form a well-studied class of perfect graphs, have important properties, and admit polynomial time solutions for several problems that are NP-complete on general graphs (see e.g. [1, 37, 49, 15]). Moreover, interval graphs have received a lot of attention due to their applicability to DNA physical mapping problems [36], and find many applications in several fields and disciplines such as genetics, molecular biology, scheduling, VLSI circuit design, archaeology and psychology [37].

The rest of this chapter is organized as follows. In Section 4.2, we review some properties of interval graphs and introduce the notion of normal paths, which is central for our algorithm. In Section 4.3, we present our algorithm for solving the longest path problem on an interval graph, which includes three phases. In Section 4.4 we prove the correctness and compute the time complexity of our algorithm. Finally, some concluding remarks are given in Section 4.5.

4.2 Theoretical Framework

For basic definitions in graph theory refer to [10, 37, 56]. Recall that by $V(P)$ we denote the set of vertices in a path P, and within this chapter we consider the *length* of the path P to be the number of vertices in P, i.e., $|P| = |V(P)|$.

4.2.1 Structural Properties of Interval Graphs

A graph G is an *interval graph* if its vertices can be put in a one-to-one correspondence with a family F of intervals on the real line such that two vertices are adjacent in G if and only if the corresponding intervals intersect; F is called an *intersection model* for G [1]. The class of interval graphs is *hereditary*, that is, every induced subgraph of an interval graph G is also an interval graph. Ramalingam and Rangan [61] proposed a numbering of the vertices of an interval graph; they stated the following lemma.

Lemma 4.1. *(Ramalingam and Rangan [61]): The vertices of any interval graph G can be numbered with integers $1, 2, \ldots, |V(G)|$ such that if $i < j < k$ and $ik \in E(G)$, then $jk \in E(G)$.*

This numbering, which also results after sorting the intervals of the intersection model of an interval graph G on their right ends [1], can be obtained in $O(|V(G)| + |E(G)|)$ time [61]. An ordering of the vertices according to this numbering is found to be quite useful in solving some graph-theoretic problems on interval graphs [1, 61]. Throughout this work, such an ordering is called a *right-end ordering* of G. Let u and v be two vertices of G; if π is a right-end ordering of G, denote $u <_\pi v$ if u appears before v in π. In particular, if $\pi = (u_1, u_2, \ldots, u_{|V(G)|})$ is a right-end ordering of G, then $u_i <_\pi u_j$ if and only if $i < j$.

The following lemma appears to be useful in obtaining some important results.

Lemma 4.2. *Let G be an interval graph, and let π be a right-end ordering of G. Let $P = (v_1, v_2, \ldots, v_k)$ be a path of G, and let $v_\ell \notin V(P)$ be a vertex of G such that $v_1 <_\pi v_\ell <_\pi v_k$ and $v_\ell v_k \notin E(G)$. Then, there exist two consecutive vertices v_{i-1} and v_i in P, $2 \leq i \leq k$, such that $v_{i-1}v_\ell \in E(G)$ and $v_\ell <_\pi v_i$.*

Proof. Consider the intersection model F of G, from which we obtain the right-end ordering π of G. Let I_i denote the interval which corresponds to the vertex v_i in F, and let $l(I_i)$ and $r(I_i)$ denote the left and the right endpoint of the interval I_i, respectively. Without loss of generality, we may assume that all values $l(I_i)$ and $r(I_i)$ are distinct. Since $P = (v_1, v_2, \ldots, v_k)$ is a path from v_1 to v_k, it is clear from the intersection model F of G that at least one vertex of P sees v_ℓ. Recall that $v_k v_\ell \notin E(G)$; let v_{i-1}, $2 \leq i \leq k$, be the last vertex of P such that $v_{i-1}v_\ell \in E(G)$, i.e., $v_j v_\ell \notin E(G)$ for every index j, $i \leq j \leq k$. Thus, since $v_\ell <_\pi v_k$, it follows that $r(I_\ell) < l(I_j) < r(I_j)$ for every index j, $i \leq j \leq k$ and, thus, $v_\ell <_\pi v_j$. Therefore, in particular, $v_\ell <_\pi v_i$. This completes the proof. ∎

4.2.2 Normal Paths

Our algorithm for constructing a longest path of an interval graph G uses a specific type of paths, namely normal paths. We next define the notion of a normal path of an interval graph G.

Definition 4.1. Let G be an interval graph, and let π be a right-end ordering of G. The path $P = (v_1, v_2, \ldots, v_k)$ of G is called *normal*, if v_1 is the leftmost vertex of $V(P)$ in π, and for every i, $2 \leq i \leq k$, the vertex v_i is the leftmost vertex of $N(v_{i-1}) \cap \{v_i, v_{i+1}, \ldots, v_k\}$ in π.

The notion of a normal path of an interval graph G is a generalization of the notion of a typical path of G; the path $P = (v_1, v_2, \ldots, v_k)$ of an interval graph G is called a *typical* path, if v_1 is the leftmost vertex of $V(P)$ in π. The notion of a typical path was introduced by Arikati and Rangan [1], in order to solve the path cover problem on interval graphs; they proved the following result.

Lemma 4.3. *(Arikati and Rangan [1]): Let P be a path of an interval graph G. Then, there exists a typical path P' in G such that $V(P') = V(P)$.*

The following lemma is the basis of our algorithm for solving the longest path problem on interval graphs.

Lemma 4.4. *Let P be a path of an interval graph G. Then, there exists a normal path P' of G, such that $V(P') = V(P)$.*

Proof. Let G be an interval graph, let π be a right-end ordering of G, and let $P = (v_1, v_2, \ldots, v_k)$ be a path of G. If $k = 1$, the lemma clearly holds. Suppose that $k \geq 2$. We will prove that for every index i, $2 \leq i \leq k$, there exists a path $P_i = (v'_1, v'_2, \ldots, v'_k)$, such that $V(P_i) = V(P)$, v'_1 is the leftmost vertex of $V(P_i)$ in π, and for every index j, $2 \leq j \leq i$, the vertex v'_j is the leftmost vertex of $N(v'_{j-1}) \cap \{v'_j, v'_{j+1}, \ldots, v'_k\}$ in π. The proof will be done by induction on i.

Due to Lemma 4.3, we may assume that $P = (v_1, v_2, \ldots, v_k)$ is typical, i.e., that v_1 is the leftmost vertex of $V(P)$ in π. Let $i = 2$. Assume that $v_j \in V(P)$, $j > 2$, is the leftmost vertex of $N(v_1) \cap \{v_2, v_3, \ldots, v_k\}$ in π. Then, since $G[V(P)]$ is an interval graph, and since $v_1 <_\pi v_j <_\pi v_2$ and $v_1v_2, v_1v_j \in E(G)$, it follows that $N[v_j] \cap \{v_1, v_2, \ldots, v_k\} \subseteq N[v_2] \cap \{v_1, v_2, \ldots, v_k\}$. Thus, there exists a path

$$P_2 = (v'_1, v'_2, \ldots, v'_k) = (v_1, v_j, v_{j-1}, \ldots, v_3, v_2, v_{j+1}, v_{j+2} \ldots, v_k)$$

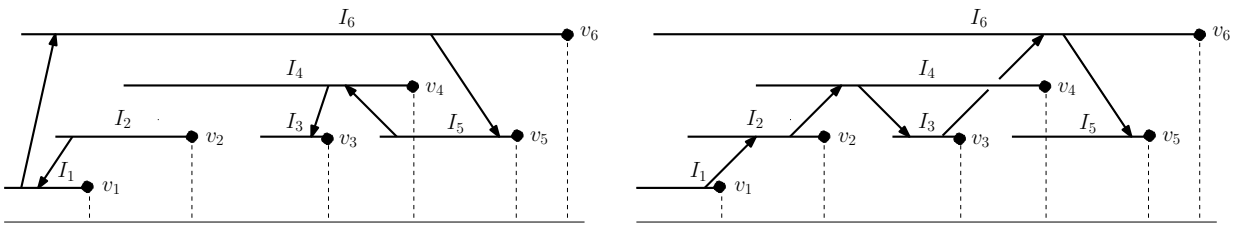

Figure 4.1: Illustrating an intersection model of an interval graph G. The path $P = (v_2, v_1, v_6, v_5, v_4, v_3)$, which is Hamiltonian for the graph G, is not a normal path. The path $P' = (v_1, v_2, v_4, v_3, v_6, v_5)$ is a normal path such that $V(P') = V(P)$.

of G, such that $V(P_2) = V(P)$, v'_1 is the leftmost vertex of $V(P_2)$ in π, and v'_2 is the leftmost vertex of $N(v'_1) \cap \{v'_2, v'_3, \ldots, v'_k\}$ in π. This proves the induction basis.

Consider now an arbitrary index i, $2 \leq i \leq k-1$, and let $P_i = (v'_1, v'_2, \ldots, v'_k)$ be a path of G, such that $V(P_i) = V(P)$, v'_1 is the leftmost vertex of $V(P_i)$ in π, and for every index j, $2 \leq j \leq i$, the vertex v'_j is the leftmost vertex of $N(v'_{j-1}) \cap \{v'_j, v'_{j+1}, \ldots, v'_k\}$ in π. In particular, it follows that the subpath $(v'_1, v'_2, \ldots, v'_i)$ of P_i is normal. We will now prove that for any vertex $v'_\ell \in \{v'_{i+1}, v'_{i+2}, \ldots, v'_k\}$, where $v'_\ell <_\pi v'_i$, it holds $v'_\ell v'_i \in E(G)$. Indeed, suppose otherwise that $v'_\ell v'_i \notin E(G)$, for such a vertex v'_ℓ. Then, since $v'_1 <_\pi v'_\ell <_\pi v'_i$, it follows by Lemma 4.2 that there are two consecutive vertices v'_{j-1} and v'_j in P_i, $2 \leq j \leq i$, such that $v'_{j-1}v'_\ell \in E(G)$ and $v'_\ell <_\pi v'_j$. Thus, v'_j is not the leftmost vertex of $N(v'_{j-1}) \cap \{v'_j, v'_{j+1}, \ldots, v'_\ell, \ldots, v'_k\}$ in π, which is a contradiction. Therefore, for any vertex $v'_\ell \in \{v'_{i+1}, v'_{i+2}, \ldots, v'_k\}$, where $v'_\ell <_\pi v'_i$, it holds $v'_\ell v'_i \in E(G)$.

Assume that $v'_j \in V(P_i)$, $j > i+1$, is the leftmost vertex of $N(v'_i) \cap \{v'_{i+1}, v'_{i+2}, \ldots, v'_k\}$ in π. Consider first the case where $v'_i <_\pi v'_j$. Then, for every vertex $v'_\ell \in \{v'_{i+1}, v'_{i+2}, \ldots, v'_k\}$ it holds $v'_i <_\pi v'_\ell$. Indeed, suppose otherwise that $v'_\ell <_\pi v'_i <_\pi v'_j$ for such a vertex v'_ℓ. Then, as we have proved above, $v'_\ell v'_i \in E(G)$, which is a contradiction, since v'_j is the leftmost vertex of $N(v'_i) \cap \{v'_{i+1}, v'_{i+2}, \ldots, v'_k\}$ in π and $v'_\ell <_\pi v'_j$. Thus, $v'_i <_\pi v'_\ell$ for every vertex $v'_\ell \in \{v'_{i+1}, v'_{i+2}, \ldots, v'_k\}$. Therefore, since $G[V(P_i)]$ is an interval graph, and since $v'_i <_\pi v'_j <_\pi v'_{i+1}$ and $v'_i v'_{i+1}, v'_i v'_j \in E(G)$, it follows that $N[v'_j] \cap \{v'_i, v'_{i+1}, \ldots, v'_k\} \subseteq N[v'_{i+1}] \cap \{v'_i, v'_{i+1}, \ldots, v'_k\}$. Then, there exists the path

$$P_{i+1} = (v''_1, v''_2, \ldots, v''_i, v''_{i+1}, \ldots, v''_k) = (v'_1, v'_2, \ldots, v'_i, v'_j, v'_{j-1}, \ldots, v'_{i+2}, v'_{i+1}, v'_{j+1}, \ldots, v'_k)$$

of G, such that $V(P_{i+1}) = V(P_i)$, v''_1 is the leftmost vertex of $V(P_{i+1})$ in π, and for every index j, $2 \leq j \leq i+1$, the vertex v''_j is the leftmost vertex of $N(v''_{j-1}) \cap \{v''_j, v''_{j+1}, \ldots, v''_k\}$ in π.

Consider now the case where $v'_j <_\pi v'_i$. Then, v'_j is the leftmost vertex of $\{v'_{i+1}, v'_{i+2}, \ldots, v'_k\}$ in π. Indeed, suppose otherwise that $v'_\ell <_\pi v'_j <_\pi v'_i$ for a vertex $v'_\ell \in \{v'_{i+1}, v'_{i+2}, \ldots, v'_k\}$. Then, as we have proved above, $v'_\ell v'_i \in E(G)$, which is a contradiction, since v'_j is the leftmost vertex of $N(v'_i) \cap \{v'_{i+1}, v'_{i+2}, \ldots, v'_k\}$ in π and $v'_\ell <_\pi v'_j$. Thus, there exists by Lemma 4.3 a typical path P_0, such that $V(P_0) = \{v'_{i+1}, v'_{i+2}, \ldots, v'_k\}$. Since P_0 is typical and v'_j is the leftmost vertex of $V(P_0)$ in π, it follows that v'_j is the first vertex of P_0. Then, since $v'_i v'_j \in E(G)$, there exists the path

$$P_{i+1} = (v''_1, v''_2, \ldots, v''_i, v''_{i+1}, \ldots, v''_k) = (v'_1, v'_2, \ldots, v'_i, P_0)$$

of G, such that $V(P_{i+1}) = V(P_i)$, v''_1 is the leftmost vertex of $V(P_{i+1})$ in π, and for every index j, $2 \leq j \leq i+1$, the vertex v''_j is the leftmost vertex of $N(v''_{j-1}) \cap \{v''_j, v''_{j+1}, \ldots, v''_k\}$ in π. This proves the induction step.

Thus, the path $P' = P_k$ is a normal path of G, such that $V(P') = V(P)$. ∎

4.3 Interval Graphs and the Longest Path Problem

In this section we present our algorithm, which we call Algorithm LP_Interval, for solving the longest path problem on interval graphs; it consists of three phases and works as follows:

- Phase 1: it takes an interval graph G and constructs the auxiliary interval graph H;
- Phase 2: it computes a longest path P on H using Algorithm LP_on_H;
- Phase 3: it computes a longest path $\widehat{P}$ on G from the path P;

The proposed algorithm computes a longest path P of the graph H using dynamic programming techniques and, then, computes a longest path $\widehat{P}$ of G from the path P. We next describe in detail the three phases of our algorithm and prove properties of the constructed graph H which will be used for proving the correctness of the algorithm.

4.3.1 The interval graph H

In this section we present Phase 1 of the algorithm: given an interval graph G and a right-end ordering π of G, we construct the interval graph H and a right-end ordering σ of H.

► **Construction of H and σ:** Let G be an interval graph and let $\pi = (v_1, v_2, \ldots, v_{|V(G)|})$ be a right-end ordering of G. Initially, set $V(H) = V(G)$, $E(H) = E(G)$, $\sigma = \pi$, and $A = \emptyset$. Traverse the vertices of π from left to right and do the following: for every vertex v_i add two vertices $a_{i,1}$ and $a_{i,2}$ to $V(H)$ and make both these vertices to be adjacent to every vertex in $N_G[v_i] \cap \{v_i, v_{i+1}, \ldots, v_{|V(G)|}\}$; add $a_{i,1}$ and $a_{i,2}$ to A. Update σ such that $a_{1,1} <_\sigma a_{1,2} <_\sigma v_1$, and $v_{i-1} <_\sigma a_{i,1} <_\sigma a_{i,2} <_\sigma v_i$ for every i, $2 \leq i \leq |V(G)|$.

We call the constructed graph H the *stable-connection graph* of the graph G. Hereafter, we will denote by n the number $|V(H)|$ of vertices of the graph H and by $\sigma = (u_1, u_2, \ldots, u_n)$ the constructed ordering of H. By construction, the vertex set of the graph H consists of the vertices of the set $C = V(G)$ and the vertices of the set A. We will refer to C as the set of the *connector vertices* c of the graph H and to A as the set of *stable vertices* a of the graph H; we denote these sets by $C(H)$ and $A(H)$, respectively. Note that $|A(H)| = 2|V(G)|$.

By the construction of the stable-connection graph H, all neighbors of a stable vertex $a \in A(H)$ are connector vertices $c \in C(H)$, such that $a <_\sigma c$. Moreover, observe that all neighbors of a stable vertex form a clique in G and, thus, also in H. For every connector vertex $u_i \in C(H)$, we denote by $u_{f(u_i)}$ and $u_{h(u_i)}$ the leftmost and rightmost neighbor of u_i in σ, respectively, which appear before u_i in σ, i.e., $u_{f(u_i)} <_\sigma u_{h(u_i)} <_\sigma u_i$. Note that $u_{f(u_i)}$ and $u_{h(u_i)}$ are distinct stable vertices, for every connector vertex u_i.

Lemma 4.5. *Let G be an interval graph. The stable-connection graph H of G is an interval graph, and the vertex ordering σ is a right-end ordering of H.*

Proof. Consider the intersection model F of G, from which we obtain the right-end ordering $\pi = (v_1, v_2, \ldots, v_{|V(G)|})$ of G. Let I_i denote the interval which corresponds to the vertex v_i in F, and let $l(I_i)$ and $r(I_i)$ denote the left and the right endpoint of the interval I_i, respectively. Without loss of generality, we may assume that all values $l(I_i)$ and $r(I_i)$ are distinct. Let ε be the smallest distance between two interval endpoints in F.

For every interval I_i which corresponds to a vertex $v_i \in C$, we replace its right endpoint $r(I_i)$ by $r(I_i) + \frac{\varepsilon}{2}$, and we add two non-intersecting intervals $I_{i,1} = [r(I_i), r(I_i) + \frac{\varepsilon}{8}]$ and $I_{i,2} = [r(I_i) + \frac{\varepsilon}{4}, r(I_i) + \frac{3\varepsilon}{8}]$ (one for each vertex $a_{i,1}$ and $a_{i,2}$ of A, respectively). The two new intervals do not intersect with any interval I_k, such that $r(I_k) < r(I_i)$. Additionally, the two new intervals

intersect with the interval I_i, and with every interval I_ℓ, such that $r(I_\ell) > r(I_i)$ and I_ℓ intersects with I_i. After processing all intervals I_i, $1 \leq i \leq |V(G)|$, of the intersection model F of G, we obtain an intersection model of H. Thus, H is an interval graph, and the ordering which results from numbering the intervals after sorting them on their right ends is identical to the vertex ordering σ of H and, thus, σ is a right-end ordering of H. ∎

Definition 4.2. Let H be the stable-connection graph of an interval graph G, and let $\sigma = (u_1, u_2, \ldots, u_n)$ be the right-end ordering of H. For every pair of indices i, j, $1 \leq i \leq j \leq n$, we define the graph $H(i,j)$ to be the subgraph $H[S]$ of H, induced by the set $S = \{u_i, u_{i+1}, \ldots, u_j\} \setminus \{u_k \in C(H) : u_{f(u_k)} <_\sigma u_i\}$.

The following properties hold for every induced subgraph $H(i,j)$, $1 \leq i \leq j \leq n$, and they are used for proving the correctness of Algorithm LP_on_H.

Observation 4.1. Let u_k be a connector vertex of $H(i,j)$, i.e., $u_k \in C(H(i,j))$. Then, for every vertex $u_\ell \in V(H(i,j))$, such that $u_k <_\sigma u_\ell$ and $u_k u_\ell \in E(H(i,j))$, u_ℓ is also a connector vertex of $H(i,j)$.

Observation 4.2. No two stable vertices of $H(i,j)$ are adjacent.

Lemma 4.6. *Let $P = (v_1, v_2, \ldots, v_k)$ be a normal path of $H(i,j)$. Then:*

(a) For any two stable vertices v_r and v_ℓ in P, v_r appears before v_ℓ in P if and only if $v_r <_\sigma v_\ell$.

(b) For any two connector vertices v_r and v_ℓ in P, if v_ℓ appears before v_r in P and $v_r <_\sigma v_\ell$, then v_r does not see the previous vertex $v_{\ell-1}$ of v_ℓ in P.

Proof. The proof will be done by contradiction.

(a) Let v_r and v_ℓ be any two stable vertices of $H(i,j)$ that belong to the normal path $P = (v_1, v_2, \ldots, v_k)$, such that v_r appears before v_ℓ in P, and assume that $v_\ell <_\sigma v_r$. Then, clearly $v_\ell \neq v_1$, since v_r appears before v_ℓ in P. Since P is a normal path of $H(i,j)$, v_1 is the leftmost vertex of $V(P)$ in σ. Thus, $v_1 <_\sigma v_\ell <_\sigma v_r$, and since no two stable vertices of $H(i,j)$ are adjacent due to Observation 4.2, it follows that $v_r v_\ell \notin E(H(i,j))$. Thus, by Lemma 4.2 there exist two consecutive vertices u and u' in P that appear between v_1 and v_r in P, such that $uv_\ell \in E(H(i,j))$ and $v_\ell <_\sigma u'$. Thus, since P is a normal path, v_ℓ should be the next vertex of u in P instead of u', which is a contradiction. Therefore, $v_r <_\sigma v_\ell$.

(b) Let v_r and v_ℓ be any two connector vertices of $H(i,j)$ that belong to the normal path $P = (v_1, v_2, \ldots, v_k)$, such that v_ℓ appears before v_r in P and $v_r <_\sigma v_\ell$. Since P is a normal path of $H(i,j)$, v_1 is the leftmost vertex of $V(P)$ in σ. Since $v_r <_\sigma v_\ell$, it follows that $v_\ell \neq v_1$ and, thus, there exists a vertex $v_{\ell-1}$ which appears before v_ℓ in P. Assume that $v_r v_{\ell-1} \in E(H(i,j))$. Since $v_r <_\sigma v_\ell$, and since P is a normal path, v_r should be the next vertex of $v_{\ell-1}$ in P instead of v_ℓ, which is a contradiction. Therefore, $v_r v_{\ell-1} \notin E(H(i,j))$.

∎

4.3.2 Finding a longest path on H

In this section we present Phase 2 of Algorithm LP_Interval. Let G be an interval graph and let H be the stable-connection graph of G constructed in Phase 1. We next present Algorithm LP_on_H, which computes a longest path of the graph H. Let us first give some definitions and notations necessary for the description of the algorithm.

Definition 4.3. Let H be a stable-connection graph, and let P be a path of $H(i,j)$, $1 \leq i \leq j \leq n$. The path P is called *binormal* if P is a normal path of $H(i,j)$, both endpoints of P are stable vertices, and no two connector vertices are consecutive in P.

ALGORITHM LP_ON_H

Input: a stable-connection graph H, a right-end ordering $\sigma = (u_1, u_2, \ldots, u_n)$ of H.
Output: a longest binormal path of H.

```
for j = 1 to n
    for i = j downto 1
      if i = j and u_i ∈ A(H) then
          ℓ(u_i; i, i) ← 1;   P(u_i; i, i) ← (u_i);
      if i ≠ j then
          for every stable vertex u_k ∈ A(H), i ≤ k ≤ j − 1
              ℓ(u_k; i, j) ← ℓ(u_k; i, j − 1);   P(u_k; i, j) ← P(u_k; i, j − 1);   {initialization}
          if u_j is a stable vertex of H(i, j), i.e., u_j ∈ A(H) then
              ℓ(u_j; i, j) ← 1;   P(u_j; i, j) ← (u_j);
          if u_j is a connector vertex of H(i, j), i.e., u_j ∈ C(H) and i ≤ f(u_j) then
              execute process(H(i, j));
compute the max{ℓ(u_k; 1, n) : u_k ∈ A(H)} and the corresponding path P(u_k; 1, n);
```

where the procedure `process()` is as follows:

```
process(H(i, j))

for y = f(u_j) + 1 to j − 1
    for x = f(u_j) to y − 1      {u_x and u_y are adjacent to u_j}
       if u_x, u_y ∈ A(H) then
          w_1 ← ℓ(u_x; i, j − 1);   P'_1 ← P(u_x; i, j − 1);
          w_2 ← ℓ(u_y; x + 1, j − 1);   P'_2 ← P(u_y; x + 1, j − 1);
          if w_1 + w_2 + 1 > ℓ(u_y; i, j) then
              ℓ(u_y; i, j) ← w_1 + w_2 + 1;   P(u_y; i, j) ← (P'_1, u_j, P'_2);
return the value ℓ(u_k; i, j) and the path P(u_k; i, j), ∀ u_k ∈ A(H(f(u_j) + 1, j − 1));
```

Algorithm 4: Algorithm LP_on_H for finding a longest binormal path of H.

Notation 4.1. *Let H be a stable-connection graph, and let $\sigma = (u_1, u_2, \ldots, u_n)$ be the right-end ordering of H. For every stable vertex $u_k \in A(H(i,j))$, we denote by $P(u_k; i, j)$ a longest binormal path of $H(i,j)$ with u_k as its right endpoint, and by $\ell(u_k; i, j)$ the length of $P(u_k; i, j)$.*

Since any binormal path is a normal path, Lemma 4.6 also holds for binormal paths. Moreover, since $P(u_k; i, j)$ is a binormal path, it follows that its right endpoint u_k is also the rightmost stable vertex of P in σ, due to Lemma 4.6(a).

Algorithm LP_on_H computes for every induced subgraph $H(i,j)$ and for every stable vertex $u_k \in A(H(i,j))$, the length $\ell(u_k; i, j)$ and the corresponding path $P(u_k; i, j)$. Since $H(1, n) = H$, it follows that the maximum among the values $\ell(u_k; 1, n)$, where $u_k \in A(H)$, is the length of a longest binormal path $P(u_k; 1, n)$ of H. In Section 4.4.2 we prove that the length of a longest path of H equals to the length of a longest binormal path of H. Thus, the binormal path $P(u_k; 1, n)$ computed by Algorithm LP_on_H is also a longest path of H.

4.3.3 Finding a longest path on G

During Phase 3 of our Algorithm LP_Interval, we compute a path $\widehat{P}$ from the longest binormal path P of H, computed by Algorithm LP_on_H, by simply deleting all the stable vertices of P. In Section 4.4.2 we prove that the resulting path $\widehat{P}$ is a longest path of the interval graph G.

ALGORITHM LP_INTERVAL

Input: an interval graph G and a right-end ordering π of G.
Output: a longest path $\widehat{P}$ of G.

1. Construct the stable-connection graph H of G and the right-end ordering σ of H;
   let $V(H) = C \cup A$, where $C = V(G)$ and A are the sets of the connector and stable vertices of H, respectively;
2. Compute a longest binormal path P of H, using Algorithm LP_on_H;
   let $P = (v_1, v_2, \ldots, v_{2k}, v_{2k+1})$, where $v_{2i} \in C$, $1 \leq i \leq k$, and $v_{2i+1} \in A$, $0 \leq i \leq k$;
3. Compute a longest path $\widehat{P} = (v_2, v_4, \ldots, v_{2k})$ of G, by deleting all stable vertices $\{v_1, v_3, \ldots, v_{2k+1}\}$ from the longest binormal path P of H;

Algorithm 5: Algorithm LP_Interval for solving the longest path problem on an interval graph G.

In this section we present our Algorithm LP_Interval for solving the longest path problem on an interval graph G; note that Steps 1, 2, and 3 of the algorithm correspond to the presented Phases 1, 2, and 3, respectively.

