Edge Modification on Perfect and Reducible Graphs with Application to Watermarking

A Dissertation

submitted to the designated

by the Assembly

of the Department of Computer Science and Engineering

Examination Committee

by

Anna Mpanti

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

University of Ioannina

School of Engineering

Ioannina 2022

Advisory Committee:

- **Stavros D. Nikolopoulos**, Professor, Department of Computer Science and Engineering, University of Ioannina
- **Leonidas Palios**, Professor, Department of Computer Science and Engineering, University of Ioannina
- **Charis Papadopoulos**, Assoc. Professor, Department of Computer Science and Engineering, University of Ioannina

Examining Committee:

- **Stavros D. Nikolopoulos**, Professor, Department of Computer Science and Engineering, University of Ioannina
- **Leonidas Palios**, Professor, Department of Computer Science and Engineering, University of Ioannina
- **Charis Papadopoulos**, Assoc. Professor, Department of Computer Science and Engineering, University of Ioannina
- **Loukas Georgiadis**, Assoc. Professor, Department of Computer Science and Engineering, University of Ioannina
- **Christos Nomikos**, Assoc. Professor, Department of Computer Science and Engineering, University of Ioannina
- **Vassilis Zissimopoulos**, Professor, Department of Informatics and Telecommunications, University of Athens
- **Antonios Symvonis**, Professor, School of Applied Mathematical and Physical Sciences, National Technical University of Athens

Dedication

To my beloved family,
my father, my mother, my sister
who supported and helped me by all means to my career and my life.

"Find your why and you'll find your way."
John Maxwell

Acknowledgements

Finishing this dissertation, I would like to begin by thanking my parents, Napoleon and Asimoula, that would not have been possible without the support and nurturing of them. Especially, I would like to thank my sister Christina whom I deeply love and who encourange me to keep going when I was losing my courrage.

Then, from the deepest of my heart, I would like to thank my supervisor, Stavros D. Nikolopoulos, Professor of the Department of Computer Science and Engineering of the University of Ioannina, my "spiritual father", initially for the trust he has shown to me from the first moment before seven years, in my first steps in research area as Master student till now, and secondly for all lessons about research and career that he taught me. I would like to express my deepest appreciation to professor Leonidas Palios, for his patience and help he provided me throughout this PhD thesis. I would also like to thank the members of the advisory and examining committee, professor Charis Papadopoulos, and professor Loukas Georgiadis, for their valuable advice and the support in all duration of my studies, as well as professor Christos Nomikos, professor Vassilios Zissimopoulos and professor Antonios Symvonis, for their valuable remarks, which contributed to the improvement of this thesis.

Finally, I would like to thank my colleagues, but mostly friends, Sifis, Maria, Sakis for all the good cooperation we had all these years, the patience and the tolerance they shown the latest years in my character and all my friends for all the beautiful moments that we had together against through difficult times.

Table of Contents

List of Figures

Abstract

Anna Mpanti, Ph.D., Department of Computer Science and Engineering, School of Engineering, University of Ioannina, Greece, 2022.
Edge Modification on Perfect and Reducible Graphs with Application to Watermarking.
Advisor: Stavros D. Nikolopoulos, Professor.

In graph modification problems, we have to repair, improve, or adjust a graph to satisfy appropriate properties while minimizing the cost of the modification. The study of graph modification problems is crucial to computer science as they find applications in different areas, such as biology, mathematics, sociology, machine learning, data mining, and computer vision.

This PhD thesis has a theoretical and a more applied part, both related to edge modification problems on classes of graphs.

For the theoretical part, we study and present polynomial algorithms for the minimum completion problem (i) of a graph with a "tail" for four subclasses of perfect graphs and (ii) of a graph and an added edge for the class of P_4-sparse graphs. The minimum completion (fill-in) problem is defined as follows: Given a graph family $\mathcal{F}$ (more generally, a property Π) and a graph G, the completion problem asks for the minimum number of non-edges needed to be added to G so that the resulting graph belongs to the graph family $\mathcal{F}$ (or has property Π). This problem is NP-complete for many subclasses of perfect graphs and polynomial solutions are available only for minimal completion sets.

Given a graph G, a *tail* uw is an edge connecting a vertex $w \notin V(G)$ to a vertex $u \in V(G)$. We study the minimum completion problem for the graph $G+uw$ for the classes of split, quasi-threshold, threshold, and P_4-sparse graphs. Based on properties of the structure of split graphs and of the tree representation of quasi-threshold, threshold, and P_4-sparse graphs, we present linear-time algorithms to solve this problem.

Additionally, for the class of P_4-sparse graphs, we study the minimum completion problem of a P_4-sparse graph G with an added edge. For any optimal solution of the problem, we prove that there is an optimal solution whose form is of one of a small number of possibilities. This along with the solution of the problem when the added edge connects two non-adjacent vertices of a spider or connects two vertices in different connected components of the graph enables us to present a linear-time algorithm for the problem.

The applied part of this thesis focuses on the study of malicious edge modifications of reducible graphs used to encode an integer number as a watermark in a specific software watermarking codec system. The most important step in any software watermarking method is the choice of the right watermark, which, in the watermarking system we study, is a watermark producing a reducible graph in which edge modifications can be detected. Through the study of such edge modifications, we classify watermarks as strong, intermediate, or weak and we are able to give recommendations for the best choice of watermarks to use.

Εκτεταμενη Περιληψη

Άννα Μπαντή, Δ.Δ., Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πολυτεχνική Σχολή, Πανεπιστήμιο Ιωαννίνων, 2022.
Τροποποίηση Ακμών σε Τέλεια και Αναγώγιμα Γραφήματα με Εφαρμογή στην Υδατογράφηση.
Επιβλέπων: Σταύρος Δ. Νικολόπουλος, Καθηγητής.

Σε προβλήματα τροποποίησης γραφήματος, πρέπει να διορθώσουμε, να βελτιώσουμε ή να προσαρμόσουμε ένα γράφημα προκειμένου να πληροί συγκεκριμένες κατάλληλες ιδιότητες, ελαχιστοποιώντας ταυτόχρονα το κόστος της τροποποίησης. Η μελέτη των προβλημάτων τροποποίησης γραφημάτων είναι ύψιστης σημασίας για την επιστήμη των υπολογιστών. Τα προβλήματα τροποποίησης γραφημάτων έχουν πολλές εφαρμογές σε διαφορετικούς τομείς, όπως η βιολογία, τα μαθηματικά, η κοινωνιολογία, η μηχανική μάθηση, η εξόρυξη δεδομένων, η υπολογιστική όραση και πολλοί άλλοι τομείς.

Το κύριο σημείο εστίασης αυτής της διδακτορικής διατριβής έγκειται σε δύο μέρη, το θεωρητικό και το εφαρμοσμένο μέρος των προβλημάτων τροποποίησης ακμών σε κατηγορίες τέλειων γραφημάτων και αναγώγιμων γραφημάτων.

Στο θεωρητικό μέρος, μελετάμε και παρουσιάζουμε αλγόριθμους πολυωνυμικού χρόνου για το πρόβλημα ελάχιστης συμπλήρωσης (i) ενός γραφήματος και την προσθήκη μιας "ουράς" (tail) για τέσσερις υποκατηγορίες τέλειων γραφημάτων και (ii) ενός γραφήματος και την προσθήκη μιας ακμής για τη κατηγορία P_4-sparse γραφημάτων. Το πρόβλημα ελάχιστης συμπλήρωσης ορίζεται ως εξής: Δεδομένης μιας οικογένειας γραφημάτων $\mathcal{F}$ (ή γενικότερα, μια ιδιότητα Π) και ενός γραφήματος G, το πρόβλημα συμπλήρωσης ζητά τον ελάχιστο αριθμό μη ακμών που χρειάζεται να προστεθούν στο G, ώστε το γράφημα που προκύπτει να ανήκει στην οικογένεια γραφημάτων $\mathcal{F}$ (ή να έχει την ιδιότητα Π). Αυτό το πρόβλημα είναι NP-complete

για πολλές κλάσεις τέλειων γραφημάτων και υπάρχουν πολυωνυμικές λύσεις μόνο για ελάχιστα σύνολα συμπλήρωσης.

Δεδομένου ενός γραφήματος G, *ουρά* uw είναι μια ακμή που συνδέει έναν κόμβο $w \notin V(G)$ με έναν κόμβο $u \in V(G)$. Μελετάμε το πρόβλημα ελάχιστης συμπλήρωσης για το γράφημα $G + uw$ σε κατηγορίες τέλειων γραφημάτων, όπως τα split, quasi-threshold, threshold και P_4-sparse γραφήματα. Με βάση τις ιδιότητες της δομής των split γραφημάτων και της δεντρικής αναπαράστασης των quasi-threshold, threshold και P_4-sparse γραφημάτων, παρουσιάζουμε αλγόριθμους γραμμικού χρόνου για την επίλυση αυτού του προβλήματος.

Επιπρόσθετα, για την κατηγορία των P_4-sparse γραφημάτων, μελετάμε το πρόβλημα ελάχιστης συμπλήρωσης ενός P_4- sparse γραφήματος G με την πρόσθεση μιας μη-ακμής. Συγκεκριμένα, δοθέντος ενός P_4- sparse γραφήματος G και μιας μη-ακμής xy (δηλαδή, ένα ζεύγος των μη γειτονικών κόμβων x και y) του G, υπολογίζεται ο ελάχιστος αριθμός μη ακμών του G που πρέπει να προστεθούν στο G έτσι ώστε το γράφημα που προκύπτει να είναι P_4-sparse γράφημα και να περιέχει την xy ως ακμή. Για κάθε βέλτιστη λύση του προβλήματος, αποδεικνύουμε ότι υπάρχει μια βέλτιστη λύση της οποίας η μορφή είναι μία από ένα μικρό πλήθος πιθανοτήτων. Αυτό μαζί με τη λύση του προβλήματος όταν η προστιθέμενη ακμή συνδέει δύο μη γειτονικού κόμβους ενός spider γραφήματος ή συνδέει δύο κόμβους σε δυο διαφορετικές συνεκτικές συνιστώσες του γραφήματος, μας δίνει τη δυνατότητα να παρουσιάσουμε έναν αλγόριθμο γραμμικού χρόνου για το πρόβλημα.

Τέλος, το εφαρμοσμένο μέρος της παρούσας διατριβής επικεντρώνεται στη μελέτη των κακόβουλων τροποποιήσεων ακμών των αναγώγιμων (reducible) γραφημάτων που χρησιμοποιούνται για την κωδικοποίηση ενός ακέραιου αριθμού ως υδατογράφημα σε ένα συγκεκριμένο σύστημα κωδικοποίησης υδατογραφήματος λογισμικού. Σε οποιαδήποτε μέθοδο υδατογράφησης λογισμικού, το πιο σημαντικό μέρος είναι η επιλογή του σωστού υδατογραφήματος, δηλαδή στην προτεινόμενη μέθοδο μας, ένα υδατογράφημα, της μορφής του αναγώγιμου γραφήματος, που είναι ανθεκτικό σε επιθέσεις τροποποίησης ακμών. Μέσω της μελέτης τέτοιων τροποποιήσεων ακμών, ταξινομούμε τα υδατογραφήματα ως ισχυρά, ενδιάμεσα ή αδύναμα και είμαστε σε θέση να δώσουμε συστάσεις για την καλύτερη επιλογή υδατογραφήματος προς χρήση.

Chapter 1

Introduction

1.1 Graph Modification Problems

The study of graph modification problems is crucial to computer science. Modification problems on graphs have several applications in different areas, such as biology, mathematics, sociology, machine learning, data mining, computer vision, and many others. In graph modification problems, also known as network modification problems, we have to repair, improve, or adjust a graph to satisfy specific appropriate properties while minimizing the cost of the modification. Theoretical, these problems have been studied in a variety of classes of graphs, such as perfect graphs or reducible graphs. An important role for solving this problem in these graphs is their properties and structure.

In graph theory, the modification problems consist of:

- *vertex modification (deletion) problems*, and
- *edge modification (completion, deletion, editing) problems*.

We next briefly discuss the theoretical point of view, as well as the applications of them in several fields.

1.1.1 Edge and Vertex Modification

In this section, we give an overview of several recent results dealing with the graph modification problems, edge or vertex modification, from the angle of perfect graphs and algorithms.

A graph-based study of the modification problems can be traced to the classical work of Lewis and Yannakakis [2] in 1980. They investigated the complexity of the vertex deletion problems, where the aim is to obtain a graph that satisfies a given hereditary non-trivial property. They studied the vertex-deletion problems, which is defined as: For a fixed graph property Π, what is the minimum number of vertices which must be deleted from a given graph so that the resulting subgraph satisfies Π? They proved that the vertex-deletion problems for Π is NP-complete for directed and undirected graphs if Π is nontrivial and hereditary on induced subgraphs. Moreover, Lund and Yannakakis [3] examined the general case of hereditary problem and proved for any such property, and for every $\epsilon > 0$, the maximum induced subgraph problem cannot be approximated with ratio $2^{\log^d n}$ in quasi-polynomial time, where $d = 1/2 - \epsilon$ and n, m the number of vertices and edges, repsectively, unless $\widetilde{P} = \widetilde{NP}$.

Fujito considered a polynomial time approximation method for node-deletion problems with nontrivial and hereditary graph properties and presented a generic algorithm scheme, which can be applied to any node-deletion problem for finding approximate solutions [4]. Okun and Barak designed a $1+(\log 2)(k-1)$ approximation algorithm, by combining the local ratio and the greedy multicovering algorithms, for the problem of deleting a minimum number of nodes so that the remaining graph contains no k-bicliques [5]. In [6], approximation algorithms are developed for a few node deletion problems for non-trivial properties which can be characterized by forbidden structure, when the input is restricted to be a bipartite graph.

The edge modification problems are splited into the following problems [7]:

- The minimum completion (fill-in) problem is defined as follow: Given a property Π and a graph G, the completion problem asks the minimum number of non-edges needed to be added to G so that the resulting graph has property Π or belongs to a graph family $\mathcal{F}$. The minimum completion (fill-in) problem is NP-

complete for chain graphs (bipartite graph, which does not contain a pair of independent edges), chordal graphs, interval and unit interval graphs, which is an undirected graph G where there is an interval representation of G in which all the intervals have the same length [8, 9]. In [10], if h edges is part of the input as the problems of minimum threshold completion, weighted 2-threshold partition and weighted 2-threshold covering, then the problem of determining whether a graph G contains a threshold subgraph containing at least h edges is NP-complete.

- The edge deletion problem is defined analogously but only deletion of edges is allowed, that is the minimum number of edges whose removal of G will form from a graph with property Π or belongs to a graph family $\mathcal{F}$. In [11], El-Mallah and Colbourn showed that edge deletion problem corresponding to specific classes of graphs is NP-hard, for instance deletion problem on cograph were shown to be NP-complete. Moreover, Goldberg et al. proved that the deletion problems on interval graphs and unit interval graphs are NP-complete [9]. Yannakakis showed that the edge-deletion problem is NP-complete for the properties: without cycles of specified length l, or of any length $\leq l$, with ≥ 3, outerplanar, transitive digraph, line-invertible, bipartite (simple max-cut problem), transitively orientable (namely, comparability graph) [12]. Also, the deletion of the minimum number of edges in order to create a disjoint union of cliques is named as clique deletion problem, which was proven to be NP-complete in [13].

- In the edge editing problem, defined similarly, deletion as well as addition of edges is allowed, specifically asks the minimum set of edges you need to edit (add or delete) such that editing of these edges in the graph G will form a graph that has property Π or belongs to a graph family $\mathcal{F}$. Cirino et al. considered graph editing to the class of bipartite interval graph and they prove that the connected bipartite interval editing problem is NP-complete [14]. In [15], the editing problem is defined as splittance of a graph G in order to produce a split graph and the authors show that it is polynomial in this class of perfect graphs. Finally, Shamir et al. resulted that cluster editing (both edge additions and edge deletions are allowed) is NP-complete [16]. Cai [17] focused on the fixed-parameter tractability of the problem of deciding whether a graph can be

made into a graph with a specified hereditary property by deleting at most i vertices, at most j edges, and adding at most k edges, where i, j, k are fixed integers and he proved that this problem is fixed-parameter tractable whenever the hereditary property can be characterized by a finite set of forbidden induced subgraphs.

The edge modification problem, specifically edge completion and deletion problem, can be also defined with a parameter which will be considered constant. Let k be the edges where we add or delete from a graph $G = (V(G), E(G))$ which belongs in a class $\mathcal{C}$ and let $G' = (V(G'), E(G'))$ be the resulting graph after addition or deletion edges, i.e. $|E(G')| = |E(G)| + k$ or $|(G')| = |(G)| - k$. Hence, it is easy to be defined the problems $(\mathcal{C}, \pm k)$-MinEdgeAddition problem and $(\mathcal{C}, \pm k)$-MinEdgeDeletion problem.

- $(\mathcal{C}, +k)$-MinEdgeAddition problem: k given non-edges are added in a graph belonging to a class $\mathcal{C}$ and the aim is to compute a minimum $\mathcal{C}$-completion of the resulting graph G',
- $(\mathcal{C}, -k)$-MinEdgeAddition problem: k given edges are deleted in a graph G belonging to a class $\mathcal{C}$, where these k edges belong to $V(G)$ and the aim is to compute a minimum $\mathcal{C}$-completion of the resulting graph G' without these k edges,
- $(\mathcal{C}, +k)$-MinEdgeDeletion problem: given non-edge set E_k are added in a graph belonging to a class $\mathcal{C}$ and the aim is the minimum number of edges to be deleted to G' so that the resulting graph belongs to $\mathcal{C}$ without delete any edge of E_k,
- $(\mathcal{C}, -k)$-MinEdgeDeletion problem: k given edges are deleted in a graph G belonging to a class $\mathcal{C}$, where these k edges belong to $V(G)$ and the aim is the minimum number of edges to be deleted to G' so that the resulting graph belongs to $\mathcal{C}$.

Figure 1.1 schematically depicts the problems which are included in the general problem of graph modification problem.

1.1.2 Applications

The study of graph modification problems plays an important role in designing algorithms for solving real problems. Many types of relations and processes are modeled

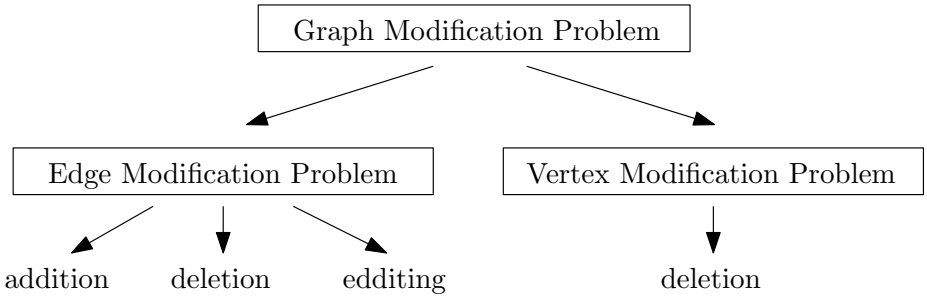

Figure 1.1: All problems which are included in the general problem of graph modification problem.

by graphs from physical, biological, social and information systems where the experimental data is very important to classify or figure with their properties. Graphs are used to represent applications to real-world systems, such as networks of communication, social network analysis, dataset of molecular biology and genomics, ecological networks, etc. As we mention above, the graph modification problems consist of edge and vertex modification problems and find application in several and different areas because the input graphs arise from experiments and edge modification approach serves to correct the few errors.

The main obstacle for analyzing and modeling data through graphs, is the lacking of data, equivalently the deletion some vertices or edges of a initial graph. On the contrary, sometimes, graphs can undergo modifications from their initial structure by a malicious user or by incorrect measurements and adjustments. The main purpose is to have the ability, regardless of the changes to the original graph, to add/delete the minimum number of appropriate elements (vertices and/or edges), in order that the final graph has similar properties to the original.

The graph modification methods change graph's structure and release the entire anonymous network and these methods allow researchers and third-parties to apply any graph-mining technique, from local to global information extraction, on anonymous data [18]. Networked systems must be resilient against the removal of network nodes, whether due to random node failure or targeted attack. The network must have optimal robustness according to a given measure, which is affected by several different strategies that alter the network by rewiring a fraction of the edges or by adding new edges [19].

Fomin et al. [20] presented some examples with application of graph modification

problems. The connectivity augmentation problem refers to enhance the network to ensure resilience against link failures, that is the addition of a few links (edges) between nodes in order to obtain a network with better connectivity. Furthermore, according to the same authors, a special case of the graph modification problem is graph clustering in which aim is to identify a set of low-cost edges (interactions) which removal partition the graph into clusters (objects). Bruckner et al. [21] suggested an algorithm to determine whether a graph is a split-cluster graph and if that is not the case, produce a forbidden subgraph in $O(n + m)$ time, developing a core periphery identification technique for protein-protein interaction (PPI) networks using graph modification.