4.4 Correctness and Time Complexity

In this section we prove the correctness of our algorithm and compute its time complexity. More specifically, in Section 4.4.1 we show that Algorithm LP_on_H computes a longest binormal path P of the graph H (in Lemma 4.13 we prove that this path is also a longest path of H), while in Section 4.4.2 we show that the length of a longest binormal path P of H is equal to $2k+1$, where k is the length of a longest path of G. Finally, we show that the path $\widehat{P}$ constructed at Step 3 of Algorithm LP_Interval is a longest path of G.

4.4.1 Correctness of Algorithm LP_on_H

We next prove that Algorithm LP_on_H correctly computes a longest binormal path of the graph H. The following lemmas appear useful in the proof of the algorithm's correctness.

Lemma 4.7. *Let H be a stable-connection graph, and let $\sigma = (u_1, u_2, \ldots, u_n)$ be the right-end ordering of H. Let P be a longest binormal path of $H(i,j)$ with u_y as its right endpoint, let u_k be the rightmost connector vertex of $H(i,j)$ in σ, and let $u_{f(u_k)+1} \leq_\sigma u_y \leq_\sigma u_{h(u_k)}$. Then, there exists a longest binormal path P' of $H(i,j)$ with u_y as its right endpoint, which contains the connector vertex u_k.*

Proof. Let P be a longest binormal path of $H(i,j)$ with u_y as its right endpoint, which does not contain the connector vertex u_k. Assume that $P = (u_y)$. Since u_k is a connector vertex of $H(i,j)$ and $u_{f(u_k)}$ is a stable vertex of $H(i,j)$, we have that $u_i \leq_\sigma u_{f(u_k)} <_\sigma u_y <_\sigma u_k$. Thus, there exists a binormal path $P_1 = (u_{f(u_k)}, u_k, u_y)$ such that $|P_1| > |P|$. However, this is a contradiction to the assumption that P is a longest binormal path of $H(i,j)$.

Therefore, assume now that $P = (u_p, \ldots, u_q, u_\ell, u_y)$. By assumption, P is a longest binormal path of $H(i,j)$ with u_y as its right endpoint that does not contain the connector vertex u_k. Since the connector vertex u_ℓ sees the stable vertex u_y and, also, since u_k is the rightmost connector vertex of $H(i,j)$ in σ, it follows by Observation 4.1 that $u_{f(u_k)} <_\sigma u_y <_\sigma u_\ell <_\sigma u_k$. Thus, u_k sees the connector vertex u_ℓ. Consider first the case where u_k does not see the stable vertex u_q, i.e., $u_q <_\sigma u_{f(u_k)} <_\sigma u_y <_\sigma u_\ell <_\sigma u_k$. Then, it is easy to see that the connector vertex u_ℓ sees $u_{f(u_k)}$, where $u_{f(u_k)}$ is always a stable vertex,

and also, from Lemma 4.6(a) it follows that the vertex $u_{f(u_k)}$ does not belong to the path P. Therefore, there exists a binormal path $P_2 = (u_p, \ldots, u_q, u_\ell, u_{f(u_k)}, u_k, u_y)$ in $H(i,j)$, such that $|P_2| > |P|$. This is a contradiction to our assumption that P is a longest binormal path.

Consider now the case where u_k sees the stable vertex u_q. Then, there exists a path $P' = (u_p, \ldots, u_q, u_k, u_y)$ of $H(i,j)$ with u_y as its right endpoint that contains the connector vertex u_k, such that $|P| = |P'|$; since P is a binormal path, it is easy to see that P' is also a binormal path. Thus, the path P' is a longest binormal path of $H(i,j)$ with u_y as its right endpoint, which contains the connector vertex u_k. ∎

Lemma 4.8. *Let H be a stable-connection graph, and let σ be the right-end ordering of H. Let $P = (P_1, v_\ell, P_2)$ be a binormal path of $H(i,j)$, and let v_ℓ be a connector vertex of $H(i,j)$. Then, P_1 and P_2 are binormal paths of $H(i,j)$.*

Proof. Let $P = (v_1, v_2, \ldots, v_{\ell-1}, v_\ell, v_{\ell+1}, \ldots, v_k)$ be a binormal path of $H(i,j)$. Then, from Definition 4.1, v_1 is the leftmost vertex of $V(P)$ in σ, and for every index r, $2 \leq r \leq k$, the vertex v_r is the leftmost vertex of $N(v_{r-1}) \cap \{v_r, v_{r+1}, \ldots, v_k\}$ in σ. It is easy to see that $P_1 = (v_1, v_2, \ldots, v_{\ell-1})$ is a normal path of $H(i,j)$. Indeed, since $V(P_1) \subset V(P)$, then v_1 is also the leftmost vertex of $V(P_1)$ in σ, and additionally, v_r is the leftmost vertex of $N(v_{r-1}) \cap \{v_r, v_{r+1}, \ldots, v_{\ell-1}\}$ in σ, for every index r, $2 \leq r \leq \ell-1$. Furthermore, since P is binormal and v_ℓ is a connector vertex, it follows that $v_{\ell-1}$ is a stable vertex and, thus, P_1 is a binormal path of $H(i,j)$ as well.

Consider now the path $P_2 = (v_{\ell+1}, v_{\ell+2}, \ldots, v_k)$ of $H(i,j)$. Since P is a binormal path and v_ℓ is a connector vertex, it follows that $v_{\ell+1}$ is a stable vertex and, thus, $v_{\ell+1} <_\sigma v_\ell$ due to Observation 4.1. We first prove that $v_{\ell+1}$ is the leftmost vertex of $V(P_2)$ in σ. Since P is a binormal path, we obtain from Lemma 4.6(a) that $v_{\ell+1}$ is the leftmost stable vertex of $V(P_2)$ in σ. Moreover, consider a connector vertex v_t of P_2. Then, its previous vertex v_{t-1} in P_2 is a stable vertex and, thus, $v_{t-1} <_\sigma v_t$ due to Observation 4.1. Since $v_{\ell+1}$ is the leftmost stable vertex of $V(P_2)$ in σ, we have that $v_{\ell+1} \leq_\sigma v_{t-1}$ and, thus, $v_{\ell+1} <_\sigma v_t$. Therefore, $v_{\ell+1}$ is the leftmost vertex of $V(P_2)$ in σ. Additionally, since P is a binormal path, it is straightforward that for every index r, $\ell+2 \leq r \leq k$, the vertex v_r is the leftmost vertex of $N(v_{r-1}) \cap \{v_r, v_{r+1}, \ldots, v_k\}$ in σ. Thus, P_2 is a normal path. Finally, since P is binormal and $v_{\ell+1}$ is a stable vertex, P_2 is a binormal path as well. ∎

Lemma 4.9. *Let H be a stable-connection graph, and let $\sigma = (u_1, u_2, \ldots, u_n)$ be the right-end ordering of H. Let P_1 be a binormal path of $H(i, j-1)$ with u_x as its right endpoint, and let P_2 be a binormal path of $H(x+1, j-1)$ with u_y as its right endpoint, such that $V(P_1) \cap V(P_2) = \emptyset$. Suppose that u_j is a connector vertex of H and that $u_i \leq_\sigma u_{f(u_j)} \leq_\sigma u_x$. Then, $P = (P_1, u_j, P_2)$ is a binormal path of $H(i,j)$ with u_y as its right endpoint.*

Proof. Let P_1 be a binormal path of $H(i, j-1)$ with u_x as its right endpoint, and let P_2 be a binormal path of $H(x+1, j-1)$ with u_y as its right endpoint, such that $V(P_1) \cap V(P_2) = \emptyset$. Let u_z be the first vertex of P_2. Since u_j is a connector vertex of H such that $u_i \leq_\sigma u_{f(u_j)} \leq_\sigma u_x$ it follows that u_j sees the right endpoint u_x of P_1. Additionally, since $u_z \in V(H(x+1, j-1))$, we have $u_{f(u_j)} \leq_\sigma u_x <_\sigma u_{x+1} \leq_\sigma u_z <_\sigma u_j$ and, thus, u_j sees u_z. Therefore, since $V(P_1) \cap V(P_2) = \emptyset$, it follows that $P = (P_1, u_j, P_2)$ is a path of H. Additionally, since $H(i, j-1)$ and $H(x+1, j-1)$ are induced subgraphs of $H(i,j)$, it follows that P is a path of $H(i,j)$. Hereafter, in the rest of this proof $P_1 = (v_1, v_2, \ldots, v_{p-1})$, $P_2 = (v_{p+1}, v_{p+2}, \ldots, v_\ell)$, $u_x = v_{p-1}$, $u_y = v_\ell$, and $u_j = v_p$.

We first show that $P = (v_1, v_2, \ldots, v_p, \ldots, v_\ell)$ is a normal path. Since v_1 is the leftmost vertex of $V(P_1)$ in σ, it follows that $v_1 \leq_\sigma u_x$. Furthermore, since for every vertex $v_k \in V(P_2)$ it holds $u_x <_\sigma u_{x+1} \leq_\sigma v_k$, it follows that v_1 is the leftmost vertex of $V(P)$ in σ. We next show that for every k, $2 \leq k \leq \ell$, the vertex v_k is the leftmost vertex of $N(v_{k-1}) \cap \{v_k, v_{k+1}, \ldots, v_\ell\}$ in σ.

Consider first the case where $2 \leq k \leq p-1$, i.e., $v_k \in V(P_1)$. Since P_1 is a normal path, v_k is the leftmost vertex of $N(v_{k-1}) \cap \{v_k, v_{k+1}, \ldots, v_{p-1}\}$ in σ. Assume that v_{k-1} is a stable vertex. Then, Lemma 4.6(a) implies that $v_{k-1} <_\sigma u_x$ and, due to Observation 4.2, it follows that

$N(v_{k-1}) \cap \{v_k, v_{k+1}, \ldots, v_\ell\}$ is a set of connector vertices. Since every connector vertex $v_r \in V(P_2)$ is a vertex of $H(x+1, j-1)$, it follows that $v_{k-1} <_\sigma u_{x+1} \leq_\sigma u_{f(v_r)}$ and, thus, $v_r \notin N(v_{k-1})$. Additionally, since v_p is the rightmost vertex of $H(i,j)$ in σ, it follows that $v_k <_\sigma v_p$. Therefore, since v_k is the leftmost vertex of $N(v_{k-1}) \cap \{v_k, v_{k+1}, \ldots, v_{p-1}\}$ in σ, it follows that v_k is the leftmost vertex of $N(v_{k-1}) \cap \{v_k, v_{k+1}, \ldots, v_\ell\}$ in σ. Assume now that v_{k-1} is a connector vertex. Since P_1 is a binormal path, v_k is a stable vertex such that $v_k \leq_\sigma u_x$ and v_k is the leftmost vertex of $N(v_{k-1}) \cap \{v_k, v_{k+1}, \ldots, v_{p-1}\}$ in σ. Since for every r, $p+1 \leq r \leq \ell$, the vertex $v_r \in V(H(x+1, j-1))$, it follows that $v_k \leq_\sigma u_x <_\sigma v_r$. Additionally, $v_k <_\sigma u_{x+1} <_\sigma v_p$. Therefore, v_k is the leftmost vertex of $N(v_{k-1}) \cap \{v_k, v_{k+1}, \ldots, v_\ell\}$ in σ.

Consider now the case where $k = p$. Since P_1 is a normal path and v_{p-1} is a stable vertex, $N(v_{p-1}) \cap \{v_p, v_{p+1}, \ldots, v_\ell\}$ is a set of connector vertices, due to Observation 4.2. Additionally, since every connector vertex $v_r \in V(P_2)$ is a vertex of $H(x+1, j-1)$, it follows that $v_{p-1} <_\sigma u_{x+1} \leq_\sigma u_{f(v_r)}$ and, thus, $v_r \notin N(v_{p-1})$. Therefore, $N(v_{p-1}) \cap \{v_p, v_{p+1}, \ldots, v_\ell\} = \{v_p\}$ and, thus, v_p is the leftmost vertex of $N(v_{p-1}) \cap \{v_p, v_{p+1}, \ldots, v_\ell\}$ in σ. Now, in the case where $k = p+1$, we have that v_{p+1} is the leftmost vertex of $V(P_2) = \{v_{p+1}, v_{p+2}, \ldots, v_\ell\}$ in σ, since P_2 is a normal path. Therefore, it easily follows that v_{p+1} is the leftmost vertex of $N(v_p) \cap \{v_{p+1}, v_{p+2}, \ldots, v_\ell\}$ in σ. Finally, in the case where $p+2 \leq k \leq \ell$, since P_2 is a normal path it directly follows that v_k is the leftmost vertex of $N(v_{k-1}) \cap \{v_k, v_{k+1}, \ldots, v_\ell\}$ in σ.

Concluding, we have shown that P is a normal path of $H(i,j)$. Additionally, since P_1 and P_2 are binormal paths of $H(i,j)$, the path P has stable vertices as endpoints and no two connector vertices are consecutive in P. Therefore, P is a binormal path of $H(i,j)$ with u_y as its right endpoint. ∎

Next, we prove the correctness of Algorithm LP_on_H.

Lemma 4.10. *Let H be a stable-connection graph, and let σ be the right-end ordering of H. For every induced subgraph $H(i,j)$ of H, $1 \leq i \leq j \leq n$, and for every stable vertex $u_y \in A(H(i,j))$, Algorithm LP_on_H computes the length $\ell(u_y; i, j)$ of a longest binormal path of $H(i,j)$ which has u_y as its right endpoint and, also, the corresponding path $P(u_y; i, j)$.*

Proof. Let P be a longest binormal path of the stable-connection graph $H(i,j)$, which has a vertex $u_y \in A(H(i,j))$ as its right endpoint. Consider first the case where $C(H(i,j)) = \emptyset$; the graph $H(i,j)$ is consisted of a set of stable vertices $A(H(i,j))$, which is an independent set, due to Observation 4.2. Therefore, in this case Algorithm LP_on_H sets $\ell(u_y; i, j) = 1$ for every vertex $u_y \in A(H(i,j))$, which is indeed the length of the longest binormal path $P(u_y; i, j) = (u_y)$ of $H(i,j)$ which has u_y as its right endpoint. Therefore, the lemma holds for every induced subgraph $H(i,j)$, for which $C(H(i,j)) = \emptyset$.

We examine next the case where $C(H(i,j)) \neq \emptyset$. Let $C(H) = \{c_1, c_2, \ldots, c_k, \ldots, c_t\}$ be the set of connector vertices of H, where $c_1 <_\sigma c_2 <_\sigma \cdots <_\sigma c_k <_\sigma \cdots <_\sigma c_t$. Let $\sigma = (u_1, u_2, \ldots, u_n)$ be the vertex ordering of H constructed in Phase 1. Recall that, by the construction of H, $n = 3t$, and $A(H) = V(H) \setminus C(H)$ is the set of stable vertices of H.

Let $H(i,j)$ be an induced subgraph of H, and let c_k be the rightmost connector vertex of $H(i,j)$ in σ. The proof of the lemma is done by induction on the index k of the rightmost connector vertex c_k of $H(i,j)$. More specifically, given a connector vertex c_k of H, we prove that the lemma holds for every induced subgraph $H(i,j)$ of H, which has c_k as its rightmost connector vertex in σ. To this end, in both the induction basis and the induction step, we distinguish three cases on the position of the stable vertex u_y in the ordering σ: $u_i \leq_\sigma u_y \leq_\sigma u_{f(c_k)}$, $u_{h(c_k)} <_\sigma u_y \leq_\sigma u_j$, and $u_{f(c_k)+1} \leq_\sigma u_y \leq_\sigma u_{h(c_k)}$. In each of these three cases, we examine first the length of a longest binormal path of $H(i,j)$ with u_y as its right endpoint and, then, we compare this value to the length of the path computed by Algorithm LP_on_H. Moreover, we prove that the path computed by Algorithm LP_on_H is a binormal path with u_y as its right endpoint.

We first show that the lemma holds for $k = 1$. In the case where $u_i \leq_\sigma u_y \leq_\sigma u_{f(c_1)}$ or $u_{h(c_1)} <_\sigma u_y \leq_\sigma u_j$, it is easy to see that the length $\ell(u_y; i, j)$ of a longest binormal path P of $H(i, j)$ with u_y as its right endpoint is equal to 1. Indeed, in these cases, if $u_y \neq u_{f(c_1)}$, then u_y does not see the unique connector vertex c_1 of $H(i, j)$ and, thus, the longest binormal path with u_y as its right endpoint is consisted of the vertex u_y. Now, in the case where $u_y = u_{f(c_1)}$, the connector vertex c_1 sees u_y, however, c_1 does not belong to any binormal path with u_y as its right endpoint, since u_y is the leftmost neighbor of c_1 in σ. Therefore, in the case where $u_i \leq_\sigma u_y \leq_\sigma u_{f(c_1)}$ or $u_{h(c_1)} <_\sigma u_y \leq_\sigma u_j$, Algorithm LP_on_$H$ computes the length of the longest binormal path $P(u_y; i, j) = (u_y)$ of $H(i, j)$ with u_y as its right endpoint. In the case where $u_{f(c_1)+1} \leq_\sigma u_y \leq_\sigma u_{h(c_1)}$, Algorithm LP_on_$H$ computes (in the subroutine `process()`) for every stable vertex u_x of $H(i, j)$, such that $u_{f(c_1)} \leq_\sigma u_x \leq_\sigma u_{y-1}$, the value $\ell(u_x; i, j-1) + \ell(u_y; x+1, j-1) + 1 = 1 + 1 + 1 = 3$ and sets $\ell(u_y; i, j) = 3$. It is easy to see that the path $P(u_y; i, j) = (u_x, c_1, u_y)$, computed by Algorithm LP_on_H in this case, is indeed a longest binormal path of $H(i, j)$ with u_y as its right endpoint.

Let now c_k be a connector vertex of H, such that $k \leq t$. Assume that the lemma holds for every induced subgraph $H(i, j)$ of H, which has c_ℓ as its rightmost connector vertex in σ, where $1 \leq \ell \leq k-1$. That is, we assume that for every such graph $H(i, j)$, the value $\ell(u_y; i, j)$ computed by Algorithm LP_on_H is the length of a longest binormal path $P(u_y; i, j)$ of $H(i, j)$ with u_y as its right endpoint. We will show that the lemma holds for every induced subgraph $H(i, j)$ of H, which has c_k as its rightmost connector vertex in σ.

Case 1: $u_i \leq_\sigma u_y \leq_\sigma u_{f(c_k)}$. In this case, it holds $\ell(u_y; i, j) = \ell(u_y; i, h(c_k))$ (note that $u_{h(c_k)}$ is the previous vertex of c_k in σ). Indeed, on the one hand, using similar arguments as in the induction basis, it easily follows that the connector vertex c_k does not belong to any binormal path of $H(i, j)$ with u_y as its right endpoint. On the other hand, since c_k is the rightmost connector vertex of $H(i, j)$, it follows that every vertex u_ℓ of $H(i, j)$, where $c_k <_\sigma u_\ell \leq_\sigma u_j$, is a stable vertex and, thus, u_ℓ does not see u_y, due to Observation 4.2. Therefore, we obtain that $\ell(u_y; i, j) = \ell(u_y; i, h(c_k))$.

Next, we show that this is the result computed by Algorithm LP_on_H in this case. Note first that, since $h(c_k) < j$, Algorithm LP_on_H has already computed the value $\ell(u_y; i, h(c_k))$ at a previous iteration, where j was equal to $h(c_k)$. Additionally, this computed value $\ell(u_y; i, h(c_k))$ equals indeed to the length of a longest binormal path $P(u_y; i, h(c_k))$ of $H(i, h(c_k))$ with u_y as its right endpoint. Indeed, consider first the case where $H(i, h(c_k))$ is a graph for which $C(H(i, h(c_k))) = \emptyset$, i.e., $H(i, h(c_k))$ has only stable vertices. Then, as we have shown in the first paragraph of the proof, the computed value $\ell(u_y; i, h(c_k)) = 1$ equals to the length of a longest binormal path of $H(i, h(c_k))$ with u_y as its right endpoint. Consider now the case where $H(i, h(c_k))$ is a graph for which $C(H(i, h(c_k))) \neq \emptyset$, i.e., $H(i, h(c_k))$ has at least one connector vertex; let c_ℓ be its rightmost connector vertex in σ. Then, $c_\ell <_\sigma c_k$, since $u_{h(c_k)} <_\sigma c_k$. Therefore, by the induction hypothesis, the value $\ell(u_y; i, h(c_k))$ computed by Algorithm LP_on_H equals indeed to the length of a longest binormal path of $H(i, h(c_k))$ with u_y as its right endpoint.

We now show that in Case 1 Algorithm LP_on_H computes $\ell(u_y; i, j) = \ell(u_y; i, h(c_k))$. Consider first the case where u_j is a connector vertex of $H(i, j)$, i.e., $u_j = c_k$. Then, Algorithm LP_on_H computes $\ell(u_y; i, j) = \ell(u_y; i, j-1)$, which equals to $\ell(u_y; i, h(c_k))$, since in this case $j - 1 = h(c_k)$. Consider now the case where u_j is a stable vertex; then $j - 1 > h(c_k)$. If $j - 1 = h(c_k) + 1$, then Algorithm LP_on_H computes $\ell(u_y; i, j) = \ell(u_y; i, j-1)$, which is equal to $\ell(u_y; i, h(c_k) + 1)$; moreover, since $u_{h(c_k)+1} = c_k$ is a connector vertex, it follows that $\ell(u_y; i, h(c_k)+1) = \ell(u_y; i, h(c_k))$ and, thus, $\ell(u_y; i, j) = \ell(u_y; i, h(c_k))$. Similarly, if $j - 1 > h(c_k) + 1$, then Algorithm LP_on_H computes $\ell(u_y; i, j) = \ell(u_y; i, j-1)$, which is again equal to $\ell(u_y; i, h(c_k))$. Therefore, in Case 1, where $u_i \leq_\sigma u_y \leq_\sigma u_{f(c_k)}$, Algorithm LP_on_$H$ computes $\ell(u_y; i, h(c_k))$ as the length of a longest binormal path of $H(i, j)$ with u_y as its right endpoint and, also, computes $P(u_y; i, j) = P(u_y; i, h(c_k))$. Then, by the induction hypothesis, this path is also binormal. Thus, in Case 1 the lemma holds.

Case 2: $u_{h(c_k)} <_\sigma u_y \leq_\sigma u_j$. Since c_k is the rightmost connector vertex of $H(i, j)$, and since u_y is a stable vertex, it follows that u_y does not see any vertex of $H(i, j)$. Thus, the longest binormal path of

$H(i,j)$ with u_y as its right endpoint is consisted of the vertex u_y, i.e., $\ell(u_y;i,j)=1$. One can easily see that in this case Algorithm LP_on_H computes the length $\ell(u_y;i,j)=1$, and the path $P(u_y;i,j)=(u_y)$, which is clearly a binormal path. Thus, in Case 2 the lemma holds.

Case 3: $u_{f(c_k)+1} \leq_\sigma u_y \leq_\sigma u_{h(c_k)}$. In this case, the connector vertex c_k sees u_y. Let $P=(u_{x'},\ldots,u_x,c_k,u_{y'},\ldots,u_y)$ be a longest binormal path of $H(i,j)$ with u_y as its right endpoint, which contains the connector vertex c_k; due to Lemma 4.7, such a path always exists. Let u_x be the previous vertex of c_k in the path P; thus, $u_{f(c_k)} \leq_\sigma u_x <_\sigma u_y$. Since P is a binormal path, the vertices $u_{x'}$, u_x, $u_{y'}$, and u_y are all stable vertices. Also, since c_k sees u_y, which is the rightmost stable vertex of P in σ, all stable vertices of P belong to the graph $H(i,h(c_k))$. Additionally, since c_k is the rightmost connector vertex of $H(i,j)$ in σ, all connector vertices of P belong to the graph $H(i,h(c_k)+1)$. Therefore, all vertices of P belong to the graph $H(i,h(c_k)+1)$. Thus, the path P is a longest binormal path of $H(i,h(c_k)+1)$ with u_y as its right endpoint, which contains the connector vertex c_k. Therefore, for every graph $H(i,j)$, for which c_k is its rightmost connector vertex in σ and $h(c_k)+1 \leq j$, we have that $\ell(u_y;i,j)=\ell(u_y;i,h(c_k)+1)$. Thus, we will examine only the case where $h(c_k)+1=j$, that is, c_k is the rightmost vertex u_j of $H(i,j)$ in σ.

Next, we examine the length $\ell(u_y;i,j)$ of a longest binormal path of $H(i,j)$ with u_y as its right endpoint, in the case where $h(c_k)+1=j$. Consider removing the connector vertex c_k from the path P. Then, we obtain the paths $P_1=(u_{x'},\ldots,u_x)$ and $P_2=(u_{y'},\ldots,u_y)$. Since P is a binormal path of $H(i,j)$, from Lemma 4.8 we obtain that P_1 and P_2 are binormal paths of $H(i,j)$. Since, as we have shown, all vertices of P belong to $H(i,h(c_k)+1)$, and since $c_k=u_j$ is the rightmost vertex of $H(i,j)$ in σ, it follows that all vertices of P_1 and P_2 belong to the graph $H(i,h(c_k))=H(i,j-1)$. Since P is a binormal path, from Lemma 4.6(a) it follows that for every stable vertex $u_{\ell_1} \in V(P_1)$, we have $u_i \leq_\sigma u_{x'} \leq_\sigma u_{\ell_1} \leq_\sigma u_x$. Additionally, for every stable vertex $u_{\ell_2} \in V(P_2)$, we have $u_x <_\sigma u_{\ell_2} \leq_\sigma u_y \leq_\sigma u_{j-1}$, where $u_{j-1}=u_{h(c_k)}$ is the rightmost vertex of $H(i,j-1)$ in σ, since $u_j=c_k$. Therefore, for every stable vertex $u_{\ell_1} \in V(P_1)$ it holds $u_{\ell_1} \in A(H(i,x))$, and for every stable vertex $u_{\ell_2} \in V(P_2)$ it holds $u_{\ell_2} \in A(H(x+1,j-1))$.

Similarly, since P_1 is a binormal path, u_x is the rightmost stable vertex of $V(P_1)$ in σ, due to Lemma 4.6(a). Moreover, since P_1 is binormal, every connector vertex $c_{\ell_1} \in V(P_1)$ sees at least two stable vertices of P_1 and, thus, $u_i \leq_\sigma u_{f(c_{\ell_1})} <_\sigma u_x$. Therefore, for every connector vertex $c_{\ell_1} \in V(P_1)$, we have that $c_{\ell_1} \in C(H(i,j-1)) \setminus \{c_\ell \in C(H(i,j-1)) : u_x \leq_\sigma u_{f(c_\ell)}\} \subseteq C(H(i,j-1)) \setminus C(H(x+1,j-1))$. Additionally, from Lemma 4.6(b) we have that every connector vertex $c_{\ell_2} \in V(P_2)$ does not see the vertex u_x, i.e., $u_x <_\sigma u_{f(c_{\ell_2})} <_\sigma c_{\ell_2} \leq_\sigma u_{j-1}$; thus, $c_{\ell_2} \in C(H(x+1,j-1))$. Summarizing, let H_1 and H_2 be the induced subgraphs of $H(i,j-1)$, with vertex sets $V(H_1)=A(H(i,x)) \cup C(H(i,j-1)) \setminus C(H(x+1,j-1))$ and $V(H_2)=A(H(x+1,j-1)) \cup C(H(x+1,j-1))$, respectively. Note that the graphs H_1 and H_2 are defined with respect to a stable vertex u_x, where $u_{f(c_k)} \leq_\sigma u_x <_\sigma u_{j-1}$, and that $H_2=H(x+1,j-1)$. Now, it is easy to see that $V(H_1) \cap V(H_2)=\emptyset$. Moreover, P_1 and P_2 belong to the graphs H_1 and H_2, respectively and, therefore, $V(P_1) \cap V(P_2)=\emptyset$.

Since $P=(P_1,c_k,P_2)$ is a longest binormal path of $H(i,j)$ with u_y as its right endpoint, and since the paths P_1 and P_2 belong to two disjoint induced subgraphs of $H(i,j)$, it follows that P_1 is a longest binormal path of H_1 with u_x as its right endpoint, and that P_2 is a longest binormal path of H_2 with u_y as its right endpoint. Thus, since $H_2=H(x+1,j-1)$, we obtain that $|P_2|=\ell(u_y;x+1,j-1)$. We will now show that $|P_1|=\ell(u_x;i,j-1)$. To this end, consider a longest binormal path P_0 of $H(i,j-1)$ with u_x as its right endpoint. Due to Lemma 4.6(a), u_x is the rightmost stable vertex of P_0 in σ and, thus, all stable vertices of P_0 belong to $A(H_1)=A(H(i,x))$. Furthermore, since P_0 is binormal, every connector vertex c_ℓ of P_0 sees at least two stable vertices of P_0 and, thus, $u_{f(c_\ell)} <_\sigma u_x$, i.e., $c_\ell \in C(H_1)=C(H(i,j-1)) \setminus C(H(x+1,j-1))$. It follows that $V(P_0) \subseteq V(H_1)$ and, thus, $|P_0| \leq |P_1|$. On the other hand, $|P_1| \leq |P_0|$, since H_1 is an induced subgraph of $H(i,j-1)$. Thus, $|P_1|=|P_0|=\ell(u_x;i,j-1)$. Therefore, for the length $|P|=\ell(u_y;i,j)$ of a longest binormal path P of $H(i,j)$ with u_y as its right endpoint, it follows that $\ell(u_y;i,j)=\ell(u_x;i,j-1)+\ell(u_y;x+1,j-1)+1$.

Hereafter, we examine the results computed by Algorithm LP_on_H in Case 3. Let P' be the path of the graph $H(i,j)$ with u_y as its right endpoint computed by Algorithm LP_on_H, in the case where $u_{f(c_k)+1} \leq_\sigma u_y \leq_\sigma u_{h(c_k)}$. Consider first the case where u_j is a connector vertex of $H(i,j)$, i.e., $u_j = c_k$. It is easy to see that the path P' constructed by Algorithm LP_on_H (in the subroutine `process()`) contains the connector vertex c_k. Algorithm LP_on_H computes the length of the path $P' = (P'_1, c_k, P'_2)$, for two paths P'_1 and P'_2 as follows. The path $P'_1 = P(u_x; i, j-1)$ is a path of $H(i, j-1)$ with u_x as its right endpoint, where u_x is a neighbor of c_k, such that $u_{f(c_k)} \leq_\sigma u_x <_\sigma u_y$. The path $P'_2 = P(u_y; x+1, j-1)$ is a path of $H(x+1, j-1)$ with u_y as its right endpoint, where $u_{f(c_k)+1} \leq_\sigma u_y \leq_\sigma u_{h(c_k)}$. Actually, in this case, Algorithm LP_on_H computes (in the subroutine `process()`) the value $w_1 + w_2 + 1 = |P'_1| + |P'_2| + 1$, for every stable vertex u_x, where $u_{f(c_k)} \leq_\sigma u_x <_\sigma u_y$, and sets $|P'|$ to be equal to the maximum among these values. Additionally, Algorithm LP_on_H computes the corresponding path $P' = (P'_1, c_k, P'_2)$.

Note that the path $P'_1 = P(u_x; i, j-1)$ (resp. $P'_2 = P(u_y; x+1, j-1)$) has already been computed by Algorithm LP_on_H at a previous iteration. Additionally, the computed path $P(u_x; i, j-1)$ (resp. $P(u_y; x+1, j-1)$) is indeed a longest binormal path of $H(i, j-1)$ (resp. of $H(x+1, j-1)$) with u_x (resp. with u_y) as its right endpoint. Indeed, consider first the case where $H(i, j-1)$ (resp. $H(x+1, j-1)$) is a graph for which $C(H(i, j-1)) = \emptyset$ (resp. $C(H(x+1, j-1)) = \emptyset$), i.e., $H(i, j-1)$ (resp. $H(x+1, j-1)$) has only stable vertices. Then, as we have shown in the first paragraph of the proof, the computed path $P(u_x; i, j-1)$ (resp. $P(u_y; x+1, j-1)$) is a longest binormal path of $H(i, j-1)$ (resp. of $H(x+1, j-1)$) with u_x (resp. with u_y) as its right endpoint. Consider now the case where $H(i, j-1)$ (resp. $H(x+1, j-1)$) is a graph for which $C(H(i, j-1)) \neq \emptyset$ (resp. $C(H(x+1, j-1)) \neq \emptyset$), i.e., $H(i, j-1)$ (resp. $H(x+1, j-1)$) has at least one connector vertex; let c_ℓ be its rightmost connector vertex in σ. Then, $c_\ell <_\sigma c_k$, since $u_{j-1} <_\sigma u_j = c_k$. Therefore, by the induction hypothesis, the path $P(u_x; i, j-1)$ (resp. $P(u_y; x+1, j-1)$) computed by Algorithm LP_on_H is indeed a longest binormal path of $H(i, j-1)$ (resp. of $H(x+1, j-1)$) with u_x (resp. with u_y) as its right endpoint.

Since by the induction hypothesis, P'_1 and P'_2 are binormal paths of $H(i, j-1)$ with u_x and u_y as their right endpoints, respectively, it follows similarly to the above that P'_1 and P'_2 belong to the graphs H_1 and H_2, respectively. Recall that, the graphs H_1 and H_2 are defined with respect to a stable vertex u_x, where $u_{f(c_k)} \leq_\sigma u_x <_\sigma u_{j-1}$. Since, as we have shown, $V(H_1) \cap V(H_2) = \emptyset$, it follows that $V(P'_1) \cap V(P'_2) = \emptyset$. Therefore, from Lemma 4.9 we obtain that the computed path $P' = (P'_1, u_j, P'_2)$ is a binormal path as well. Moreover, Algorithm LP_on_H computes (in the subroutine `process()`) for every stable vertex u_x, where $u_{f(c_k)} \leq_\sigma u_x <_\sigma u_y$, the value $\ell(u_x; i, j-1) + \ell(u_y; x+1, j-1) + 1$, and sets $|P'|$ to be equal to the maximum among these values. Thus, the computed path P' is a longest binormal path of $H(i,j)$ with u_y as its right endpoint.

Consider now the case where u_j is a stable vertex of $H(i,j)$. Let c_k be the rightmost connector vertex of $H(i,j)$ in σ; then $h(c_k) + 1 < j$. Assume first that $h(c_k) + 1 = j - 1$. Since u_j is a stable vertex and also the rightmost vertex of $H(i,j)$, u_j does not see any vertex of $H(i, h(c_k)+1)$. In this case, Algorithm LP_on_H correctly computes the path $P' = P(u_y; i, j-1) = P(u_y; i, h(c_k)+1)$, with length $|P'| = \ell(u_y; i, h(c_k)+1)$. Similarly, in the case where $h(c_k)+1 < j-1$, Algorithm LP_on_H again computes the path $P' = P(u_y; i, j-1) = P(u_y; i, h(c_k)+1)$, with length $|P'| = \ell(u_y; i, j-1) = \ell(u_y; i, h(c_k)+1)$. Algorithm LP_on_$H$ has already computed the value $\ell(u_y; i, h(c_k)+1)$ at a previous iteration where j was equal to $h(c_k)+1$ (i.e., $u_j = c_k$) and, also, the computed path $P' = P(u_y; i, h(c_k)+1)$ is binormal.

Concluding, in both cases where u_j is a connector or a stable vertex of $H(i,j)$, the path P' of $H(i,j)$ with u_y as its right endpoint computed by Algorithm LP_on_H is a longest binormal path $P(u_y; i, j)$ of $H(i,j)$ with u_y as its right endpoint, and $|P'| = \ell(u_y; i, j)$. Thus, the lemma holds in Case 3 as well. ∎

Due to Lemma 4.10, and since the output of Algorithm LP_on_H is the maximum among the lengths $\ell(u_y; 1, n)$, $u_y \in A(H(1,n))$, along with the corresponding path, it follows that Algorithm LP_on_H computes a longest binormal path of $H(1,n)$ with right endpoint a vertex $u_y \in A(H(1,n))$. Thus, since $H(1,n) = H$, we obtain the following result.

Lemma 4.11. *Let G be an interval graph. Algorithm LP_on_H computes a longest binormal path of the stable-connection graph H of the graph G.*

4.4.2 Correctness of Algorithm LP_Interval

We next show that Algorithm LP_Interval correctly computes a longest path of an interval graph G. The correctness proof is based on the following property: for any longest path P of G there exists a longest binormal path P' of H, such that $|P'| = 2|P| + 1$ and vice versa (this property is proved in Lemma 4.12). Therefore, we obtain that the length of a longest binormal path P of H computed by Algorithm LP_on_H, is equal to $2k + 1$, where k is the length of a longest path $\widehat{P}$ of G. Next, we show that the length of a longest binormal path of H equals to the length of a longest path of H. Finally, we show that the path $\widehat{P}$ computed at Step 3 of Algorithm LP_Interval is indeed a longest path of the interval graph G.

Lemma 4.12. *Let H be the stable-connection graph of an interval graph G. Then, for any longest path P of G there exists a longest binormal path P' of H, such that $|P'| = 2|P| + 1$ and vice versa.*

Proof. Let σ be the right-end ordering of H, constructed in Phase 1.

($\Longrightarrow$) Let $P = (v_1, v_2, \ldots, v_k)$ be a longest path of G, i.e., $|P| = k$. We will show that there exists a binormal path P' of H such that $|P'| = 2k + 1$. Since G is an induced subgraph of H, the path P of G is a path of H as well. We construct a path $\widehat{P}$ of H from P, by adding to P the appropriate stable vertices, using the following procedure. Initially, set $\widehat{P} = P$ and for every subpath (v_i, v_{i+1}) of the path $\widehat{P}$, $1 \leq i \leq k-1$, do the following: consider first the case where $v_i <_\sigma v_{i+1}$; then, by the construction of H, v_{i+1} is adjacent to both stable vertices $a_{i,1}$ and $a_{i,2}$ associated with the connector vertex v_i. If $a_{i,1}$ has not already been added to $\widehat{P}$, then replace the subpath (v_i, v_{i+1}) by the path $(v_i, a_{i,1}, v_{i+1})$; otherwise, replace the subpath (v_i, v_{i+1}) by the path $(v_i, a_{i,2}, v_{i+1})$. Similarly, in the case where $v_{i+1} <_\sigma v_i$, replace the subpath (v_i, v_{i+1}) by the path $(v_i, a_{i+1,1}, v_{i+1})$ or $(v_i, a_{i+1,2}, v_{i+1})$, respectively. Finally, consider the endpoint v_1 (resp. v_k) of $\widehat{P}$. If $a_{1,1}$ (resp. $a_{k,1}$) has not already been added to $\widehat{P}$, then add $a_{1,1}$ (resp. $a_{k,1}$) as the first (resp. last) vertex of $\widehat{P}$; otherwise, add $a_{1,2}$ (resp. $a_{k,2}$) as the first (resp. last) vertex of $\widehat{P}$.

By the construction of $\widehat{P}$ it is easy to see that for every connector vertex v of P we add two stable vertices as neighbors of v in $\widehat{P}$, and since in H there are exactly two stable vertices associated with every connector vertex v, it follows that every stable vertex of H appears at most once in $\widehat{P}$. Furthermore, since we add in total $k + 1$ stable vertices to P, where $|P| = k$, it follows that $|\widehat{P}| = 2k + 1$. Denote now by P' a normal path of H such that $V(P') = V(\widehat{P})$. Such a path exists, due to Lemma 4.4. Due to the above construction, the path $\widehat{P}$ is consisted of $k + 1$ stable vertices and k connector vertices. Thus, since no two stable vertices are adjacent in H due to Observation 4.2, and since P' is a normal path of H, it follows that P' is a binormal path of H. Thus, for any longest path P of G there exists a binormal path P' of H, such that $|P'| = 2|P| + 1$.

($\Longleftarrow$) Consider now a longest binormal path $P' = (v_1, v_2, \ldots, v_\ell)$ of H. Since P' is binormal, it follows that $\ell = 2k + 1$, and that P' has k connector vertices and $k + 1$ stable vertices, for some $k \geq 1$. We construct a path P by deleting all stable vertices from the path P' of H. By the construction of H, all neighbors of a stable vertex a are connector vertices and form a clique in G; thus, for every subpath (v, a, v') of P', v is adjacent to v' in G. It follows that P is a path of G. Since we removed all the $k + 1$ stable vertices of P', it follows that $|P| = k$, i.e., $|P'| = 2|P| + 1$.

Summarizing, we have constructed a binormal path P' of H from a longest path P of G such that $|P'| = 2|P| + 1$, and a path P of G from a longest binormal path P' of H such that $|P'| = 2|P| + 1$. This completes the proof. ∎

In the next lemma, we show that the length of a longest path of H is equal to the length of a longest binormal path of H.

Lemma 4.13. *For any longest path P and any longest binormal path P' of H, it holds $|P'| = |P|$.*

Proof. Let P be a longest path of H, and let P' be a longest binormal path of H, i.e., a binormal path of H with maximum length. Then, clearly $|P'| \leq |P|$. Suppose that P has k connector and ℓ stable vertices. Since no two stable vertices of H are adjacent due to Observation 4.2, it holds clearly that $\ell \leq k+1$. Similarly to the second part of the proof of Lemma 4.12, we can obtain a path $\widehat{P}$ of H with k vertices, by removing all ℓ stable vertices from P. Then, similarly to the first part of the proof of Lemma 4.12, there exists a binormal path P'' of H, where $|P''| = 2k+1 \geq k+\ell = |P| \geq |P'|$. However, $|P''| \leq |P'|$, since P' be a longest binormal path of H. Therefore, $|P'| = |P|$. This completes the proof. ∎

Let P be the longest binormal path of H computed in Step 2 of Algorithm LP_Interval, using Algorithm LP_on_H. Then, in Step 3 Algorithm LP_Interval computes the path $\widehat{P}$ by deleting all stable vertices from P. By the construction of H, all neighbors of a stable vertex a are connector vertices and form a clique in G; thus, for every subpath (v, a, v') of P, v is adjacent to v' in G. It follows that $\widehat{P}$ is a path of G. Moreover, since P is binormal, it has k connector vertices and $k+1$ stable vertices, i.e., $|P| = 2k+1$, where $k \geq 1$. Thus, since we have removed all $k+1$ stable vertices of P, it follows that $|\widehat{P}| = k$ and, thus, $\widehat{P}$ is a longest path of G due to Lemma 4.12. Therefore, we have proved the following result.

Theorem 4.1. *Algorithm LP_Interval computes a longest path of an interval graph G.*

4.4.3 Time Complexity

Let G be an interval graph on $|V(G)| = n$ vertices and $|E(G)| = m$ edges. It has been shown that we can obtain the right-end ordering π of G, which results from numbering the intervals after sorting them on their right ends, in $O(n+m)$ time [1, 61].

First, we show that Step 1 of Algorithm LP_Interval, which constructs the stable-connection graph H of the graph G, takes $O(n^2)$ time. Indeed, for every connector vertex u_i, $1 \leq i \leq n$, we can add two stable vertices in $V(H)$ in $O(1)$ time and we can compute the specific neighborhood of u_i in $O(n)$ time.

Step 2 of Algorithm LP_Interval includes the execution of Algorithm LP_on_H. The subroutine `process()` takes $O(n^2)$ time, due to the $O(n^2)$ pairs of the neighbors u_x and u_y of the connector vertex u_j in the graph $H(i,j)$. Additionally, the subroutine `process()` is executed at most once for each subgraph $H(i,j)$ of H, $1 \leq i \leq j \leq n$, i.e., it is executed $O(n^2)$ times. Thus, Algorithm LP_on_H takes $O(n^4)$ time.

Step 3 of Algorithm LP_Interval can be executed in $O(n)$ time since we simply traverse the vertices of the path P, constructed by Algorithm LP_on_H, and delete every stable vertex.

Therefore, we obtain the following result concerning the time complexity of the algorithm.

Theorem 4.2. *A longest path of an interval graph can be computed in $O(n^4)$ time.*

In order to compute the length of a longest path, we need to store one value for every induced subgraph $H(i,j)$ and for every stable vertex u_y of $H(i,j)$. Thus, since there are in total $O(n^2)$ such subgraphs $H(i,j)$, $1 \leq i \leq j \leq n$, and since each one has at most $O(n)$ stable vertices, we can compute the length of a longest path in $O(n^3)$ space. Furthermore, in order to compute and report a longest path, instead of its length only, we have to store a path of at most n vertices for every one of the $O(n^3)$ computed values. Therefore, the space complexity of Algorithm LP_Interval is $O(n^4)$.

4.5 Concluding Remarks

In this work we presented a polynomial-time algorithm for solving the longest path problem on interval graphs, which runs in $O(n^4)$ time and, thus, provided a solution to the open problem stated by Uehara and Uno in [63] asking for the complexity status of the longest path problem on interval graphs. It

would be interesting to see whether the ideas presented in this work can be applied to find a polynomial solution to the longest path problem on convex and biconvex graphs, the complexities of which still remain open [63].

Chapter 5

The Longest Path Problem on Cocomparability Graphs

5.1 Introduction

The longest path problem, i.e., the problem of finding a path of maximum length in a graph, is a generalization of the Hamiltonian path problem. The Hamiltonian path problem is the problem of determining whether a graph is Hamiltonian; a graph is said to be Hamiltonian if it contains a simple path in which every vertex of the graph appears exactly once. The longest path problem or, equivalently, the problem of finding a maximum Hamiltonian induced subgraph of a graph, is NP-complete on general graphs and, in fact, on every class of graphs that the Hamiltonian path problem is NP-complete. Thus, it is interesting to study the longest path problem on classes of graphs where the Hamiltonian path problem is polynomial, since even if a graph is not Hamiltonian, it makes sense in several applications to search for a longest path of the graph. Although the Hamiltonian path problem has received a great deal of attention the past two decades, only recently did the longest path problem start receiving attention.

Additionally, the longest path problem has also received attention the recent years in the direction of approximation related results, some of which imply that finding a longest path seems to be more difficult than deciding whether or not a graph admits a Hamiltonian path. Indeed, it has been proved that even if a graph has a Hamiltonian path, the problem of finding a path of length $n - n^{\varepsilon}$ for any $\varepsilon < 1$ is NP-hard, where n is the number of vertices of the graph [47]. Moreover, there is no polynomial-time constant-factor approximation algorithm for the longest path problem unless P=NP [47]. For related results see also [29, 31, 32, 66, 68].

As we have mentioned, the longest path problem is NP-hard on every class of graphs on which the Hamiltonian path problem is NP-complete. The Hamiltonian path problem is known to be NP-complete in general graphs [33, 34], and remains NP-complete even when restricted to some small classes of graphs

such as split graphs [37], chordal bipartite graphs, split strongly chordal graphs [58], circle graphs [22], planar graphs [34], and grid graphs [46]. However, it makes sense to investigate the tractability of the longest path problem on the classes of graphs for which the Hamiltonian path problem admits polynomial time solutions. Such classes include interval graphs [1], circular-arc graphs [24], biconvex graphs [3], and cocomparability graphs [23, 39]. Note that the problem of finding a longest path on proper interval graphs is easy, since all connected proper interval graphs have a Hamiltonian path which can be computed in linear time [6]; on the contrary, not all interval graphs are Hamiltonian.

In contrast to the Hamiltonian path problem, the known polynomial time solutions for the longest path problem are rather recent, and restrict to smaller graph classes. Specifically, a linear time algorithm for finding a longest path in a tree was proposed by Dijkstra around 1960, a formal proof of which can be found in [12]. Later, through a generalization of Dijkstra's algorithm for trees, Uehara and Uno [63] solved the longest path problem for weighted trees and block graphs in linear time and space, and for cacti in $O(n^2)$ time and space, where n and m denote the number of vertices and edges of the input graph, respectively. More recently, polynomial algorithms have been proposed that solve the longest path problem on bipartite permutation graphs in $O(n)$ time and space [64], and on ptolemaic graphs in $O(n^5)$ time and $O(n^2)$ space [65]. Furthermore, Uehara and Uno in [63] solved the longest path problem on a subclass of interval graphs in $O(n^3(m + n \log n))$ time, and as a corollary they showed that a longest path on threshold graphs can be found in $O(n + m)$ time and space. In Chapter 4 we presented a polynomial solution of the longest path problem on interval graphs, answering thus the question left open in [63].

In this chapter we present a polynomial solution for the longest path problem on cocomparability graphs. Cocomparability graphs form an important and well-known class of perfect graphs [37], which is a superclass of interval graphs and permutation graphs. As interval and permutation graphs have linear structures, so do cocomparability graphs: a graph G is a cocomparability graph if and only if its vertices can be put in an order $v_1, v_2, \ldots, v_{|V(G)|}$ such that if $i < k < j$ and $v_i v_j \in E(G)$, then $v_j v_k \in E(G)$ or $v_i v_k \in E(G)$ [56]. The Hamiltonian path problem on cocomparability graphs has been proved to be polynomial [23], while the status of the longest path problem on such graphs is unknown; actually, the status of the longest path problem is unknown even on the more special class of permutation graphs. In this work, we present a polynomial-time algorithm for solving the longest path problem on the class of cocomparability graphs. Therefore, we resolve the open question for the status of the problem on cocomparability graphs, and thus on permutation graphs. This result extends our polynomial solution of the longest path problem on interval graphs presented in Chapter 4.

The rest of this chapter is organized as follows. In Section 5.2, we first review some properties of partial orders, comparability and cocomparability graphs and, then, introduce the notion of a normal antipath on a cocomparability graph, which is needed for our algorithm. In Section 5.3, we present our algorithm for solving the longest path problem on a cocomparability graph, and in Section 5.4 we prove the correctness and compute the time complexity of our algorithm. Finally, some concluding remarks are given in Section 5.5.

5.2 Theoretical Framework

For basic definitions in graph theory refer to [10, 37, 56]. Recall that by $V(P)$ we denote the set of vertices in a path (resp. antipath) P, and within this chapter we consider the *length* of the path (resp. antipath) P to be the number of vertices in P, i.e., $|P| = |V(P)|$.

5.2.1 Partial Orders and Cocomparability Graphs

A *partial order* will be denoted by $P = (V, <_P)$, where V is the finite ground set of elements or vertices and $<_P$ is an irreflexive, antisymmetric, and transitive binary relation on V. Two elements $a, b \in V$ are comparable in P (denoted by $a \sim_P b$) if $a <_P b$ or $b <_P a$. Otherwise, they are said to be incomparable

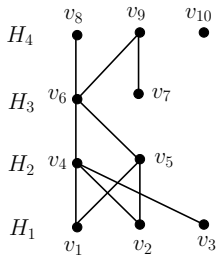

Figure 5.1: Illustrating a Hasse diagram of a cocomparability graph G, with layers H_1, H_2, H_3, H_4. Note that $\sigma = (v_1, v_2, \ldots, v_{10})$ is a layered ordering for G.

(denoted by $a \parallel b$). An *extension* of a partial order $P = (V, <_P)$ is a partial order $L = (V, <_L)$ on the same ground set that *extends* P, i.e., $a <_P b \Rightarrow a <_L b$, for all $a, b \in V$. The *dual partial order* P^d of $P = (V, <_P)$ is a partial order $P^d = (V, <_{P^d})$ such that for any two elements $a, b \in V$, $a <_{P^d} b$ if and only if $b <_P a$. A *linear order* is a partial order without incomparable elements. A *linear extension* of a partial order $P = (V, <_P)$ is a linear order $L = (V, <_L)$ on the same ground set that extends P.

The graph G, edges of which are exactly the comparable pairs of a partial order P on $V(G)$, is called the *comparability graph* of P, and is denoted by $G(P)$. The complement graph $\overline{G}$, whose edges are the incomparable pairs of P, is called the *cocomparability graph* of P, and is denoted by $\overline{G}(P)$. Alternatively, a graph G is a cocomparability graph if its complement graph $\overline{G}$ has a transitive orientation, corresponding to the comparability relations of a partial order $P_{\overline{G}}$. Note that a partial order P uniquely determines its comparability graph $G(P)$ and its cocomparability graph $\overline{G}(P)$, but the reverse is not true, i.e., a cocomparability graph G has as many partial orders $P_{\overline{G}}$ as is the number of the transitive orientations of $\overline{G}$. Furthermore, the class of cocomparability graphs is *hereditary*, i.e., every induced subgraph of a cocomparability graph G is also a cocomparability graph.

Let G be a comparability graph, and let P_G be a partial order which corresponds to G. The graph G can be represented by a directed covering graph with layers $H_1, H_2, \ldots, H_h$, in which each vertex is on the highest possible layer. That is, the maximal vertices of the partial order P_G are on the highest layer H_h, and for every vertex v on layer H_{i-1} there exists a vertex u on layer H_i such that $v <_{P_G} u$; such a layered representation of G (respectively P_G) is a called the *Hasse diagram* of G (respectively P_G). Let $\sigma = (V(G), <_\sigma)$ be a partial order on the vertices of a comparability graph G, such that for any two vertices $v, u \in V(G)$, $v <_\sigma u$ if and only if $v \in H_i$, $u \in H_j$, and $i < j$; we may, equivalently, denote $v <_\sigma u$ by $u >_\sigma v$. For vertices $v, u \in V(G)$ which belong to the same layer H_i of the Hasse diagram of G, for simplicity sometimes we shall write $v =_\sigma u$; $v \neq_\sigma u$ denotes that vertices $v, u \in V(G)$ belong to different layers. Also, $v \leq_\sigma u$ implies that either $v <_\sigma u$ or $v =_\sigma u$; again we may, equivalently, denote $v \leq_\sigma u$ by $u \geq_\sigma v$. Throughout the chapter, such an ordering σ is called a *layered ordering* of G. Note that, the partial order σ is an extension of the partial order P_G; in particular, it holds that for any two vertices $u, v \in V(G)$, $v <_{P_G} u$ if and only if $v <_\sigma u$ and $vu \in E(G)$.

Since a comparability graph G does not uniquely determine a partial order, hereafter, for clarity, we will consider a comparability graph G represented by its Hasse diagram and we will denote by P_G the partial order $(V(G), <_{P_G})$ to which the Hasse diagram of G corresponds, i.e., the vertices which are on the highest layer H_h of the Hasse diagram are the maximal vertices of the partial order P_G, and for two vertices $u, v \in V(G)$, $v <_{P_G} u$ if $v \in H_{i-1}$, $u \in H_i$ and $uv \in E(G)$. Thus, we will say that P_G is the partial order which *corresponds* to the comparability graph G. Also note that the *transitivity property* holds for vertices in the Hasse diagram; for any three vertices $v, u, w \in V(G)$ such that $v \in H_i$, $u \in H_j$, $w \in H_k$, and $i < j < k$ (or, equivalently, $v <_\sigma u <_\sigma w$), if $vu \in E(G)$ and $uw \in E(G)$, then $vw \in E(G)$.