Moreover, there are several interactions between the development of fixed-parameter algorithms and the design of heuristics for graph modification problems, featuring quite different aspects of mutual benefits [22]. Finally, the finding of the optimal elimination order to minimizing the number of fill elements can be formulated as the problem of adding the minimum number of edges to convert a graph into a chordal graph. A recurring topic in many applications is graph modification problems that produce a graph with a nice combinatorial characterisation, such as being a class of perfect graphs, i.e. interval or planar graph [20]. All above problems of graph modifications are NP-hard problems.

Rajabzadeh et al. [23] proposed a k-degree anonymization method (or genetic k-degree edge modification), in which includes partitioning of vertices and community detection in the graph in order to increase in edges for every vertex in each society and using a genetic algorithm by adding some edges between vertices in each community. In the same research area, Kiabod et al. [24] suggested a algorithm for increase the anonymization speed, using number factorization to remove the best edges from the graph in the graph modification step of the algorithm and using NaFa algorithm to add all the appropriate edges.

1.2 Basic Graph Definitions

We consider finite undirected graphs with no loops or multiple edges. For a graph G, we denote by $V(G)$ and $E(G)$ the vertex set and edge set of G, respectively. Let S be a subset of the vertex set $V(G)$ of a graph G. Then, the subgraph of G induced

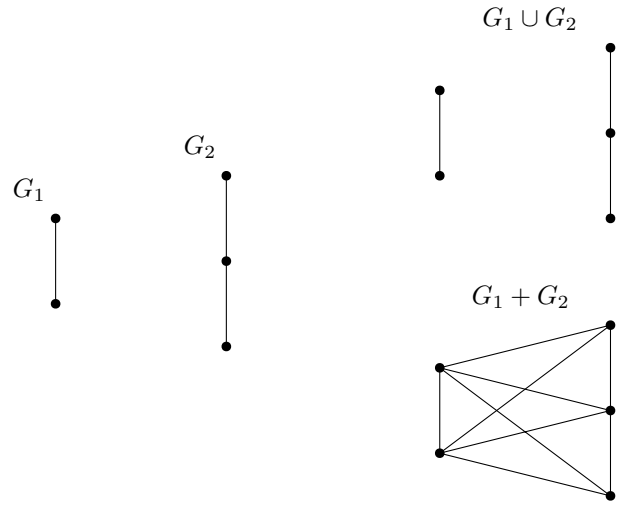

Figure 1.2: The union $G_1 \cup G_2$ and the join $G_1 + G_2$ of graphs G_1 and G_2.

by S is denoted by $G[S]$. A maximal connected subgraph of a graph G is a *connected component* of G. For any two graphs G_1 and G_2 with disjoint vertex sets, we define

- the *union* $G_1 \cup G_2$ which is the graph with vertex set $V(G_1) \cup V(G_2)$ and edge set $E(G_1) \cup E(G_2)$, and
- the *join* $G_1 + G_2$ which is the graph union $G_1 \cup G_2$ together with all the edges joining $V(G_1)$ and $V(G_2)$.

An example of union $G_1 \cup G_2$ and join $G_1 + G_2$ of graphs G_1 and G_2 is depicted in Figure 1.2.

The *neighborhood* $N(x)$ of a vertex x of the graph G is the set of all the vertices of G which are adjacent to x. The *closed neighborhood* of x is defined as $N[x] := N(x) \cup \{x\}$. The neighborhood of a subset S of vertices is defined as $N(S) := \left(\bigcup_{x \in S} N(x)\right) - S$ and its closed neighborhood as $N[S] := N(S) \cup S$. For an edge $e = xy$, the *neighborhood* (*closed neighborhood*) of e is the vertex set $N(\{x, y\})$ (resp. $N[\{x, y\}]$) and is denoted by $N(e)$ (resp. $N[e]$).

The *degree* of a vertex x in G, denoted $deg(x)$, is the number of vertices adjacent to x in G; thus, $deg(x) = |N(x)|$. A vertex of a graph is *universal* if it is adjacent to all other vertices of the graph. We extend this notion to a subset of the vertices of a graph G and we say that a subset $S \subseteq V(G)$ is *universal in* G, if every vertex in S is adjacent to every vertex in $V(G) \setminus S$.

The chordless path on k vertices is denoted by P_k. In each P_4, edge connecting the second and third vertex is called the *rib*, while the remaining two edges are called *wings*; moreover, the endpoints of the rib are called *middle* vertices of the P_4.

The coloring (of vertices) of a graph is an assignment of "colors" to its vertices such that, every two adjacent vertices always have different colors. The chromatic number of a graph is the smallest number of colors that suffice to color it. A graph is called a clique if its vertices are pairwise adjacent. The clique number of a graph is the size of the largest clique in this graph. We denote the chromatic number and the clique number of graph G by $\chi(G)$ and $\omega(G)$, respectively [25]. The chromatic number of a graph is at least its clique number, since every two adjacent vertices must receive different colors. Perfect graph is defined as a graph in which every induced subgraph H has $\chi(H) = \omega(H)$. At present, no polynominal-time algorithm to recognize perfect graph is known, although several large classes of perfect graphs, with polynomial-time, recognition algorithms, have been found [26].

1.3 Motivation

A main part of this thesis is some classes of perfect graphs and the edge modification of them. The motivation of our work is that many classes of perfect graphs arise quite naturally in real-world applications. More specifically, split, cographs, threshold, quasi-threshold, interval, permutation, P_4-sparse graphs are used to optimization of computer storage, analysis of genetic structure, hiding information, synchronization of parallel processes, etc [27].

To be specific, one application is that threshold covered partitions are unigraphic [28]. In addition, the classes of threshold and quasi-threshold graphs have applications in set-packing problems, parallel processing, and resource allocation problems [29, 30, 31]. The importance and significance of the study of P_4-sparse graphs in practical application is obvious from local density properties, which are featured in real-life applications. In particular, graphs that are unlikely to have more than a few chordless paths of length three appear in a number of contexts [32]. Applications in scheduling, clustering, and computational semantics were the driving forces behind the study of P4-sparse graphs, which attracted attention due to their natural generalization of cographs, which has a nice tree structure and bounded clique-width,

implying efficient algorithms for some problems [1, 33, 34, 35].

In [36], Nikolopoulos and Palios proposed a linear-time algorithm for the (Cograph, +1) - MinEdgeAddition problem based on the properties of the component-partition of a cograph. Given a cograph G and a non-edge xy, (Cograph, +1) - MinEdgeAddition is the problem of finding the minimum number of non-edges of G that need to be added to G so that the resulting graph is a cograph and contains xy as an edge. The same authors let the (P_4-sparse graph, +1) - MinEdgeAddition problem as an open problem since class of P_4-sparse graphs is a superclass of the class of cographs.

Today, modifications on networks are a common problem that needs to be addressed. Network diagrams can be defined as a graph, which figures the interconnections between a set of entities. The vertices of this graph are each entity and its edges are the connections between vertices. Graph modification is an attack on the integrity of network. Modification means an unauthorized party not only accesses the data but tampers it, for instance, by modifying the data packets being transmitted or causing a denial of service attack such as flooding the network with bogus data. Similarly, edge and vertex modification mean addition, deletion or editing of elements of graph.

Furthermore, the calling relationships between subroutines in a computer program are represented by a control-flow graph called a call graph (also known as a call multigraph). The most important part of a security of a call graph is its resilience to edge-modification attacks. A technique of hiding additional data or hiding information is watermarking. More precisely, the software watermarking is the embedding of a signature, i.e., an identifier reliably representing the owner, in the code. The resilience of software watermarking in the form of call graph, is proved by the resilience of call graph to attacks against edge and/or vertex modifications. The main purpose is the finding of the strongest watermark through a specific software codec watermarking system.

To sum up, the basis of motivation for this thesis is the importance of classes of perfect graphs and the graph modification problems applied in real-life applications. It is crucial to have the ability to characterize a software watermark, namely how strong the embedding watermark is in a graph.

1.4 Structure of the Thesis

This thesis consists of two main parts: The first (Chapters 2 and 3), the theoretical part studies the complexity and approximability of edge modification problems. The second (Chapter 4), applied part highlights the application of watermarking.

Chapter 1 provides an introduction to perfect graphs and an outline of the graph modification problems under consideration, while it briefly presents our research contribution. Additionally in the same chapter, we summarize preliminary remarks and basic definitions of graph theory terms, which serve as the primary corpus for our study, and conclude with the results of the thesis.

In Chapter 2, graph properties and the graph modification problem concerning the scope of the thesis are extensively described. More precisely, the fundamental structural components and properties of specific classes of perfect graphs required for solution of edge modification problem are presented, analyzed and discussed over the aspect of their utilization as also how they are combined to consist the basis of this thesis. In this chapter, we study the completion problem, especially adding a tail in some classes of perfect graphs, such as split, threshold, quasi-threshold and P_4-sparse graphs.

Next, in Chapter 3, it is presented the proposed approach for an algorithm, which solves the (P_4-sparse graph, +1)-MinEdgeAddition problem. Following the similar structure as in the previous section, the main theoretical assumptions regarding the properties of tree representation of P_4-sparse graphs are all described and extensively discussed, followed by lemmas and theorems that proved the efficiency and correctness of proposed Algorithm P_4-sparse-Edge-Addition.

In addition, Chapter 4 describes all basic parts of an already proposed software watermarking codec system (encoding watermark members as graph structures) and through edge modification theory, we characterize the watermarks according to their resilience against attacks by malicious user. Based on graph structure of the embedded watermark in software, the minimum number of edges modification is calculated in order to clasiffy the watermarks as strong, intermediate and weak in a specific range.

Chapter 5, the last chapter of the PhD thesis, summarizes the main results presented in Chapters 2 to 4, and discusses possible future extensions, also similar problems that could be studied on the horizon of further research.

1.5 Main Results

From the above partitioning of the five Chapters, it becomes obvious that the main research results of our Thesis are presented in two parts, the theoretical and applied parts. We next give the organization of the chapters of these two parts emphasizing the theoretical results and its application.

In the first theoretical part (Chapters 2 and 3), we first present the results of addition a tail to graphs, which are included in classes of perfect graphs, such as split, quasi-threshold, threshold and P_4-sparse graphs. To be specific, we study the problem of given a graph G and a tail uw, where $w \notin V(G)$ and $u \in V(G)$, and we compute the minimum number of non-edges to be added to $G + uw$. Based on properties and the structure of P_4-sparse graph and its tree representation, we first present some lemmas and theorems in order to prove the correctness of description the **Algorithm P_4-sparse-Tail-Addition**.

Moreover, we study the (P_4-sparse graph, +1)-MinEdgeAddition problem, namely given a P_4-sparse graph G and a non-edge xy (i.e., a pair of non-adjacent vertices x and y) of G, find the minimum number of non-edges of G that need to be added to G so that the resulting graph is a P_4-sparse graph and contains xy as an edge. For optimal solution H and its P_4-sparse tree T, there exist two cases: the root node of T to be 1-node, and the root node of T to be 2-node with two options, thin or thick spider. Also, we present the cases, according to what node is the least common ancestor of the leaves corresponding to u, v in $T(G)$: if it is a 0-node, then the graph G is a disconnected graph, which consists of 2 connected components each containing one of the endpoints of the added non-edge uv, and if it is a 2-node with at least one of the leaves corresponding to u, v being a child of the 2-node, the vertices u, v belong to the same spider subgraph. As a consequence of lemmas and theorems about P_4-sparse graphs that we present in the same chapter, we prove the efficiency and correctness of **Algorithm P_4-sparse-Edge-Addition**.

The applied part of this thesis is referred to graph modification to software watermarking method. We analyze a software watermarking codec system [37], which embeds a graph encoded by an integer number as watermark, in order to mention the properties of this process. In any software watermarking method, the most important part is the choice of the right watermark, that is, in our proposed method, a watermark which is resilient under edge-modification attacks. Thus, we classify each

watermark into one of three categories, which are strong, intermediate and weak watermarks in a specific range. Through this classification, we suggest the best choice of the strongest watermark, which user has to embed in his software.

Chapter 2

Adding a Tail to Classes of Perfect Graphs

2.1 Introduction

Given a graph G, an edge connecting a vertex $w \notin V(G)$ to a vertex u of G is a *tail* added to G; let us denote the resulting graph as $G + uw$. If G belongs to a class $\mathcal{C}$ of graphs, this may not hold for the graph $G + uw$. Hence, we are interested in computing a minimum $\mathcal{C}$-completion of $G + uw$, i.e., the minimum number of non-edges (in addition to the tail uw) to be added to $G + uw$ so that the resulting graph belongs to $\mathcal{C}$; such non-edges are called *fill edges*. The above problem is an instance of the more general $(\mathcal{C}, +k)$-MinEdgeAddition problem [36] in which we add k given non-edges in a graph belonging to a class $\mathcal{C}$ and we want to compute a minimum $\mathcal{C}$-completion of the resulting graph.

Yannakakis [38] defined the node (edge) deletion problem as the finding of the minimum number of nodes (edges), whose deletion results in a subgraph satisfying

property π and show that if graph property π belongs to a rather broad class of properties (the class of properties that are hereditary on induced subgraphs) then the node-deletion problem is NP-complete, and the same is true for several restrictions of it.

A related field is that of the dynamic recognition (or on-line maintenance) problem on graphs: a series of requests for the addition or the deletion of an edge or a vertex (potentially incident on a number of edges) are submitted and each is executed only if the resulting graph remains in the same class of graphs. Several authors have studied this problem for different classes of graphs and have given algorithms supporting some or all the above operations; we mention the edges-only fully dynamic algorithm of Ibarra [39] for chordal and split graphs, and the fully dynamic algorithms of Hell et al. [40] for proper interval graphs, of Shamir and Sharan [41] for cographs, of Heggernes and Mancini for split graphs [42], and of Nikolopoulos et al. for P_4-sparse graphs [34].

Given an Hermitian matrix A whose graph G is a simple undirected graph and its eigenvalues, Toyonagaet al. [43] suppose the status of each vertex in the graph is known for each eigenvalue of A. They investigate the change of the multiplicity of each eigenvalue, adding a pendent vertex with given value to a particular vertex in the graph via an edge with given weight and show how each multiplicity changes based on this information. Bevis et al. [44] examine several ways in which a single vertex and some positive number of edges can be added to a graph, and the resulting effect on the rank of the adjacency matrix of a graph G. In the general case of adding a vertex and any number of edges to an arbitrary graph G, they prove that the rank depends on the relationship of the neighborhood vector to the range space of A.

Recently, Lopes and Carvalho [45] present an integer programming formulation for solving the minimum interval graph completion problem recurring to a characterization of interval graphs that produces a linear ordering of the maximal cliques of the solution graph. Chimani et al. [46] consider the problem of computing a crossing minimum drawing of a given planar graph $G = (V, E)$ augmented by a star, i.e., an additional vertex v together with its incident edges $E_v = \{(v, u)|u \in V\}$, in which all crossings involve E_v. They prove that the star insertion problem is polynomially solvable and decribe an efficient algorithm for this problem using the SPQR-tree data structure to handle the exponential number of possible embeddings, in conjunction with dynamic programming schemes. Brandes et al. [47] propose an algorithm to

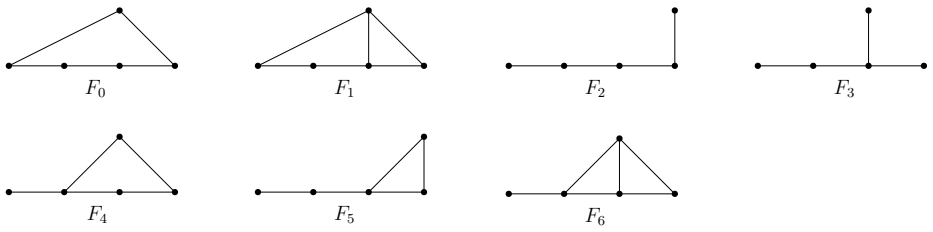

Figure 2.1: The forbidden subgraphs of the class of P_4-sparse graphs (the naming follows [1]).

solve the quasi-threshold graph editing problem with a minimum number of edge insertions and deletions, called Quasi-Threshold Mover (QTM).

Our work in this chapter also focuses on P_4-sparse graphs; the P4-sparse graphs are defined as the graphs for which every set of five vertices induces at most one chordless path on four vertices [25] (Figure 2.1 depicts the 7 forbidden subgraphs for the class of P_4-sparse graphs). The P_4-sparse graphs are perfect and also perfectly orderable [25], and properly contain many graph classes, such as, the cographs, the P_4-reducible graphs, etc. (see [48, 35, 1]). The P_4-sparse graphs have received considerable attention in recent years and they find applications in applied mathematics and computer science (e.g., communications, transportation, clustering, scheduling, computational semantics) in problems that deal with graphs featuring "local density" properties; note that the notion of local density is often associated with the absence of P_4s.

In this chapter, we study the properties of split, threshold and quasi-threshold graphs and their representation tree and we compute the minimum added edges that are needed to remain in the same class of initial graph after an added tail. Furthermore, we exploit the structure of the P_4-sparse tree of the P_4-sparse graphs in order to present an algorithm for computing a minimum P_4-sparse completion of a given graph G to which we have added a tail. Given the P_4-sparse tree of G, our algorithm runs in optimal $O(n)$ time where n is the number of vertices of G. This algorithm is the first step towards the solution of the (P_4-sparse,$+1$)-MinEdge-Addition Problem [36].

Our Contribution. In this chapter, in order to connect a node $w \notin V(G)$ to graph G by a single edge uw where $u \in V(G)$, which is named as *tail*, we compute a minimum

$\mathcal{C}$-completion of G', where the graph G' resulting from G after the addition of the tail need not belong to the class $\mathcal{C}$. More specifically, we refer to the minimum number of non-edges (in addition to the tail uw) to be added to G' so that the resulting graph belongs to $\mathcal{C}$.

We first present the above problem for the class of split graphs and we study its graph properties to compute the minimum Split-completion after addition of a tail. In class of split graphs, a split graph $G = (V, E)$ has a partition $V = K + S$ of its vertex set V into a clique (complete) set K and a independent (stable) set S. Hence, the addition of tail can be performed in two sets, in vertex set K or in vertex set S. Based on the structure of this class we figure the minimum number of edges that need to be added.

Furthermore, we study the same problem of classes of Threshold and Quasi-Threshold graphs. The main idea behind the computation of minimum Threshold-completion and (Quasi-Threshold)-completion is the structure of an A-free graph, named cent-tree $T_c(G)$, which incorporates several important properties and characteristics. The proof of minimum completion of these classes are based on properties of tree representation $T_c(G)$.

We next consider the above problem for the class of P_4-sparse graphs and we describe an algorithm for it which, given the P_4-sparse tree of the given graph G, runs in $O(n)$ time where n is the number of vertices of G.

Road Map. The chapter is organized as follows: In Section 2.2 we establish the notation and and related terminology, and present background results. In Section 2.3 we define the minimum number of edges that needed to be added in a graph G after the addition of a tail uw on a node u of G and the graph G can be split, threshold or quasi-threshold graph. In Section 2.4 we describe the algorithm for counting of added edges for a P_4-sparse completion of the graph $G + uw$ and show the main results of our work. Finally, in Section 2.5 we conclude the chapter and discuss possible future extensions.

2.2 Theoretical Framework

Some classes of perfect graphs are split, threshold, quasi-threshold and P_4-sparse graphs, which are presented below.

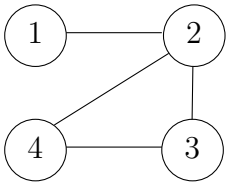

Figure 2.2: A split graph.

Split Graphs. An undirected graph $G = (V, E)$ is *split* if there is a partition $V = K + S$ of its vertex set V into a clique (complete) set K and a independent (stable) set S. These graphs were first proposed in [49], where it was shown that G is split if and only if it does not have an induced subgraph isomorphic to one of the three forbidden graphs, C_4, C_5, or $2K_2$. The complement and every induced subgraph of a split graph are split as a result of the definition (or the forbidden subgraph characterization) [50, 28]. Let $G = (V, E)$ be the graph with vertex set $V = \{1, 2, 3, 4\}$ illustrated in Fig. 2.2. Evidently, G is a split graph. One way to partition V into a disjoint union of an independent set and a clique is $V = \{1\} \cup \{2, 3, 4\}$. Other possibilities are $V = \{1, 4\} \cup \{2, 3\}$ and $V = \{1, 3\} \cup \{2, 4\}$.

Threshold Graphs. A well-known subclass of perfect graphs called threshold graphs are those whose independent (stable) vertex set subsets can be distinguished by using a single linear inequality. Equivalently, a graph $G = (V, E)$ is threshold if there exists a threshold assignment $[\alpha, t]$ consisting of a labeling α of the vertices by non-negative integers and an integer threshold t such that: S is a independent set if and only if $\alpha(v_1) + \alpha(v_2) + \cdots + \alpha(v_p) \leq t$, where $v_i \in S, 1 \leq i \leq p$, and $S \subseteq V$. Chvátal and Hammer first proposed threshold graphs in 1973 [29] and have proved the following:

Theorem 2.1. *Let $G = (V, E)$ be an undirected graph. Then, the following statements are equivalent:*

(i) G is a threshold graph.