The following definition and results where given by Damaschke *et al.* in [23] for providing an alternative proof for their algorithm for finding a Hamiltonian path of a cocomparability graph; note that the algorithm and the original correctness proof were first presented in [39], in order to provide a polynomial

solution for the bump number problem of a partial order.

Definition 5.1. (Damaschke *et al.* [23]): Let G be a comparability graph, and let P_G be the partial order which corresponds to G. A path $P = (v_1, v_2, \ldots, v_k)$ of the cocomparability graph $\overline{G}$ is monotone if $v_i <_{P_G} v_j$ implies $i < j$, i.e., v_i appears before v_j in the path P.

The following results appear to be useful in the sequence.

Lemma 5.1. *(Damaschke et al. [23]): Let G be a comparability graph, and let P_G be the partial order which corresponds to G. Let $P = (v_1, v_2, \ldots, v_k)$ be a Hamiltonian path of the cocomparability graph $\overline{G}$ such that v_1 is a minimal element of P_G. Then there exists a monotone Hamiltonian path P' of $\overline{G}$ starting with vertex v_1.*

Theorem 5.1. *(Damaschke et al. [23]): Let G be a cocomparability graph. Then, G has a Hamiltonian path if and only if G has a monotone Hamiltonian path.*

Note that the above two results were proved in [23] for Hamiltonian paths of a cocomparability graph G. In fact, it appears that the two results hold not only for Hamiltonian paths of G, but also for any path of G. Indeed, let P be a path of the cocomparability graph G, and let $G' = G[V(P)]$ be the subgraph of G induced by the vertices of P. Also, let $P_{G'}$ be the partial order which corresponds to G', such that P_G is an extension of $P_{G'}$, i.e., for any two vertices $u, v \in V(G)$, if $u <_{P_G} v$ and $u, v \in V(G')$, then $u <_{P_{G'}} v$. Then since P is a Hamiltonian path of G', then from Lemma 5.1 and Theorem 5.1, there exists a monotone path P' of G' (with respect to $P_{G'}$) such that $V(P') = V(P)$. From Definition 5.1 it is easy to see that P' is also a monotone path of G (with respect to P_G), since P_G is an extension of $P_{G'}$.

Additionally, since a path P of a cocomparability graph $\overline{G}$ is an antipath on the comparability graph G, and since our algorithm for computing a longest path of a cocomparability graph $\overline{G}$ computes, in fact, a longest antipath of the comparability graph G, we restate the above definition and results and whenever P denotes a path of a cocomparability graph $\overline{G}$, we refer to P as an antipath of the comparability graph G.

We first restate Definition 5.1 as follows: an antipath $P = (v_1, v_2, \ldots, v_k)$ of a comparability graph G is monotone if $v_i <_{P_G} v_j$ implies $i < j$, where P_G is the partial order which corresponds to G. We next restate Lemma 5.1 and Theorem 5.1 in a form stronger than the one stated in [23], to assist us obtain some important for the correctness of our algorithm results, in the sequence.

Lemma 5.2. *Let G be a comparability graph, and let P_G be the partial order which corresponds to G. Let $P = (v_1, v_2, \ldots, v_k)$ be an antipath of G such that v_1 is a minimal element of $V(P)$ in P_G. Then there exists a monotone antipath P' of G starting with vertex v_1 such that $V(P') = V(P)$.*

Theorem 5.2. *Let G be a comparability graph. If P is an antipath of G, then there exists a monotone antipath P' of G such that $V(P') = V(P)$.*

The following lemma derives from properties of comparability graphs, and appears to be useful in obtaining some important results.

Lemma 5.3. *Let G be a comparability graph, and let σ be the layered ordering of G. Let $P = (v_1, v_2, \ldots, v_k)$ be an antipath of G, and let $v_\ell \notin V(P)$ be a vertex of G such that $v_1 \leq_\sigma v_\ell <_\sigma v_k$ and $v_\ell v_k \in E(G)$. Then there exist two consecutive vertices v_{i-1} and v_i in P, $2 \leq i \leq k$, such that $v_{i-1} v_\ell \notin E(G)$ and $v_\ell <_\sigma v_i$.*

Proof. Let $P = (v_1, v_2, \ldots, v_k)$ be an antipath of G, and let $v_\ell \notin V(P)$ be a vertex of G such that $v_1 \leq_\sigma v_\ell <_\sigma v_k$ and $v_\ell v_k \in E(G)$. We first show that at least one vertex of P does not see v_ℓ. In the case where $v_1 =_\sigma v_\ell$, then v_1 is such a vertex, i.e., $v_1 v_\ell \notin E(G)$. Consider now that case where $v_1 <_\sigma v_\ell <_\sigma v_k$, and assume that $v_\ell v_i \in E(G)$ for every vertex $v_i \in V(P)$, $1 \leq i \leq k$. Then for every vertex $v_i \in V(P)$, $1 \leq i \leq k$, it follows that $v_\ell \neq_\sigma v_i$, since vertices belonging to the same layer of the Hasse diagram of G form an independent set. If $v_2 <_\sigma v_1$, then obviously $v_2 <_\sigma v_\ell$. Assume now

that $v_1 <_\sigma v_2$; recall that $v_1 <_\sigma v_\ell$. If $v_1 <_\sigma v_\ell <_\sigma v_2$, from the transitivity property it follows that $v_2v_1 \in E(G)$, since $v_2v_\ell \in E(G)$ and $v_\ell v_1 \in E(G)$; this is a contradiction to our assumption that v_1 and v_2 are consecutive in the antipath P. Thus, $v_2 <_\sigma v_\ell$. Similarly, we can easily show by induction that for every pair v_{x-1}, v_x of consecutive vertices in P, $2 \leq x \leq k-1$, if $v_{x-1} <_\sigma v_\ell$ then $v_x <_\sigma v_\ell$, otherwise $v_{x-1}v_x \in E(G)$ due to the transitivity property. In particular, the same holds for the pair v_{k-2} and v_{k-1}, i.e., from $v_{k-2} <_\sigma v_\ell$, we obtain $v_{k-1} <_\sigma v_\ell$. Recall that $v_\ell <_\sigma v_k$; thus, $v_{k-1} <_\sigma v_\ell <_\sigma v_k$, and since $v_kv_\ell \in E(G)$ and $v_\ell v_{k-1} \in E(G)$, from the transitivity property we obtain that $v_kv_{k-1} \in E(G)$. This comes to a contradiction to our assumption that P is an antipath of G. Thus, there exists at least one vertex of P which does not see v_ℓ.

Let v_{i-1} be the last vertex from left to right in P (i.e., $i-1$ is the greatest index) such that $v_{i-1}v_\ell \notin E(G)$, $2 \leq i \leq k$. Therefore, for every index j, $i \leq j \leq k$, we have $v_jv_\ell \in E(G)$ and, thus, $v_j \neq_\sigma v_\ell$. If $i = k$, then v_{k-1}, v_k is a pair of consecutive vertices in P such that $v_{k-1}v_\ell \notin E(G)$ and $v_\ell <_\sigma v_k$, and the lemma holds. Assume that $2 \leq i \leq k-1$. We will show that $v_\ell <_\sigma v_j$ for every j, $i \leq j \leq k$. For $j = k$, $v_\ell <_\sigma v_k$ holds by assumption. Consider now the case where $i \leq j \leq k-1$. Assume that there exists a vertex v_p, $i \leq p \leq k-1$, such that $v_p <_\sigma v_\ell$; let v_p be the last such vertex from left to right in P. Thus, $v_\ell <_\sigma v_{p+1}$, by the choice of v_p. Then, $v_p <_\sigma v_\ell <_\sigma v_{p+1}$, and since $v_{p+1}v_\ell \in E(G)$ and $v_\ell v_p \in E(G)$, we obtain that $v_{p+1}v_p \in E(G)$. This is a contradiction to our assumption that v_p and v_{p+1} are consecutive in the antipath P of G. Therefore, there exists no vertex v_p, $i \leq p \leq k-1$, such that $v_p <_\sigma v_\ell$. Thus, we have shown that $v_\ell <_\sigma v_j$ for every j, $i \leq j \leq k$. In particular, $v_\ell <_\sigma v_i$. Therefore, the vertices v_{i-1} and v_i are a pair of consecutive vertices in P such that $v_{i-1}v_\ell \notin E(G)$ and $v_\ell <_\sigma v_i$. ∎

5.2.2 Normal Antipaths on Comparability Graphs

It is easy to see that P is a longest antipath of a comparability graph G if and only if P is a longest path of the cocomparability graph $\overline{G}$. Our algorithm computes a longest path P of a cocomparability graph $\overline{G}$, by computing in fact a longest antipath P of the comparability graph G. In particular, our algorithm uses a specific type of antipaths of comparability graphs, which we call normal antipaths. We next define the notion of a normal antipath of a comparability graph G. Recall that by $N_{\overline{G}}(v)$ we denote the set of the antineighbors of a vertex v in the graph G.

Definition 5.2. Let G be a comparability graph, and let σ be a layered ordering of G. The antipath $P = (v_1, v_2, \ldots, v_k)$ of G is called normal, if v_1 is a leftmost (i.e., minimal) vertex of $V(P)$ in σ, and for every i, $2 \leq i \leq k$, the vertex v_i is a leftmost vertex of $N_{\overline{G}}(v_{i-1}) \cap \{v_i, v_{i+1}, \ldots, v_k\}$ in σ.

Using Lemma 5.3 and Definition 5.2, we prove the following result.

Lemma 5.4. *Let G be a comparability graph, and let σ be the layered ordering of G. Let $P = (v_1, v_2, \ldots, v_k)$ be a normal antipath of G, and let v_ℓ, and v_j be two vertices of P such that $v_\ell <_\sigma v_j$ and $v_\ell v_j \in E(G)$. Then $\ell < j$, i.e., v_ℓ appears before v_j in P.*

Proof. Let $P = (v_1, v_2, \ldots, v_k)$ be a normal antipath of a comparability graph G, and let v_ℓ, and v_j be two vertices of P such that $v_\ell <_\sigma v_j$ and $v_\ell v_j \in E(G)$. Assume that $j < \ell$, i.e., $P = (v_1, \ldots, v_j, \ldots, v_\ell, \ldots, v_k)$. Since P is a normal antipath, then v_1 is a leftmost vertex of $V(P)$ in σ; thus, $v_1 \leq_\sigma v_\ell <_\sigma v_j$. Since $P' = (v_1, v_2, \ldots, v_j)$ is an antipath, $v_\ell \notin V(P')$, $v_1 \leq_\sigma v_\ell <_\sigma v_j$, and $v_\ell v_j \in E(G)$, then from Lemma 5.3, we obtain that there exist two consecutive vertices v_{i-1} and v_i in P', $2 \leq i \leq j$, such that $v_{i-1}v_\ell \notin E(G)$ and $v_\ell <_\sigma v_i$. However, this comes to a contradiction to our assumption that P is a normal antipath, since from Definition 5.2 we obtain that v_ℓ should be the next vertex of v_{i-1} in P, instead of v_i. Therefore, we obtain $\ell < j$. ∎

Recall that, if G is a comparability graph, P_G is the partial order corresponding to G, and σ is the layered ordering of G, then $v_\ell <_{P_G} v_j$ if and only if $v_\ell <_\sigma v_j$ and $v_\ell v_j \in E(G)$, for any two vertices $v_\ell, v_j \in V(G)$. Therefore, the definition of a monotone antipath can be paraphrased as follows: an

antipath $P = (v_1, v_2, \ldots, v_k)$ of a comparability graph G is monotone if $v_\ell <_\sigma v_j$ and $v_\ell v_j \in E(G)$ implies that v_ℓ appears before v_j in P. Then, from Lemma 5.4 we obtain that the notion of a normal antipath of a comparability graph G is a generalization of the notion of a monotone antipath of G. In particular we obtain the following result. Note that the inverse of Corollary 5.1 is not always true.

Corollary 5.1. *Let G be a comparability graph. If P is a normal antipath of G, then P is a monotone antipath of G.*

In [23], for proving that for any Hamiltonian path P of a cocomparability graph $\overline{G}$ there exists a monotone Hamiltonian path of $\overline{G}$, Damaschke *et al.* first show that there exists a path $P' = (v_1, v_2, \ldots, v_{|V(G)|})$ of $\overline{G}$ such that v_1 is a minimal vertex of either P_G or P_G^d. Using the same arguments, we show the following lemma, which is useful for obtaining some important results.

Lemma 5.5. *Let G be a comparability graph, and let P_G be the partial order which corresponds to G. If P is an antipath of G, then there exists an antipath P' of G such that $V(P') = V(P)$ which starts with a minimal vertex of $V(P)$ in P_G.*

Proof. Let $P = (v_1, v_2, \ldots, v_x)$ be an antipath of a comparability graph G. Let k be the smallest index such that v_k is either a minimal or a maximal vertex of $V(P)$ in P_G^d. First, consider the case where v_k is a minimal vertex of $V(P)$ in P_G^d. We apply Lemma 5.2 to the antipath $P_1 = (v_k, \ldots, v_x)$, and we obtain a monotone antipath $P_1{}' = (v_k{}', \ldots, v_x{}')$ (with respect to P_G^d) such that $V(P_1') = V(P_1)$ and $v_k' = v_k$. Therefore, $P_2 = (v_1, v_2, \ldots, v_{k-1}, v_k{}', \ldots, v_x{}')$ is an antipath of G such that $V(P_2) = V(P)$. Since $(v_1, v_2, \ldots, v_{k-1})$ contains no maximal vertex of $V(P)$ in P_G^d, and $(v_k{}', \ldots, v_x{}')$ is monotone (with respect to P_G^d), it follows that v_x' is a maximal vertex of $V(P) = \{v_1, v_2, \ldots, v_k', \ldots, v_x'\}$ in P_G^d (and not only of $\{v_k{}', \ldots, v_x{}'\}$). Now, consider the reversed antipath $P' = (v_x', v_{x-1}', \ldots, v_k', v_{k-1}, \ldots, v_1)$, where v_x' is a minimal vertex of $V(P')$ in P_G. Thus, P' is an antipath of G such that $V(P') = V(P)$ which starts with a minimal vertex of $V(P)$ in P_G.

Consider now the case there v_k is a maximal vertex of $V(P)$ in P_G^d. Thus, v_k is a minimal vertex of $V(P)$ in P_G. Then following the above same maner we can obtain an antipath $P' = (v_1', v_2', \ldots, v_x')$ of G such that $V(P') = V(P)$ and v_1' is a minimal vertex of $V(P)$ in P_G^d. Thus, by applying again the same above procedure to P', we can obtain an antipath P'' of G such that $V(P'') = V(P)$ which starts with a minimal vertex of $V(P)$ in P_G. ∎

The following result is very important for proving the correctness of our algorithm for solving the longest path problem on cocomparability graphs.

Lemma 5.6. *Let P be a longest antipath of a comparability graph G. Then, there exists a normal antipath P' of G, such that $V(P') = V(P)$.*

Proof. Let G be a comparability graph, P_G be the partial order that corresponds to G, σ be the layered ordering of G, and let $P = (v_1, v_2, \ldots, v_k)$ be a longest antipath of G. If $k = 1$, the lemma holds. Suppose that $k \geq 2$. We will prove that for every index i, $2 \leq i \leq k$, there exists an antipath $P_i = (v_1', v_2', \ldots, v_k')$, such that $V(P_i) = V(P)$, v_1' is a leftmost vertex of $V(P_i)$ in σ, and for every index j, $2 \leq j \leq i$, the vertex v_j' is a leftmost vertex of $N_{\overline{G}}(v_{j-1}') \cap \{v_j', v_{j+1}', \ldots, v_k'\}$ in σ. The proof will be done by induction on i.

From Lemma 5.5, we may assume that v_1 is a minimal vertex of $V(P)$ in P_G, and then from Lemma 5.2 we may assume that P is a monotone antipath of G. Thus, for every vertex v_i, $2 \leq i \leq k$, such that $v_i <_\sigma v_1$, we have $v_i v_1 \notin E(G)$. If v_1 is a leftmost vertex of $V(P)$ in σ, then $P_1 = P$. Consider now the case where v_1 is not a leftmost vertex of $V(P)$ in σ. Let j, $2 \leq j \leq k$, be the greatest index such that v_j is a leftmost vertex of $V(P)$ in σ. If $v_1 v_{j+1} \notin E(G)$ then $P_1 = (v_j, v_{j-1}, \ldots, v_1, v_{j+1}, \ldots, v_k)$ is an antipath of G such that $V(P_1) = V(P)$ and v_1 is a leftmost vertex of $V(P_1)$ in σ.

Consider now the case where $v_1 v_{j+1} \in E(G)$. Since P is monotone and v_1 appears in P before v_{j+1}, we obtain that $v_1 <_\sigma v_{j+1}$. Since $v_j <_\sigma v_1 <_\sigma v_{j+1}$, $v_j v_{j+1} \notin E(G)$, and $v_1 v_{j+1} \in E(G)$, from the transitivity

property it follows that $v_1v_j \notin E(G)$. Therefore, by the construction of the Hasse diagram of G (and, thus, of σ), there exists a vertex v_x in G, such that $v_x =_\sigma v_1$ and $v_jv_x \in E(G)$; thus, $v_{j+1}v_x \notin E(G)$ due to the transitivity property. If $v_x \notin V(P)$, then $P' = (v_j, v_{j-1}, \ldots, v_1, v_x, v_{j+1}, \ldots, v_k)$ is an antipath of G longer than P. This is a contradiction to our assumption that P is a longest antipath of G, thus, $v_x \in V(P)$. Since P is monotone, $v_jv_x \in E(G)$, and $v_j <_\sigma v_x =_\sigma v_1$, it follows that v_j appears in P before v_x, i.e., $j+1 \leq x \leq k$. In fact, $j+2 \leq x \leq k$, since $v_x =_\sigma v_1 <_\sigma v_{j+1}$. Then $P' = (v_j, v_{j-1}, \ldots, v_1, v_x, v_{x-1}, \ldots, v_{j+1})$ is an antipath of G such that $V(P') = V(P) \setminus \{v_{x+1}, v_{x+2}, \ldots, v_k\}$. If $v_{j+1}v_{x+1} \notin E(G)$ then $P_1 = (v_j, v_{j-1}, \ldots, v_1, v_x, v_{x-1}, \ldots, v_{j+1}, v_{x+1}, \ldots, v_k)$ is an antipath of G such that $V(P_1) = V(P)$ and v_j is a leftmost vertex of $V(P_1)$ in σ.

Consider now the case where $v_{j+1}v_{x+1} \in E(G)$. Since P is monotone, $v_{j+1}v_{x+1} \in E(G)$ and v_{j+1} appears in P before v_{x+1}, we have that $v_{j+1} <_\sigma v_{x+1}$; thus, $v_x <_\sigma v_{j+1} <_\sigma v_{x+1}$. Since $v_xv_{j+1} \notin E(G)$, it follows by the construction of the Hasse diagram, that there exists a vertex v_y in G such that $v_y =_\sigma v_{j+1}$ and $v_xv_y \in E(G)$; thus, $v_{x+1}v_y \notin E(G)$ due to the transitivity property. Similarly to the above, $v_y \in V(P)$, since P is a longest antipath of G. Since P is monotone, $v_xv_y \in E(G)$ and $v_x <_\sigma v_y =_\sigma v_{j+1}$, it follows that v_x appears in P before v_y, i.e., $x+1 \leq y \leq k$ and, in fact, $x+2 \leq y \leq k$. Therefore, $P' = (v_j, v_{j-1}, \ldots, v_1, v_x, v_{x-1}, \ldots, v_{j+1}, v_y, v_{y-1}, \ldots, v_{x+1})$ is an antipath of G such that $V(P') = V(P) \setminus \{v_{y+1}, v_{y+2}, \ldots, v_k\}$. Again, if $v_{x+1}v_{y+1} \notin E(G)$, then using the above transformation we obtain an antipath P_1. If $v_{x+1}v_{y+1} \in E(G)$, then we can repeat the above procedure until we find a pair of vertices v_{x+1} and v_{y+1} in P such that $v_y <_\sigma v_{x+1}$, $x+2 \leq y \leq k$, and $v_{x+1}v_{y+1} \notin E(G)$.

Assume that such a pair of vertices v_{x+1} and v_{y+1} does not exists in P, i.e., v_{y+1} is the last vertex v_k of P, $v_y <_\sigma v_{x+1}$, $x+2 \leq y = k-1$, and $v_{x+1}v_{y+1} \in E(G)$. Therefore, $P' = (v_j, v_{j-1}, \ldots, v_1, v_x, v_{x-1}, \ldots, v_{j+1}, v_y, v_{y-1}, \ldots, v_{x+1})$ is an antipath of G such that $V(P') = V(P) \setminus \{v_{y+1}\}$ and $y+1 = k$. Since P is monotone, $v_{x+1}v_{y+1} \in E(G)$, and v_{x+1} appears in P before v_{y+1}, it follows that $v_{x+1} <_\sigma v_{y+1}$; thus, $v_y <_\sigma v_{x+1} <_\sigma v_{y+1}$. Then, similarly to the above, it follows that $v_yv_{x+1} \notin E(G)$, and thus there exists a vertex v_ℓ in G such that $v_{x+1} =_\sigma v_\ell$ and $v_yv_\ell \in E(G)$; thus $v_\ell v_{y+1} \notin E(G)$. Since P is monotone, $v_y <_\sigma v_\ell$ and $v_yv_\ell \in E(G)$, it follows that if $v_\ell \in V(P)$, then v_ℓ appears in P after v_y and, in fact, after v_{y+1}, i.e., $y+1 < \ell \leq k$. This comes to a contradiction to our assumption that $y+1 = k$, i.e., v_{y+1} is the last vertex v_k of P. Thus, $v_\ell \notin V(P)$ and, therefore, $P' = (v_j, v_{j-1}, \ldots, v_1, v_x, v_{x-1}, \ldots, v_{j+1}, v_y, v_{y-1}, \ldots, v_{x+1}, v_\ell, v_{y+1})$ is an antipath of G longer that P, since $vy+1 = k$ and, thus, $V(P') = V(P) \cup \{v_\ell\}$. This comes to a contradiction to our assumption that P is a longest antipath of G. Therefore, there exists a pair of vertices v_{x+1} and v_{y+1} in P such that $v_y <_\sigma v_{x+1}$, $x+2 \leq y \leq k$, and $v_{x+1}v_{y+1} \notin E(G)$. Then, $P_1 = (v_j, v_{j-1}, \ldots, v_1, v_x, v_{x-1}, \ldots, v_{j+1}, v_y, v_{y-1}, \ldots, v_{x+1}, v_{y+1}, v_{y+2}, \ldots, v_k)$ is an antipath such that $V(P_1) = V(P)$ and v_j is a leftmost vertex of $V(P_1)$ in σ. This completes the proof for the induction basis.

Consider now an arbitrary index i, $2 \leq i \leq k-1$, and let $P_i = (v'_1, v'_2, \ldots, v'_i, v'_{i+1}, \ldots, v'_k)$ be an antipath of G, such that $V(P_i) = V(P)$, v'_1 is a leftmost vertex of $V(P_i)$ in σ, and for every index j, $2 \leq j \leq i$, the vertex v'_j is a leftmost vertex of $N_{\overline{G}}(v'_{j-1}) \cap \{v'_j, v'_{j+1}, \ldots, v'_k\}$ in σ. Therefore, the antipath $(v'_1, v'_2, \ldots, v'_i)$ is normal. We now show that v'_i is a minimal vertex of $\{v'_i, v'_{i+1}, \ldots, v'_k\}$ in P_G. Assume otherwise that there exists a vertex $v'_x \in \{v'_{i+1}, v'_{i+2}, \ldots, v'_k\}$, such that $v'_x <_{P_G} v'_i$ or, equivalently, $v'_x <_\sigma v'_i$ and $v'_xv'_i \in E(G)$. By the induction hypothesis, v'_1 is a leftmost vertex of $V(P)$ in σ and, thus, $v'_1 \leq_\sigma v'_x <_\sigma v'_i$. Since $P' = (v'_1, v'_2, \ldots, v'_i)$ is an antipath of G, $v'_x \notin V(P')$, $v'_xv'_i \in E(G)$, and $v'_1 \leq_\sigma v'_x <_\sigma v'_i$, from Lemma 5.3 we obtain that there exist two consecutive vertices v'_{y-1} and v'_y in P', $2 \leq y \leq i$, such that $v'_{y-1}v'_x \notin E(G)$ and $v'_x <_\sigma v'_y$. This comes to a contradiction to our assumptions, since by the induction hypothesis v'_y is a leftmost vertex of $N_{\overline{G}}(v'_{y-1}) \cap \{v'_y, v'_{y+1}, \ldots, v'_i, \ldots, v'_x, \ldots, v'_k\}$, while $v'_x \in N_{\overline{G}}(v'_{y-1}) \cap \{v'_y, v'_{y+1}, \ldots, v'_i, \ldots, v'_x, \ldots, v'_k\}$ and $v'_x <_\sigma v'_y$. Therefore, we conclude that v'_i is a minimal vertex of $\{v'_i, v'_{i+1}, \ldots, v'_k\}$ in P_G. From Lemma 5.2, for any antipath P of a comparability graph G which starts with a minimal element v of $V(P)$ in P_G, there exists a monotone antipath P'' of G starting with the same vertex v such that $V(P'') = V(P)$. Therefore, without loss of generality we may assume that $\{v'_i, v'_{i+1}, \ldots, v'_k\}$ is a monotone antipath of G. Therefore, by the induction hypothesis

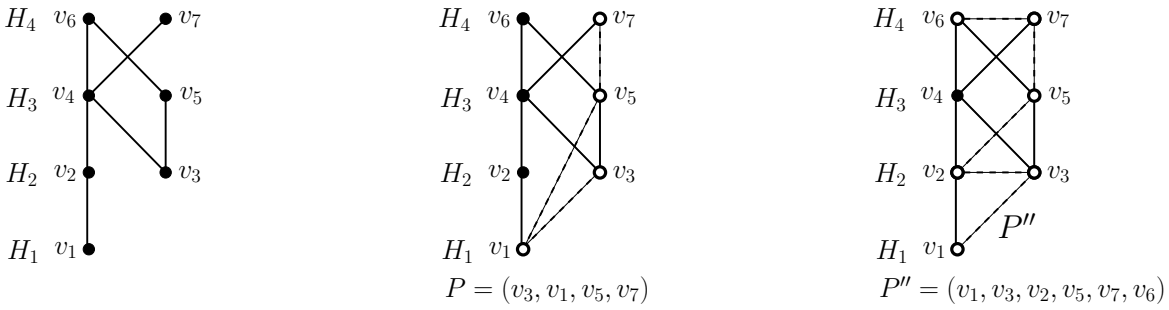

Figure 5.2: Illustrating a Hasse diagram of a comparability graph G, an antipath $P = (v_3, v_1, v_5, v_7)$ of G which is not normal and a normal antipath $P'' = (v_1, v_3, v_2, v_5, v_7, v_6)$ of G.

it is easy to obtain that the path P_i is a monotone path.

If v'_{i+1} is a leftmost vertex of $N_{\overline{G}}(v'_i) \cap \{v'_{i+1}, v'_{i+2}, \ldots, v'_k\}$ in σ, then $P_{i+1} = P_i$. Consider now the case where v_{i+1} is not a leftmost vertex of $N_{\overline{G}}(v'_i) \cap \{v'_{i+1}, v'_{i+2}, \ldots, v'_k\}$ in σ. Let j, $i+2 \leq j \leq k$, be the greatest index for which v'_j is a leftmost vertex of $N_{\overline{G}}(v'_i) \cap \{v'_{i+1}, v'_{i+2}, \ldots, v'_k\}$ in σ. Then, $P' = (v'_1, v'_2, \ldots, v'_i, v'_j, v'_{j-1}, \ldots, v'_{i+1})$ is an antipath of G such that $V(P') = V(P) \setminus \{v'_{j+1}, v'_{j+2}, \ldots, v'_k\}$. If $v'_{i+1}v'_{j+1} \notin E(G)$, then $P_{i+1} = (v''_1, v''_2, \ldots, v''_i, v''_{i+1}, \ldots, v''_k) = (v'_1, v'_2, \ldots, v'_i, v'_j, v'_{j-1}, \ldots, v'_{i+1}, v'_{j+1}, v'_{j+2}, \ldots, v'_k)$ is an antipath of G such that $V(P_{i+1}) = V(P)$, v''_1 is a leftmost vertex of $V(P_{i+1})$ in σ, and for every index ℓ, $2 \leq \ell \leq i+1$, the vertex v''_ℓ is a leftmost vertex of $N_{\overline{G}}(v''_{\ell-1}) \cap \{v''_\ell, v''_{\ell+1}, \ldots, v''_k\}$ in σ. In the case where $v'_{i+1}v'_{j+1} \in E(G)$, then we repeat exactly the same procedure described in the induction basis until we find a pair of vertices v'_{x+1} and v'_{y+1} in P such that $v'_y <_\sigma v'_{x+1}$, $x+2 \leq y \leq k$, and $v'_{x+1}v'_{y+1} \notin E(G)$; such a pair of vertices exists, as we have proven in the induction basis. Then, $P_{i+1} = (v''_1, v''_2, \ldots, v''_i, v''_{i+1}, \ldots, v''_k) = (v'_1, v'_2, \ldots, v'_i, v'_j, v'_{j-1}, \ldots, v'_{i+1}, \ldots, v'_{x+1}, v'_{y+1}, v'_{y+2}, \ldots, v'_k)$ is an antipath of G such that $V(P_{i+1}) = V(P)$, v''_1 is a leftmost vertex of $V(P_{i+1})$ in σ, and for every index ℓ, $2 \leq \ell \leq i+1$, the vertex v''_ℓ is a leftmost vertex of $N_{\overline{G}}(v''_{\ell-1}) \cap \{v''_\ell, v''_{\ell+1}, \ldots, v''_k\}$ in σ. This completes the proof for the induction step.

Thus, the antipath $P' = P_k$ is a normal antipath of G such that $V(P') = V(P)$. ∎

Figure 5.2 illustrates a Hasse diagram of a comparability graph G. The antipath $P = (v_3, v_1, v_5, v_7)$ of G is not normal, and there exists no normal antipath P' of G such that $V(P') = V(P)$. Also, P is not a longest antipath of G, since there exists an antipath $P'' = (v_1, v_3, v_2, v_5, v_7, v_6)$ of G such that $|P''| > |P|$; note that P'' is a normal antipath of G.

5.3 The Algorithm

In this section we present our algorithm, which we call Algorithm LP_Cocomparability, for solving the longest path problem on cocomparability graphs. The proposed algorithm computes a longest path P of a cocomparability graph G, by computing actually a longest antipath P of the comparability graph $\overline{G}$.

Let G be a comparability graph, given by its Hasse diagram with layers $H_1, H_2, \ldots, H_k$. For simplifying our notations, we add a dummy vertex u_0 to G, such that u_0 belongs to a layer H_0 in the Hasse diagram of G, and $u_0u_i \in E(G)$, for every i, $1 \leq i \leq n$; let G' be the resulting graph after adding the vertex u_0. Note that, G' is a comparability graph, having a Hasse diagram with layers $H_0, H_1, H_2, \ldots, H_k$, and let $\sigma = (u_0, u_1, u_2, \ldots, u_n)$ be the layered ordering of G'. Note that for any longest antipath of G' there exists a longest antipath of G' which does not contain the dummy vertex u_0; such an antipath is also a longest antipath of G, since G is an induced subgraph of G'. Algorithm LP_Cocomparability computes a longest antipath of G' which does not contain the vertex u_0 and, thus, it is a longest antipath

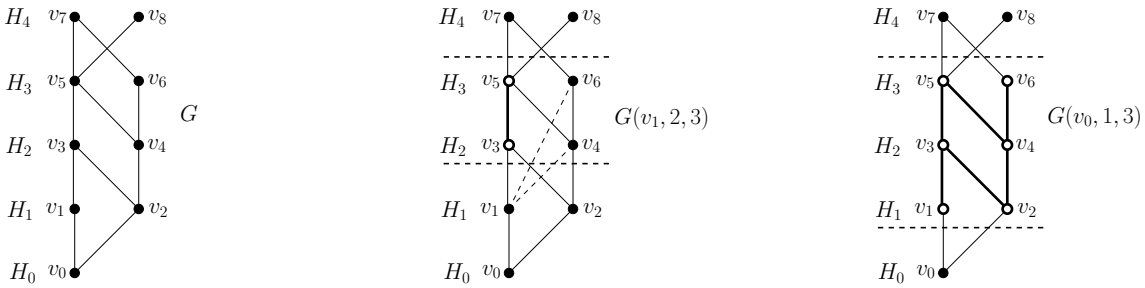

Figure 5.3: Illustrating a Hasse diagram of a comparability graph G and the induced subgraphs $G(v_1, 2, 3)$ and $G(v_0, 1, 3)$ of G.

of the original graph G as well. Hereafter, we consider comparability graphs having assumed that we have already added the dummy vertex u_0, which we denote by G, and the antipaths we compute in G are also antipaths of the graph $G \setminus \{u_0\} = G[V(G) \setminus \{u_0\}]$. We next give some definitions and notations necessary for the description of the algorithm.

Let $L_j = (v_1, v_2, \ldots, v_k)$ be an arbitrary ordering of the set $\{v_1, v_2, \ldots, v_k\}$. We denote by $V(L_j)$ the set of vertices in the ordering L_j and by $|L_j|$ the cardinality of the set $V(L_j)$, i.e., $|L_j| = |V(L_j)|$. For every vertex $v_z \in L_j$, we denote by $L_j(v_z)$ the ordering $(v_1, v_2, \ldots, v_{z-1}, v_{z+1}, v_{z+2}, \ldots, v_{|L_j|}, v_z)$, and for every index r, $0 \leq r \leq |L_j|$, we denote by $L_j^r(v_z)$ the ordering containing the first r vertices of $L_j(v_z)$; that is:

- $L_j = (v_1, v_2, \ldots, v_{|L_j|})$,
- $L_j(v_z) = (v_1, v_2, \ldots, v_{z-1}, v_{z+1}, v_{z+2}, \ldots, v_{|L_j|}, v_z)$,
- $L_j^0(v_z) = \emptyset$,
- $L_j^r(v_z) = (v_1, v_2, \ldots, v_r)$ if $1 \leq r \leq z-1$,
- $L_j^r(v_z) = (v_1, v_2, \ldots, v_{z-1}, v_{z+1}, v_{z+2}, \ldots, v_{r+1})$ if $z \leq r \leq |L_j| - 1$, and
- $L_j^r(v_z) = L_j(v_z)$ if $r = |L_j|$.

Definition 5.3. Let G be a comparability graph, given by its Hasse diagram with layers $H_0, H_1, H_2, \ldots, H_k$, and let $\sigma = (u_0, u_1, u_2, \ldots, u_n)$ be the layered ordering of G. For every triple p, i, and j, where $1 \leq i \leq j \leq k$ and $u_p \in H_{i-1}$, we define the graph $G(u_p, i, j)$ to be the subgraph $G[S]$ of G induced by the set $S = \{u_x : u_x \in H_\ell, i \leq \ell \leq j\} \setminus \{u_x : u_p u_x \notin E(G)\}$.

Definition 5.4. Let L_j be an ordering of the set $H_j \cap V(G(u_p, i, j))$. We define the graph $G_{u_z}^r(u_p, i, j)$, where $u_z \in L_j$ and $0 \leq r \leq |L_j|$, to be the subgraph $G[S]$ of G induced by the set $S = V(G(u_p, i, j-1)) \cup L_j^r(u_z)$ if $i < j$, and $S = L_j^r(u_z)$ if $i = j$.

Note that, since the dummy vertex u_0 is adjacent to every other vertex of G, the graph $G(u_p, 1, j)$, $1 \leq j \leq k$, is the subgraph $G[S]$ of G, induced by the set $S = \{u_x : u_x \in H_\ell, 1 \leq \ell \leq j\}$. Additionally, $G_{u_z}^{|L_j|}(u_p, i, j) = G(u_p, i, j)$, and if $i < j$, then $G_{u_z}^0(u_p, i, j) = G(u_p, i, j-1)$.

Figure 5.3 illustrates two examples which correspond to Definition 5.3. In particular, the figure to the left illustrates a Hasse diagram of a comparability graph G with layers $H_0, H_1, \ldots, H_4$. The figure in the middle illustrates the subgraph $G(v_1, 2, 3)$ of G induced by the vertices $\{v_3, v_5\}$. The figure to the right illustrates the subgraph $G(v_0, 1, 3)$ of G induced by the vertices $\{v_1, v_2, v_3, v_4, v_5, v_6\}$, which are all the vertices belonging to the layers H_1, H_2, H_3; recall that no vertex of G is an antineighbor of the dummy vertex v_0.

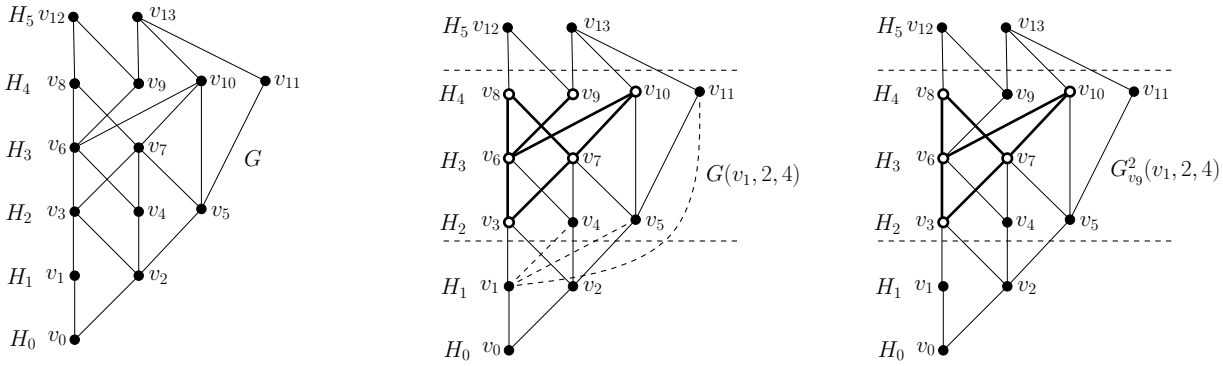

Figure 5.4: Illustrating a Hasse diagram of a comparability graph G and the induced subgraphs $G(v_1, 2, 4)$ and $G^2_{v_9}(v_1, 2, 4)$ of G.

Figure 5.4 illustrates an example which corresponds to Definition 5.4. In particular, the figure to the left illustrates a Hasse diagram of a comparability graph G with layers $H_0, H_1, \ldots, H_5$. The figure in the middle illustrates the subgraph $G(v_1, 2, 4)$ of G induced by the vertices $\{v_3, v_6, v_7, v_8, v_9, v_{10}\}$. The figure to the right illustrates the subgraph $G^2_{v_9}(v_1, 2, 4)$ of G, if we consider the ordering $L_4 = (v_8, v_9, v_{10})$ for the vertices of $H_4 \cap V(G(v_1, 2, 4))$. The subgraph $G^2_{v_9}(v_1, 2, 4)$ of G is induced by the vertices $\{v_3, v_6, v_7, v_8, v_{10}\}$, and it is actually an induced subgraph of $G(v_1, 2, 4)$.

Notation 5.1. *Let G be a comparability graph, given by its Hasse diagram with layers $H_0, H_1, H_2, \ldots, H_k$, and let $\sigma = (u_0, u_1, u_2, \ldots, u_n)$ be the layered ordering of G. For every vertex $u_t \in V(G^r_{u_z}(u_p, i, j))$, if $u_t \in H_j$, then we denote by $f(u_t)$ the smallest index such that $f(u_t) < j$, for which there exists a vertex u_x of $G^r_{u_z}(u_p, i, j)$ such that $u_x \in H_{f(u_t)}$ and $u_x u_t \notin E(G)$; in the case where no such vertex u_x exists in $G^r_{u_z}(u_p, i, j)$, where $u_x <_\sigma u_t$, we set $f(u_t) = j$.*

Notation 5.2. *Let G be a comparability graph, and let $\sigma = (u_0, u_1, u_2, \ldots, u_n)$ be the layered ordering of G. For every vertex $u_y \in V(G^r_{u_z}(u_p, i, j))$ we denote by $P(u_y; G^r_{u_z}(u_p, i, j))$ a longest normal antipath of $G^r_{u_z}(u_p, i, j)$ with right endpoint the vertex u_y, and by $\ell(u_y; G^r_{u_z}(u_p, i, j))$ the length of $P(u_y; G^r_{u_z}(u_p, i, j))$.*

Notation 5.2 is also used by substituting $G^r_{u_z}(u_p, i, j)$ by $G(u_p, i, j)$. Note that, when we refer to an antipath $P = P(u_y; G(u_p, i, j))$ as a longest normal antipath, it follows that P is a normal antipath of $G(u_p, i, j)$ with right endpoint the vertex u_y, and that P is a longest such antipath; thus, P is not necessarily a longest antipath of $G(u_p, i, j)$. However, if P' is a longest antipath of $G(u_p, i, j)$, from Lemma 5.6 we may assume that P' is normal; let u_y be the last vertex of the normal antipath P'. Thus, there exists a longest normal antipath $P' = P(u_y; G(u_p, i, j))$ which is also a longest antipath of $G(u_p, i, j)$ for some vertex $u_y \in V(G(u_p, i, j))$.

Given a comparability graph G, Algorithm LP_Cocomparability computes for every induced subgraph $G(u_p, i, j)$ of G, and for every vertex $u_y \in V(G(u_p, i, j))$, the length $\ell(u_y; G(u_p, i, j))$ and the corresponding antipath $P(u_y; G(u_p, i, j))$. Since $G(u_0, 1, k) = G \setminus \{u_0\}$, it follows that the maximum among the values $\ell(u_y; G(u_0, 1, k))$ is the length of a longest normal antipath $P(u_y; G(u_0, 1, k))$ of $G \setminus \{u_0\}$ and, thus, of G. In Section 5.4.1, we prove that the normal antipath $P(u_y; G(u_0, 1, k))$ computed by Algorithm LP_Cocomparability is also a longest antipath of G and, thus, a longest path of the cocomparability graph $\overline{G}$.

Before giving Algorithm LP_Cocomparability in detail, which is presented in Figures Algorithm 6 and Algorithm 7, we give a high level description of our algorithm.

Algorithm LP_Cocomparability. Let G be a comparability graph, given by a Hasse diagram with $H_0, H_1, \ldots, H_k$. Let P be a longest antipath of G. Since there exists a longest antipath of G which does not contain the vertex u_0 we may assume that P belongs to the graph $G \setminus \{u_0\}$. Due to Lemma 5.6, we may assume without loss of generality that P is a normal antipath; let the vertex u be the right endpoint of P.

(A) For every vertex $u_y \in V(G(u_0, 1, k))$

`compute` a longest normal antipath of $G(u_0, 1, k)$ with right endpoint the vertex u_y, where $G(u_0, 1, k) = G \setminus \{u_0\}$.

(B) `Compute` the longest antipath among the n antipaths computed in (A).

Step (B) is trivial, while for executing Step (A) we do the following:

(A.1) For every subgraph $G(u_p, i, j)$ and

for every vertex $u_y \in V(G(u_p, i, j))$

`compute` a longest normal antipath of $G(u_p, i, j)$ with right endpoint the vertex u_y.

Let L_j be an ordering of $H_j \cap V(G(u_p, i, j))$.

(A.1.1) For every subgraph $G^r_{u_z}(u_p, i, j)$ and

for every vertex $u_y \in V(G^r_{u_z}(u_p, i, j))$ such that $u_y \notin L_j \setminus \{u_t\}$ (where u_t is the last vertex of $L^r_j(u_z)$)

`compute` a longest normal antipath of $G^r_{u_z}(u_p, i, j)$ with right endpoint the vertex u_y, where $G^{|L_j|}_{u_z}(u_p, i, j) = G(u_p, i, j)$, $\forall u_z \in L_j$.

5.4 Correctness and Time Complexity

In this section we prove the correctness of our algorithm and compute its time complexity. In particular, in Section 5.4.1 we show that Algorithm LP_Cocomparability computes a longest normal antipath P of the comparability graph G which is, in fact, a longest antipath of G and, thus, a longest path of the cocomparability graph $\overline{G}$. Finally, in Section 5.4.2 we analyze the time complexity of our algorithm.

5.4.1 Correctness of Algorithm LP_Cocomparability

We next prove that Algorithm LP_Cocomparability correctly computes a longest antipath of the comparability graph G. The following lemmas appear useful in the proof of the algorithm's correctness.

Lemma 5.7. *Let G be a comparability graph, given by its Hasse diagram with layers $H_0, H_1, H_2, \ldots, H_k$, let σ be the layered ordering of G, and let L_j be an ordering of the set $H_j \cap V(G(u_p, i, j))$. Let $P = (P_1, v_\ell, P_2)$ be a normal antipath of $G^r_{u_z}(u_p, i, j)$, and let v_ℓ be the last vertex of $L^r_j(u_z)$. Then, P_1 and P_2 are normal antipaths of $G^r_{u_z}(u_p, i, j)$.*

ALGORITHM LP_COCOMPARABILITY

Input: a comparability graph G, given by its Hasse diagram with layers $H_0, H_1, H_2, \ldots, H_k$, and a layered ordering $\sigma = (u_0, u_1, u_2, \ldots, u_n)$ of G.

Output: a longest normal antipath of G.

1. `for` $j = 1$ to k
2. `for` $i = j$ downto 1
3. `for` every vertex $u_p \in H_{i-1}$
4. let L_j be an ordering of $H_j \cap V(G(u_p, i, j))$
5. `for` every vertex $u_z \in L_j$
6. `for` $r = 1$ to $|L_j|$
7. let u_t be the last vertex of $L_j^r(u_z)$
8. `for` every vertex $u_y \in V(G^r_{u_z}(u_p, i, j))$ and $y \neq t$ {*initialization for* $u_y \neq u_t$}
9. `if` $r = 1$ `then`
10. $\ell(u_y; G^0_{u_z}(u_p, i, j)) \leftarrow \ell(u_y; G(u_p, i, j-1))$;
11. $P(u_y; G^0_{u_z}(u_p, i, j)) \leftarrow P(u_y; G(u_p, i, j-1))$;
12. $\ell(u_y; G^r_{u_z}(u_p, i, j)) \leftarrow \ell(u_y; G^{r-1}_{u_z}(u_p, i, j))$;
13. $P(u_y; G^r_{u_z}(u_p, i, j)) \leftarrow P(u_y; G^{r-1}_{u_z}(u_p, i, j))$;
14. `end_for`
15. `if` $i = j$ `then` {*case* $i = j$}
16. $\ell(u_t; G^r_{u_z}(u_p, j, j)) \leftarrow |L_j^r(u_z)|$;
17. $P(u_t; G^r_{u_z}(u_p, j, j)) \leftarrow L_j^r(u_z)$;
18. `if` $i \neq j$ `then`
19. $\ell(u_t; G^r_{u_z}(u_p, i, j)) \leftarrow 1$; {*initialization for* $u_y = u_t$}
20. $P(u_t; G^r_{u_z}(u_p, i, j)) \leftarrow (u_t)$;
21. `execute` process$(G^r_{u_z}(u_p, i, j))$;
22. `end_for`
23. $\ell(u_z; G(u_p, i, j)) \leftarrow \ell(u_z; G^{|L_j|}_{u_z}(u_p, i, j))$; {*for the vertex* $u_z \in L_j$}
24. $P(u_z; G(u_p, i, j)) \leftarrow P(u_z; G^{|L_j|}_{u_z}(u_p, i, j))$;
25. `end_for`
26. `for` every vertex $u_y \in V(G(u_p, i, j))$ and $u_y \notin L_j$ {*for the vertices* $u_y \notin L_j$}
27. $\ell(u_y; G(u_p, i, j)) \leftarrow \ell(u_y; G^{|L_j|}_{u_z}(u_p, i, j))$;
28. $P(u_y; G(u_p, i, j)) \leftarrow P(u_y; G^{|L_j|}_{u_z}(u_p, i, j))$;
29. `end_for`
30. `end_for`
31. `end_for`
32. `end_for`
33. `compute` the $max\{\ell(u_y; G(u_0, 1, k)) : u_y \in G(u_0, 1, k)\}$ and the corresponding antipath $P(u_y; G(u_0, 1, k))$;

Algorithm 6: Algorithm LP_Cocomparability for finding a longest antipath of G.

where the procedure `process()` is as follows:

PROCESS($G^r_{u_z}(u_p, i, j)$)

```
procedure bridge(G^r_{u_z}(u_p, i, j))
if f(u_t) < j then          {u_t is the last vertex of L^r_j(u_z)}
   for h = f(u_t) + 1 to j
      for ℓ = f(u_t) to h − 1
         for every vertex u_x ∈ H_ℓ ∩ V(G^{r−1}_{u_z}(u_p, i, j)) and u_x u_t ∉ E(G)
            for every vertex u_y ∈ H_h ∩ V(G^{r−1}_{u_z}(u_x, ℓ + 1, j))
               w_1 ← ℓ(u_x; G^{r−1}_{u_z}(u_p, i, j));   P'_1 ← P(u_x; G^{r−1}_{u_z}(u_p, i, j));
               w_2 ← ℓ(u_y; G^{r−1}_{u_z}(u_x, ℓ + 1, j));   P'_2 ← P(u_y; G^{r−1}_{u_z}(u_x, ℓ + 1, j));
               if w_1 + w_2 + 1 > ℓ(u_y; G^r_{u_z}(u_p, i, j)) then
                  ℓ(u_y; G^r_{u_z}(u_p, i, j)) ← w_1 + w_2 + 1;
                  P(u_y; G^r_{u_z}(u_p, i, j)) ← (P'_1, u_t, P'_2);
            end_for
         end_for
      end_for
   end_for

procedure append(G^r_{u_z}(u_p, i, j))
for ℓ = f(u_t) to j          {u_t is the last vertex of L^r_j(u_z)}
    for every vertex u_x ∈ H_ℓ ∩ (V(G^{r−1}_{u_z}(u_p, i, j)) and u_x u_t ∉ E(G)
       w_1 ← ℓ(u_x; G^{r−1}_{u_z}(u_p, i, j));   P'_1 ← P(u_x; G^{r−1}_{u_z}(u_p, i, j));
       if w_1 + 1 > ℓ(u_t; G^r_{u_z}(u_p, i, j)) then
          ℓ(u_t; G^r_{u_z}(u_p, i, j)) ← w_1 + 1;
          P(u_t; G^r_{u_z}(u_p, i, j)) ← (P'_1, u_t);
   end_for
end_for

return (the value ℓ(u_y; G^r_{u_z}(u_p, i, j)) and the antipath P(u_y; G^r_{u_z}(u_p, i, j)), for every vertex
        u_y ∈ V(G^r_{u_z}(u_p, f(u_t) + 1, j)) if f(u_t) < j, and for u_y = u_t if f(u_t) = j);
```

Algorithm 7: The procedure process().

Proof. Let $P = (v_1, v_2, \ldots, v_{\ell-1}, v_\ell, v_{\ell+1}, \ldots, v_y)$ be a normal antipath of $G^r_{u_z}(u_p, i, j)$. Then, from Definition 5.2, v_1 is a leftmost vertex of $V(P)$ in σ, and for every index x, $2 \leq x \leq y$, the vertex v_x is a leftmost vertex of $N_{\overline{G}}(v_{x-1}) \cap \{v_x, v_{x+1}, \ldots, v_y\}$ in σ. It is easy to see that $P_1 = (v_1, v_2, \ldots, v_{\ell-1})$ is a normal antipath of $G^r_{u_z}(u_p, i, j)$. Indeed, since $V(P_1) \subset V(P)$, then v_1 is also a leftmost vertex of $V(P_1)$ in σ and, additionally, v_x is a leftmost vertex of $N_{\overline{G}}(v_{x-1}) \cap \{v_x, v_{x+1}, \ldots, v_{\ell-1}\}$ in σ, for every index x, $2 \leq x \leq \ell - 1$.

Consider now the antipath $P_2 = (v_{\ell+1}, v_{\ell+2}, \ldots, v_y)$ of $G^r_{u_z}(u_p, i, j)$. We first prove that $v_{\ell+1}$ is a leftmost vertex of $V(P_2)$ in σ. By assumption $v_\ell \in L_j$, thus, $v_x \leq_\sigma v_\ell$ for every index x, $\ell + 1 \leq x \leq y$. We will show that $v_x v_\ell \notin E(G)$, for every index x, $\ell + 1 \leq x \leq y$. Let v_x be a vertex of $V(P_2)$. Consider first the case where $v_x =_\sigma v_\ell$; then it is straightforward that $v_x v_\ell \notin E(G)$. Consider now the case where $v_x <_\sigma v_\ell$. Since P is a normal antipath, $v_x <_\sigma v_\ell$, and v_ℓ appears before v_x in P, from Lemma 5.4 we obtain that $v_x v_\ell \notin E(G)$. Thus, we have proved that $v_x v_\ell \notin E(G)$ for every vertex $v_x \in V(P_2)$. Since $v_{\ell+1}$ is a leftmost vertex of $N_{\overline{G}}(v_\ell) \cap \{v_{\ell+1}, v_{\ell+2}, \ldots, v_y\}$ in σ, and since $N_{\overline{G}}(v_\ell) \cap \{v_{\ell+1}, v_{\ell+2}, \ldots, v_y\} = V(P_2)$, it follows that $v_{\ell+1}$ is a leftmost vertex of $V(P_2)$ in σ. Additionally, since P is a normal antipath, it is straightforward that v_x is a leftmost vertex of $N_{\overline{G}}(v_{x-1}) \cap \{v_x, v_{x+1}, \ldots, v_y\}$ in σ, for every index x, $\ell + 2 \leq x \leq y$. Therefore, from Definition 5.2 it follows that P_2 is a normal antipath of $G^r_{u_z}(u_p, i, j)$. ∎

Lemma 5.8. *Let G be a comparability graph, given by its Hasse diagram with layers $H_0, H_1, H_2, \ldots, H_k$, let $\sigma = (u_0, u_1, u_2, \ldots, u_n)$ be the layered ordering of G, let L_j be an ordering of the set $H_j \cap V(G(u_p, i, j))$, and let u_t be the last vertex of $L^r_j(u_z)$. Let P_1 be a normal antipath of $G^{r-1}_{u_z}(u_p, i, j)$ with right endpoint a vertex u_x such that $u_x \in H_\ell$, $f(u_t) \leq \ell \leq j - 1$, and $u_t u_x \notin E(G)$. Let P_2 be a normal antipath of $G^{r-1}_{u_z}(u_x, \ell+1, j)$ with right endpoint a vertex u_y such that $u_y \in H_h$, $\ell+1 \leq h \leq j$, and $V(P_1) \cap V(P_2) = \emptyset$. Then, $P = (P_1, u_t, P_2)$ is a normal antipath of $G^r_{u_z}(u_p, i, j)$ with right endpoint the vertex u_y.*

Proof. Let P_1 be a normal antipath of $G^{r-1}_{u_z}(u_p, i, j)$ with right endpoint a vertex u_x such that $u_x \in H_\ell$, $f(u_t) \leq \ell \leq j - 1$, and $u_t u_x \notin E(G)$, and let P_2 be a normal antipath of $G^{r-1}_{u_z}(u_x, \ell + 1, j)$ with right endpoint a vertex u_y such that $u_y \in H_h$, $\ell + 1 \leq h \leq j$, and $V(P_1) \cap V(P_2) = \emptyset$. Since $u_t u_x \notin E(G)$, $u_x <_\sigma u_s \leq_\sigma u_t$ and $u_s u_x \in E(G)$ for every vertex $u_s \in V(P_2)$, it follows that $u_t u_s \notin E(G)$ for every vertex $u_s \in V(P_2)$. Thus, the first vertex of P_2 is an antineightbor of u_t. Therefore, since $V(P_1) \cap V(P_2) = \emptyset$, it follows that $P = (P_1, u_t, P_2)$ is an antipath of G. Additionally, since $u_p <_\sigma u_x <_\sigma u_s$, $u_p u_x \in E(G)$, and $u_x u_s \in E(G)$ for every vertex $u_s \in V(G^{r-1}_{u_z}(u_x, \ell + 1, j))$, from the transitivity property we obtain that $u_p u_s \in E(G)$, for every vertex $u_s \in V(P_2)$; thus, for every vertex $u_s \in V(P_2)$, we obtain $u_s \in V(G^{r-1}_{u_z}(u_p, i, j))$. Therefore, since $G^{r-1}_{u_z}(u_p, i, j)$ and $G^{r-1}_{u_z}(u_x, \ell + 1, j)$ are induced subgraphs of $G^r_{u_z}(u_p, i, j)$, it follows that P is a antipath of $G^r_{u_z}(u_p, i, j)$. Hereafter, in the rest of this proof $P_1 = (v_1, v_2, \ldots, v_{q-1})$, $P_2 = (v_{q+1}, v_{q+2}, \ldots, v_s)$, $u_x = v_{q-1}$, $u_y = v_s$, and $u_t = v_q$.