(ii) G has no induced subgraph isomorphic to $2K_2$, P_4, or C_4.

Given a graph $G = (V, E)$, Nikolopoulos [51] defines three classes of edges in G, denoted by actual edges (AE), free edges (FE) and semi-free edges (SE) according

to relationship of the closed neighborhoods of the endpoints of its edges, where $E = FE + SE + AE$. Let $x = (u, v)$ be an edge of G. Then, $(u, v) \in FE$ if $N[u] = N[v]$, $(u, v) \in SE$ if $N[u] \subset N[v]$, $(u, v) \in AE$ if $N[u] - N[v] = \emptyset$ and $N[v] - N[u] = \emptyset$.

Definition 2.1. [51] An undirected graph $G = (V, E)$ is called A-free if every edge of G is either free or semi-free edge.

Lemma 2.1. *A graph $G = (V, E)$ is an A-free graph if and only if it contains no induced subgraph isomorphic to P_4 or C_4.*

Thus, the following result directly follows from Lemma 6 and Theorem 2.

Theorem 2.2. *The threshold graphs are precisely those A-free graphs containing no induced subgraph isomorphic to $2K_2$.*

Let $G = (V, E)$ be a connected A-free graph and is defined as
$cent(G) = x \in V(G) | N[x] = V(G)$.

Theorem 2.3. *(see [52]). The following two statements hold.*

1. *A graph G is A-free if and only if $G - cent(G)$ is an A-free graph.*
2. *Let G be a connected A-free graph. Then $cent(G) = \emptyset$. Moreover, if $G - cent(G) = \emptyset$, then $G - cent(G)$ contains at least two components.*

Theorem 2.4. *(see [52]). Let G be a connected A-free graph, and let $V(G) = V_1 + V_2 + \cdots + V_k$ be the partition of vertex set, in particular, $V_1 := cent(G)$. Then this partition and the partially ordered set $(\{Vi\}, \preceq)$ have the following properties:*

(P_1) If $V_i \preceq V_j$, then every vertex of V_i and every vertex of V_j are joined by an edge of G.

(P_2) For every V_i, $cent(G[\{\cup V_i | V_j \succeq V_i\}]) = V_i$.

(P_3) For every two V_s and V_t such that $V_s \preceq V_t$, $G[\{\cup V_i | V_s \preceq V_i \preceq V_t\}]$ is a complete graph. Moreover, for every maximal element V_t of $(\{V_i\}, \preceq)$, $G[\{\cup V_i | V_1 \preceq V_i \preceq V_t\}]$ is a maximal complete subgraph of G.

(P_4) Every edge with both endpoints in V_i is a free edge.

(P_5) Every edge with one endpoint in V_i and the other endpoint in V_j, where $V_i \neq V_j$, is a semi-free edge.

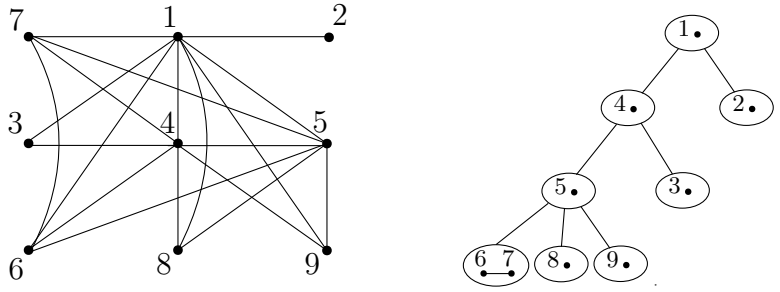

Figure 2.3: (a) Threshold Graph G. (b) The cent-tree $T_c(G)$.

The typical structure of an A-free graph, named $T_c(G)$, meets the properties of above Theorem. The elements of a cent-tree $T_c(G)$ are called nodes; that is, the vertex sets $V_{i,j}$, where $0 \leq i \leq h$ and $1 \leq j \leq k_i$, of the partition of $V(G)$ of an A-free graph G. Every node $V_{i,j}$ of the cent-tree, which is a rooted tree with root $V_{0,1}$, is either a leaf or has at least two children. Nodes $V_{i,j}$ with $j > 1$ contain only one vertex. Furthermore, $V_{i,j} \preceq V_{s,t}$ if and only if $V_{i,j}$ is an ancestor of $V_{s,t}$. If $V_{i,j}$ and $V_{s,t}$ are disjoint vertex sets of an A-free graph G, $V_{i,j}$ and $V_{s,t}$ are defined as clique-adjacent and $V_{i,j} \approx V_{s,t}$ if $V_{i,j} \preceq V_{s,t}$ or $V_{s,t} \preceq V_{i,j}$.

The cent-tree $T_c(G)$ of an A-free graph G has the following properties:

- The vertex set $K = V_{0,1} \cup V_{1,1} \cup \cdots \cup V_{h,1}$ is a clique.
- The vertex set $S = V - \{V_{0,1} \cup V_{1,1} \cup \cdots \cup V_{h,1}\}$ is an independent set.
- For every pair of nodes $x, y \in S$ such that $level(x) < level(y), N(x) \subseteq N(y)$.

In Figure 2.3, it is depicted an example of a threshold graph G and its cent-tree $T_c(G)$.

Quasi-Threshold Graphs. A graph G is called quasi-threshold, or QT-graph for short, if G contains no induced subgraph isomorphic to P_4 or C_4 (cordless path or cycle on 4 vertices) [53, 31, 54, 55]. The class of quasi-threshold-graphs is a subclass of the class of cographs and contains the class of threshold graphs [56, 57, 26, 58]. Nikolopoulos and Papadopoulos [59] have shown, among other properties, a unique tree representation of such graphs which is similar with threshold graphs $cent(G)$.

Theorem 2.5. *[52, 60] Let G be an undirected graph.*

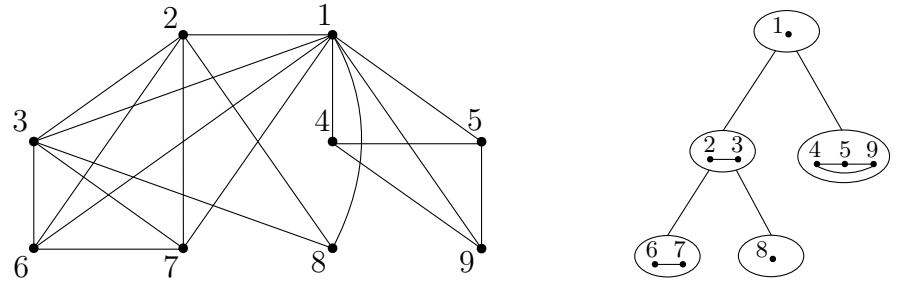

Figure 2.4: (a) Quasi-threshold Graph G. (b) The cent-tree $T_c(G)$.

1. *G is a QT-graph if and only if every connected induced subgraph $G[S], S \subseteq V(G)$, satisfies $cent(G[S] \neq \emptyset)$.*

2. *G is a QT-graph if and only if $G[V(G) - cent(G)]$ is a QT-graph.*

3. *Let G be a connected QT-graph. If $V(G) - cent(G) \neq \emptyset$, then $G[V(G) - cent(G)]$ contains at least two connected components.*

Corollary 2.1. *[52, 60] A graph G is a QT-graph if and only if G has a cent-tree $T_c(G)$.*

The structure of a cent-tree $T_c(G)$, which is a tree representation of QT-graph, is similar to tree representation of a threshold graph. Every node $V_{i,j}$ of the cent-tree $T_c(G)$, which is a rooted tree with root $V_{0,1}$, is either a leaf or has at least two children and all nodes $V_{i,j}$ can be a clique vertex set, since it contains induced subgraph isomorphic to $2K_2$. If $u \in V_{i,j}$ and p_i is a pair of (k, l) where $V_0 \preceq V_{p_1} \preceq V_{p_2} \preceq \cdots \preceq V_{p_i} \preceq V_{i,j}$, we assume, without any loss of generality, that $V_{0,1} \preceq V_{1,1} \preceq V_{2,1} \preceq \cdots \preceq V_{i-1,1} \preceq V_{i,1}$. We define all children of node $V_{i,j}$ in $T_c(G)$ as $V_{i+1,1}, V_{i+1,2}, \cdots, V_{i+1,k_{i+1}}$ (see, Figure 2.4).

P_4-sparse Graphs. As we referred above, a graph G is called P_4-sparse graph, if for every set of five vertices induces at most one cordless path on four vertices [25]. Furthermore, a graph H is called a *spider* if its vertex set $V(H)$ admits a partition into sets S, K, R such that:

- the set S is an independent (stable) set, the set K is a clique, and $|S| = |K| \geq 2$;
- every vertex in R is adjacent to every vertex in K and to no vertex in S;

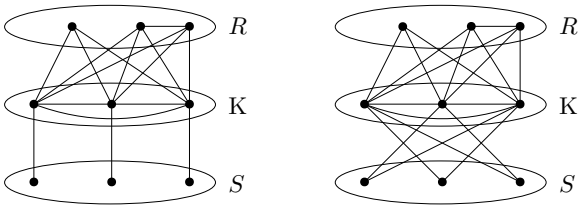

Figure 2.5: (left) A thin spider; (right) a thick spider.

- there exists a bijection $f : S \to K$ such that either $N_G(s) \cap K = \{f(s)\}$ for each vertex $s \in S$ or else, $N_G(s) \cap K = K - \{f(s)\}$ for each vertex $s \in S$; in the former case, the spider is *thin*, in the latter it is *thick*; see Figure 2.5.

The triple (S, K, R) is called the *spider partition*. Note that for $|S| = |K| = 2$, the spider is simultaneously thin and thick.

In [1], Jamison and Olariu showed that each P_4-sparse graph G admits a unique tree representation, up to isomorphism, called the P_4-*sparse tree* $T(G)$ of G which is a rooted tree such that:

(i) each internal node of $T(G)$ has at least two children provided that $|V(G)| \geq 2$;

(ii) the internal nodes are labeled by either 0, 1, or 2 (0-, 1-, 2-*nodes*, *resp.*) and the parent-node of each internal node t has a different label than t;

(iii) the leaves of the P_4-sparse tree are in a 1-to-1 correspondence with the vertices of G; if the least common ancestor of the leaves corresponding to two vertices v_i, v_j of G is a 0-node (1-node, resp.) then the vertices v_i, v_j are non-adjacent (adjacent, resp.) in G, whereas the vertices corresponding to the leaves of a subtree rooted at a 2-node induce a spider.

The structure of the P_4-sparse tree implies the following lemma.

Lemma 2.2. *Let G be a P_4-sparse graph and let $H = (S, K, R)$ be a thin spider of G. Moreover, let $s \in S$ and $k \in K$ be vertices that are adjacent in the spider.*

***P1**. Every vertex of the spider is adjacent to all vertices in $N_G(s) - \{k\}$.*

***P2**. Every vertex in $K - \{k\}$ is adjacent to all vertices in $N_G(k) - \{s\}$.*

Also the same authors [1] have shown how this tree representation of a P_4-sparse graph can be computed in linear time. Figure 2.6 shows an example of a P_4-sparse graph along with its tree representation.

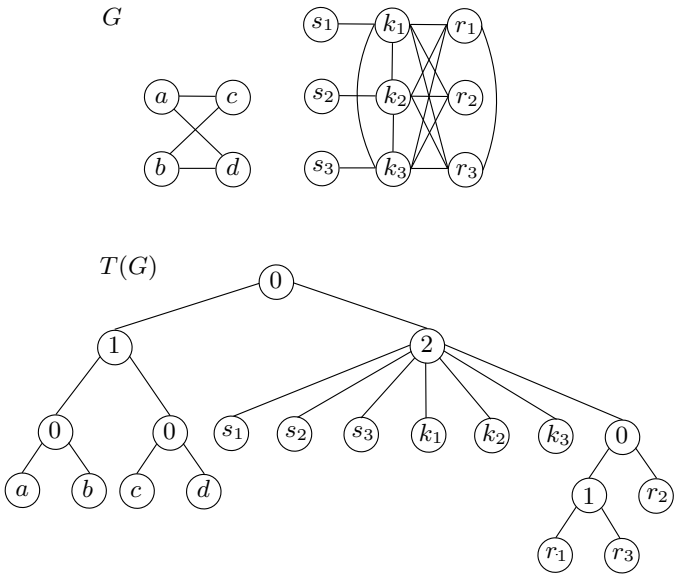

Figure 2.6: An example of a P_4-sparse graph G and the tree representation $T(G)$ of G.

Note. With a slight abuse of terminology, in the following, we will simply use the term *edges* instead of fill edges, which in fact are non-edges of the given graph.

Given the forbidden subgraphs of P_4-sparse graphs (see, Figure 2.1), an interesting problem that arises is how they can be transformed into P_4-sparse graphs with as few edges as possible. In Figure 2.7, the minimum edges that must be added to become it into a P_4-sparse graph are shown in green color. For the graph F_2 (which is isomorphic to the path P_5) there is a unique solution by adding exactly one edge. The forbidden subgraphs F_3, F_4, F_6 need the addition of 1 edge and as seen their solution is not unique, i.e. there is more than one edge choice that can become it into a P_4-sparse graph. For the F_5 graph, 2 edge additions are needed to be a P_4-sparse graph, and this pair is not unique.

In the same Figure 2.7, we show the edge additions for the House graph and for the cycle C_5, i.e., the graphs F_1 and F_0 respectively. In the same image, the edges that should be added to these two graphs so that they are in the class of P_4-sparse graphs are shown in green color. For the House graph, adding an edge is necessary in order to belong the class of P_4-sparse graphs, without the solution being unique since two nodes can become a universal node in the final graph. For the cycle C_5, i.e., the F_0 forbidden subgraph, the minimum additions of edges needed to belong to P_4-sparse

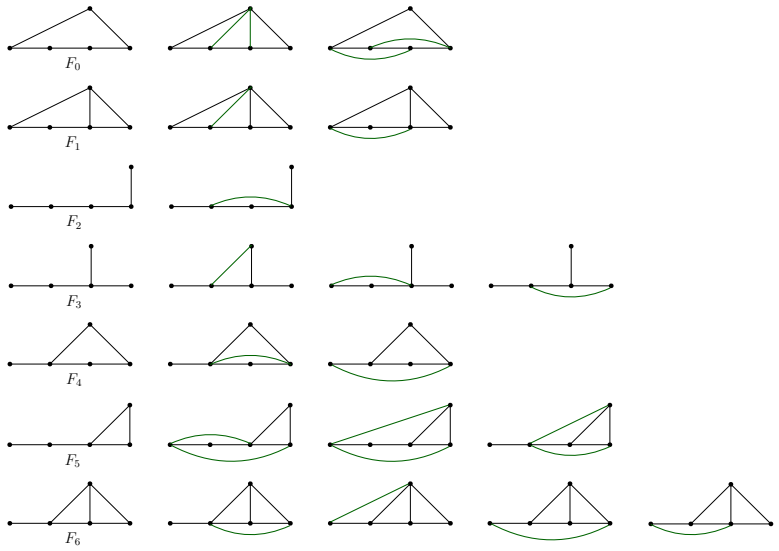

Figure 2.7: All choices of the minimum number of added edges (green edges) in order to forbidden graphs of P_4-sparse graphs to be P_4-sparse graphs.

graphs is again 2 edges, without this pair of edges being unique.

2.3 Adding a Tail to Split, Threshold and Quasi-Threshold Graphs

In this section, we present the results of calculation of the minimum number of fill edges that needed after an addition of a tail in the initial graph which is split, threshold or quasi-threshold graph and we present the following lemmas for this purpose.

Split Graphs. The first class of perfect graphs that we will study, in order to add a tail, is the split graphs. Based on its structural properties and its vertex partition, we conclude the calculation of the minimum number of fill edges needed if a tail uw is added on a node u of G.

Let G be a given graph to which we want to add the tail uw with $u \in V(G)$.

Lemma 2.3. *Let $G = (V, E)$ be a split graph with vertex partition $V = K + S$, where K is a clique (complete) set and S is a independent (stable) set. Consider the addition of a tail uw incident on a node u of G. Then, for a minimum split completion of the graph $G + uw$, it holds:*

1. *If $u \in K$, the minimum number of fill edges needed (in addition to the tail uw) is 0.*

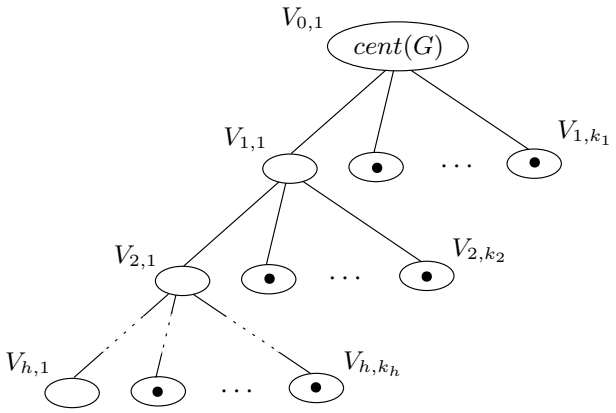

Figure 2.8: The structure of an A-free graph which contains no induced subgraph isomorphic to $2K_2$.

2. *If $u \in S$ and $degree(u) = k_u$, the minimum number of fill edges needed (in addition to the tail uw) is $|K| - k_u$.*

Proof. 1. No fill edge (in addition to uw) is needed, since the addition of uw yields a new split graph $G' = (V', E')$, where $E' = E \cup \{uw\}$ and $V' = S' + K$ with independent set $S' = S \cup \{w\}$.

2. Let $G' = (V', E')$ be a new split graph after the addition of the minimum number of fill edges and let the vertex partition be $V' = S' + K'$. We consider three cases:

(i) $u \in K'$ and $w \in S'$. In this case, the number of fill edges (in addition to the tail uw) needed is $|K| - degree(u)$ since the node u needs to be adjacent to all vertices of K.

(ii) $u \in S'$ and $w \in K'$. Then, $|K|$ edges need to be added.

(iii) $u, w \in K'$. It means that $S' = S$ and $K = K' \cup \{u, w\}$; which implies that, $2|K| - degree(u)$ fill edges are needed.

Hence, the minimum number of edges (in addition to the tail uw) needed is $|K| - k_u$, where $k_u = degree(u)$ and $u \in K'$, $w \in S'$.

□

Threshold Graphs. Based on the structural of an A-free graph G and its tree representation, cent-tree $T_c(G)$, we show how we can calculate the minimum number of edges that need to be added in $G + uv$ graph, where uv is a tail.

Lemma 2.4. *Let $G = (V, E)$ be a A-free graph, where $V(G) = K \cup S$ and let $T_c(G)$ be its cent-tree. Consider the addition of a tail uw incident on a node u of G. Then, there exists a minimum A-free completion of the graph $G + uw$ then:*

(i) If $u \in V_{i,1} \subseteq K$, the minimum number of fill edges needed (in addition to the tail uw) is $\min_{0 \leq l \leq i} A(l)$ where

$$A(l) = \sum_{j=0}^{l-1} |V_{j,1}| + \sum_{w=l+1}^{i} \sum_{j=2}^{k_w} |V_{w,j}|.$$

(ii) If $u \in V_{i,j} \subseteq S$, where $2 \leq j \leq k_i$, the minimum number of fill edges needed (in addition to the tail uw) is $\min(A_m, \sum_{j=0}^{i-1} |V_{j,1}| + \sum_{w=i+1}^{h} \sum_{j=1}^{k_w} |V_{w,j}| + |V_{i,1}|)$, where $A_m = \min_{0 \leq l < i} A(l)$ and

$$A(l) = \sum_{j=0}^{l-1} |V_{j,1}| + \sum_{w=l+1}^{i-1} \sum_{j=2}^{k_w} |V_{w,j}| + \sum_{w=i}^{h} \sum_{j=1}^{k_w} |V_{w,j}| - 1$$

.

Proof. 1. Let $u \in V_{i,1} \subseteq K$ and add the tail uw. We can get a A-free graph $G' = (V', E')$ with following structure of $T_c(G')$ after having added the edges $|V_{0,1}| + |V_{1,1}| + \cdots + |V_{i-1,1}|$ such that for all $V'_{j,1} = V_{j,1}$, where $0 \leq l < i$ and $V'_{i,1} = \{u\}$, $V'_{i+1,1} = V_{i,1} \setminus \{u\}$ and $V'_{j+2,1} = V_{j,1}$, where $i < j \leq h$. All independent sets remain with the same number of vertices and degree except $V'_{i+1,2} = \{w\}$. Also, we can get a A-free graph $G' = (V', E')$ after having added the edges $|V_{0,1}| + |V_{1,1}| + \cdots + |V_{i-2,1}| + \sum_{j=2}^{k_i} |V_{i,j}|$. Aiming the minimality of this solution, we have to minimize the edges from vertices u and w. Thus, the minimum number of fill edges needed (in addition to the tail uw) is $\min_{0 \leq l \leq h} A(l)$ where $A(l) = \sum_{j=0}^{l-1} |V_{j,1}| + \sum_{j=2}^{k_{l+1}} |V_{l+1,j}| + \sum_{j=2}^{k_{l+2}} |V_{l+2,j}| + \cdots + \sum_{j=2}^{k_i} |V_{i,l}| = \sum_{j=0}^{l-1} |V_{j,1}| + \sum_{w=l+1}^{i} \sum_{j=2}^{k_w} |V_{k,j}|$.

2. Let $u \in V_{i,j} \subseteq S$, where $2 \leq j \leq k_i$ and add the tail uw. We can get a A-free graph $G' = (V', E')$, if the vertex u is the only one member of clique set $V'_{i,1}$ of cent-tree $T_c(G')$, and it holds, with this solution, the total fill edges to be $\sum_{j=0}^{i-1} |V_{j,1}| + \sum_{w=i+1}^{h} \sum_{j=1}^{k_w} |V_{w,j}| + |V_{i,1}|$. In other case, if the cent-tree of G' has the following structure $V'_{i+1,1} = \{u\}$ and $V'_{i+2,1} = V_{i+1,1}$ and $V'_{i+2,2} = \{w\}$, then the number of fill edges needed (in addition to the tail uw) is increased by $|V_{i,1}|$. Thus, the minimum number of fill edges needed will be resulted by vertex u, which belongs

to a clique set $V'_{l,1}$, where $0 \leq l \leq i$. If $l = i$, it holds that the minimum number of fill edges needed is $\sum_{j=0}^{i-1} |V_{j,1}| + \sum_{w=i+1}^{h} \sum_{j=1}^{k_w} |V_{w,j}| + |V_{i,1}|$ and if $0 \leq l < i$, it holds $A_l = \sum_{j=0}^{l-1} |V_{j,1}| + \sum_{w=l+1}^{i-1} \sum_{j=2}^{k_w} |V_{w,j}| + \sum_{w=i}^{h} \sum_{j=1}^{k_w} |V_{w,j}| - 1$. Thus the minimum number of fill edges needed is $A_m = \min(A_m, \sum_{j=0}^{i-1} |V_{j,1}| + \sum_{w=i+1}^{h} \sum_{j=1}^{k_w} |V_{w,j}| + |V_{i,1}|)$, where $A_m = \min_{0 \leq l < i} A(l)$. □

Quasi-threshold Graphs. Let G be a given quasi-threshold graph and we want to add a tail uw where $u \in V(G)$ and $w \notin V(G)$. We can easily prove the following lemma about minimum number of fill edges.

Lemma 2.5. *Let $G = (V, E)$ be a QT-graph, and let $T_c(G)$ be its cent-tree. Consider the addition of a tail uw incident on a node u of G. Then, there exists a minimum QT completion of the graph $G + uw$, and if $u \in V_{i,j}$, the minimum number of fill edges needed (in addition to the tail uw) is $\min(A_m, \sum_{w=0}^{i-1} |V_{w,1}| + |V_{i,j}| - 1)$, where $A_m = \min_{0 \leq l < i} A(l)$ and $A(l) = \sum_{j=0}^{l-1} |V_{j,1}| + \sum_{w=l}^{h} \sum_{j=2}^{k_w} |V_{w,j}| + \sum_{w=i}^{h} |V_{w,1}| - |V_{i,j}| - 1$.*

Proof. It is easy to see that this case of QT-graph can be proved with similar way of Lemma 7.(i). In the case where a tail uw is added, where $u \in V_{i,j}$, and there exists a new induced subgraph isomorphic to $2K_2$ included the new node w, then the number of edges that need to be added is $\sum_{w=0}^{i-1} |V_{w,1}||V_{i,j}| -$. Therefore, the minimum number of edges (in addition to the tail uw) that need to be added is the minimum value of case that created a new induced subgraph isomorphic to $2K_2$ and the case that created a new vertex set $V'_{l,1} = u$ and $V'_{l+1,2} = w$ where $0 \leq l < i$. □

2.4 Adding a Tail to a P_4-sparse Graph

Let G be a given graph to which we want to add the tail uw with $u \in V(G)$. Let $t_0 t_1 \cdots t_h u$ be the path from the root in the P_4-sparse tree T_G of G to u and let V_i $(0 \leq i < h)$ be the set of vertices associated with the leaves of the subtrees rooted at the children of t_i except for t_{i+1} and V_h be the set of vertices associated with the leaves of the subtrees rooted at the children of t_h except for u (see Figure 2.9).

We first show that there exist minimum P_4-sparse completions of the graph $G+uw$ in which u and w appear together in a small number of different formations.

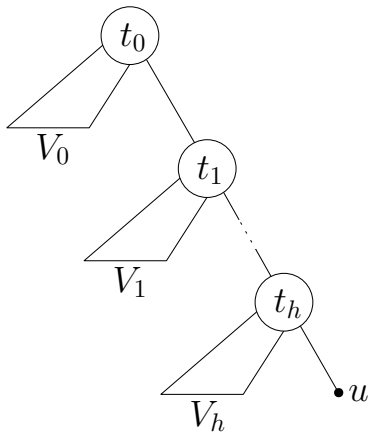

Figure 2.9: The P_4-sparse tree of the given graph G in terms of the leaf corresponding to vertex u.

Lemma 2.6. *Let G be a P_4-sparse graph and T_G be its P_4-sparse tree. Consider the addition of a tail uw incident on a node u of G. Then, there exists a minimum P_4-sparse completion of the graph $G + uw$ such that for the P_4-sparse tree $T_{G'}$ of the resulting graph G', one of the following three cases holds:*

1. *The nodes u, w in $T_{G'}$ have the same parent-node which is a 2-node corresponding to a thin spider with partition (S, K, R) with $u \in K$ and $w \in S$.*

2. *The P_4-sparse tree $T_{G'}$ contains a 3-treenode subtree with an 1-node t at the root and the nodes for u and w as children (see Formation 1 in Figure 2.10). Moreover, the parent-node of t, if it exists, is either a 0-node or a 2-node corresponding to a spider (S, K, R) where $R = \{u, w\}$.*

3. *The P_4-sparse tree $T_{G'}$ is the same as T_G without the node for u except that an 1- or a 2-node t in the path from the root of T_G to the node for u is replaced by a 5-treenode subtree with an 1-node at the root with children the node for u and a 0-node which in turn has children the node for w and the node t (see Formation 2 in Figure 2.10).*

Proof. Let G_{OPT} be the P_4-sparse graph that is a minimum P_4-sparse completion of the graph $G + uw$ and let T_{OPT} be its P_4-sparse tree. We consider the following cases:
A. *The leaves associated with u, w in T_{OPT} have the same parent-node*: Then, since u, w are adjacent, the parent-node is either a 1-node or a 2-node.

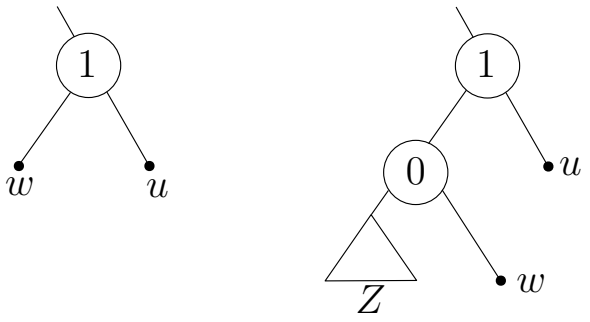

Figure 2.10: (left) Formation 1; (right) Formation 2 (if the root node of the P_4-sparse tree for Z is a 0-node than it is merged with its parent 0-node).

(i) *The parent-node of u, w in T_{OPT} is a 2-node*: Let $H = (S, K, R)$ be the corresponding spider. The spider H cannot be thick with $|K| \geq 3$: if H were a thick spider, no matter whether the tail uw was an S-K, K-K, or R-K edge, the sum of degrees of u, w would be at least $|V(H)| - 2 + |K| - 1$ (consider a vertex in K and a vertex in S); however, we would have added fewer edges to $G + uw$ if we made u universal in H and then replaced it by Formation 2 with $Z = V(H) - \{u, w\}$. The same holds if H is a thin spider and the edge uw were a K-K or R-K edge, whereas if it were an S-K edge with $u \in S$ and $w \in K$, then we can exchange u and w for the same total number of added edges. Thus, we have a thin spider with $u \in K$ and $w \in S$.

(ii) *The parent-node of u, w in T_2 is a 1-node*: Then, if t is the parent-node of u, w, the leaves associated to u and w are the only children of t (Formation 1), otherwise we can use Formation 2 as shown in Figure 2.11 for a smaller number of added edges. Additionally, the parent-node of t in T_{OPT} is a 0-node or a 2-node in which case the set R of the corresponding spider is precisely $\{u, w\}$.

B. *The nodes for u, w in T_{OPT} do not have the same parent-node*: Let T_R be the P_4-sparse tree obtained from T_{OPT} by using Formation 2 as shown in Figure 2.12. Then, the graph corresponding to T_R uses no more added edges than T_{OPT}: note that because w is adjacent to u, in T_{OPT}, w is also adjacent to all vertices in Z whereas in T_R, w is not adjacent to any vertex in Z but u is adjacent to all of them.

Now, in the right P_4-sparse tree T_R in Figure 2.12, let T_A be its part resulting after the removal of the subtree rooted at the 1-node that is the parent of the leaf u and let T_Z be the subtree corresponding to Z; if A is the set of vertices of the graph G

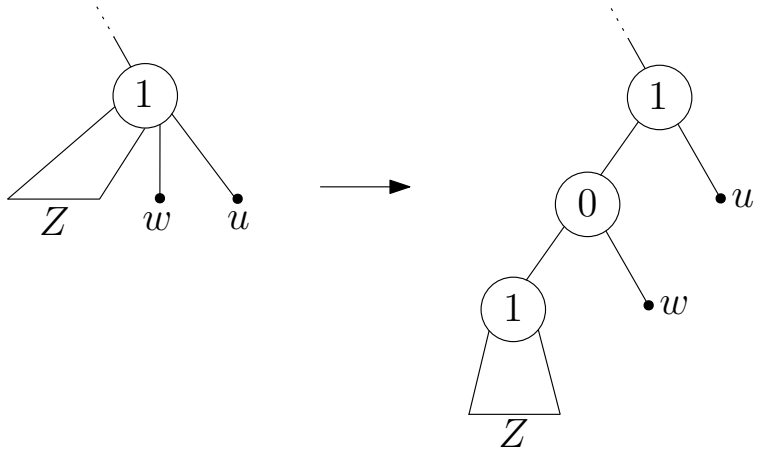

Figure 2.11: A transformation that saves added edges.

associated with the leaves of T_A, clearly, A and Z partition the set of vertices $V(G) - \{u\}$.

Recall that $t_0 t_1 \cdots t_h u$ is the path from the root to u in the P_4-sparse tree T_G of G and V_i is the set of vertices associated with the leaves of the subtrees rooted at the children of t_i except for t_{i+1} (where $t_{h+1} = u$); see Figure 2.9.

Next, we show that we can construct a P_4-sparse tree containing a Formation 2 in which the part above the Formation 2 consists of the subpath $t_0 \cdots t_p$ for $0 \leq p < h$ with the associated complete subtrees for the V_is in T_G and the part below the Formation 2 consists of the remaining subpath $t_{p+1} \cdots t_h$ with the associated complete subtrees in T_G. Suppose for contradiction that this is not the case. First, consider a neighbor x of u in G such that $x \in V_j$, $x \notin A$ and $(V_j \cup \cdots \cup V_h) \cap A \neq \emptyset$; in fact, without loss of generality, let x be such a vertex belonging to the *lowest-index* such set V_j. Since u, x are neighbors in G, then because of the P_4-sparse tree T_G, x is adjacent to all the vertices in $V_{j+1} \cup \cdots \cup V_p$. Since $x \in Z$, then in T_R, u is adjacent to all vertices in $(V_{j+1} \cup \cdots \cup V_p) \cap A$, otherwise x would not be adjacent to these vertices, a contradiction. Then, since u is adjacent to all of them, then we can change T_R by moving all these vertices in Z and replace T_Z by the P_4-sparse tree of the induced subgraph $G[Z]$. Additionally, there are no vertices in $(V_0 \cup \cdots \cup V_j) - N_G(u)$ in Z, because if they were, we could move them in A and replace T_A by the P_4-sparse tree of the induced subgraph $G[A]$, thus reducing the number of added edges. Finally, in the worst case, there may exist q such that for each $i = 0, \ldots, q-1$ it holds that $V_i \subseteq A$, and for each $i = q+1, \ldots, h$ it holds that $V_i \subseteq Z$ whereas $V_q \cap A \neq \emptyset$ and $V_q \cap Z \neq \emptyset$.

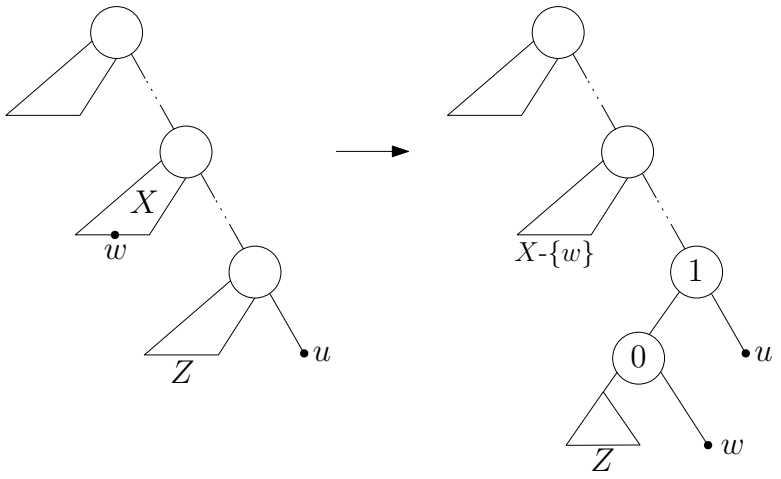

Figure 2.12: (left) The P_4-sparse tree T_{OPT} in which the leaves associated with u, w do not have the same parent; (right) The P_4-sparse tree T_R obtained by using Formation 2 that uses no more added edges than T_{OPT}.

Then, if t_q is a 0-node, we move all vertices in V_q to A and reduce the number of added edges whereas if t_q is a 1- or 2-node, we move all vertices in V_q to Z, since the number of non-neighbors of u among the elements of V_q is no more than the number of neighbors (thus, it is worth adding edges connecting u to its non-neighbors in V_q than adding edges connecting w to u's neighbors in V_q). The final P_4-sparse tree is obtained using the P_4-sparse tree of $G[A \cup \{u\}]$ and of $G[Z]$. □

It is interesting to note that Formation 1 is a special case of Formation 2 when $Z = \emptyset$.

2.4.1 A Special Case

In this section, we consider a special case that will be useful for our algorithm for the general problem. In particular, we consider that the given graph is a spider $H = (S_H, K_H, R_H)$ and we add a tail uw to a vertex u of H such that the parent-node of u in the P_4-sparse tree T_H of H is the 2-node corresponding to H. We distinguish the cases whether H is thin or thick.

Lemma 2.7. *Suppose that the spider $H = (S_H, K_H, R_H)$ is thin. Then:*

1. *If $u \in S_H$, the minimum number of fill edges needed (in addition to the tail uw) is $|K_H| - 1$ if $R_H = \emptyset$ and K_H otherwise.*

2. *If $u \in K_H$, the minimum number of fill edges needed (in addition to the tail uw) is $|K_H| - 1$.*

3. *If $u \in R_H$, then $R_H = \{u\}$ and no edge (in addition to the tail uw) needs to be added.*

Proof. 1. Let $v \in K_H$ be the neighbor of u in H. Then, we can get a P_4-sparse graph as follows: if $R_H = \emptyset$, we connect u to all vertices in $K_H - \{v\}$ (we get a thin spider with $S = (S_H - \{u\}) \cup \{w\}$, $K = (K_H - \{v\}) \cup \{u\}$, and $R = \{v\}$, that is, the tail uw is a leg of a P_4 of a thin spider), otherwise we connect v to all vertices in $\{w\} \cup (S_H - \{u\})$, which makes v universal in $V(H) \cup \{w\}$ and u, w are moved to the set R where they form a separate connected component; the total number of added edges (excluding the edge uw) is precisely $|K_H| - 1$ if $R_H = \emptyset$ and K_H otherwise.
Moreover, this is the minimum number of edges (in addition to the tail uw) that need to be added. First, we note that for each pair k_i, s_i where $k_i \in K_H - \{v\}$ and $s_i \in S_H - \{u\}$, the vertices v, u, w, k_i, s_i define an F_5 or an F_3 depending on whether the vertices v, w are adjacent or not, which implies that at least $|K_H| - 1$ added edges (in addition to the tail uw) are needed. Then, if there is a way of getting a P_4-sparse graph by adding fewer than the number of added edges mentioned in case 1 of the statement of the lemma, it has to be the case that (i) $R_H \neq \emptyset$, (ii) each pair k_i, s_i where $k_i \in K_H - \{v\}$ and $s_i \in S_H - \{u\}$ is incident on exactly 1 added edge, and (iii) no more edges are added. Let $r \in R_H$ and $k \in K_H - \{v\}$. Then, the vertices v, u, w, k, r induce a en subgraph (an F_5 if k is non-adjacent to both u, w, or an F_6 (F_1, resp.) if k becomes adjacent to u (w, resp.) by means of an added edge); thus, at least K_H added edges are needed in this case.

2. Let $v \in S_H$ be the neighbor in H of $u \in K_H$. Then, by connecting u to all vertices in $S_H - \{v\}$ or by connecting w to all vertices in $K_H - \{u\}$ yields a P_4-sparse graph. Moreover, this is the minimum number of edges (in addition to the tail uw) that need to be added. Suppose, for contradiction, that we get a P_4-sparse graph after having added fewer than $|K_H| - 1$ edges (in addition to the edge uw) to the thin spider H. Then, there exists a pair of adjacent vertices s, k with $s \in S_H - \{v\}$ and $k \in K_H - \{u\}$ such that neither s nor k is incident on any of the added edges. Then the vertices u, v, w, s, k induce a forbidden subgraph F_5 if w and v are adjacent and a forbidden subgraph F_3 if they are not; a contradiction.

3. No edge (in addition to uw) needs to be added, since the addition of uw yields a thin spider with $S = S_H \cup \{w\}$, $K = K_H \cup \{u\}$, and $R = \emptyset$. □

In turn, for a thick spider we have the following lemma.

Lemma 2.8. *Suppose that the spider $H = (S_H, K_H, R_H)$ is thick and $|K_H| \geq 3$. Then:*

1. *If $u \in S_H$, the minimum number of fill edges needed (in addition to the tail uw) is*
   - $|K_H|-1 = 2$ *if* $|K_H| = 3$ *and* $R_H = \emptyset$;
   - $|K_H| = 3$ *if* $|K_H| = 3$ *and* $|R_H| = 1$;
   - $|K_H|+1 = 4$ *if* $|K_H| = 3$ *and* $|R_H| \geq 2$;
   - $|K_H|$ *if* $|K_H| \geq 4$ *and* $R_H = \emptyset$;
   - $|K_H| + 1$ *if* $|K_H| \geq 4$ *and* $|R_H| \geq 1$.
2. *If $u \in K_H$, the minimum number of fill edges needed (in addition to the tail uw) is* 1.
3. *If $u \in R_H$, then, $R_H = \{u\}$, and the minimum number of fill edges needed (in addition to the tail uw) is $|K_H|$.*

Proof. 1. Let $v \in K_H$ be the non-neighbor of u in H. Let us first consider the case $|K_H| = 3$. If $|R_H| \leq 2$, we can get a P_4-sparse graph after having added the edges vu and vw (this implies that v becomes universal in $(V(H) - \{v\}) \cup \{w\}$) and the edge connecting u to the vertices in R_H if R_H is non-empty (then the vertices in $(V(H)-\{v\})\cup\{w\}$ induce a thin spider with $K = (K_H-\{v\})\cup\{u\}$, $S = (S_H-\{u\})\cup\{w\}$, and $R = R_H$, for a total of $|K_H| - 1 + |R_H|$ added edges (excluding the edge uw). If $|R_H| \geq 2$[1], a P_4-sparse graph is obtained after in addition to the tail uw vu and vw (again v is universal in $(V(H) - \{v\}) \cup \{w\}$) and the edges connecting w to the vertices in $K_H - \{v\}$ (then the vertices in $(V(H) - \{v\}) \cup \{w\}$ induce a thin spider with $K = K_H - \{v\}$, $S = S_H - \{u\}$, and $R = R_H \cup \{u, w\}$), for a total of $|K_H| + 1$ added edges (in addition to uw).

Now, consider the case that $|K_H| \geq 4$. If $|R_H| \leq 1$, we get a P_4-sparse graph after having made u universal by adding $|K_H| - 1$ edges to the remaining vertices in S_H, the edge uv, and the edge connecting u to the vertex in R_H if R_H is non-empty, for a total of $|K_H| + |R_H|$ added edges (in addition to uw). If $|R_H| \geq 1$[2], a P_4-sparse graph is obtained after having made v universal (by adding the edges vu and vw) and then having connected w to all vertices in $K_H - \{v\}$ (then the vertices in $(V(H)-\{v\})\cup\{w\}$ induce a thick spider with $K = K_H - \{v\}$, $S = S_H - \{u\}$, and $R = R_H \cup \{u, w\}$) for a total of $|K_H| + 1$ added edges (in addition to uw).