We first show that $P = (v_1, v_2, \ldots, v_q, \ldots, v_s)$ is a normal antipath. Since v_1 is a leftmost vertex of $V(P_1)$ in σ, it follows that $v_1 \leq_\sigma u_x$. Furthermore, since for every vertex $v_k \in V(P_2)$ it holds $u_x <_\sigma v_k$, it follows that v_1 is a leftmost vertex of $V(P)$ in σ. We next show that for every k, $2 \leq k \leq s$, the vertex v_k is a leftmost vertex of $N_{\overline{G}}(v_{k-1}) \cap \{v_k, v_{k+1}, \ldots, v_s\}$ in σ.

Consider first the case where $2 \leq k \leq q - 1$, i.e., $v_k \in V(P_1)$. Since P_1 is a normal antipath, it follows that v_k is a leftmost vertex of $N_{\overline{G}}(v_{k-1}) \cap \{v_k, v_{k+1}, \ldots, v_{q-1}\}$ in σ. Consider first the case where $v_k \leq_\sigma u_x$. Since $u_x <_\sigma v_{k'}$ for every vertex $v_{k'}$, $q \leq k' \leq s$, it follows that $v_k <_\sigma v_{k'}$. Therefore, in the case where $v_k \leq_\sigma u_x$, we obtain that v_k is also a leftmost vertex of $N_{\overline{G}}(v_{k-1}) \cap \{v_k, v_{k+1}, \ldots, v_s\}$ in σ. Consider now the case where $u_x <_\sigma v_k$. Since v_q is a rightmost vertex of $V(P)$ is σ, it follows that v_k is a leftmost vertex of $N_{\overline{G}}(v_{k-1}) \cap \{v_k, v_{k+1}, \ldots, v_{q-1}, v_q\}$ in σ. Now, since $u_x <_\sigma v_k$, and v_k is the next vertex of v_{k-1} in P_1, it follows that $v_{k-1} u_x \in E(G)$. Also, since P_1 is normal, $v_{k-1} u_x \in E(G)$, and v_{k-1} appears before u_x in P_1, from Lemma 5.4 it follows that $v_{k-1} <_\sigma u_x$. Now, since $v_{k-1} <_\sigma u_x <_\sigma v_{k'}$ for every vertex $v_{k'} \in V(P_2)$, $v_{k-1} u_x \in E(G)$, and $u_x v_{k'} \in E(G)$, from the transitivity property it follows that $v_{k-1} v_{k'} \in E(G)$. Thus, for every vertex $v_{k'}$ of P_2, it follows that $v_{k-1} v_{k'} \in E(G)$. Therefore, in the case where $u_x <_\sigma v_k$, we obtain again that v_k is a leftmost vertex of $N_{\overline{G}}(v_{k-1}) \cap \{v_k, v_{k+1}, \ldots, v_s\}$ in σ. Therefore, in the case where $2 \leq k \leq q - 1$, we have proved that v_k is a leftmost vertex of $N_{\overline{G}}(v_{k-1}) \cap \{v_k, v_{k+1}, \ldots, v_s\}$ in σ.

Consider now the case where $k = q$. Since P_1 is a normal antipath, and for every vertex $v_{k'} \in V(P_2)$ we have that $v_{k'} \in V(G_{u_z}^{r-1}(u_x, \ell+1, j))$, it follows that $v_{k'}u_x \in E(G)$. Therefore, v_q is the only antineighbor of v_{q-1} in $\{v_q, v_{q+1}, \ldots, v_s\}$ and, thus, v_q is a leftmost vertex of $N_{\overline{G}}(v_{q-1}) \cap \{v_q, v_{q+1}, \ldots, v_s\}$ in σ. Now, in the case where $k = q+1$, we have that v_{q+1} is a leftmost vertex of $V(P_2) = \{v_{q+1}, v_{q+2}, \ldots, v_s\}$ in σ, since P_2 is a normal antipath. Therefore, it easily follows that v_{q+1} is a leftmost vertex of $N_{\overline{G}}(v_q) \cap \{v_{q+1}, v_{q+2}, \ldots, v_s\}$ in σ. Finally, in the case where $q+2 \leq k \leq s$, since P_2 is a normal antipath it directly follows that v_k is a leftmost vertex of $N_{\overline{G}}(v_{k-1}) \cap \{v_k, v_{k+1}, \ldots, v_s\}$ in σ. ∎

We next prove the correctness of Algorithm LP_Cocomparability.

Lemma 5.9. *Let G be a comparability graph, given by its Hasse diagram with layers $H_0, H_1, H_2, \ldots, H_k$, and let $\sigma = (u_0, u_1, u_2, \ldots, u_n)$ be the layered ordering of G. For every induced subgraph $G(u_p, i, j)$ of G, and for every vertex $u_y \in V(G(u_p, i, j))$, Algorithm LP_Cocomparability computes the length $\ell(u_y; G(u_p, i, j))$ of a longest normal antipath of $G(u_p, i, j)$ with right endpoint the vertex u_y and, also, the corresponding antipath $P(u_y; G(u_p, i, j))$.*

Proof. The proof of the lemma for every subgraph $G(u_p, i, j)$, $1 \leq i \leq j \leq k$, will be done by induction on the index j, $1 \leq j \leq k$.

We first prove the lemma for $j = 1$, i.e., for the subgraph $G(u_p, 1, 1)$, where $u_p = u_0$ in this case. Let L_1 be an ordering of the set $H_1 \cap V(G(u_p, 1, 1))$. It is easy to see that the length $\ell(u_z; G(u_p, 1, 1))$, of a longest normal antipath of $G(u_p, 1, 1)$ with right endpoint a vertex $u_z \in L_1$, equals to $|L_1|$. Let us now compare this value to the value computed by Algorithm LP_Cocomparability. Firstly, since in this case $i = j$, it easy to see that for every graph $G_{u_z}^r(u_p, 1, 1)$, $1 \leq r \leq |L_1|$, Algorithm LP_Cocomparability correctly computes and sets $\ell(u_t; G_{u_z}^r(u_p, 1, 1)) = |L_1^r(u_z)|$ and $P(u_t; G_{u_z}^r(u_p, 1, 1)) = L_1^r(u_z)$, where u_t is the last vertex of $L_1^r(u_z)$ (lines 15-17). Finally, for $r = |L_1|$, it is easy to see that $\ell(u_t; G_{u_z}^{|L_1|}(u_p, 1, 1)) = |L_1(u_z)|$ and $P(u_t; G_{u_z}^{|L_1|}(u_p, 1, 1)) = L_1(u_z)$, and Algorithm LP_Cocomparability sets $\ell(u_z; G(u_p, 1, 1)) = \ell(u_z; G_{u_z}^{|L_1|}(u_p, 1, 1))$ and $P(u_z; G(u_p, 1, 1)) = P(u_z; G_{u_z}^{|L_1|}(u_p, 1, 1))$, for every vertex u_z of L_1 (lines 23-24). Therefore, the lemma holds for every subgraph $G_{u_z}^r(u_p, 1, 1)$, $1 \leq r \leq |L_1|$. This proves the induction basis.

Assume now that the lemma holds for every index j', $1 \leq j' \leq j-1 \leq k-1$. That is, assume that for every induced subgraph $G(u_p, i', j')$ of G, $1 \leq i' \leq j' \leq j-1 \leq k-1$, and for every vertex $u_y \in V(G(u_p, i', j'))$, Algorithm LP_Cocomparability computes the length $\ell(u_y; G(u_p, i', j'))$ of a longest normal antipath of $G(u_p, i, j')$ with right endpoint the vertex u_y and, also, the corresponding antipath $P(u_y; G(u_p, i', j'))$.

We will next show that the lemma holds for $j' = j$, $1 \leq i < j \leq k$, i.e., for every induced subgraph $G(u_p, i, j)$ of G.

Consider first the case where $1 \leq i = j \leq k$. Let L_j be an ordering of the set $H_j \cap V(G(u_p, j, j))$. It is easy to see that the length $\ell(u_z; G(u_p, j, j))$, of a longest normal antipath of $G(u_p, j, j)$ with right endpoint a vertex $u_z \in L_j$, equals to $|L_j|$. Let us now compare this value to the value computed by Algorithm LP_Cocomparability. Let u_t be the last vertex of $L_j^r(u_z)$. We first show that for every graph $G_{u_z}^r(u_p, j, j)$, $1 \leq r \leq |L_j|$, Algorithm LP_Cocomparability correctly computes the values $\ell(u_t; G_{u_z}^r(u_p, j, j))$ and $P(u_t; G_{u_z}^r(u_p, j, j))$. It is easy to see that the length $\ell(u_t; G_{u_z}^r(u_p, j, j))$, of a longest normal antipath of $G_{u_z}^r(u_p, j, j)$ which has u_t as its right endpoint, equals to $|L_j^r(u_z)|$. In the case where $i = j$, Algorithm LP_Cocomparability correctly computes and sets $\ell(u_t; G_{u_z}^r(u_p, j, j)) = |L_j^r(u_z)|$ and $P(u_t; G_{u_z}^r(u_p, j, j)) = L_j^r(u_z)$ (lines 15-17); note that for $r = |L_j|$, we have $|L_j^r(u_z)| = |L_j|$. Since Algorithm LP_Cocomparability computes these values for every vertex $u_z \in L_j$, i.e., for every subgraph $G_{u_z}^r(u_p, j, j))$, and since $G(u_p, j, j) = G_{u_z}^{|L_j|}(u_p, j, j)$, for any vertex $u_z \in L_j$, it follows that Algorithm LP_Cocomparability correctly computes and sets $\ell(u_z, G(u_p, j, j)) = \ell(u_z, G_{u_z}^{|L_j|}(u_p, j, j))$ and $P(u_z, G(u_p, j, j)) = P(u_z, G_{u_z}^{|L_j|}(u_p, j, j))$ (lines 23-24), for every vertex $u_z \in L_j$. Thus, the lemma holds for every subgraph $G(u_p, i, j)$ of G such that $1 \leq i = j \leq k$.

Consider now the case where $1 \leq i < j \leq k$. To prove that the lemma holds in this case, we use the following claim.

Claim 5.1. *For every induced subgraph $G^r_{u_z}(u_p, i, j)$ of G, $1 \leq i < j \leq k$, and for every vertex $u_y \in V(G^r_{u_z}(u_p, i, j))$ such that $u_y \notin L_j \setminus \{u_t\}$, where u_t be the last vertex of $L^r_j(u_z)$, Algorithm LP_Cocomparability correctly computes $\ell(u_y; G^r_{u_z}(u_p, i, j))$ and $P(u_y; G^r_{u_z}(u_p, i, j))$.*

Recall that $G(u_p, i, j) = G^{|L_j|}_{u_z}(u_p, i, j)$ for any vertex $u_z \in L_j$. Then, for the length of a longest normal antipath of $G(u_p, i, j)$ with right endpoint a vertex $u_y \in V(G(u_p, i, j))$ such that $u_y \notin L_j$, from Claim 5.1 we obtain that $\ell(u_y; G(u_p, i, j)) = \ell(u_y; G^{|L_j|}_{u_z}(u_p, i, j))$, where u_z is any vertex of L_j. It is easy to see that Algorithm LP_Cocomparability sets $\ell(u_y; G(u_p, i, j)) = \ell(u_y; G^{|L_j|}_{u_z}(u_p, i, j))$ and $P(u_y; G(u_p, i, j)) = P(u_y; G^{|L_j|}_{u_z}(u_p, i, j))$, where u_z is any vertex of L_j, for every vertex u_y of $G(u_p, i, j)$ such that $u_y \notin L_j$ (lines 26-28). Also, for the length of a longest normal antipath of $G(u_p, i, j)$ with right endpoint a vertex $u_z \in L_j$, from Claim 5.1 we obtain that $\ell(u_z; G(u_p, i, j)) = \ell(u_z; G^{|L_j|}_{u_z}(u_p, i, j))$. Since the procedure `process()` is executed for every vertex $u_z \in L_j$, i.e., for every subgraph $G^r_{u_z}(u_p, i, j))$, it follows that Algorithm LP_Cocomparability correctly computes and sets $\ell(u_z; G(u_p, i, j)) = \ell(u_z; G^{|L_j|}_{u_z}(u_p, i, j))$ and $P(u_z; G(u_p, i, j)) = P(u_z; G^{|L_j|}_{u_z}(u_p, i, j))$ for every vertex $u_z \in L_j$ (lines 23-24). It is now clear that Algorithm LP_Cocomparability correctly computes the length of a longest normal antipath of $G(u_p, i, j)$ with right endpoint a vertex u_y, for every vertex $u_y \in V(G(u_p, i, j))$. This proves the lemma. ∎

Proof of Claim 5.1. For proving the claim we use the induction hypothesis of Lemma 5.9. That is, we assume that for every induced subgraph $G(u_p, i', j')$ of G, $1 \leq i' \leq j' \leq j-1 \leq k-1$, and for every vertex $u_y \in V(G(u_p, i', j'))$, Algorithm LP_Cocomparability correctly computes the length $\ell(u_y; G(u_p, i', j'))$ of a longest normal antipath of $G(u_p, i', j')$ with right endpoint the vertex u_y and, also, the corresponding antipath $P(u_y; G(u_p, i', j'))$.

Let $G^r_{u_z}(u_p, i, j)$ be an induced subgraph of G such that $1 \leq i < j \leq k$. We prove the claim by induction on j, $2 \leq j \leq k$, i.e., given a specific index j, we prove that the claim holds for every induced subgraph $G^r_{u_z}(u_p, i, j)$ of G, where $1 \leq i < j \leq k$ and $0 \leq r \leq |L_j|$, and for every vertex $u_y \in V(G^r_{u_z}(u_p, i, j))$ such that $u_y \notin L_j \setminus \{u_t\}$, where u_t is the last vertex of $L^r_j(u_z)$. To this end, we distinguish three cases on the position of the vertex u_y in the ordering σ: $u_y \in H_\ell$ where $\ell \leq f(u_t)$, $u_y \in H_\ell$ where $f(u_t) + 1 \leq \ell \leq j - 1$, and $u_y = u_t$. In each of these cases, we examine first the length of a longest normal antipath of $G^r_{u_z}(u_p, i, j)$ with right endpoint the vertex u_y and, then, we compare this value to the length of the antipath computed by Algorithm LP_Cocomparability. The proof is done by induction on the index r, $0 \leq r \leq |L_j|$.

Consider first the case where $r = 0$, i.e., $L^0_j = \emptyset$. Since here we examine the case where $i \neq j$, from Definition 5.4 we obtain that $G^0_{u_z}(u_p, i, j) = G(u_p, i, j-1)$. Therefore, it is easy to see that for every subgraph $G^0_{u_z}(u_p, i, j)$, and for every vertex $u_y \in V(G^0_{u_z}(u_p, i, j))$, the length $\ell(u_y; G^0_{u_z}(u_p, i, j))$ equals to $\ell(u_y; G(u_p, i, j-1))$. It is easy to see that Algorithm LP_Cocomparability sets $\ell(u_y; G^0_{u_z}(u_p, i, j)) = \ell(u_y; G(u_p, i, j-1))$ and $P(u_y; G^0_{u_z}(u_p, i, j)) = P(u_y; G(u_p, i, j-1))$, for every vertex $u_y \in V(G^0_{u_z}(u_p, i, j))$ (lines 8-11). Since by the induction hypothesis of Lemma 5.9, Algorithm LP_Cocomparability correctly computes the values of $\ell(u_y; G(u_p, i, j-1))$ and $P(u_y; G(u_p, i, j-1))$, it follows that the algorithm also correctly computes the values of $\ell(u_y; G^0_{u_z}(u_p, i, j))$ and $P(u_y; G^0_{u_z}(u_p, i, j))$. Therefore, the claim holds for $r = 0$.

Suppose now that the claim holds for every index ℓ, $0 \leq \ell \leq r-1 \leq |L_j| - 1$. We will now prove that the claim holds for the subgraph $G^r_{u_z}(u_p, i, j)$ of G, $1 \leq r \leq |L_j|$.

Case 1. For every vertex u_y of $G^r_{u_z}(u_p, i, j)$ such that $u_y \in H_\ell$ and $i \leq \ell \leq f(u_t)$, it is easy to see that $\ell(u_y; G^r_{u_z}(u_p, i, j)) = \ell(u_y; G^{r-1}_{u_z}(u_p, i, j))$, since u_t does not belong to any normal antipath with right endpoint a vertex $u_y \in H_\ell$, $i \leq \ell \leq f(u_t)$. In this case, Algorithm LP_Cocomparability computes and sets $\ell(u_y; G^r_{u_z}(u_p, i, j)) = \ell(u_y; G^{r-1}_{u_z}(u_p, i, j))$ for the length of a longest normal antipath of $G^r_{u_z}(u_p, i, j)$ with right endpoint a vertex $u_y \in H_\ell$, $i \leq \ell \leq f(u_t)$; the algorithm also computes the corresponding

antipath. This computation is done during the initialization (lines 8-14), and these values do not change during the `process()` of the algorithm, since $u_y \in H_\ell$ and $\ell < f(u_t) + 1$. Since by the induction hypothesis Algorithm LP_Cocomparability correctly computes the value of $\ell(u_y; G^{r-1}_{u_z}(u_p, i, j)$, for every vertex $u_y \in G^{r-1}_{u_z}(u_p, i, j)$ such that $u_y \notin L_j$, it follows that Algorithm LP_Cocomparability correctly computes the values of $\ell(u_y; G^r_{u_z}(u_p, i, j))$ and $P(u_y; G^r_{u_z}(u_p, i, j))$.

Case 2. We consider next the case where $u_y \in H_h$ and $f(u_t) + 1 \leq h \leq j - 1$. Let $P = (u_{x'}, \ldots, u_y)$ be a longest normal antipath of $G^r_{u_z}(u_p, i, j)$ with right endpoint the vertex u_y.

(I) Consider first the case where P contains the vertex u_t. Assume first that $P = (u_t, u_y)$ is a longest normal antipath of $G^r_{u_z}(u_p, i, j)$ with right endpoint the vertex u_y. Since $u_y \in H_h$, $f(u_t) + 1 \leq h \leq j - 1$, it follows that there exists at least one vertex $u_x \in H_{f(u_t)}$ such that $u_x u_t \notin E(G)$. Thus, $P' = (u_x, u_t, u_y)$ is a normal antipath of $G^r_{u_z}(u_p, i, j)$ with right endpoint the vertex u_y which is longer than P. This is a contradiction to our assumption on P.

Assume now that $P = (u_{x'}, \ldots, u_x, u_t, u_{y'}, \ldots, u_y) = (P_1, u_t, P_2)$ is a longest normal antipath of $G^r_{u_z}(u_p, i, j)$ with right endpoint the vertex u_y. From Lemma 5.7, we obtain that $P_1 = (u_{x'}, \ldots, u_x)$ and $P_2 = (u_{y'}, \ldots, u_y)$ are normal antipaths of $G^r_{u_z}(u_p, i, j)$, and in fact of $G^{r-1}_{u_z}(u_p, i, j)$.

We will now show that $u_x <_\sigma u_s$ and $u_x u_s \in E(G)$, for every vertex $u_s \in V(P_2)$, where u_x is the right endpoint of P_1. Since $u_t \in L_j$ and P is an antipath of $G^r_{u_z}(u_p, i, j)$, it follows that $u_s \leq_\sigma u_t$, for every vertex $u_s \in V(P_2)$. Consider first the case where u_s is a vertex of P_2 such that $u_s <_\sigma u_t$. Since P is normal and u_t is the next vertex of u_x in P, it follows that $u_x u_s \in E(G)$ for every vertex $u_s \in V(P_2)$ such that $u_s <_\sigma u_t$. Since P is normal, $u_x u_s \in E(G)$, and u_x appears before u_s in P, from Lemma 5.4 we obtain that $u_x <_\sigma u_s$, for every vertex $u_s \in V(P_2)$ such that $u_s <_\sigma u_t$. Therefore, we have proved that for every vertex $u_s \in V(P_2)$ such that $u_s <_\sigma u_t$, we have $u_x <_\sigma u_s$ and $u_x u_s \in E(G)$.

Consider now the case where u_s is a vertex of P_2 such that $u_s =_\sigma u_t$. Since u_y is a vertex of P_2 such that $u_y <_\sigma u_t$, from the above we obtain that $u_x <_\sigma u_y$. Since $u_x <_\sigma u_y <_\sigma u_t =_\sigma u_s$, it follows that $u_x <_\sigma u_s$, for every vertex $u_s \in V(P_2)$ such that $u_s =_\sigma u_t$. It is left to show that the property $u_x u_s \in E(G)$ for every vertex $u_s \in V(P_2)$ such that $u_s =_\sigma u_t$ holds. Assume that P is an antipath for which this property does not hold. We next show that there exists a longest normal antipath P' of $G^r_{u_z}(u_p, i, j)$ with right endpoint the vertex u_y, such that $P' = (P_1, P_2')$ and $V(P') = V(P)$, for which this property holds.

Assume now that there exists a vertex $u_s \in V(P_2)$, such that $u_s =_\sigma u_t$ and $u_x u_s \notin E(G)$. Let $P = (P_1, u_t, P_2) = (P_1, u_t, u_{y'} \ldots, u_{s'}, u_s, u_{s''}, \ldots, u_y)$, and let u_s be the last such vertex in P. Then $P' = (P_1, P_2') = (P_1, u_s, u_{y'} \ldots, u_{s'}, u_t, u_{s''}, \ldots, u_y)$ is an antipath, since we next prove that u_t and u_s have an antiedge with every vertex of P_2. To this end, let u_q be a vertex of P_2 such that $q \neq s$. If $u_q =_\sigma u_t$, then indeed $u_q u_t \notin E(G)$ and $u_q u_s \notin E(G)$. If $u_q <_\sigma u_t$, then from the above we have proved that $u_x <_\sigma u_q$ and $u_x u_q \in E(G)$. Since $u_x <_\sigma u_q <_\sigma u_t$, $u_x u_q \in E(G)$, and $u_x u_t \notin E(G)$, from the transitivity property we obtain $u_q u_t \notin E(G)$; using the same arguments we obtain that $u_q u_s \notin E(G)$. Therefore, since $u_{y'}, u_{s'}, u_{s''} \in V(P_2)$, we obtain that $P' = (P_1, u_s, u_{y'} \ldots, u_{s'}, u_t, u_{s''}, \ldots, u_y)$ is a longest antipath of $G^r_{u_z}(u_p, i, j)$ with right endpoint the vertex u_y. It is easy to see that P' is normal, since P is normal and $u_t =_\sigma u_s$. By repeating the above procedure we can obtain a longest normal antipath $P' = (P_1', u_t, P_2')$ with right endpoint the vertex u_y such that $u_x u_s \in E(G)$ for every vertex $u_s \in V(P_2')$ such that $u_s =_\sigma u_t$, where u_x is the last vertex of P_1'. Therefore, we may assume without loss of generality that $P = (u_{x'}, \ldots, u_x, u_t, u_{y'}, \ldots, u_y) = (P_1, u_t, P_2)$ is a longest normal antipath of $G^r_{u_z}(u_p, i, j)$ with right endpoint the vertex u_y, with the property that $u_x u_s \in E(G)$ for every vertex $u_s \in V(P_2)$ such that $u_s =_\sigma u_t$.

Therefore, we have proved that $u_x <_\sigma u_s$ and $u_x u_s \in E(G)$, for every vertex $u_s \in V(P_2)$. Since $u_x <_\sigma u_y$ and by assumption $u_y \in H_h$, $f(u_t) + 1 \leq h \leq j - 1$, we obtain that $u_x \in H_\ell$, where $f(u_t) \leq \ell \leq j - 2$. Then, for every vertex $u_s \in V(P_2)$ we obtain that $u_s \in H_h$, where $\ell + 1 \leq h \leq j$. Additionally, since we have shown that $u_x u_s \in E(G)$ for every vertex $u_s \in V(P_2)$, from Definition 5.4 we obtain that $u_s \in V(G^{r-1}_{u_z}(u_x, \ell + 1, j))$ for every vertex $u_s \in V(P_2)$. Note that, every vertex u_s of $G^{r-1}_{u_z}(u_x, \ell+1, j)$ is also a vertex of $G^{r-1}_{u_z}(u_p, i, j)$. Indeed, since $i < \ell+1 \leq j$, and since $u_p <_\sigma u_x <_\sigma u_s$,

$u_pu_x \in E(G)$, and $u_xu_s \in E(G)$, from the transitivity property we obtain that $u_pu_s \in E(G)$. Let H_2 be the subgraph of $G^r_{u_z}(u_p, i, j)$ induced by $V(H_2) = V(G^{r-1}_{u_z}(u_x, \ell+1, j))$. Therefore, we have shown that every vertex of P_2 belongs to H_2.