[1] For $|R_H| = 2$, we have two optimal solutions.

[2] For $|R_H| = 1$, we have two optimal solutions.

Below we show the minimality of this solution. Let $v \in K_H$ be the non-neighbor of u in H. We consider each of the five cases.

(i) $|K_H| = 3$ *and* $R_H = \emptyset$: Suppose, for contradiction, that there is a solution with at most $|K_H| - 2 = 1$ added edge (in addition to uw). Then, there exists at least one vertex $s \in S_H - \{u\}$ that is not incident on this added edge. If v is incident on the unique added edge (which connects v to u or w), then the vertices u, v, w, s, s' (where $\{s'\} = S_H - \{u, s\}$) induce an F_3. If v is not incident on the added edge, then the vertices u, v, w, s, k (where $k \in K_H$ is the non-neighbor of s in H) induce an F_5 if k, w are connected by the added edge, or an F_2 otherwise.

(ii) $|K_H| = 3$ *and* $|R_H| = 1$: Let $R_H = \{r\}$. Suppose, for contradiction, that there is a solution with at most $|K_H| - 1 = 2$ added edge (in addition to uw). We distinguish three cases depending on whether v is incident on 0, 1, or 2 added edges:

- v *is not incident on an added edge*: If there exists a pair s, k of non-neighbors with $s \in S_H - \{u\}$ and $k \in K_H - \{v\}$ such that none of s, k is incident on an added edge to u or w, the vertices u, v, w, s, k induce an F_2. Otherwise, since the number of such pairs is 2, each such pair s, k is incident on 1 added edge to u or w, and no other added edges exist. If there exists a vertex $k \in K_H - \{v\}$ not incident on an added edge to w, the vertices u, v, w, k, r induce an F_5, otherwise each of the added edges connects w to each of the vertices in $K_H - \{v\}$ to w and then u, v, w, s, k (for any pair s, k of non-neighbors with $s \in S_H - \{u\}$ and $k \in K_H - \{v\}$) induce an F_6.
- v *is incident on* 1 *added edge (to* u *or* w*)*: Then, there is 1 more added edge; hence, there exist 2 vertices in the set $(S_H - \{u\}) \cup \{r\}$ that are not incident on an added edge connecting them to u or w, and let these vertices be p_1, p_2. Then, the vertices u, v, w, p_1, p_2 induce an F_5 if p_1, p_2 are connected by an added edge or an F_3 otherwise.
- v *is incident on* 2 *added edges connecting it to* u *and* w: Then, there is no other added edge. Then, the vertices u, w, k, k', r (where $\{k, k'\} = K_H - \{v\}$) induce an F_6.

(iii) $|K_H| = 3$ *and* $|R_H| \geq 2$: Let r_1, r_2 be two vertices in R_H. Suppose, for contradiction, that there is a solution with at most $|K_H| = 3$ added edge (in addition to

uw). Again, we distinguish three cases depending on whether v is incident on 0, 1, or 2 added edges:

- *v is not incident on an added edge*: Consider the case that there exists a vertex $k \in K_H - \{v\}$ that is not incident on an added edge to u or w. Let $s \in S_H$ be the non-neighbor of k in H and $A = (S_H - \{u, s\}) \cup \{r_1, r_2\}$; the set A contains 3 vertices which are common neighbors of v, k. If at least one of these 3 vertices (say, p) is not incident on an added edge to u, w, then the vertices u, v, w, k, p induce an F_5, otherwise all 3 of these vertices are incident on an added edge to u, w (then these are all the added edges) and the vertices u, v, w, s, k induce an F_2. On the other hand, if no such vertex k exists, then both vertices in $K_H - \{v\}$ are incident on an added edge to w, accounting for 2 of the 3 added edges; then there exists a vertex $s' \in S_H - \{u\}$ that is not incident on an added edge and the vertices u, v, w, s', k' (where $k' \in K_H$ is the non-neighbor of s') induce an F_5.
- *v is incident on 1 added edge (to u or w)*: There are 2 more added edges; hence, there exist 2 vertices in the set $(S_H - \{u\}) \cup \{r_1, r_2\}$ that are not incident on an added edge connecting them to u or w, and let these vertices be p_1, p_2. Then, the vertices u, v, w, p_1, p_2 induce an F_5 if p_1, p_2 are connected by an added edge or an F_3 otherwise.
- *v is incident on 2 added edges connecting it to u and w*: Then, there is 1 more added edge; hence, there exists a vertex $k \in K_H - \{v\}$ that is not incident on the added edge. Moreover, there exist 2 vertices in the set $(S_H - \{u, s\}) \cup \{r_1, r_2\}$ that are not incident on an added edge connecting them to u or w (where $s \in S_H$ is the non-neighbor of k); let these vertices be p_1, p_2. Then, the vertices u, w, k, p_1, p_2 induce an F_5 if p_1, p_2 are connected by an added edge or an F_3 otherwise.

(iv) *$|K_H| \geq 4$ and $R_H = \emptyset$*: Suppose, for contradiction, that there is a solution with at most $|K_H| - 1$ added edge (in addition to the tail uw). Again, we distinguish three cases depending on whether v is incident on 0, 1, or 2 added edges:

- *v is not incident on an added edge*: If there exists a vertex $s \in S_H - \{u\}$ not incident on an added edge to u or w, the vertices u, v, w, s, k (where $k \in K_H$ is the non-neighbor of s in H) induce an F_5 if k, w are connected by an

added edge, or an F_2 otherwise; if all vertices in $S_H - \{u\}$ are incident on an added edge to u or w, then there are no more added edges and the vertices u, v, w, k, k' (for any $k, k' \in K_H - \{v\}$) induce an F_6.

- *v is incident on 1 added edge (to u or w)*: Then, the remaining added edges are at most $|K_H| - 2$ in total. If there exist two vertices $s, s' \in S_H - \{u\}$ not incident on an added edge to u or w, the vertices u, v, w, s, s' induce an F_5 or an F_3 depending on whether s, s' are adjacent or not. Otherwise, the remaining $|K_H| - 2$ added edges connect each of $|K_H| - 2$ vertices in $S_H - \{u\}$ to u or w; let s be the unique vertex in $S_H - \{u\}$ not incident on an added edge. Then, the vertices u, v, w, s, k' (where $k' \in K_H - \{v\}$ is a neighbor of s in H) induce an F_6 or an F_1 if the added edge incident on v connects it to u or w respectively.
- *v is incident on 2 added edges connecting it to u and w*: Then, the remaining added edges are at most $|K_H|-3$ in total; hence, there exist two pairs of non-adjacent vertices s_1, k_1 and s_2, k_2 with $s_1, s_2 \in S_H - \{u\}$ and $k_1, k_2 \in K_H - \{v\}$ such that none of s_1, s_2, k_1, k_2 is incident on an added edge to w or u. Let $A = S_H - \{u, s_1, s_2\}$; the set A is the set of $|K_H| - 3$ common neighbors of k_1, k_2 in S_H other than u. If there exists a vertex $s \in A$ not incident on an added edge to u or w, then the vertices u, w, k_1, k_2, s induce an F_6, otherwise, the remaining $|K_H| - 3$ added edges connect each of the vertices in A to u or w, that is, none of the vertices in $K_H - \{v\}$ is incident on an added edge. Then, the vertices u, w, s_1, s_2, k (where k is any vertex in $K_H - \{v, k_1, k_2\}$) induce an F_3.

(v) *$|K_H| \geq 4$ and $|R_H| \geq 1$*: Let $r \in R_H$. Suppose, for contradiction, that there is a solution with at most $|K_H|$ added edge (in addition to the tail uw). Again, w distinguish three cases depending on whether v is incident on 0, 1, or 2 added edges:

- *v is not incident on an added edge*: If there exists a vertex $s \in S_H - \{u\}$ not incident on an added edge to u or w, the vertices u, v, w, s, k (where $k \in K_H$ is the non-neighbor of s in H) induce an F_5 if k, w are connected by an added edge, or an F_2 otherwise; if all vertices in $S_H - \{u\}$ are incident on an added edge to u or w, which account for the $|K_H| - 1$ of the $|K_H|$ added edges, there exist vertices $k, k' \in K_H - \{v\}$ which are not incident on an

added edge and then the vertices u, v, w, k, k' induce an F_6.

- *v is incident on 1 added edge (to u or w)*: Then, the remaining added edges are at most $|K_H| - 1$ in total. If all vertices in $K_H - \{v\}$ are incident on an added edge to w, then no more added edges exist and the vertices u, v, w, s, s' (for any $s, s' \in S_H - \{u\}$) induce an F_3. So, assume that there exists $k \in K_H - \{v\}$ which is not incident on an added edge to w. The number of common neighbors of v, k in $(S_H - \{u\}) \cup r$ is $|K_H| - 1$. If each of these vertices is incident on an added edge to u or w, then no more added edges exist and the vertices u, v, w, s, k' induce an F_6 or an F_1 if the added edge incident on v connects it to u or w, respectively, where $s \in S_H$ is the non-neighbor of k and k' is any vertex in $K_H - \{v, k\}$; otherwise, there exists a common neighbor p not incident on an added edge to u or w and the vertices u, v, w, k, p induce an F_6 or an F_1 if the added edge incident on v connects it to u or w, respectively.
- *v is incident on 2 added edges connecting it to u and w*: Then, the remaining added edges are at most $|K_H| - 2$ in total; hence, there exists a pair of non-adjacent vertices s, k (where $s \in S_H - \{u\}$ and $k \in K_H - \{v\}$) which are not incident on an added edge to w or u. Let $A = (S_H - \{u, s\}) \cup r$; the set A is a set of $|K_H| - 1$ neighbors of k other than u. Then, there exists 1 vertex p_1 in A which is not incident on an added edge to u or w. If there exists a second vertex p_2 in A not incident on an added edge to u or w, then the vertices u, w, k, p_1, p_2 induce an F_5 if p_1, p_2 are connected by an added edge or an F_3 otherwise. If each vertex in $A - \{p_1\}$ is incident on an added edge to u or w, then the added edges incident on these vertices account for the remaining $|K_H| - 2$ added edges. Then, the vertices u, w, s, k_1, k_2 (for any vertices $k_1, k_2 \in K_H - \{v, k\}$) induce an F_6.

Therefore, if we use fewer than the stated number of added edges, in each case, the resulting graph contains an induced forbidden subgraph; a contradiction.

2. Let $v \in S_H$ be the non-neighbor of u in H. Then, we get a P_4-sparse graph by connecting u to v; thus, u becomes universal in $V(H) \cup \{w\}$. This is the minimum number of edges (in addition to the tail uw) that need to be added since for any pair of non-neighbors s, k with $s \in S_H - \{v\}$ and $k \in K_H - \{u\}$, the vertices u, v, w, s, k induce a forbidden subgraph F_3; a contradiction.

3. By connecting u to all vertices in S_H or by connecting w to all vertices in K_H, we get a P_4-sparse graph; in the former case, u becomes universal in $V(H) \cup \{w\}$, in the latter case, we get a thick spider with $S = S_H$, $K = K_H$, and $R = \{u, w\}$, and in either case the number of added edges (in addition to the edge uw) is equal to $|K_H|$. In particular, for $|K_H| = 3$, we can also get a P_4-sparse graph, by connecting a vertex $k \in K_H$ to w and to its non-neighbor s in S_H (thus making k universal in $V(H) \cup \{w\}$) and by connecting s to u; the resulting P_4-sparse graph is a thin spider with $S = (S_H - \{s\}) \cup \{w\}$, $K = (K_H - \{k\}) \cup \{u\}$, and $R = \{s\}$.
To prove the minimality of this number of added edges, suppose, for contradiction, that we can get a P_4-sparse graph after having added at most $|K_H| - 1$ edges (in addition to the tail uw). Then, there exists a pair s_1, k_1 of non-neighbors in H with $s_1 \in S_H$ and $k_1 \in K_H$ none of which is incident on an added edge to u or w. We distinguish the following two cases that cover all possibilities.

- *Each of the vertices in $K_H - \{k_1\}$ is incident on an added edge to w.* These are precisely all the $|K_H| - 1$ added edges; hence none of the vertices in $S_H - \{s_1\}$ is incident on an added edge. Then, the vertices u, w, k_1, s_2, s_3 (for any $s_2, s_3 \in S_H - \{s_1\}$) induce an F_3.

- *There exists at least one vertex in $K_H - \{k_1\}$ that is not incident on an added edge to w.* Let that vertex be k_2. Then, if there exists another vertex $k_3 \in K_H - \{k_1, k_2\}$ that is not incident on an added edge to w as well, the vertices u, w, k_2, k_3, s_1 induce an F_6. On the other hand, if each of the vertices in $K_H - \{k_1, k_2\}$ is incident on an added edge to w (which implies that k_3 is adjacent to w), then these added edges are $|K_H| - 2$ in total, with only 1 remaining. If the non-neighbor s_3 of k_3 in S_H is not incident on an added edge to u or w, then the vertices u, w, k_1, k_2, s_3 induce an F_6 whereas if it is adjacent to w, the vertices u, w, k_2, s_1, s_3 induce an F_4 and if it is adjacent to u, the vertices u, k_1, k_3, s_1, s_3 induce an F_6.

In each case, we get a contradiction. □

If the (thin or thick) spider H belongs to a more general P_4-sparse graph, then Lemmas 2.7 and 2.8 imply the following result.

Corollary 2.2. *If the spider H of Lemmas 2.7 and 2.8 belongs to a more general P_4-sparse graph G, then the minimum number of added edges needed for a minimum P_4-sparse*

completion of the graph $G + uw$ does not exceed the minimum number respectively given by Lemmas 2.7 and 2.8 augmented by $|N_G(u) \cap (V_0 \cup \cdots \cup V_{h-1})|$.

The number suggested in the corollary corresponds to doing the minimum P_4-completion of the graph $H + uw$ and not changing the rest of the P_4-sparse tree of G.

2.4.2 The Algorithm

Recall that $t_0 t_1 \cdots t_h u$ is the path from the root to u in the P_4-sparse tree T_G of G and V_i $(0 \leq i < h)$ is the set of vertices associated with the leaves of the subtrees rooted at the children of t_i except for t_{i+1} and V_h is the set of vertices associated with the leaves of the subtrees rooted at the children of t_h except for u. See Figure 2.9.

In the following lemma, we show how a P_4 with the tail uw as a wing may be formed in a minimum completion of the graph $G + uw$.

Lemma 2.9. *In addition to the cases of Lemmas 2.7 and 2.8, a P_4 with the tail uw as a wing may be formed in a minimum completion of the graph $G + uw$, only if there exists j $(0 \leq j < h)$ such that t_j is a 1-node, t_{j+1} is a 0-node, there exists a vertex $a \in V_i$ that is universal in V_i, and there exists a vertex $b \in V_{i+1}$ that is adjacent to no vertex in V_{i+1}. Then, the vertices u, w, a, b belong to a spider with (S, K, R) where $S = \{w, b\}$, $K = \{u, a\}$ and $R = (V_{i+1} - \{b\}) \cup V_{i+2} \cup \cdots \cup V_h$.*

Proof. For the tail uw to be the wing of a P_4 $wuxy$, it has to be the case that u, x, y form a P_3. If the P_4 $wuxy$ belongs to a spider with clique size equal to 2, then with just the additional added edge uy, we get a P_4-sparse graph as well (then, u is universal in the vertex set of the spider). This implies that a minimum completion involving such a P_4 will be better if u, x, y form a P_3 in G so that no added edge is required for building the P_3. The same argument applies for more than one P_4. For example, consider that the minimum completion of $G + uw$ contains a spider with $|K| = |S| = k = 3$ where $K = \{u, k_1, k_2\}$ and $S = \{w, s_1, s_2\}$, with s_1, s_2 being adjacent to k_1, k_2, respectively. In G, the vertices u, k_1, k_2, s_1, s_2 should induce a connected subgraph for otherwise, we add edges connecting u to its non-neighbors only in the connected component of $G[\{u, k_1, k_2, s_1, s_2\}]$ to which u belongs, thus getting a P_4-sparse graph with fewer added edges. Then, the edges s_1k_1 and s_2k_2 should be edges of G; moreover, the edge k_1k_2 should belong to G for otherwise, to ensure connectivity, u would be adjacent

to both k_1 and k_2, and u, k_1, k_2, s_1, s_2 would induce an $F_2 = P_5$. Since k_1k_2 is an edge of G, then the vertices s_1, k_1, k_2, s_2 induce a P_4 in G and thus the formation of a P_4 with the tail uw as a wing occurs as described in the proofs of Lemmas 2.7 and 2.8. We work similarly, for larger spiders.

Therefore, in the following, we consider the generation of a single P_4 $wuab$ where the graph G contains the P_3 uab. If any edge of the P_3 uab connects a vertex of a set S to a vertex of a set K of a spider, then the 3rd vertex would belong to the spider, and thus Lemmas 2.7 and 2.8 cover this case. Then, the only way, to have a P_3 uab is to have a node t_i which is a 1- or a 2-node such that $a \in V_i$ (in the latter case, a is a vertex of the clique of the spider) and a node t_j with $j > i$ such that t_j is a 0- or a 2-node with $b \in V_j$ (in the latter case, b is a vertex of the independent set of the spider). If we use Formation 2 right after node t_i, then the number of added edges are $|(V_{i+1} \cup \cdots \cup V_h) - N_G(u)| + |(V_0 \cup \cdots \cup V_i) \cap N_G(u)|$; the former term corresponds to edges incident on u, the latter to edges incident on w.

Let us now try forming a P_4 $wuab$, and thus a spider of clique size equal to 2; clearly, this is a thin spider. Then, Property P1 in Lemma 2.2 implies that w and a (as well as u) should be adjacent to all the neighbors of b except for a, that is, w should be adjacent to at least the vertices in $[(V_0 \cup \cdots \cup V_{j-1}) - \{a\}] \cap N_G(b) = [(V_0 \cup \cdots \cup V_{j-1}) - \{a\}] \cap N_G(u)$. Additionally, Property P2 in Lemma 2.2 implies that because a is adjacent to all the vertices in $V_{i+1} \cup \cdots \cup V_h$ and to its neighbors in V_i, then so must be vertex u. Clearly, $(V_{i+1} \cup \cdots \cup V_h) - N_G(u) \subseteq V_{i+1} \cup \cdots \cup V_h$. On the other hand, $|(V_0 \cup \cdots \cup V_i) \cap N_G(u)| \leq |[(V_0 \cup \cdots \cup V_{j-1}) - \{a\}] \cap N_G(u)|$ unless $j = i+1$. Moreover, if a is universal in V_i and b is adjacent to no vertex in $V_j = V_{i+1}$, then by removing a from V_i and b from V_j, and by placing the new spider involving just the P_4 $wuab$ in between the nodes t_i and t_{i+1} requires 1 added edge less than the solution using Formation 2. This advantage is lost if t_i is a 2-node in which case a has a neighbor in V_i that is not a neighbor of u; then Property P2 in Lemma 2.2 implies that an added edge is needed to connect u to it. Now consider that t_i is a 1-node. Then, the advantage is lost again if a has a non-neighbor in V_i; this non-neighbor is a neighbor of u and Property P2 in Lemma 2.2 implies that an added edge is needed to connect a to it. The advantage is also lost if t_j is a 0-node and b has a neighbor in V_j or if t_j is a 2-node, which also implies that b has a neighbor in V_j; in either case, Property P1 in Lemma 2.2 yields that w must be adjacent to it, costing an additional edge. □

Now we are ready to describe our algorithm for counting the minimum number of added edges for a P_4-completion of the graph $G + uw$.

Algorithm P_4**-sparse-Tail-Addition**

Input: A P_4-sparse graph G and a tail uw to be added to G.

Output: The minimum number of added edges (in addition to the tail uw) needed so that the resulting graph is P_4-sparse.

if $|V(G)| = 1$ **then** $\{V(G) = \{u\}\}$

the graph resulting from the addition of the tail uw is P_4-sparse;

no further added edge is needed;

return(0);

Let $t_0 t_1 ... t_h$ ($h \geq 1$) be the path from the root t_0 of the P_4-sparse tree of G to the parent-node t_h of the leaf corresponding to u;

Let V_i ($0 \leq i < h$) be the set of vertices associated with the leaves of the subtrees rooted at the children of t_i except for t_{i+1} and V_h be the set of vertices associated with the leaves of the subtrees rooted at the children of t_h except for u (see Figure 2.9);

$min \leftarrow |N_G(u)|$; $\{$*corresponds to Formation* 1$\}$

$\{$*check for Formation* 2 *(Lemma 2.6(iii))*$\}$

for each t_i ($i = 0, 1, \ldots, h$) that is a 1- or a 2-node **do**

$\ell \leftarrow |N_G(u) \cap (V_0 \cup \cdots \cup V_{i-1})| + |(V_i \cup \cdots \cup V_h) - N_G(u)|$;

if $\ell < min$ **then**

$min \leftarrow \ell$;

$\{$*check for new* P_4 *formation (Lemma 2.9)*$\}$

for each $i = 0, 1, \ldots, h-1$ such that t_i is a 1-node and T_{i+1} is a 0-node **do**

if there exists a vertex $a \in V_i$ such that a is universal in V_i **and**

there exists a vertex $b \in V_{i+1}$ such that b has non neighbors in V_{i+1} **then**

$\ell \leftarrow |N_G(u) \cap (V_0 \cup \cdots \cup V_{i-1})| + |V_i - \{a\}| + |V_{i+1} - \{b\}| +$
$|(V_{i+2} \cup \cdots \cup V_h) - N_G(u)|$;

if $\ell < min$ **then**

$min \leftarrow \ell$;

$\{$*check the cases if* t_h *is a* 2*-node and apply Corollary 2.2*$\}$

if t_h is a 2-node **then**

$\ell \leftarrow$ number of added edges according to cases of Lemmas 2.7 or 2.8;

$\ell \leftarrow \ell + |N_G(u) \cap (V_0 \cup \cdots \cup V_{h-1})|$;

if $\ell < min$ **then**

$min \leftarrow \ell$;

return(min);

Algorithm P_4-sparse-Tail-Addition can be easily augmented to return a minimum cardinality set of added edges.

The correctness of the algorithm follows from Lemmas 2.6, 2.7, 2.8, 2.9, and Corollary 2.2. Let G be the given graph and let n be the number of its vertices. If the P_4-sparse tree T_G of G is given, an $O(n)$-time traversal of the tree enables us to compute the number of neighbors and non-neighbors of u in each of the sets $V_0, \ldots, V_h$; additionally, the height of T_G is $O(n)$ which implies that $h = O(n)$. Since the conditions of Lemmas 2.7 and 2.8 can be checked in $O(1)$-time, the entire algorithm runs in $O(n)$ time.

Theorem 2.6. *Let G be a P_4-sparse graph on n vertices and let uw be tail attached at node u of G. If the P_4-sparse tree of G is given, Algorithm P_4-sparse-Tail-Addition computes the minimum number of edges to be added to $G+uw$ so that the resulting graph is P_4-sparse in $O(n)$ time.*

If the P_4-sparse tree T_G of G is not given, then it can be computed in $O(n+m)$ time where m is the number of edges of G [35], and the entire algorithm takes $O(n+m)$ time.

2.5 Concluding Remarks

In this chapter, we study the minimum $\mathcal{C}$-completion problem. We consider a graph G which belongs to a graph class $\mathcal{C}$ and we are interested in connecting a node $w \notin V(G)$ to G by a single edge uw where $u \in V(G)$; we call such an edge a *tail*. Our aim is the calculation of a minimum $\mathcal{C}$-completion of G'; in other words, the minimum number of non-edges (in addition to the tail uw) that must be added in order to the resulting graph to belong to the class $\mathcal{C}$, because the graph G' resulting from G after the addition of the tail need not necessarily belong to the class $\mathcal{C}$.

In particular, we study this completion problem in some classes of perfect graphs, such as split, threshold, quasi-threshold and P_4-sparse graphs. In every class, we study their properties and graph structure in order to calculate the minimum number of

fill edges that needed after an addition of a tail in the initial graph G. Firstly, we assume a split graph G, and we add a tail uw, where $u \in V(G)$ and $w \notin V(G)$. The minimum number of fill edges depends on the vertex set, independent set S or clique set K, to which node u belongs. Secondly, we consider a quasi-threshold graph G, which is A-free graph or a threshold graph G, which is precisely a A-free graph containing no induced subgraph isomorphic to $2K_2$. In these classes, the minimum quasi-threshold/threshold completion of the graph $G + uw$, where uw is a tail, is defined as the minimum value of some functions, which depends on the position and the level of vertex u in the tree representation $T_c(G)$. Finally, since we study the properties and the structure of P_4-sparse graph and its tree representation, we describe the **Algorithm P_4-sparse-Tail-Addition** in order to count the added edges for a P_4-sparse completion of the graph $G + uw$. The correctness of this algorithm based on a series of lemmas.

Chapter 3

Adding an Edge in a P_4-sparse Graph

3.1 Introduction

Other one instance of the general $(\mathcal{C}, +k)$-MinEdgeAddition problem [36] is the (P_4-sparse,$+1$)-MinEdgeAddition Problem. In this problem, we add 1 given non-edge uv in a P_4-sparse graph and we want to compute a minimum P_4-sparse-completion of the resulting graph $G + uv$.

The above problem are motivated by the dynamic recognition (or on-line maintenance) problem on graphs: a series of requests for the addition or the deletion of an edge or a vertex (potentially incident on a number of edges) are submitted and each is executed only if the resulting graph remains in the same class of graphs. Several authors have studied this problem for different classes of graphs and have given algorithms supporting some or all the above operations; we mention the edges-only fully dynamic algorithm of Ibarra [39] for chordal and split graphs, and the fully dynamic

algorithms of Hell et al. [40] for proper interval graphs, of Shamir and Sharan [41] for cographs, and of Nikolopoulos et al. for P_4-sparse graphs [51].

As referred in [61], the class of integrally completable graphs are those Laplacian integral graphs having the property that one can add in a sequence of edges, presenting Laplacian integrality with each addition, and that such edge additions can continue until a complete graph is obtained. According to [62], the energy of a complete multipartite graph, i.e., the sum of the absolute values of its eigenvalues, increases if a new edge added or an old edge is deleted. Papagelis [63] study the problem of edge modification on social graphs and consider the problem of adding a small set of non existing edges in a social graph with the main objective of minimizing its characteristic path length, i.e., the average distance between pairs of vertices that controls how broadly information can propagate through a network.

More specifically about $\mathcal{C}$-completion problems, Yannakakis [8] showed that the computing the minimum fill-In of chordal graphs is NP-Complete. Nikolopoulos and Palios [59] establish structural properties of cographs and they present an algorithm which, for a cograph G and a non-edge xy (i.e., two non-adjacent vertices x and y) of G, finds the minimum number of edges that need to be added to the edge set of G such that the resulting graph is a cograph and contains the edge xy. Their proposed algorithm could be a suitable addition to the algorithm of Shamir and Sharan [40] for the online maintenance of cographs and it runs in time linear in the size of the input graph and requires linear space.

Our Contribution. A typical graph modification problem aims to modify a graph G, via a small number of operations from a specified set S, into some other graph G' that has a certain desired property, which usually describes a certain graph class $\mathcal{C}$ to which G' must belong. Based on the idea behind the proof of problem of adding a tail to a graph G, which belongs to a class of perfect graphs, we present an efficient algorithm for calculation the minimum fill edges in $G + uv$ such that belongs in the same class with initial graph G. For this purpose, we investigate the class of P_4-sparse graphs, and we study the above problem: given a P_4-sparse graph G and non edge uv, where $u, v \in V(G)$, we compute the minimum fill edges for $G+uv$ to be P_4-sparse graph.

What is important of the proposed method of solution is to distinguish the two cases of addition of a non-edge. More precisely, the first case is the least common

ancestor of the leaves corresponding to u, v in P_4-sparse tree T is a 0-node, i.e., the given P_4-sparse graph G consists of 2 connected components each containing one of the endpoints of the added non-edge uv. Furthermore, there exists the case where the addition of a non-edge is incident on a vertex of the clique or the independent set of a spider graph (either thin or thick spider). Our approach is summarized finding the optimal solution H in each case and what root node (1-node, 2-node thin or 2-node thick) of P_4-sparse tree $T(H)$ is.

Finally, we would like to point out that the primary purpose of our approach is not to fill a gap of the existing C-completion methods by proposing a new algorithm, but to expand the idea used on the previous chapter and show that it can be efficiently applied for other classes of perfect graphs depicting thus the high versatility of the whole concept.

Road Map. The chapter is organized as follows: In Section 3.2 we define the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem, namely we consider the special case in which the given P_4-sparse graph G consists of 2 connected components each containing one of the endpoints of the added non-edge uv. In Section 3.3 we introduce the same problem in the case where the adding a non-edge is incident on a vertex of the clique or the independent set of a spider graph, i.e., the addition of a new edge is done inside in a spider graph. In Section 3.4 we analyze the steps of general algorithm, which calculate the minimum number of fill edges in (P_4-sparse,+1)-MinEdgeAddition Problem. Finally, in Section 3.5 we summarize our work with the main parts of it.

3.2 Connecting two Connected Components

In this section, we will consider the special case in which the given P_4-sparse graph G consists of 2 connected components each containing one of the endpoints of the added non-edge uv; we will cal this problem *(P_4-sparse-2CC,+1)-MinEdgeAddition*. Let C_u (C_v respectively) be the connected component of G containing u (v respectively). Clearly $V(G) = C_u \cup C_v$. It is not difficult to see that:

Observation 3.1. *Let G be a disconnected graph consisting of 2 connected components C_u and C_v such that $u \in C_u$ and $v \in C_v$, and consider the instance of the (P_4-sparse-2CC,+1)-*

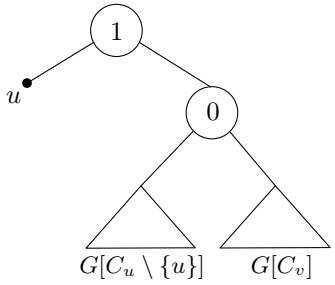

Figure 3.1: The tree representation $T(G)$ with the vertex u as universal in G.

MinEdgeAddition Problem for the graph G and the non-edge uv. Then

(i) Each of the induced subgraphs $G[C_u]$ and $G[C_v]$ is connected.

(ii) In any optimal solution H to the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for the graph G and the added non-edge uv it holds that each of the induced subgraphs $H[C_u]$ and $H[C_v]$ is connected and the entire graph H is connected.

Observation 3.1(ii) implies that the root node of the P_4-sparse tree of any optimal solution H of the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for G and uv is a 1-node or a 2-node (for a thin or a thick spider) and these are the cases that we consider in the following subsections.

Before that, however, we note that we can get a P_4-sparse graph G' where $V(G') = V(G)$ and $E(G) \cup \{uv\} \subseteq E(G')$ by making u universal in G (Figure 3.1) which requires $|V(G)| - 1 - deg_G(u)$ fill edges (including uv). A similar statement holds for v.

Also, it is important to note that for any two positive integers i_1, i_2, it holds that

$$\begin{aligned} & i_1 \cdot i_2 \ \geq \ i_1 + i_2 - 1; \\ & \text{equality holds if } i_1 = 1 \text{ or } i_2 = 1. \end{aligned} \tag{3.1}$$

Note that $i_1 \cdot i_2 = i_1 + i_2 - 1 \iff (i_1 - 1) \cdot (i_2 - 1) = 0$.

Our algorithm for the (P_4-sparse,+1)-MinEdgeAddition Problem relies on the structure of the P_4-sparse tree of the given graph. In particular, for a P_4-sparse graph G and a vertex u of G, we define the subtrees $T_{u,1}(G)$, $T_{u,2}(G)$, ... Let $t_1 t_2 \cdots t_r$ be the path in the P_4-sparse tree T_G of G from the root node t_1 to the leaf t_r corresponding to u. Then,

- $T_{u,1}(G)$ is the subtree of T_G containing t_1 after we have removed the tree edge $t_1 t_2$;

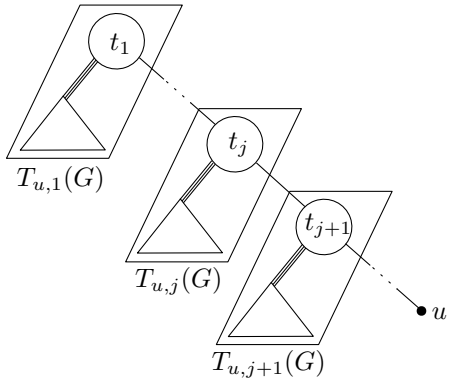

Figure 3.2: The subtrees $T_{u,1}(G)$, $T_{u,2}(G)$, ..., $T_{u,j}(G)$, $T_{u,j+1}(G)$, ... which contain the vertices $t_1, t_2, \ldots t_j, t_{j+1} \ldots$ respectively.

- for $j = 2, 3, \ldots, r-1$, $T_{u,j}(G)$ is the subtree of T_G containing t_j after we have removed the tree edges $t_{j-1}t_j$ and t_jt_{j+1}.

In Figure 3.2, it depicts the path $t_1t_2 \cdots t_jt_{j+1} \cdots u$ and the subtrees $T_{u,1}(G)$, $T_{u,2}(G)$, ..., $T_{u,j}(G)$, $T_{u,j+1}(G)$, ...

3.2.1 Case 1: The root node of the P_4-sparse tree T_H of the solution H is a 1-node

If the treenode corresponding to u (v resp.) in T_H is a child of the root of T_H, then u (v resp.) is universal in H. So, in the following, assume that the treenodes corresponding to u, v are not children of T_H's root. Let T_u, T_v be the subtrees rooted at the children of the root of T_H containing the treenodes corresponding to u and v, respectively. Next, we consider the cases whether $T_u = T_v$ and $T_u \neq T_v$.

Case 1a. The vertices u, v belong to the same subtree T.

We show the following lemma.

Lemma 3.1. *Suppose that the root node of the P_4-sparse tree T_H of an optimal solution H of the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for the graph G and the non-edge uv is a 1-node and that the vertices u, v belong to the same subtree of the root of T_H. Then, there exists an optimal solution H' of the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for the graph G and the non-edge uv (a) which results from making u or v universal in G or (b) in which the subtree $T_{u,1}(H')$ is identical to $T_{u,1}(G[C_u])$ or $T_{v,1}(G[C_v])$ or (c) $G[C_u]$*

($G[C_v]$ respectively) is a thin spider (S', K', R') with $u \in S'$ ($v \in S'$ respectively) in which case H' results from making the unique neighbor of u (v respectively) universal in G.

Proof. We distinguish the following cases depending on the treenode type of the root of the subtree $T_{u,1}(G[C_u])$; since the subgraph $G[C_u]$ of G induced by C_u is connected (Observation 3.1(i)), the root of $T_{u,1}(G[C_u])$ is a 1-node or a 2-node.

A. *The root of the subtree $T_{u,1}(G[C_u])$ is a 1-node.* If $V(T_{u,1}(G[C_u])) \subseteq V(T_{u,1}(H))$, then in H, every vertex in $V(T_{u,1}(G[C_u]))$ is adjacent to all the vertices in $V(G) \setminus V(T_{u,1}(G[C_u]))$; then, an optimal solution of the problem can be constructed from the join $G[V(T_{u,1}(G[C_u]))] + F$ where F is an optimal solution after the addition of the non-edge uv in $G[V(G) \setminus V(T_{u,1}(G[C_u]))$.

Let $Q = V(T_{u,1}(G[C_u])) \setminus V(T_{u,1}(H))$ and consider now the case in which $Q \neq \emptyset$; in particular, assume that H is such that $|Q|$ is minimum. Then, Q is universal in $G[C_u \setminus V(T_{u,1}(G[C_u]))]$, which includes u. Additionally, since u is adjacent to all the vertices in $V(T_{u,1}(G[C_u]))$, the graph $G[C_u \setminus V(T_{u,1}(H))$ is connected and so is $H[V(G) \setminus V(T_{u,1}(H))$; since the root of the P_4-sparse tree of H is a 1-node, then any non-leaf child of the root is a 2-node.

If the least common ancestor t of u, v is not a child of the root of the P_4-sparse tree T_H of H, then the subtree $T_{u,2}(H)$ is well defined and its root is a 2-node; let (S_1, K_1, R_1) be the corresponding spider and $u, v \in R_1$.

- *The spider (S_1, K_1, R_1) is thin.* If $S_1 \cup K_1$ contains vertices from both C_u and C_v, Lemma 3.4 implies that there exists an optimal solution of the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for the subgraph $G[V(G) \setminus V(T_{u,1}(H))]$ and the non-edge uv in which u or v is universal or the induced subgraphs $G[C_u \setminus V(T_{u,1}(H))]$ and $G[C_v]$ are as shown in Figure 3.5. In the latter case, we cannot have that $|R_1 \cap C_v| \leq |R_1 \cap C_u|$ since then there exists an optimal solution H' of the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for G and uv in which $Q \cup \{u'\}$ is universal in $G[C_u \setminus V(T_{u,1}(G[C_u]))]$, in contradiction to the minimality of Q. Now, if $|R_1 \cap C_u| < |R_1 \cap C_v|$ then $Q \cup \{v'\}$ is universal in $G[C_u \setminus V(T_{u,1}(G[C_u]))]$ and Lemma 3.2 implies that there is an optimal solution with u or v is universal or the induced subgraphs $G[C_u \setminus V(T_{u,1}(H))]$. But if u or v is universal in $G[C_u \setminus V(T_{u,1}(H))]$, there exists an optimal solution of the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for G and uv in which u or v is universal in G.

Let us now consider that $S_1 \cup K_1 \subseteq C_u$ or $S_1 \cup K_1 \subseteq C_v$. However, it is not possible that $S_1 \cup K_1 \subseteq C_u$, otherwise $S_1 \cap Q = \emptyset$ (no vertex in S_1 is adjacent to u) and then no vertex is adjacent to all the vertices in $G[S_1]$. Hence $S_1 \cup K_1 \subseteq C_v$ but then exchanging $T_{u,1}(H)$ and $T_{u,2}(H)$, we get an optimal solution with fewer fill edges.

- *The spider (S_1, K_1, R_1) is thick*. Lemma 3.8 implies that either $S_1 \cup K_1 \subseteq C_u$ or $S_1 \cup K_1 \subseteq C_v$; the former case is impossible otherwise $S_1 \cap Q = \emptyset$ (no vertex in S_1 is adjacent to u) and then no vertex is adjacent to all the vertices in $G[S_1]$, whereas in the latter case, by exchanging $T_{u,1}(H)$ and $T_{u,2}(H)$, we get an optimal solution with fewer fill edges.

If the least common ancestor t of u, v is a child of the root of the P_4-sparse tree T_H of H, then t is a 2-node. If $G[C_u \setminus V(T_{u,1}(H))]$ is a P_2 then $|C_u| \geq 3$ and u is universal in $G[C_u]$. Then the number of fill edges in H is at least $(|C_u|-2)\cdot|C_v| \geq |C_v|$ and thus there exists an optimal solution of the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for the graph G and the non-edge uv with u being universal in H''. On the other hand, $G[C_u \setminus V(T_{u,1}(H))]$ is not a spider since no subset of vertices are adjacent to all remaining vertices in a spider. Let $(S_2.K_2.R_2)$ be the spider corresponding to the treenode t.

- *The spider (S_2, K_2, R_2) is thin*. If one of u, v belongs to $S_2 \cup K_2$ and the other belongs to R_2, the subgraph $G[C_u]$ cannot be a P_3 with u as an endpoint and Lemma 3.5 implies that there exists an optimal solution H' with u or v being universal in $H'[V(G) \setminus V(T_{u,1}(H))]$. On the other hand, if u, v in $S_2 \cup K_2$, since $G[C_u \setminus V(T_{u,1}(H))]$ is neither a P_2 nor a thin spider, then Lemma 3.6 implies that there exists an optimal solution H' with u or v being universal in $H'[V(G) \setminus V(T_{u,1}(H))]$.
- *The spider (S_2, K_2, R_2) is thick*. Since at least one of u, v belongs to $S_2 \cup K_2$, Lemma 3.9 implies that there exists an optimal solution H' with u or v being universal in $H'[V(G) \setminus V(T_{u,1}(H))]$.

Therefore, if the least common ancestor t of u, v is a child of the root of T_H there exists an optimal solution H' with u or v being universal in $H'[V(G)\setminus V(T_{u,1}(H))]$ directly implies that there exists an optimal solution H'' (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for the graph G and the non-edge uv with u or v

being universal in H''.

B. *The root of the subtree $T_{u,1}(G[C_u])$ is a 2-node corresponding to a thin or a thick spider (S_G, K_G, R_G).* Let $S_G = \{s_1, \ldots, s_{|K_G|}\}$ and $K_G = \{k_1, \ldots, k_{|K_G|}\}$ where s_i, k_i ($i = 1, \ldots, |K_G|$) are adjacent (non-adjacent resp.) if $G[C_u]$ is a thin (thick, resp.) spider. If $K_G \not\subseteq V(T_{u,1}(H))$ and there exist vertices in $V(T_{u,1}(H)) \setminus (S_G \cup K_G)$, then we exchange vertices in $K_G \setminus V(T_{u,1}(H))$ with vertices in $V(T_{u,1}(H)) \setminus (S_G \cup K_G)$; note that for any vertex $k_i \in K_G$ and any vertex $w \in V(T_{u,1}(H)) \setminus (S_G \cup K_G)$, it holds that $N_G[w] \subseteq N_G[k_i]$. Additionally, for any i ($i = 1, \ldots, |K_G|$) such that $s_i \in V(T_{u,1}(H))$ and $k_i \notin V(T_{u,1}(H))$, we exchange s_i and k_i; note that again $N_G[s_i] \subseteq N_G[k_i]$. After these exchanges, which do not increase the number of fill edges, we have constructed an optimal solution H' of the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for the graph G and the non-edge uv, and in H', there is no vertex s_j in the resulting $V(T_{u,1}(H))$ such that $k_j \notin V(T_{u,1}(H))$.