By assumption, for every vertex $u_q \in V(P_1)$ we have $u_q \in V(G^{r-1}_{u_z}(u_p, i, j))$. Let now u_q be any vertex of P_1. If $u_q \leq_\sigma u_x$, then $u_q \in H_d$ and $d \leq \ell$; thus, from Definition 5.4 we obtain that $u_q \notin V(H_2) = V(G^{r-1}_{u_z}(u_x, \ell+1, j))$. Consider now the case where u_q is a vertex of P_1 such that $u_x <_\sigma u_q$. Since P_1 is a normal antipath, $u_x <_\sigma u_q$, and u_q appears before u_x in P_1, from Lemma 5.4 we obtain that $u_xu_q \notin E(G)$. Therefore, from Definition 5.4 we obtain again that $u_q \notin V(H_2) = V(G^{r-1}_{u_z}(u_x, \ell+1, j))$. Therefore, we have proved that no vertex of P_1 can belong to H_2. Let H_1 be the subgraph of $G^r_{u_z}(u_p, i, j)$ induced by $V(H_1) = V(G^{r-1}_{u_z}(u_p, i, j)) \setminus V(G^{r-1}_{u_z}(u_x, \ell+1, j))$. Thus, we have shown that every vertex of P_1 belongs to H_1. Therefore, we have shown that $V(P_1) \subseteq V(H_1)$, $V(P_2) \subseteq V(H_2)$, and $V(H_1) \cap V(H_2) = \emptyset$. It is easy to see that $V(P_1) \cap V(P_2) = \emptyset$.

Since $P = (P_1, u_t, P_2)$ is a longest normal antipath of $G^r_{u_z}(u_p, i, j)$ with right endpoint the vertex u_y, and since the antipaths P_1 and P_2 belong to two disjoint induced subgraphs of $G^r_{u_z}(u_p, i, j)$, it follows that P_1 is a longest normal antipath of H_1 with right endpoint the vertex u_x, and that P_2 is a longest normal antipath of H_2 with right endpoint the vertex u_y. Thus, since $H_2 = G^{r-1}_{u_z}(u_x, \ell+1, j)$, we obtain that $|P_2| = \ell(u_y; G^{r-1}_{u_z}(u_x, \ell+1, j))$. We will now show that $|P_1| = \ell(u_x; G^{r-1}_{u_z}(u_p, i, j))$. To this end, let P_0 be a longest normal antipath of $G^{r-1}_{u_z}(u_p, i, j)$ with right endpoint the vertex u_x. Assume that there exists a vertex $u_s \in V(P_0)$ such that $u_s \in V(H_2) = V(G^{r-1}_{u_z}(u_x, \ell+1, j))$. Since $u_x \in H_\ell$, then $u_x <_\sigma u_s$ and $u_xu_s \in E(G)$. Since P_0 is normal, from Lemma 5.4 we obtain that u_x appears before u_s in P_0. This comes to a contradiciton to our assumption that u_x is the right endpoint of P_0. Thus, no vertex of P_0 belongs to H_2. Thus, $V(P_0) \subseteq V(H_1)$, and since P_1 is a longest normal antipath of H_1 with right endpoint the vertex u_x, we obtain that $|P_0| \subseteq |P_1|$. Additionally, since H_1 is an induced subgraph of $G^{r-1}_{u_z}(u_p, i, j)$, we obtain that $|P_1| \subseteq |P_0|$. Thus, $|P_0| = |P_1|$ and, therefore, P_1 is a longest normal antipath of $G^{r-1}_{u_z}(u_p, i, j)$ with right endpoint the vertex u_x. Thus, $|P_1| = \ell(u_x; G^{r-1}_{u_z}(u_p, i, j))$.

Therefore, for a longest normal antipath $P = (P_1, u_t, P_2)$ of $G^r_{u_z}(u_p, i, j)$ with right endpoint a vertex $u_y \in H_h$, $f(u_t) + 1 \leq h \leq j-1$, we have shown that $|P| = \ell(u_y; G^r_{u_z}(u_p, i, j)) = \ell(u_x; G^{r-1}_{u_z}(u_p, i, j)) + \ell(u_y; G^{r-1}_{u_z}(u_x, \ell+1, j)) + 1$ and, also, $P = P(u_y; G^r_{u_z}(u_p, i, j)) = (P(u_x; G^{r-1}_{u_z}(u_p, i, j)), u_t, P(u_y; G^{r-1}_{u_z}(u_x, \ell+1, j)))$.

Hereafter, we examine the results computed by Algorithm LP_Cocomparability in Case 2. Let P' be the antipath of $G^r_{u_z}(u_p, i, j)$ with right endpoint a vertex u_y computed by Algorithm LP_Cocomparability, in the case where $u_y \in H_h$, $f(u_t) + 1 \leq h \leq j-1$. Note that the antipath P' which is constructed by the algorithm with the `procedure(bridge)` contains the vertex u_t. Algorithm LP_Cocomparability computes the length of $P' = (P'_1, u_t, P'_2)$, where u_t is the last vertex of $L^r_j(u_z)$, for two antipaths P'_1 and P'_2 as follows. The antipath $P'_1 = P(u_x; G^{r-1}_{u_z}(u_p, i, j))$ is a longest normal antipath of $G^{r-1}_{u_z}(u_p, i, j)$ with right endpoint a vertex u_x such that $u_x \in H_\ell$, $f(u_t) \leq \ell \leq j-2$, and $u_xu_t \notin E(G)$. The antipath $P'_2 = P(u_y; G^{r-1}_{u_z}(u_x, \ell+1, j))$ is a longest normal antipath of $G^{r-1}_{u_z}(u_x, \ell+1, j)$ with right endpoint a vertex u_y such that $u_y \in H_h$, $\ell + 1 \leq h \leq j-1$. Actually, in this case, Algorithm LP_Cocomparability computes with the `procedure(bridge)` the value $w_1 + w_2 + 1 = |P'_1| + |P'_2| + 1$, for every vertex u_x such that $u_x \in H_\ell$, $f(u_t) \leq \ell \leq j-2$, and $u_xu_t \notin E(G)$, and sets $|P'|$ to be equal to the maximum among these values. Also, Algorithm LP_on_H computes the corresponding antipath $P = (P'_1, u_t, P'_2)$.

By the induction hypothesis, Algorithm LP_Cocomparability has correctly computed the values $P'_1 = P(u_x; G^{r-1}_{u_z}(u_p, i, j))$ and $P'_2 = P(u_y; G^{r-1}_{u_z}(u_x, \ell+1, j))$. Since, by the induction hypothesis, the computed antipaths P'_1 and P'_2 are normal antipaths of $G^r_{u_z}(u_p, i, j)$ with right endpoints the vertices u_x and u_y, respectively, it follows similarly to the above that P'_1 belongs to the graph H_1 and P'_2 belongs to the graph H_2, where $V(H_1) = V(G^{r-1}_{u_z}(u_p, i, j)) \setminus V(G^{r-1}_{u_z}(u_x, \ell+1, j))$, $V(H_2) = V(G^{r-1}_{u_z}(u_x, \ell+1, j))$, and u_x is the right endpoint of P'_1. Since $V(H_1) \cap V(H_2) = \emptyset$, it follows that $V(P'_1) \cap V(P'_2) = \emptyset$. Then, from Lemma 5.8 we obtain that the antipath $P' = (P'_1, u_t, P'_2)$ computed by Algorithm LP_Cocomparability is a normal antipath of $G^r_{u_z}(u_p, i, j)$ with right endpoint the vertex u_y. Moreover, since Algorithm LP_Cocomparability computes with the `procedure(bridge)`

the value $\ell(u_x; G^{r-1}_{u_z}(u_p,i,j)) + \ell(u_y; G^{r-1}_{u_z}(u_x, \ell+1, j)) + 1$, for every vertex u_x such that $u_x \in H_\ell$, $f(u_t) \leq \ell \leq j-2$, and $u_x u_t \notin E(G)$, and sets $\ell(u_y; G^r_{u_z}(u_p,i,j))$ to be equal to the maximum among these values, it follows that the value $\ell(u_y; G^r_{u_z}(u_p,i,j))$ computed by the algorithm equals to the length of a longest normal antipath of $G^r_{u_z}(u_p,i,j)$ with right endpoint the vertex u_y. Also, Algorithm LP_Cocomparability also correctly computes the corresponding antipath $P(u_y; G^r_{u_z}(u_p,i,j) = (P(u_x; G^{r-1}_{u_z}(u_p,i,j)), u_t, P(u_y; G^{r-1}_{u_z}(u_x, \ell+1, j)))$.

(II) Consider now the case where the longest normal antipath P of $G^r_{u_z}(u_p,i,j)$ with right endpoint the vertex u_y does not contain the vertex u_t. Then, $V(P) \subseteq V(G^{r-1}_{u_z}(u_p,i,j))$, and it easily follows that P is a longest normal antipath of $G^{r-1}_{u_z}(u_p,i,j)$ with rigth endpoint the vertex u_y. By the induction hypothesis, Algorithm LP_Cocomparability correctly computes the value $\ell(u_y; G^{r-1}_{u_z}(u_p,i,j))$, for every vertex $u_y \in G^{r-1}_{u_z}(u_p,i,j)$ such that $u_y \notin L_j \setminus \{u_t\}$. During the initialization (lines 8-14) the algorithm sets $\ell(u_y; G^r_{u_z}(u_p,i,j)) = \ell(u_y; G^{r-1}_{u_z}(u_p,i,j))$, for every vertex $u_y \in H_h$, $f(u_t)+1 \leq h \leq j-1$.

From Lemma 5.8 (since we have shown above that $V(P'_1) \cap V(P'_2) = \emptyset$), we obtain that during the execution of the `procedure(bridge)` the antipaths constructed by Algorithm LP_Cocomparability are normal antipaths of $G^r_{u_z}(u_p,i,j)$ with right endpoint a vertex u_y. Therefore, since we have assumed that P is a longest normal antipath of $G^r_{u_z}(u_p,i,j)$ with right endpoint the vertex u_y, it follows that no antipath with right endpoint the vertex u_y which is constructed with the `procedure(bridge)` is longer than P. Thus, since $|P|$ is the initial value given to $\ell(u_y; G^r_{u_z}(u_p,i,j))$, it follows that the statement $w_1 + w_2 + 1 > \ell(u_y; G^r_{u_z}(u_p,i,j))$ (in the `procedure(bridge)`) is false for every vertex $u_x \in H_\ell$ such that $f(u_t) \leq \ell \leq h-1$ and $u_t u_x \notin E(G)$. Therefore, the initial value of $\ell(u_y; G^r_{u_z}(u_p,i,j))$ is not changed during the execution of the `process()`. Therefore, Algorithm LP_Cocomparability correctly computes and sets $\ell(u_y; G^r_{u_z}(u_p,i,j)) = \ell(u_y; G^{r-1}_{u_z}(u_p,i,j))$ for the vertex $u_y \in H_h$, $f(u_t)+1 \leq h \leq j-1$.

Concluding, in both Cases 2(I) and 2(II), we have proved that the antipath P' computed by Algorithm LP_Cocomparability is a longest normal antipath $P(u_y; G^r_{u_z}(u_p,i,j))$ of $G^r_{u_z}(u_p,i,j)$ with u_y as its right endpoint, and $|P'| = \ell(u_y; G^r_{u_z}(u_p,i,j))$. Thus, the claim holds in Case 2.

Case 3. Consider now the case where $u_y = u_t$. Assume first that u_t has no antineighbors in $G^r_{u_z}(u_p,i,j)$. Then (u_t) is a longest normal antipath of $G^r_{u_z}(u_p,i,j)$ with right endpoint the vertex u_t. Since we examine the case where $i \neq j$, it is easy to see that Algorithm LP_Cocomparability sets $\ell(u_t; G^r_{u_z}(u_p,i,j)) = 1$ and $P(u_t; G^r_{u_z}(u_p,i,j)) = (u_t)$ (lines 19-20). Since u_t has no antineighbors in $G^r_{u_z}(u_p,i,j)$, it follows that $r = 1$ and $f(u_t) = j$. Thus, the initial value of $\ell(u_t; G^r_{u_z}(u_p,i,j))$ is not changed during the execution of the `process()`. Therefore, Algorithm LP_Cocomparability correctly computes the values of $\ell(u_t; G^r_{u_z}(u_p,i,j))$ and $P(u_t; G^r_{u_z}(u_p,i,j))$ in the case where u_t has no antineighbors in $G^r_{u_z}(u_p,i,j)$.

Assume now that u_t has at least one antineighbor in $G^r_{u_z}(u_p,i,j)$. Let $P = (u_{x'}, \ldots, u_x, u_t) = (P', u_t)$ be a longest normal antipath of $G^r_{u_z}(u_p,i,j)$ with right endpoint the vertex u_t. Then, it is easy to see that P' is a longest normal antipath of $G^{r-1}_{u_z}(u_p,i,j)$ with rigth endpoint the vertex u_x. In this case, with the `procedure(append)`, Algorithm LP_Cocomparability computes the value $w_1+1 = \ell(u_x; G^{r-1}_{u_z}(u_p,i,j))+1$, for every vertex $u_x \in H_\ell \cap V(G^{r-1}_{u_z}(u_p,i,j))$ such that $f(u_t) \leq \ell \leq j$, $x \neq t$, and $u_x u_t \notin E(G)$, and sets $\ell(u_t; G^r_{u_z}(u_p,i,j))$ to be equal to the maximum amongs these values. We next show that the algorithm correctly computes the values $\ell(u_t; G^r_{u_z}(u_p,i,j))$ and $P(u_t; G^r_{u_z}(u_p,i,j))$.

(a) Assume first that $u_x \notin L_j$, where u_x is the right endpoint of P'. Since by the induction hypothesis the algorithm correctly computes the values $\ell(u_x; G^{r-1}_{u_z}(u_p,i,j)$, for every vertex $u_x \in G^{r-1}_{u_z}(u_p,i,j)$ such that $u_x \notin L_j$, it follows that Algorithm LP_Cocomparability computes, among other, the value $\ell(u_x; G^{r-1}_{u_z}(u_p,i,j)) + 1 = |P'| + 1$, and sets $\ell(u_t; G^r_{u_z}(u_p,i,j))$ to be equal to $|P'| + 1 = |P|$ which is the length of a longest normal antipath P of $G^r_{u_z}(u_p,i,j)$ with right endpoint the vertex u_y. Also, the algorithm correctly computes the corresponding antipath $P(u_t; G^r_{u_z}(u_p,i,j)) = (P(u_x; G^{r-1}_{u_z}(u_p,i,j)), u_t)$.

(b) Consider now the case where for any longest normal antipath $P = (u_{x'}, \ldots, u_x, u_t) = (P', u_t)$ of $G^r_{u_z}(u_p,i,j)$ with right endpoint the vertex $u_y = u_t$ we have $u_x \in L_j$. Then P' is a longest normal antipath of $G^{r-1}_{u_z}(u_p,i,j)$ with right endpoint any vertex of L_j, i.e., $|P'| \geq |P''|$, for any normal antipath P'' of $G^{r-1}_{u_z}(u_p,i,j)$ with right endpoint a vertex of L_j. Let u_x be the last vertex of $L^{r-1}_j(u_z)$ for which such an antipath P' exists. Since Algorithm LP_Cocomparability computes, with the `procedure(append)`,

the value $w_1 + 1 = \ell(u_x; G^{r-1}_{u_z}(u_p, i, j)) + 1$, for every vertex $u_x \in L^{r-1}_j(u_z)$, and sets $\ell(u_t; G^r_{u_z}(u_p, i, j))$ to be equal to the maximum amongs these values, it follows that it suffices to show that there exists at least one vertex $u_x \in L^{r-1}_j(u_z)$ such that Algorithm LP_Cocomparability correctly computes the value $\ell(u_x; G^{r-1}_{u_z}(u_p, i, j))$ and sets it to be equal to $|P'|$.

Assume that u_x is the last vertex of $L^{r-1}_j(u_z)$, i.e., there exists such a longest normal antipath P' for which u_x is the last vertex of $L^{r-1}_j(u_z)$. Then by the induction hypothesis, Algorithm LP_Cocomparability correctly computes the length $\ell(u_x; G^{r-1}_{u_z}(u_p, i, j))$ of a longest normal antipath of $G^{r-1}_{u_z}(u_p, i, j)$ with right endpoint the vertex u_x. Therefore, in this case the last vertex u_x of $L^{r-1}_j(u_z)$ is such a vertex, and the claim holds.

Consider now the case where u_x is not the last vertex of $L^{r-1}_j(u_z)$, i.e., $u_x \in L^{r-2}_j(u_z)$. Let u_q be the last vertex of $L^{r-1}_j(u_z)$. Since P' is a longest normal antipath of $G^{r-1}_{u_z}(u_p, i, j)$ with right endpoint any vertex of L_j, it follows that $u_q \in V(P')$, since otherwise $P'' = (P', u_q)$ is such an antipath longer than P'. Let $P' = (u_{x'}, \ldots, u_{q'}, u_q, u_{q''}, \ldots, u_x) = (P_1, u_q, P_2)$. We now prove that $u_{q'} <_\sigma u_q$. The case where $u_{q'} >_\sigma u_q$ does not exist since $u_q \in L_j$. Assume that $u_{q'} =_\sigma u_q$. Then using similar arguments as in Case 2(I), Lemma 5.4, and Definition 5.2, it is easy to obtain that $u_s =_\sigma u_q$ for every vertex $u_s \in V(P_2)$. Thus, $P' = (P_1, u_{q''}, \ldots, u_x, u_q)$ is a normal antipath such that $V(P'') = V(P')$ with right endpoint the vertex u_q which appears after u_x in $L^{r-1}_j(u_z)$; this is a contradiction to our assumption on u_x. Therefore, we obtain that $u_{q'} <_\sigma u_q$. Assume now that $P' = (u_{x'}, \ldots, u_{q'}, u_q, u_{q''}, \ldots, u_x)$ is a longest normal antipath of $G^{r-1}_{u_z}(u_p, i, j)$ with the property that for any vertex u_s of $L^{r-2}_j(u_z)$ which appears after u_q in P' we have $u_s u_{q'} \in E(G)$; since we have proved that $u_{q'} <_\sigma u_q$, then using the same arguments as in Case 2(I) we can prove that such a longest normal antipath exists. In particular, using the same arguments as in Case 2(I), we can prove that $u_{q'} <_\sigma u_s$ and $u_{q'} u_s \in E(G)$, for every vertex $u_s \in V(P_2)$.

Since $u_{q'} u_q \notin E(G)$ and $u_{q'} <_\sigma u_q$, we assume that $u_{q'} \in H_\ell$, $f(u_q) \leq \ell \leq j - 1$. Let H_2 be the subgraph of $G^{r-1}_{u_z}(u_p, i, j)$ induced by $V(H_2) = V(G^{r-2}_{u_z}(u_{q'}, \ell + 1, j))$. Similarly to Case 2(I), we can show that every vertex of P_2 belongs to H_2. Let H_1 be the subgraph of $G^{r-1}_{u_z}(u_p, i, j)$ induced by $V(H_1) = V(G^{r-2}_{u_z}(u_p, i, j)) \setminus V(G^{r-2}_{u_z}(u_{q'}, \ell + 1, j))$. Again, we can show that every vertex of P_1 belongs to H_1. Therefore, we have that $V(P_1) \subseteq V(H_1)$, $V(P_2) \subseteq V(H_2)$, and $V(H_1) \cap V(H_2) = \emptyset$. It is easy to see that $V(P_1) \cap V(P_2) = \emptyset$. Finally, we can obtain that P_1 is a longest normal antipath of $G^{r-2}_{u_z}(u_p, i, j)$ with right endpoint the vertex $u_{q'}$, i.e., $|P_1| = \ell(u_{q'}; G^{r-2}_{u_z}(u_p, i, j))$, and P_2 is a longest normal antipath of $G^{r-2}_{u_z}(u_{q'}, \ell + 1, j)$ with right endpoint the vertex u_x, i.e., $|P_2| = \ell(u_x; G^{r-2}_{u_z}(u_{q'}, \ell + 1, j))$.

Since $u_{q'} <_\sigma u_q$, it follows that $u_{q'} \notin L_j$. Therefore, from the induction hypothesis Algorithm LP_Cocomparability correctly computes the length $\ell(u_{q'}; G^{r-2}_{u_z}(u_p, i, j)) = |P_1|$. Now it is left to show that the value $\ell(u_x; G^{r-2}_{u_z}(u_{q'}, \ell+1, j)) = |P_2|$ computed by the algorithm is the length of a longest normal antipath of $G^{r-2}_{u_z}(u_{q'}, \ell+1, j)$ with right endpoint the vertex u_x. Observe that now P_2 is a longest normal antipath of $G^{r-2}_{u_z}(u_{q'}, \ell+1, j)$ with right endpoint any vertex of $L^{r-2}_j(u_z)$ and, actually, u_x is the last vertex of $L^{r-2}_j(u_z)$ for which such an antipath P_2 exists, otherwise we come to a contradiction to the choice of P'. If u_x is the last vertex of $L^{r-2}_j(u_z)$ then, similarly to the above, by the induction hypothesis the algorithm correctly computes the value $\ell(u_x; G^{r-2}_{u_z}(u_{q'}, \ell + 1, j))$. If u_x is not the last vertex of $L^{r-2}_j(u_z)$, then we repeat the above same procedure for the last vertex of $L^{r-2}_j(u_z)$. We repeat the above procedure until u_x is the last vertex of the ordering $L^{r'}_j(u_z)$, $1 \leq r' \leq r-2$. Then, using the induction hypothesis, we obtain that the algorithm correctly computes the value of the corresponding normal antipath P_2 with right endpoint the vertex u_x, since at that iteration u_x is the last vertex of the ordering $L^{r'}_j(u_z)$. Concluding Algorithm LP_Cocomparability correctly computes the length $\ell(u_x; G^{r-1}_{u_z}(u_p, i, j)) = |P'|$ and, thus, the length $\ell(u_t; G^r_{u_z}(u_p, i, j)) = |P'| + 1 = |P|$; the algorithm also computes the corresponding antipaths.

Concluding, we have proved that the claim holds for the subgraph $G^r_{u_z}(u_p, i, j)$ of G, where $1 \leq r \leq |L_j|$. ∎

Now, let P be a longest antipath of G. From Lemma 5.6 we may assume without loss of generality that P is a normal antipath of G. If $u_y \in V(G)$ is the right endpoint of P, then P is a longest normal

antipath of G with right endpoint the vertex u_y. Since there exists a longest antipath of G which does not contain the vertex u_0 we may assume that P belongs to the graph $G \setminus \{u_0\}$. Since $G(u_0, 1, k) = G \setminus \{u_0\}$, from Lemma 5.9 it follows that Algorithm LP_Cocomparability correctly computes a longest normal antipath of $G(u_0, 1, k)$ with right endpoint the vertex u_y and, thus, sets $\ell(u_y; G(u_0, 1, k)) = |P|$. Since the output of Algorithm LP_Cocomparability is the maximum among the lengths $\{\ell(u_y; G(u_0, 1, k)) : u_y \in V(G(u_0, 1, k))\}$, along with the corresponding antipath, from Lemma 5.9 it follows that Algorithm LP_Cocomparability computes a longest normal antipath P' of $G(u_0, 1, k)$ with right endpoint any vertex $u_y \in V(G(u_0, 1, k)))$ such that $|P'| = |P|$. Therefore, we obtain the following result.

Theorem 5.3. *Algorithm LP_Cocomparability computes a longest antipath of a comparability graph.*

5.4.2 Time Complexity

Let G be a comparability graph on $|V(G)| = n$ vertices and $|E(G)| = m$ edges. Given a Hasse diagram of G, the time complexity of our algorithm is as follows.

Algorithm LP_Cocomparability executes the subroutine `process()` for every induced subgraph $G^r_{u_z}(u_p, i, j)$ of G. In particular, the subroutine `process()` contains two procedures, the `procedure(bridge)` and the `procedure(append)`. The execution of the `procedure(bridge)` for the subgraph $G^r_{u_z}(u_p, i, j)$ takes $O(n^2)$ time, due to the $O(n^2)$ pairs of antineighbors u_x and u_y of the vertex u_t in the graph $G^r_{u_z}(u_p, i, j)$. The execution of the `procedure(append)` for the subgraph $G^r_{u_z}(u_p, i, j)$ takes $O(n)$ time, due to the $O(n)$ antineighbors u_x of the vertex u_t in the graph $G^r_{u_z}(u_p, i, j)$. Therefore, the execution of the subroutine `process()` for the subgraph $G^r_{u_z}(u_p, i, j)$ takes $O(n^2)$ time.

Additionally, the subroutine `process()` is executed at most once for each subgraph $G^r_{u_z}(u_p, i, j)$ of G. Since $1 \leq i \leq j \leq k$, $u_p \in H_{i-1}$, $u_z \in L_j$, and $1 \leq r \leq |L_j|$, it follows that there exist $O(n^5)$ such subgraphs $G^r_{u_z}(u_p, i, j)$ of G. Thus, Algorithm LP_Cocomparability takes $O(n^7)$ time.

In order to compute the length of a longest antipath, we need to store one value for every vertex u_y of $G^r_{u_z}(u_p, i, j)$, for every induced subgraph $G^r_{u_z}(u_p, i, j)$ of G. Thus, since there are in total $O(n^5)$ such subgraphs $G^r_{u_z}(u_p, i, j)$, and since each one has at most $O(n)$ vertices, we can compute the length of a longest antipath in $O(n^6)$ space. Furthermore, in order to compute and report a longest antipath, instead of its length only, we have to store an antipath of at most n vertices for each one of the $O(n^6)$ computed values. Therefore, the space complexity of Algorithm LP_Cocomparability is $O(n^7)$.

5.5 Concluding Remarks

In this chapter we presented a polynomial-time algorithm for solving the longest path problem on cocomparability graphs, resolving thus the open question on the complexity status of the problem on cocomparability and, also, on permutation graphs. We also help to shed some light on the borderline between P and NP, since the longest path problem is known to be NP-complete on comparability graphs and quasi-parity graphs, which are superclasses of permutation and cocomparability graphs, respectively.

It would be interesting to study the complexity of the longest path problem on distance-hereditary and bipartite distance-hereditary graphs, since they admit polynomial solutions for the Hamiltonian path problem, and also since the longest path problem has been proved to be NP-complete on chordal bipartite graphs, HHD-free graphs, and parity graphs, while it is polynomial on ptolemaic graphs and trees. Additionally, the same holds for the classes of convex and biconvex graphs, since the longest path problem has been proved to be NP-complete on chordal bipartite graphs and polynomial on bipartite permutation graphs.