   Let $K' \subseteq K_G$ be the set of vertices $k_j \in V(T_{u,1}(H))$ such that $s_j \neq u$ and let $S' = \{\, s_j \mid k_j \in K' \,\}$. Then, if $|K'| = 1$ and $K' = \{k_j\}$, let F be the graph resulting from H' after having removed the vertices s_j, k_j, having inserted s_j as an isolated vertex, and after having made k_j as a universal vertex whereas if $|K'| \geq 2$, let F be the spider $(S', K', V(G) \setminus (S' \cup K'))$ where $F[V(G) \setminus (S' \cup K')] = H'[V(G) \setminus (S' \cup K')]$ (the spider is thin or thick if and only if the spider (S_G, K_G, R_G) is thin or thick respectively). In either case, the graph F is P_4-sparse and a completion of G including the non-edge uv and has fewer fill edges than H, a contradiction. The only possibility is that $V(T_{u,1}(H)) = \{k_j\}$ such that $s_j = u$.

□

Case 1b. The vertices u, v belong to subtrees T_u, T_v, respectively, with $T_u \neq T_v$. Then, we show the following lemma.

Lemma 3.2. *Let H be an optimal solution of the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for the graph G and the non-edge uv and suppose that the vertices u, v belong to subtrees T_1, T_2, respectively, of the root of the P_4-sparse tree T_H of H. If $A = V(T_1)$ and $B = V(G) \setminus V(T_1)$, then it is not possible that $A \cap C_v \neq \emptyset$ and $B \cap C_u \neq \emptyset$ and there exists an optimal solution of the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for the graph G and the non-edge uv which results from making u or v universal in G.*

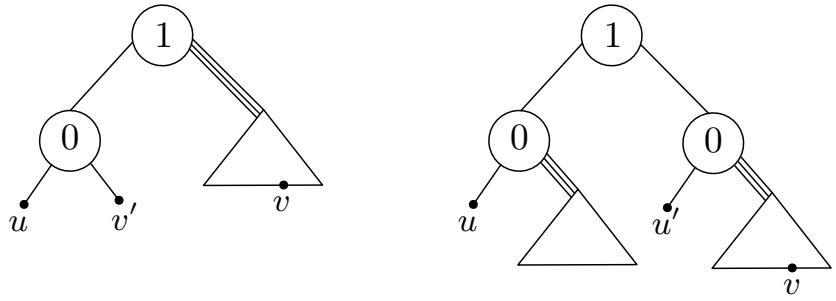

Figure 3.3: The P_4-sparse tree T_H of the optimal solution H in Cases (i) and (ii) of the proof of Lemma 3.4 respectively.

Proof. The definition of A, B implies that $u \in A$ and $v \in B$. First we prove that it is not possible that that $A \cap C_v \neq \emptyset$ and $B \cap C_u \neq \emptyset$. Suppose for contradiction that $A \cap C_v \neq \emptyset$ and $B \cap C_u \neq \emptyset$. By considering only fill edges with one endpoint in C_u and the other in C_v, we have that the number N of fill edges is

$$\begin{aligned} N \geq\ & |A \cap C_u| \cdot |B \cap C_v| + |A \cap C_v| \cdot |B \cap C_u| \\ \geq\ & (|A \cap C_u| + |B \cap C_v| - 1) + (|A \cap C_v| + |B \cap C_u| - 1) \ =\ |V(G)| - 2 \end{aligned}$$

On the other hand, if we make u or v universal, we need $|V(G)| - 1 - deg_G(u)$ and $|V(G)| - 1 - deg_G(v)$ fill edges respectively. Then the optimality of H implies that $deg_G(u) \leq 1$ and $deg_G(v) \leq 1$. Since $A \cap C_v \neq \emptyset$ and $v \in B$, we have that $|C_v| \geq 2$ which implies that $deg_G(v) \geq 1$ because the induced subgraph $G[C_v]$ is connected (Observation 3.1(i)); thus, $deg_G(v) = 1$. In a similar fashion, $deg_G(u) = 1$. Then, the number of fill edges needed to make u or v universal is $|V(G)| - 2$ and the optimality of H along with Equation 3.1 imply that

- at least one of $|A \cap C_u|$, $|B \cap C_v|$ is equal to 1;
- at least one of $|A \cap C_v|$, $|B \cap C_u|$ is equal to 1;
- no more fill edges are used in H which implies that $H[C_u] = G[C_u]$ and $H[C_v] = G[C_v]$.

We consider the following two main cases; the remaining ones are similar.

(i) $|A \cap C_u| = 1$ and $|A \cap C_v| = 1$: Then, $A \cap C_u = \{u\}$ and if $A \cap C_v = \{v'\}$, the P_4-sparse tree of H is as shown in Figure 3.3(left) which implies that u is universal in $H[C_u] = G[C_u]$ and that v' is universal in $H[C_v] = G[C_v]$. The fact that

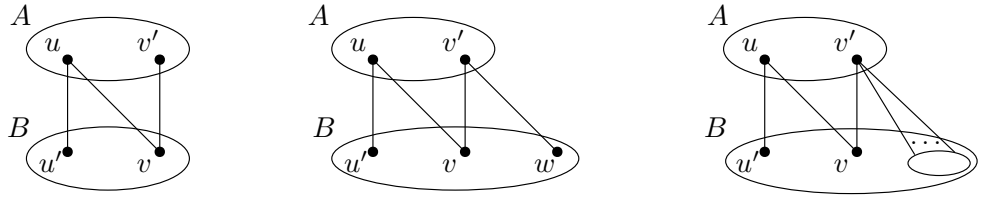

Figure 3.4: (a) $|C_v| = 2$: only the fill edge uv is needed; (b) $|C_v| = 3$: only the fill edges uv and uv' are needed; (c) $|C_v| \geq 4$: only the fill edges uv, uv', and $u'v'$ are needed ($C_u = \{u, u'\}$).

$deg_G(u) = 1$ yields $|C_u| = 2$ and the fact that $deg_G(v) = 1$ yields that v is adjacent only to v' in $G[C_v]$. Figure 3.4 shows solutions to the (P_4-sparse-2CC,+1)-Min-EdgeAddition Problem for the graph G and the non-edge uv contradicting the optimality of H which requires 2, 3, and $|C_v|$ fill edges (including the edge uv) in case (a), (b), and (c) respectively.

(ii) $|A \cap C_u| = 1$ and $|B \cap C_u| = 1$: Then, $A \cap C_u = \{u\}$ and $C_u = \{u, u'\}$ where $B \cap C_u = \{u'\}$. The optimality of H implies that the P_4-sparse tree of H is as shown in Figure 3.3(right). Moreover, since $H[C_v] = G[C_v]$ and $deg_G(v) = 1$, we conclude that $|A \cap C_v| = 1$ which leads to the setting of Case (i).

We reached a contradiction in each case. Then either $A \cap C_v = \emptyset$ or $B \cap C_u = \emptyset$. Suppose without loss of generality that $A \cap C_v = \emptyset$. If $A = \{u\}$ then H' = H and we are done. Suppose next that $|A| \geq 2$. Then the number of fill edges N in H is at least equal to

$$\begin{aligned} N \;&\geq |B \setminus N_G(u)| + |A \setminus \{u\}| \cdot |C_v| \;\geq\; |B \setminus N_G(u)| + |A| - 1 + |C_v| - 1 \\ &\geq |V(G) \setminus N_G(u)| + |C_v| - 1 \;\geq\; |V(G) \setminus N_G(u)| \end{aligned}$$

which implies that there is an optimal solution with u being a universal vertex. □

3.2.2 Case 2: The root node of the P_4-sparse tree of the solution H is a 2-node corresponding to a thin spider (S, K, R)

We first prove some important properties for the optimal solution H in this case.

Observation 3.2. *Suppose that an optimal solution H of the (P_4-sparse-2CC,+1)-Min-EdgeAddition Problem for a P_4-sparse graph G and a non-edge uv is a thin spider (S, K, R). Then:*

(i) For each edge ab in H such that $a \in K$, $b \in S$, and b is not u or v, the vertices a, b are adjacent in G (i.e., ab is not *a fill edge).*

(ii) For each edge ac in H such that $a, c \in K \cap C_u$, the vertices a, c are adjacent in G (i.e., ac is not *a fill edge); a symmetric result holds if $a, c \in K \cap C_v$.*

Proof. (i) Suppose without loss of generality that a, b are not adjacent in G; then, ab is a fill edge in H. Let H' be the graph resulting from H after we have removed the edge ab. The graph H' is P_4-sparse since it is the union of the isolated vertex b with the induced subgraph $H[V(G) \setminus \{b\}]$; in fact, since b is not u or v, it is an optimal solution of the (P_4-sparse,+1)-MinEdgeAddition Problem for the P_4-sparse graph G and the non-edge uv and it has 1 fewer fill edge than H, in contradiction to the optimality of H. Therefore, a, b are adjacent in G.

(ii) We concentrate only in the case in which $a, c \in K \cap C_u$. In H, let a' (c', resp.) be the unique neighbor of a (c, resp.) in S; by statement (i) of this observation, $a', c' \in C_u$, and a, a' and c, c' are adjacent in G. Now, suppose, for contradiction, that a, c are not adjacent in G. Since $a, c \in C_u$ and the induced subgraph $G[C_u]$ is connected (Observation 3.1(i)), there is a path connecting a' to c' in $G[C_u]$, and in fact there is a chordless such path ρ. Clearly, ρ starts with the edge $a'a$, ends at the edge cc' and has length at least 4; thus, G contains an induced chordless path on at least 5 vertices, in contradiction to the fact that G is P_4-sparse. □

Case 2a. The vertices u, v belong to R. Since $u, v \in R$, it is possible that $S \cup K \subset C_u$ or $S \cup K \subset C_v$. For these cases, we show the following lemma.

Lemma 3.3. *Suppose that the optimal solution H of the (P_4-sparse,+1)-MinEdgeAddition Problem for a P_4-sparse graph G and a non-edge uv is a thin spider (S, K, R) with $u, v \in R$. If $S \cup K \subseteq C_u$ then there exists an optimal solution H' of the (P_4-sparse-2CC,+1)-Min-EdgeAddition Problem for the graph G and the non-edge uv (a) which results from making u or v universal in G or (b) in which $T_{u,1}(H) = T_{u,1}(G[C_u])$ or $T_{u,1}(H) = T_{v,1}(G[C_v])$. A symmetric result holds if $S \cup K \subseteq C_v$.*

Proof. We consider the following cases that cover all possibilities:

A. *The root node of the tree $T_{u,1}(G[C_u])$ is a 1-node*. This implies that every vertex in $V(T_{u,1}(G[C_u]))$ is adjacent to all vertices in $C_u \setminus V(T_{u,1}(G[C_u]))$ and in particular to u. On the other hand, the vertices in S are not adjacent to u in H and consequently are not adjacent to u in G; hence, since $S \subset C_u$, it holds that $S \subset C_u \setminus V(T_{u,1}(G[C_u]))$ which in turn implies that in G, all the vertices in $V(T_{u,1}(G[C_u]))$ are adjacent to all the vertices in S and this is also true in H. But this is *impossible* since no vertex in H is adjacent to all vertices in S.

B. *The root node of the tree $T_{u,1}(G[C_u])$ is a 2-node corresponding to a thin spider (S_G, K_G, R_G)*. Since each vertex in $C_u \setminus S_G$ has degree at least 2 in G and thus it has degree at least 2 in H, and each vertex in $S \subset C_u$ has degree 1, we conclude that $S \subseteq S_G$. Then, by Observation 3.2(i), $K = N_G(S)$ and $K \subseteq K_G$. If $K = K_G$ then $S = S_G$ and $T_{u,1}(H) = T_{u,1}(G[C_u])$.

   In the following assume that $K \subset K_G$. Then if $|K_G \setminus K| \geq 2$, the subgraph $G[C_u \setminus (S \cup K)]$ is a thin spider $(S_G \setminus S, K_G \setminus K, R_G)$ whereas if $K_G \setminus K = \{w\}$ then w is universal in $G[C_u \setminus (S \cup K)]$ and the remaining vertices form a disconnected graph with connected components R_G and z where $\{z\} = N_G(w) \cap S_G$. In either case, $|C_u \setminus (S \cup K)| \geq 3$ and $G[C_u \setminus (S \cup K)]$ is connected.

   If the least common ancestor t of u, v is not a child of the root of the P_4-sparse tree T_H of H, then the subtree $T_{u,2}(H)$ is well defined and its root is a 1-node or a 2-node.

   (a) *The root node of $T_{u,2}(H)$ is a 1-node*. Then Lemma 3.1 implies that there is an optimal solution F of the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for the subgraph $G[V(G) \setminus (S \cup K)]$ and the non-edge uv in which either $T_{u,1}(F) = T_{u,1}(G[C_u \setminus (S \cup K)])$ or $T_{u,1}(F) = T_{v,1}(G[C_v])$ or u or v is universal. The former case is impossible since by replacing $H[S \cup K \cup V(T_{u,1}(F))]$ by $G[S \cup K \cup V(T_{u,1}(F))]$, we get an optimal solution with fewer fill edges in contradiction to the optimality of H.

   (b) *The root node of $T_{u,2}(H)$ is a 2-node*. Let (S_1, K_1, R_1) be the corresponding spider and $u, v \in R_1$.

      – *The spider (S_1, K_1, R_1) is thin*. If $S_1 \cup K_1$ contains vertices from both C_u and C_v, Lemma 3.4 implies that there exists an optimal solution H' of the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for the subgraph

$G[V(G) \setminus V(T_{u,1}(H))]$ and the non-edge uv in which either u or v is universal in H' or $T_{u,1}(H')$ is identical to $T_{u,1}(G[V(G) \setminus V(T_{u,1}(H))])$ or $T_{v,1}(G[C_v])$. In the latter case, by exchanging $T_{u,1}(H')$ and $T_{u,2}(H')$ we get an optimal solution of the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for G and uv in which $T_{u,1}(H')$ is identical to $T_{u,1}(G[C_u])$ or $T_{v,1}(G[C_v])$. In turn, if vertex u or v is universal in an optimal solution of the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for the induced subgraph $G[V(G)\setminus(S\cup K)$ and uv, then there exists an optimal solution of the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for G and uv in which u or v is universal; note that solution H contains fill edges connecting the vertices in K to all the vertices in $(S_G \setminus S) \cup C_v$, which, for $|C_v| \geq 2$, are more than the $|K|$ fill edges needed to connect u or v to the vertices in S.

If $S_1 \cup K_1 \subseteq C_u$ then $S_1 \subseteq S_G$ which implies that $K_1 \subseteq K_G$, and if we replace $H[(S \cup S_1) \cup (K \cup K_1)]$ by $G[(S \cup S_1) \cup (K \cup K_1)]$ we get an optimal solution with fewer fill edges than H, a contradiction. Hence $S_1 \cup K_1 \subseteq C_v$. Then, because $|K| \geq 2$, $|K_1| \geq 2$, $|C_u| \geq 5$ and $|C_v| \geq 5$, the number N of fill edges is at least equal to

$$\begin{aligned} N &\geq |K| \cdot |C_v| + |K_1| \cdot |C_u \setminus (S \cup K)| \\ &\geq |C_v| + (|K| - 1) \cdot |C_v| + 2\,|C_u \setminus (S \cup K)| \\ &= |C_v| + (|K| - 1) \cdot (|C_v| - 4) + 4\,(|K| - 1) + 2\,|C_u| - 4\,|K| \\ &\geq 2\,|C_v| - 4 + 2\,|C_u| - 4 \;\geq\; |C_v| + |C_u| + 2 \end{aligned}$$

which is greater than making u or v universal, a contradiction to the optimality of H.

– *The spider* (S_1, K_1, R_1) *is thick*. Lemma 3.8 implies that either $S_1 \cup K_1 \subseteq C_u$ or $S_1 \cup K_1 \subseteq C_v$. If $S_1 \cup K_1 \subseteq C_v$ then by working as in the previous case, we get a contradiction. If $S_1 \cup K_1 \subseteq C_u$ then no matter where the vertices in $K_G \setminus K$ are, there exists a vertex in S_1 that belongs to S_G, which implies that its neighbor in K_G belongs to K_1. Then, by removing these two vertices from the spider (S_1, K_1, R_1) and joining them to the spider (S, K, R) we get an optimal solution that requires fewer fill edges than H, a contradiction.

If the least common ancestor t of u, v is a child of the root of the P_4-sparse tree T_H of H, then t is a 1-node or a 2-node.

(a) *The root node of $T_{u,2}(H)$ is a 1-node*. Then Lemma 3.2 implies that there exists an optimal solution F of the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for the subgraph $G[V(G) \setminus (S \cup K)]$ and the non-edge uv in which u or v is universal.

(b) *The root node of $T_{u,2}(H)$ is a 2-node*. Let (S_2, K_2, R_2) be the spider corresponding to the treenode t.

– *The spider (S_2, K_2, R_2) is thin*. If one of u, v belongs to $S_2 \cup K_2$ and the other belongs to R_2, Lemma 3.5 applies. If Lemma 3.5, case (c) holds, $G[C_v]$ is a P_2 and let the resulting spider be (S', K', R'). Then, we can get an optimal solution of the (P_4-sparse-2CC,+1)-MinEdge-Addition Problem for the graph G and the non-edge uv, which is a spider with stable set $S \cup (S' \cap C_u)$ and clique $K \cup (K' \cap C_u)$, requiring fewer fill edges than H, a contradiction. A similar construction implies that Lemma 3.5, case (b) if $T_{u,2}(H) = T_{u,1}(G[C_u \setminus (S \cup K)])$ as well as Lemma 3.5, case (b), if $T_{u,2}(H) = T_{v,1}(G[C_v])$ and the root node of $T_{v,1}(G[C_v])$ is a 1-node are not possible either. If Lemma 3.5,case (b) holds with $T_{u,2}(H) = T_{v,1}(G[C_v])$ and the root node of $T_{v,1}(G[C_v])$ being a 2-node then by exchanging $T_{u,1}(H)$ and $T_{u,2}(H)$, we get an optimal solution with $T_{u,1}(H) = T_{v,1}(G[C_v])$.

On the other hand, if u, v in $S_2 \cup K_2$, then Lemma 3.6 applies. Since $G[C_u \setminus (S \cup K)$ cannot be a P_2 or a headless thin spider (which includes the P_4), then the only possibility is Lemma 3.6, case (a), i.e, there exits an optimal solution F of the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for the graph $G[V(G) \setminus (S \cup K)$ and the non-edge uv in which u or v is universal.

– *The spider (S_2, K_2, R_2) is thick*. Since at least one of u, v belongs to $S_2 \cup K_2$, Lemma 3.9 implies that there exists an optimal solution H' with u or v being universal in $H'[V(G) \setminus V(T_{u,1}(H))]$.

Therefore, if the least common ancestor t of u, v is a child of the root of T_H and t is a2-node, then there exists an optimal solution H' of the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for the graph G and the

non-edge uv in which $T_{u,1}(H') = T_{v,1}(G[C_v])$ or u or v is universal in the induced subgraph $G[V(G) \setminus (S \cup K)$.

If vertex u or v is universal in an optimal solution of the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for the graph $G[V(G) \setminus (S \cup K)$ and the non-edge uv, then there exists an optimal solution of the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for G and uv in which u or v is universal; note that solution H contains fill edges connecting the vertices in K to all the vertices in $(S_G \setminus S) \cup C_v$.

C. *The root node of the tree $T_{u,1}(G)$ is a 2-node corresponding to a thick spider $Q_G = (S_G, K_G, R_G)$.* Since Q_G is a thick spider and $|S_G| = |K_G| \geq 3$, every vertex $w \in C_u$ is adjacent to at least 2 vertices in C_u. On the other hand, in H, each vertex in $S \subset C_u$ is adjacent to exactly 1 vertex, which belongs to $K \subset C_u$. Therefore, such a case is *impossible*.

□

In addition to the above case, it is possible that $S \cup K$ contains vertices from both C_u and C_v; however, we show that this case cannot yield solutions better than having u or v being universal in H.