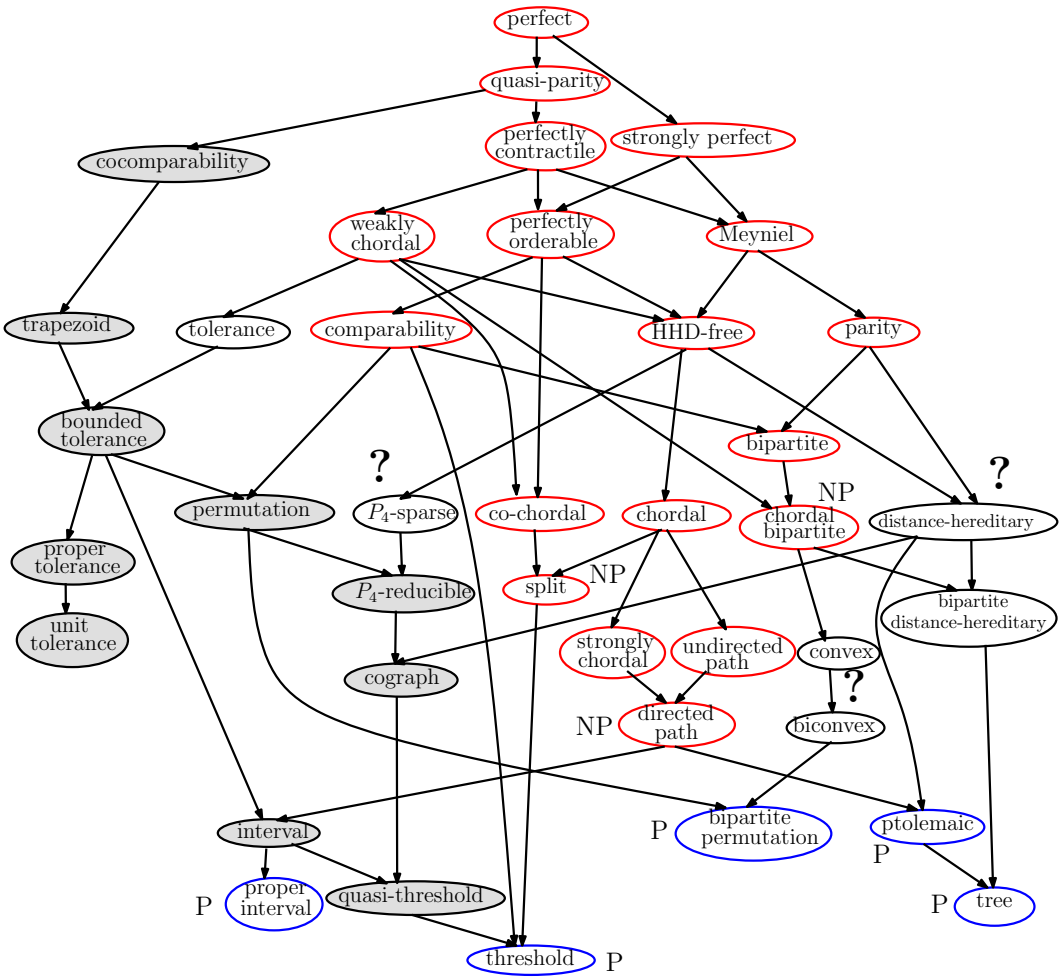

Figure 5.5: Illustrating a map of some classes of perfect graphs and the complexity status of the longest path problem.

Figure 5.5 illustrates a map of some classes of perfect graphs and the complexity status of the longest path problem.

- By NP we mark the classes for which the longest path problem has been proved to be NP-complete; in fact, we obtain these results from the NP-completeness of the Hamiltonian path problem.
- By P we mark the classes for which polynomial solutions have been presented for the longest path problem until now.
- With gray color we mark the classes of interval graphs and cocomparability graphs, as well as their subclasses, for which we have proved within this work that the longest path problem admits a polynomial solution.
- By the symbol ? we mark the classes for which the Hamiltonian path problem has been proved to be polynomial, while the complexity of the longest path problem still remains an open question.

Chapter 6

Conclusions and Further Research

6.1 Colinear and Linear Graphs

In this work we introduced the colinear coloring on graphs and proposed a colinear coloring algorithm that can be applied to any graph G. Based on the colinear coloring we defined two graph properties, namely the χ-colinear and α-colinear properties, and characterized known graph classes in terms of these properties. We also defined and studied the graphs that are characterized completely by the χ-colinear or the α-colinear property, which form two new classes of perfect graphs, and which we call colinear and linear graphs.

We also provide characterizations for colinear and linear graphs and prove structural properties. More specifically, we show that the class of colinear graphs is a subclass of co-chordal graphs, a superclass of threshold graphs, and is distinguished from the class of split graphs. Additionally, we infer that linear graphs form a subclass of chordal graphs and a superclass of quasi-threshold graphs. We also prove that any P_6-free chordal graph, which is not a linear graph, properly contains a k-sun as an induced subgraph. However, the k-sun is not a forbidden induced subgraph for the class of linear graphs and, thus, linear graphs form a superclass of the class of P_6-free strongly chordal graphs.

An interesting question would be to study structural and recognition properties of colinear and linear graphs and see whether they can be characterized by a finite set of forbidden induced subgraphs. Moreover, an obvious though interesting open question would be whether combinatorial and/or optimization problems can be efficiently solved on the classes of linear and colinear graphs. In addition, it would be interesting to study the relation between the colinear chromatic number and other coloring numbers such as the harmonious number and the achromatic number on classes of graphs.

Concerning the question of whether optimization problems can be efficiently solved on the classes of linear and colinear graphs, it would be interesting as a first step to study the complexity status of the harmonious coloring problem and the longest path problem. From the results presented in Chapter 3 it follows that the harmonious coloring problem is NP-complete on the classes of colinear graphs and disconnected linear graphs, while it still remains open on connected linear graphs. Additionally, the longest path problem is NP-complete on both colinear and linear graphs, a result which follows from

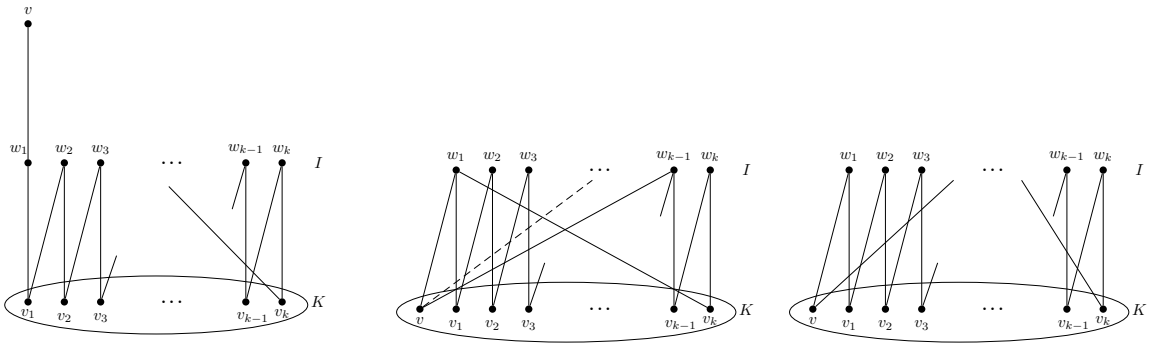

Figure 6.1: Illustrating forbidden subgraphs F_1, F_2, and F_3 (in the order they appear from left to right). Note that, in all three graphs, the set K is a clique, and I is an independent set.

the NP-completeness of the problem on split strongly chordal graphs [58], which form a subclass of both linear and colinear graphs.

Additionally, a promising and interesting area for further research is studying structural and recognition properties of colinear and linear graphs, and seeing whether they can be characterized by a finite set of forbidden induced subgraphs. In fact, we have done some progress towards this direction, and below we present some preliminary results.

Briefly, we build on our results presented in Chapter 2, where we proved some structural properties for colinear and linear graphs, including the following theorem.

Theorem 2.1. *Let $\mathcal{F}$ be the family of all the minimal forbidden induced subgraphs of the class of linear graphs, and let F_i be a member of $\mathcal{F}$. The graph F_i is either a C_n ($n \geq 4$), or a P_6, or it properly contains a k-sun ($k \geq 3$) as an induced subgraph.*

Observe now that, if S_k is a k-sun graph, $k \geq 4$, then from Theorem 2.1 and the definition of colinear and linear graphs, it is easy to see that any $\overline{P}_6$-free co-chordal graph which is not a colinear graph properly contains an $\overline{S_k}$ graph as an induced subgraph. Additionally, from the definition of strongly chordal graphs it is easy to obtain that for any k-sun graph S_k, $k \geq 4$, the graph $\overline{S_k}$ contains a k'-sun, $k' \leq k$, as an induced subgraph. From the above observations, and our study on colinear and liner graphs in terms of forbidden induced subgraphs, we conjecture the following.

Conjecture 6.1. *Any $\overline{P}_6$-free co-chordal graph which is not a colinear graph properly contains a k-sun, $k \geq 3$, as an induced subgraph.*

Let us now explain the graphs illustrated in Figures 6.1 and 6.2. We first explain the graphs illustrated in Figure 6.1. Note that in all three graphs, the set K is a clique, and I is an independent set. In F_1, the vertex v_k sees the vertex $w_k \in I$, and also another vertex $w_i \in I$ such that $i \neq k-1$. In F_2, the vertex v sees at least three vertex of I, such that for any two neighbors $w_i, w_j \in I$, $i < j$, of v we have $i \neq j-1 \pmod{k}$. In F_3, the vertex v sees $w_1 \in I$, and also another vertex $w_i \in I$ such that $i \neq 2$ and $i \neq k$. Also, the vertex v_k sees $w_k \in I$, and also another vertex $w_i \in I$ such that $i \neq k-1$ and $i \neq 1$.

We explain now the graphs illustrated in Figure 6.2. Again in all three graphs, the set K is a clique, and I is an independent set. In $\widetilde{F}_1$, the vertex v_k sees the vertex $w_k \in I$, and also another vertex $w_i \in I$ such that $i \neq k-1$. Also, the vertex v sees the vertex $w_1 \in I$ and also every vertex of the set $K \setminus N(w_1)$. In $\widetilde{F}_2$, the vertex v sees at least three vertex of K, such that for any two neighbors $v_i, v_j \in K$, $i < j$, of v we have $i \neq j-1 \pmod{k}$. In $\widetilde{F}_3$, the vertex v sees $v_k \in K$, and also another vertex $v_i \in K$ such that $i \neq k-1$ and $i \neq 1$. Also, the vertex w_1 sees $v_1 \in K$, and also another vertex $v_i \in K$ such that $i \neq 2$ and $i \neq k$.

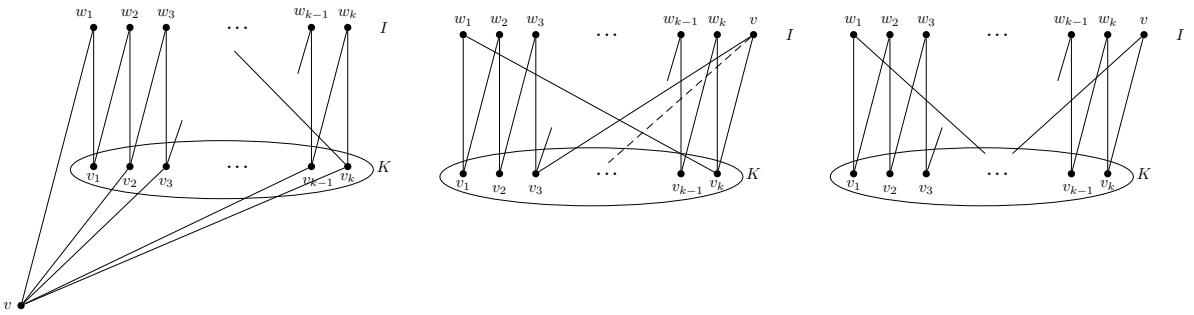

Figure 6.2: Illustrating the forbidden subgraphs $\widetilde{F}_1$, $\widetilde{F}_2$, and $\widetilde{F}_3$ (in the order they appear from left to right). Note that, in all three graphs, the set K is a clique, and I is an independent set.

In sum, from Theorem 2.1, Claim 6.1, and after much work towards the direction of characterizing colinear and linear graphs in terms of forbidden induced subgraphs, we conjecture the following.

Conjecture 6.2. *Let G be a P_6-free chordal graph. Then, G is a linear graph if and only if G is (F_1,F_2,F_3)-free and $\overline{G}$ is ($\widetilde{F}_1$,$\widetilde{F}_2$,$\widetilde{F}_3$)-free.*

6.2 The Harmonious Coloring Problem

In this work we first show that the harmonious coloring problem is NP-complete on connected interval and permutation graphs. Also we prove the NP-completeness of the problem on the class of split graphs. Extending our results, we then prove that the harmonious coloring problem is NP-complete on the classes of split undirected path graphs and colinear graphs. We also present a polynomial solution for the same problem on the class of split strongly chordal graphs. The interest of this result lies on the fact that the harmonious coloring problem is NP-complete on split graphs and strongly chordal graphs. In addition, polynomial solutions for the problem were only known until now for the classes of connected cographs, connected quasi-threshold graphs, and threshold graphs, all of which have a trivial solution; note that, the harmonious coloring problem on disconnected quasi-threshold graphs is NP-complete.

An interesting next step in the study of the harmonious coloring problem would be to see if the problem admits a polynomial solution on linear graphs. Since linear graphs form a superclass of both split strongly chordal graphs and quasi-threshold graphs, the harmonious coloring problem is NP-complete on disconnected linear graphs, while it still remains open on connected linear graphs. Obtaining a polynomial solution of the harmonious coloring problem on connected linear graphs would be interesting, since most of the known results for the harmonious coloring problem on special classes of graphs are NP-completeness results; indeed, the only known non-trivial polynomial algorithm for the problem is our solution on split strongly chordal graphs, which form a subclass of linear graphs.

6.3 The Longest Path Problem

In this work we presented a polynomial-time algorithm for solving the longest path problem on interval graphs, which runs in $O(n^4)$ time and, thus, provided a solution to the open problem stated by Uehara and Uno in [63] asking for the complexity status of the longest path problem on interval graphs.

Moreover, we studied the longest path problem on the class of cocomparability graphs, a well-known

class of perfect graphs which includes both interval and permutation graphs. Although the Hamiltonian path problem on cocomparability graphs was proved to be polynomial almost two decades ago [23], the complexity status of the longest path problem on cocomparability graphs has remained open until now; actually, the complexity status of the longest path problem has been open even on the more special class of permutation graphs. In this work, we presented a polynomial-time algorithm for solving the longest path problem on the class of cocomparability graphs. This result extends our polynomial solution of the longest path problem on interval graphs, and resolves the open question for the complexity of the problem on cocomparability graphs, and thus on permutation graphs.

Additionally, through this work we help to shed some light on the borderline between P and NP, since the longest path problem was known to be NP-complete on comparability graphs and quasi-parity graphs [58], which are superclasses of permutation and cocomparability graphs, respectively.

It would be interesting to study the complexity of the longest path problem on convex and biconvex graphs, which they admit polynomial solutions for the Hamiltonian path problem; the longest path problem is NP-complete on chordal bipartite graphs [58] which is a superclass of both convex and biconvex graphs, and polynomial on bipartite permutation graphs [64] which is subclass of convex and biconvex graphs. Additionally, the same holds for the classes of distance-hereditary and bipartite distance-hereditary graphs. Indeed, the longest path problem is NP-complete on chordal bipartite graphs [58] which is a minimal superclass of bipartite distance-hereditary graphs, and polynomial on trees [12] which is a minimal subclass of bipartite distance-hereditary graphs. Additionally, the longest path problem is NP-complete on the minimal superclasses of distance-hereditary graphs, namely HHD-free graphs and parity graphs [58], while it is polynomial on ptolemaic graphs [65] which is a minimal subclass of distance-hereditary graphs.

Therefore resolving the question concerning the complexity of the longest path problem on these graph graphs, would sharpen the demarcation line between polynomially solvable and NP-hard cases of the problem, for most of the well-known subclasses of perfect graphs.

Bibliography

[1] S.R. Arikati and C.P. Rangan, Linear algorithm for optimal path cover problem on interval graphs, *Inform. Process. Lett.* **35** (1990) 149–153.

[2] K. Asdre and S.D. Nikolopoulos, NP-completeness results for some problems on subclasses of bipartite and chordal graphs, *Theoret. Comput. Sci.* **381** (2007) 248–259.

[3] K. Asdre and S.D. Nikolopoulos, The 1-Fixed-Endpoint Path Cover Problem is Polynomial on Interval Graphs, *Algorithmica* (to appear).

[4] C. Berge, Färbung von Graphen, deren sämtliche bzw. deren ungerade Kreise starr sind, Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Natur. Reihe **10** (1961) 114–115.

[5] C. Berge, *Graphs and Hypergraphs*, American Elsevier, New York, 1973.

[6] A.A. Bertossi, Finding Hamiltonian circuits in proper interval graphs, *Inform. Proc. Lett.* **17** (1983) 97–101.

[7] A.A. Bertossi, Total domination in interval graphs, *Inform. Proc. Lett.* **23** (1986) 131–134.

[8] H.L. Bodlaender, Achromatic number is NP-complete for cographs and interval graphs, *Inform. Proc. Lett.* **31** (1989) 135–138.

[9] F.T. Boesch and J.F. Gimpel, Covering the points of a digraph with point-disjoint paths and its application to code optimization, *J. of the ACM* **24** (1977) 192–198.

[10] A. Brandstädt, V.B. Le and J.P. Spinrad, *Graph Classes: A Survey*, SIAM, Philadelphia, PA, 1999.

[11] R.L. Brooks, On colouring the nodes of a network, *Proc. Cambridge Phil. Soc.* **37** (1941) 194–197.

[12] R. Bulterman, F. van der Sommen, G. Zwaan, T. Verhoeff, A. van Gasteren, and W. Feijen, On computing a longest path in a tree, *Inform. Proc. Lett.* **81** (2002) 93–96.

[13] N. Cairnie and K. Edwards, Some results on the achromatic number, *J. Graph Theory* **26** (1997) 129–136.

[14] G.J. Chang, Labeling algorithms for domination problems in sun-free chordal graphs, *Discrete Applied Math.* **22** (1988) 21–34.

[15] M.S. Chang, S.L. Peng, and J.L. Liaw, Deferred-query: An efficient approach for some problems on interval graphs, *Networks* **34** (1999) 1–10.

[16] M. Chudnovsky, N. Robertson, P.D. Seymour, and R. Thomas, The strong perfect graph theorem, *Annnals of Math.* **164** (2006) 51–229.

[17] V. Chvátal and P.L. Hammer, Aggregation of inequalities for integer programming, *Ann. Discrete Math.* **I** (1977) 145–162.

[18] Y. Civan and E. Yalçin, Linear colorings of simplicial complexes and collapsing, *J. Comb. Theory A* **114** (2007) 1315–1331.

[19] S. Cook, The complexity of theorem-proving procedures, *Proc. 3rd Ann. ACM Symp. on Theory of Computing*, Association for Computing Machinery, New York, 151–158.

[20] D.G. Corneil, H. Lerchs, and L. Stewart Burlingham, Complement reducible graphs, *Discrete Applied Math.* **3** (1981) 163–174.

[21] P. Csorba, C. Lange, I. Schurr, and A. Wassmer, Box complexes, neighborhood complexes, and the chromatic number, *J. Comb. Theory A* **108** (2004) 159–168.

[22] P. Damaschke, The Hamiltonian circuit problem for circle graphs is NP-complete, *Inform. Proc. Lett.* **32** (1989) 1–2.

[23] P. Damaschke, J.S. Deogun, D. Kratsch, and G. Steiner, Finding Hamiltonian paths in cocomparability graphs using the bump number algorithm, *Order* **8** (1992) 383–391.

[24] P. Damaschke, Paths in interval graphs and circular arc graphs. *Discrete Math.* **112** (1993) 49–64.

[25] K.J. Edwards, The harmonious chromatic number and the achromatic number, in: *Surveys in Combinatorics* (R.A. Baily, Ed.), Cambridge University Press, Cambridge (1997) 13–47.

[26] K.J. Edwards and C. McDiarmid, The complexity of harmonious coloring for trees, *Discrete Applied Math.* **57** (1995) 133–144.

[27] M. Farber, Characterizations of strongly chordal graphs, *Discrete Math.* **43** (1983) 173–189.

[28] M. Farber, Domination, independent domination, and duality in strongly chordal graphs, *Discrete Applied Math.* **7** (1984) 115–130.

[29] T. Feder and R. Motwani, Finding large cycles in Hamiltonian graphs, *Proc. of the 16th annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, ACM (2005) 166–175.

[30] O. Frank, F. Harary, and M. Plantholt, The line-distinguishing chromatic number of a graph, *Ars Combin.* **14** (1982) 241–252.

[31] H.N. Gabow, Finding paths and cycles of superpolylogarithmic length, *Proc. of the 36th annual ACM Symp. on Theory of Computing (STOC)*, ACM (2004) 407–416.

[32] H.N. Gabow and S. Nie, Finding long paths, cycles and circuits, *Proc. of the 19th annual International Symp. on Algorithms and Computation (ISAAC)*, LNCS **5369** (2008) 752–763.

[33] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, W.H. Freeman, San Francisco, 1979.

[34] M.R. Garey, D.S. Johnson, and R.E. Tarjan, The planar Hamiltonian circuit problem is NP-complete, *SIAM J. Computing* **5** (1976) 704–714.

[35] F. Gavril, A recognition algorithm for the intersection graph of paths of a tree, *Discrete Math.* **23** (1978) 377–388.

[36] P.W. Goldberg, M.C. Golumbic, H. Kaplan, and R. Shamir, Four strikes against physical mapping of DNA, *Journal of Comp. Biology* **2** (1995) 139–152.

[37] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs* (Annals of Discrete Mathematics, Vol 57), North-Holland Publishing Co., Amsterdam, The Netherlands, 2004.

[38] U.I. Gupta, D.T. Lee, and J.Y. Leung, Efficient algorithms for interval graphs and circular-arc graphs, *Networks* **12** (1982) 459–467.

[39] M. Habib, R.H. Morhing, and G. Steiner, Computing the bump number is easy, *Order* **5** (1988) 107–129.

[40] F. Harary, *Graph Theory*, Addison-Wesley, 1969.

[41] F. Harary and S.T. Hedetniemi, The achromatic number of a graph, *J. Comb. Theory* **8** (1970) 154–161.

[42] F. Harary, S.T. Hedetniemi, and G. Prins, An interpolation theory for graphical homomorphisms, *Portugal. Math.* **26** (1967) 453–462.

[43] CT. Hoàng, *On the complexity of finding a sun in a graph*, eprint arXiv:0807.0462 (2008).

[44] J. Hopcroft and R.M. Karp, A $n^{5/2}$ algorithm for maximum matchings in bipartite graphs, *SIAM J. Computing* **2** (1973) 225–231.

[45] J.E. Hopcroft and M.S. Krishnamoorthy, On the harmonious coloring of graphs, *SIAM J. Alg. Discrete Meth.* **4** (1983) 306–311.

[46] A. Itai, C.H. Papadimitriou, and J.L. Szwarcfiter, Hamiltonian paths in grid graphs, *SIAM J. Computing* **11** (1982) 676–686.

[47] D. Karger, R. Motwani, and G.D.S. Ramkumar, On approximating the longest path in a graph, *Algorithmica* **18** (1997) 82–98.

[48] R.M. Karp, Reducibility among combinatorial problems, *Complexity of Computer Computations*, Plenum Press, New York, 85–103, 1972.

[49] J.M. Keil, Finding Hamiltonian circuits in interval graphs, *Inform. Proc. Lett.* **20** (1985) 201–206.

[50] M. Kneser, Aufgabe 300, *Jahresbericht der Deutschen Mathematiker-Vereinigung* **58** (1955) 2.

[51] D. Kratsch, Finding dominating cliques efficiently, in strongly chordal graphs and undirected path graphs, *Discrete Math.* **86** (1990) 225–238.

[52] R. Lin, S. Olariu and G. Pruesse, An optimal path cover algorithm for cographs, *Comput. Math. Appl.* **30** (1995) 75–83.

[53] L. Lovász, A characterization of perfect graphs, *J. Combinatorial Theory B* **13** (1972) 95–98.

[54] L. Lovász, Kneser's conjecture, chromatic numbers and homotopy, *J. Comb. Theory A* **25** (1978) 319–324.

[55] J. Matoušek and G.M. Ziegler, Topological lower bounds for the chromatic number: a hierarchy, *Jahresbericht der Deutschen Mathematiker-Vereinigung* **106** (2004) 71–90.

[56] T.A. McKee and F.R. McMorris, *Topics in Intersection Graph Theory*, Society for Industrial and Applied Mathematics, Philadelphia, 1999.

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

[58] H. Müller, Hamiltonian Circuits in chordal bipartite graphs, *Discrete Math.* **156** (1996) 291–298.

[59] S.D. Nikolopoulos, Recognizing cographs and threshold graphs through a classification of their edges, *Inform. Proc. Lett.* **74** (2000) 129–139.

[60] S.D. Nikolopoulos and L. Palios, Hole and antihole detection in graphs, *Proc. 15th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA'04)*, (2004) 850–859.

[61] G. Ramalingam and C.P. Rangan, A unified approach to domination problems on interval graphs, *Inform. Proc. Lett.* **27** (1988) 271–274.

[62] A.A. Schäffer, A faster algorithm to recognize undirected path graphs, *Discrete Applied Math.* **43** (1993) 261–295.

[63] R. Uehara and Y. Uno, Efficient algorithms for the longest path problem, *Proc. of the 15th annual International Symp. on Algorithms and Computation (ISAAC)*, LNCS **3341** (2004) 871–883.

[64] R. Uehara and G. Valiente, Linear structure of bipartite permutation graphs and the longest path problem, *Inform. Proc. Lett.* **103** (2007) 71–77.

[65] Y. Takahara, S. Teramoto, and R. Uehara, Longest path problems on ptolemaic graphs, *IEICE Trans. Inf. and Syst.* **91-D** (2008) 170–177.

[66] S. Vishwanathan, An approximation algorithm for finding a long path in Hamiltonian graphs, *Proc. of the 11th annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, ACM (2000) 680–685.

[67] M. Yannakakis and F. Gavril, Edge dominating sets in graphs, *SIAM J. Appl. Math.* **38** (1980) 364–372.

[68] Z. Zhang, and H. Li, Algorithms for long paths in graphs, *Theoret. Comput. Sci.* **377** (2007) 25–34.

[69] G.M. Ziegler, Generalised Kneser coloring theorems with combinatorial proofs, *Inventiones mathematicae* **147** (2002) 671–691.

Index

Author's Publications

- K. Ioannidou and S.D. Nikolopoulos, Colinear coloring on graphs, *3rd Annual Workshop on Algorithms and Computation (WALCOM'09)*, LNCS **5431** (2009) 117–128.
- K. Ioannidou and S.D. Nikolopoulos, Colinear coloring and colinear graphs, *Technical Report TR-2007-06*, Department of Computer Science, University of Ioannina, 2007 (submitted to journal).
- K. Asdre, K. Ioannidou, S.D. Nikolopoulos, The harmonious coloring problem is NP-complete for interval and permutation graphs, *Discrete Applied Math.* **155** (2007) 2377–2382.
- K. Ioannidou and S.D. Nikolopoulos, Harmonious coloring on subclasses of colinear graphs, *4rd Annual Workshop on Algorithms and Computation (WALCOM'10)*, accepted.
- K. Ioannidou, G.B. Mertzios, and S.D. Nikolopoulos, The longest path problem is polynomial on interval graphs, *34th International Symposium on Mathematical Foundations of Computer Science (MFCS'09)*, LNCS **5734** (2009) 403–414.
- K. Ioannidou and S.D. Nikolopoulos, The longest path problem is polynomial on cocomparability graphs, *Technical Report TR-2009-28*, Department of Computer Science, University of Ioannina, 2009 (submitted to journal).

Short CV

Kyriaki Ioannidou was born in Larnaka (Cyprus) on the 6th of August, 1982. She received a B.Sc in Mathematics from the Department of Mathematics, Aristotle University of Thessaloniki (2000-2004), and a M.Sc in Analysis, Design & Management of Information Systems from the London School of Economics and Political Science (2004-2005). Since 2005 she has been a Ph.D student at the Department of Computer Science, University of Ioannina, and she received her Ph.D. on the 11th of December, 2009. Her research interests are in design and analysis of algorithms, graph theory, and graph algorithms. She has studied characterizations of perfect graph classes, and algorithms for optimization problems as well as NP-complete problems on classes of perfect graphs. Her research work has been published at journals and conferences.