Lemma 3.4. *Suppose that the optimal solution H of the (P_4-sparse-2CC,+1)-MinEdge-Addition Problem for a P_4-sparse graph G and a non-edge uv is a thin spider (S, K, R) with $u, v \in R$. Then, if $S \cup K$ contains vertices from both C_u and C_v, there exists an optimal solution H' of the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for the graph G and the non-edge uv (a) which results from making u or v universal in G or (b) in which the subtree $T_{u,1}(H')$ is identical to $T_{u,1}(G[C_u])$ or $T_{v,1}(G[C_v])$ in the case shown in Figure 3.5.*

Proof. Because $H[C_u]$ and $H[C_v]$ are connected (Observation 3.1(i)), there exist vertices $u' \in K \cap C_u$ and $v' \in K \cap C_v$. Let $k_u = |K \cap C_u|$ and $k_v = |K \cap C_v|$; clearly $k_u \geq 1$ and $k_v \geq 1$. By taking into account the fill edges with one endpoint in C_u and the other in C_v, we have that the number N of fill edges is

$$N \;\geq\; k_u \cdot |R \cap C_v| + k_v \cdot |R \cap C_u| + k_u \cdot k_v + 1$$

where the term $+1$ accounts for the added non-edge uv. Then by Equation 3.1 we have

$$N \;\geq\; (k_u + |R \cap C_v| - 1) + (k_v + |R \cap C_u| - 1) + (k_u + k_v - 1) + 1 \;=\; |V(G)| - 2.$$

X

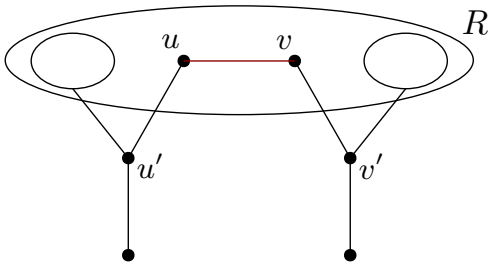

Figure 3.5: The vertex u is adjacent only to u' which is universal in $G[C_u]$, with $u' = K \cap C_u$ and $v' = K \cap C_v$.

If vertex u is universal in G then the number of fill edges is $|V(G)| - 1 - deg_G(u)$ where $deg_G(u) \geq 1$ and similarly for v. Then, the optimality of H implies that in H all of the following hold: $deg_G(u) = deg_G(v) = 1$; $k_u = 1$ or $|R \cap C_v| = 1$; $k_v = 1$ or $|R \cap C_u| = 1$; $k_u = 1$ or $k_v = 1$; no fill edges exist with both endpoints in C_u or C_v, i.e., $H[C_u] = G[C_u]$ and $H[C_v] = G[C_v]$.

Let $u' = K \cap C_u$ and $v' = K \cap C_v$. The facts that $H[C_u] = G[C_u]$, $k_u \geq 1$, and $deg_G(u) = 1$ imply that $k_u = 1$ and that in G, u is adjacent only to u' which is universal in $G[C_u]$ (Figure 3.5). Similarly, $k_v = 1$ and in G, v is adjacent only to v', which is universal in $G[C_v]$. Then, $|K| = 2$ and the number of fill edges (including uv) is $|V(G)| - 2 = |R| + 2$ (where $|R| \geq 2$) matching the number of fill edges if u or v is made universal in G.

Moreover, we can get a P_4-sparse graph by making u' or v' universal. In particular, if $|R \cap C_v| \leq |R \cap C_u|$, we make u' universal and add the fill edge uv, and if $|R \cap C_v| > 1$ we add the fill edge $u'v$ as well; the total number of fill edges is 4 if $|R \cap C_v| = 1$ and $|R \cap C_v| + 4$ if $|R \cap C_v| \geq 2$; a symmetric result holds if $|R \cap C_u| \leq |R \cap C_v|$. In summary, the number of fill edges (including uv) is 4 if $\min\{|R \cap C_u|, |R \cap C_v|\} = 1$ otherwise it is $\min\{|R \cap C_u|, |R \cap C_v|\} + 4$. Since $\min\{|R \cap C_u|, |R \cap C_v|\} \leq |R|/2$, this solution ties the solution with u or v universal if $R = \{u, v\}$ or $|R \cap C_u| = |R \cap C_v| = 2$ and is better in all other cases. The lemma follows from the fact that u' (v' respectively) is universal in $G[C_u]$ ($G[C_v]$ respectively). □

Cases 2b. One of the vertices u, v belongs to R and the other one belongs to $S \cup K$; since u, v are adjacent in the solution H, the latter vertex belongs to K.

Without loss of generality, suppose that $u \in K$ and $v \in R$. Then $k_u = |K \cap C_u| \geq 1$.

Lemma 3.5. *Suppose that an optimal solution H of the (P_4-sparse-2CC,+1)-MinEdge-Addition Problem for a P_4-sparse graph G and a non-edge uv is a thin spider (S, K, R) with one of u, v belongs to R and the other one belongs to $S \cup K$. Then, there exists an optimal solution H' which*

(a) results from making u or v universal in G or

(b) has $T_{u,1}(H') = T_{u,1}(G[C_u])$ or $T_{v,1}(H') = T_{v,1}(G[C_v])$

(c) except if in G one of C_u, C_v induces a P_2 and the other induces a P_3 with u or v being an end vertex or a thin spider (S_1, K_1, R_1) with u or v being an isolated vertex in $G[R_1]$ and $|R_1| \leq |K_1|$ in which case the optimal solution involves joining $G[C_u]$ and $G[C_v]$ into a thin spider.

Proof. We distinguish the following cases:

A. $k_v = |K \cap C_v| = 0$. Then $C_v \subseteq R$. By taking into account the number of fill edges with one endpoint in C_u and the other in C_v, we have that the number N of fill edges in H is

$$N \;\geq\; k_u \cdot |R \cap C_v| \;=\; k_u \cdot |C_v| \;\geq\; k_u + |C_v| - 1.$$

If we make u universal in G, the number of fill edges (including the fill edge uv) is precisely $k_u - 1 + |C_v|$. Then, the optimality of H implies that $N = k_u - 1 + |C_v|$ which requires that $k_u \cdot |C_v| = k_u + |C_v| - 1$ and that no additional fill edges exist; the former implies that $k_u = 1$ or $|C_v| = 1$, the latter that no fill edges exist with both endpoints in C_u or C_v. Thus, since $k_v = 0$, $G[C_u]$ is a thin spider (S_u, K_u, R_u), which implies that $k_u \geq 2$; thus $|C_v| = 1$, i.e., $C_v = \{v\}$. Then, $N = k_u + |C_v| - 1 = k_u$ and this is optimal: if there were an optimal solution H' with at most $k_u - 1$ fill edges (one of which is uv), there would exist an edge ab in $G[C_u]$ where $a \in K \setminus \{u\}$, $b \in S$, and no fill edge in H' is incident on a or b; then, the vertices u, v, a, b, c (where c is the unique neighbor of u in S) induce an F_5 or an F_2 depending on whether v, c are adjacent in H' or not, in contradiction to the fact that H' is P_4-sparse.

B. $k_v \geq 1$ and $R \cap C_u \neq \emptyset$. By taking into account the number of fill edges with one endpoint in C_u and the other in C_v, we have that the number N of fill edges in

H is

$$\begin{aligned} N &\geq k_u \cdot k_v + k_u \cdot |R \cap C_v| + k_v \cdot |R \cap C_u| \\ &\geq (k_u + k_v - 1) + (k_u + |R \cap C_v| - 1) + (k_v + |R \cap C_u| - 1) = |V(G)| - 3. \end{aligned}$$

If we make u universal in G, the number of fill edges (including uv) is precisely $|V(G)| - 1 - deg_G(u)$. By Observation 3.2 and the facts that the induced graph $G[C_u]$ is connected (Observation 3.1(i)) and that $R \cap C_u \neq \emptyset$, we have $deg_G(u) \geq k_u + 1 \geq 2$. Then, the optimality of H implies that $deg_G(u) = 2$ and $N = |V(G)| - 3$ which by Equation 3.1 requires that all of the following hold: $k_u = 1$ or $k_v = 1$; $k_u = 1$ or $|R \cap C_v| = 1$; $k_v = 1$ or $|R \cap C_u| = 1$; no additional fill edges exist, i.e., $G[C_u] = H[C_u]$ and $G[C_v] = H[C_v]$. Since $deg_G(u) = 2$, $k_u \geq 1$, $R \cap C_u \neq \emptyset$, and $G[C_u] = H[C_u]$, Observation 3.2 implies that $k_u = 1$ and $|R \cap C_u| = 1$; thus, $G[C_u]$ is a P_3 and $N = 2\,k_v + |R \cap C_v|$.

Next, if we make v universal in G, the number of fill edges (including uv) is precisely $3 + k_v + |(R \cap C_v) \setminus N_G[v]|$. The optimality of H implies that

$$2\,k_v + |R \cap C_v| \leq 3 + k_v + |(R \cap C_v) \setminus N_G[v]| \iff k_v + |(R \cap C_v) \cap N_G(v)| \leq 2.$$

Then there exist three possibilities:

(i) $k_v = 1$ and $|(R \cap C_v) \cap N_G(v)| = 0$. Let $K \cap C_v = \{a\}$. If $|R \cap C_v| = 1$, then an optimal solution requires 3 fill edges (including uv), a *tie* between the thin spider H (clique $\{u, a\}$) and making u universal; if $|R \cap C_v| = 2$, then an optimal solution requires 4 fill edges (including uv), a three-way *tie* among the thin spider H, making u universal, and making a universal; if $|R \cap C_v| \geq 3$, the optimal solution is obtained by making a universal, which requires 4 fill edges (including uv). Note that vertex a is universal in $G[C_v]$; thus, $T_{v,1}(H') = T_{v,1}(G[C_v])$ if H' is the optimal solution with a universal.

(ii) $k_v = 1$ and $|(R \cap C_v) \cap N_G(v)| = 1$. Then $|R \cap C_v| \geq 2$. Let $K \cap C_v = \{a\}$. If $|R \cap C_v| = 2$, then an optimal solution requires 4 fill edges (including uv), a four-way *tie* among the thin spider H (clique $\{u, a\}$), making u universal, making v universal, and making a universal; if $|R \cap C_v| \geq 3$, then the optimal solution is to make a universal which requires 4 fill edges (including uv). Again, vertex a is universal in $G[C_v]$ and $T_{v,1}(H') = T_{v,1}(G[C_v])$ if H' is the optimal solution with a universal.

(iii) $k_v = 2$ and $|(R \cap C_v) \cap N_G(v)| = 0$. Let $K \cap C_v = \{a, b\}$. If $|R \cap C_v| = 1$, then an optimal solution requires 5 fill edges including uv), a *tie* between the thin spider H (clique $\{u, a, b\}$) and making u universal; if $|R \cap C_v| = 2$, then an optimal solution requires 6 fill edges including uv), a three-way *tie* among the thin spider H, making u universal, and forming a thin spider with clique $\{a, b\}$; if $|R \cap C_v| \geq 3$, then an optimal solution is to form a thin spider with clique $\{a, b\}$ which requires 6 fill edges (including uv). Again, note that $G[C_v]$ is a thin spider with clique $\{a, b\}$.

C. $k_v \geq 1$ and $R \cap C_u = \emptyset$. Then $R \cap C_v = R$. By taking into account the number of fill edges with one endpoint in C_u and the other in C_v, we have that the number N of fill edges in H is

$$\begin{aligned} N &\geq k_u \cdot k_v + k_u \cdot |R \cap C_v| = k_u \cdot k_v + k_u \cdot |R| \\ &\geq (k_u + k_v - 1) + (k_u + |R| - 1) = 2\,k_u + k_v + |R| - 2. \end{aligned}$$

In accordance with Observation 3.2, if we make u universal in G then the number of fill edges is $k_u - 1 + 2\,k_v + |R|$ whereas if we make v universal in G then the number of fill edges is $2\,k_u + k_v + |R \setminus N_G[v]|$. The optimality of H implies that

$$2\,k_u + k_v + |R \setminus N_G[v]| \;\geq\; N \;\geq\; 2\,k_u + k_v + |R| - 2 \iff |R \cap N_G(v)| \leq 1$$

and in accordance with Equation 3.1 for the product $k_u \cdot |R|$, that

$$k_u - 1 + 2\,k_v + |R| \;\geq\; N \;\geq\; k_u \cdot k_v + k_u \cdot |R| \;\geq\; k_u \cdot k_v + k_u + |R| - 1$$

from which we conclude that $k_u \leq 2$. in fact, if $k_u = 2$, then from $k_u - 1 + 2\,k_v + |R| \geq k_u \cdot k_v + k_u \cdot |R|$ we conclude that $2\,k_v + |R| + 1 \geq 2\,k_v + 2\,|R| \iff |R| + 1 \geq 2\,|R| \iff |R| \leq 1 \iff |R| = 1$, i.e., $R = \{v\}$.

We distinguish two cases.

(i) v has no neighbors in R. If $k_u = 1$ then $G[C_u]$ is a P_2. If $k_v = 1$ then if $|R| = 1$ the optimal solution is the thin spider H which requires 2 fill edges (including uv), if $|R| \geq 3$ the optimal solution is to make the single vertex in $K \cap C_v$ universal which requires 3 fill edges (including uv), and there is a *tie* between these two possibilities if $|R| = 2$ (3 fill edges including uv); note that the single vertex in $K \cap C_v$ is universal in $G[C_v]$. Let us now

consider that $k_v \geq 2$. We note that in this case the thin spider H requires fewer fill edges than making v universal which in turn requires fewer fill edges than making u universal. Then, if $|R| \leq k_v$, the optimal solution is the thin spider H which requires $|R| + k_v$ fill edges (including uv), if $|R| \geq k_v + 2$ the optimal solution is the thin spider with clique $K \cap C_v$ (the vertices in C_u are placed in the R-set of the spider) which requires $2\,k_v + 1$ fill edges (including uv), and there is a *tie* between these two possibilities if $|R| = k_v + 1$ in which case $|R| + k_v = 2\,k_v + 1$ fill edges (including uv) are required.

If $k_u = 2$ then $G[C_u]$ is a P_4 and $G[C_v]$ is a P_3 if $k_v = 1$ or else a thin spider (S_v, K_v, R_v) where $|S_v| = |K_v| = k_v \geq 2$ and $R_v = \{v\}$. If $G[C_v]$ is a P_3 then an optimal solution requires 4 fill edges (including uv), a *tie* between the thin spider H and making u universal; if $G[C_v]$ is a thin spider $(S_v, K_v, \{v\})$, then if $k_v = 2$ an optimal solution requires 6 fill edges (including uv), a *tie* between the thin spider H and making u or v universal whereas if $k_v \geq 3$, the optimal solution is to make v universal which requires $k_v + 4$ fill edges (including uv).

(ii) v has 1 neighbor in R. Let z be the neighbor of v in R and let S_z be the connected component in $H[R]$ to which v, z belong. The fact that v has 1 neighbor in R implies that $|R| = |R \cap C_v| \geq 2$ and hence $k_u = 1$; then, due to Observation 3.2, the induced subgraph $G[C_u]$ is a P_2. If $k_v = 1$, G Moreover, $|R \setminus N_G[v]| = |R| - 2$ and the optimality of H implies that $N = 2\,k_u + k_v + |R| - 2 = k_v + |R|$ which by Equation 3.1 requires that no additional fill edges exist, i.e., $H[C_u] = G[C_u]$ and $H[C_v] = G[C_v]$. Then, the thin spider H and making v universal tie in the number of fill edges required. If $k_v = 1$, then the optimal solution is making the single vertex in $K \cap C_v$ universal which requires 3 fill edges (including uv); we note that there is a tie with making v universal if $|R| = 2$ and that the single vertex in $K \cap C_v$ is universal in $G[C_v]$. If $k_v \geq 2$ then the induced subgraph $G[C_v]$ is a thin spider (S_v, K_v, R) with $K_v = K \cap C_v$. Then, a thin spider $(S_v, k_v, R_v \cup C_u)$ can be built which requires

$2\,k_v + 1$ fill edges if $S_z = \{v, z\}$,

$2\,k_v + 2$ fill edges if $G[S_z]$ is a P_3,

$2\,k_v + 3$ fill edges if z is universal in S_z but S_z is not a P_2 or a P_3,

$2\,k_v + \kappa + 2$ fill edges if zv is a "leg" of a thin spider with clique size equal to κ

where the above number of fill edges includes uv. The optimal solution is one of the above possibilities and depends on the difference of $|R| - kv$.

□

Case 2c. The vertices u, v belong to $S \cup K$. Since u, v are adjacent in H and because of Observation 3.2(i), then $u, v \in K$ and thus $k_u = |K \cap C_u| \geq 1$ and $k_v = |K \cap C_v| \geq 1$. We show the following lemma.

Lemma 3.6. *Suppose that an optimal solution H of the (P_4-sparse-2CC,+1)-MinEdge-Addition Problem for a P_4-sparse graph G and a non-edge uv is a thin spider (S, K, R) with $u, v \in S \cup K$. Then, there exists an optimal solution which*

(a) results from making either u or v universal in G

(b) except if in G

(i) one of C_u, C_v induces a P_2 and the other induces a P_2 or a headless thin spider $(S_1, K_1, \emptyset)$ with u or v in $G[K_1]$ or

(ii) both C_u and C_v induce a P_4 with u, v being middle vertices,

in which cases the optimal solution involves joining $G[C_u]$ and $G[C_v]$ into a thin spider.

Proof. Due to the symmetry of u, v, it suffices to consider the following cases.

A. $R \cap C_u \neq \emptyset$ and $R \cap C_v \neq \emptyset$: By counting the fill edges with one endpoint in C_u and the other in C_v, we have that the total number N of fill edges in H is

$$N \;\geq\; k_u \cdot k_v + k_u \cdot |R \cap C_v| + k_v \cdot |R \cap C_u|$$

which by Equation 3.1 gives

$$N \;\geq\; (k_u + k_v - 1) + (k_u + |R \cap C_v| - 1) + (k_v + |R \cap C_u| - 1) \;=\; |V(G)| - 3.$$

If we make u universal in G, the number of fill edges needed (including uv) is $|V(G)|-1-deg_G(u)$; then, the optimality of H implies that $deg_G(u) \geq 2$. Moreover, since the induced graph $G[C_u]$ is connected (Observation 3.1(i)), $deg_G(u) \geq 2$ and thus $deg_G(u) = 2$. Similarly, we get that $deg_G(v) = 2$. The optimality of H

implies that $N = |V(G)| - 3$ and Equation 3.1 requires that all of the following hold: $k_u = 1$ or $k_v = 1$; $k_u = 1$ or $|R \cap C_v| = 1$; $k_v = 1$ or $|R \cap C_u| = 1$; no additional fill edges exist, i.e., $G[C_u] = H[C_u]$ and $G[C_v] = H[C_v]$. Note that if $k_u > 1$, then because $|R \cap C_u| \geq 1$ we would have $deg_G(u) \geq 3$, in contradiction to $deg_G(u) \leq 2$; thus, $k_u = 1$ and similarly $k_v = 1$, which implies that each of $G[C_u], G[C_v]$ is a P_3. Then the optimal solution requires 3 fill edges (including uv) and there is a *tie* between the thin spider H and making u or v universal.

B. $R \cap C_u \neq \emptyset$ but $R \cap C_v = \emptyset$: Then $R \cap C_u = R$. By counting the fill edges with one endpoint in C_u and the other in C_v, we have that the total number N of fill edges in H is

$$N \;\geq\; k_u \cdot k_v + k_v \cdot |R \cap C_u| \;=\; k_u \cdot k_v + k_v \cdot |R|$$

which by Equation 3.1 gives

$$N \;\geq\; (k_u + k_v - 1) + (k_v + |R| - 1) \;=\; |V(G)| - k_u - 2.$$

If we make u universal in G, the number of fill edges needed (including uv) is $|V(G)| - 1 - deg_G(u)$. By Observation 3.2 and the facts that the induced graph $G[C_u]$ is connected (Observation 3.1(i)) and that $R \cap C_u \neq \emptyset$, we have $deg_G(u) \geq k_u + 1$ and then, the optimality of H implies that $deg_G(u) = k_u + 1$; similarly, we get that $k_v \leq deg_G(v) \leq k_u + 1$. The optimality of H implies that $N = |V(G)| - k_u - 2$ and Equation 3.1 requires that all of the following hold: $k_u = 1$ or $k_v = 1$; $k_v = 1$ or $|R| = 1$; no additional fill edges exist, i.e., $G[C_u] = H[C_u]$ and $G[C_v] = H[C_v]$. The facts $deg_G(u) = k_u + 1$ and $H[C_u] = G[C_u]$ imply that $|R| = |R \cap C_u| = 1$, whereas the fact $H[C_v] = G[C_v]$ implies that $deg_G(v) = k_v$ from which we get that $k_v \leq k_u + 1$. We distinguish the following cases.

- $k_u = k_v = 1$: Then $G[C_u]$ is a P_3 and $G[C_v]$ is a P_2; an optimal solution requires 2 fill edges (including uv), a *tie* between the thin spider H and making u universal.
- $k_u = 1$ and $k_v > 1$: Since $k_v \leq k_u + 1 = 2$, $k_v = 2$. Then $G[C_u]$ is a P_3 and $G[C_v]$ is a P_4; an optimal solution requires 4 fill edges (including uv), a *tie* between the thin spider H and making u or v universal.
- $k_v = 1$ and $k_u > 1$: Then $G[C_v]$ is a P_2 whereas $G[C_u]$ is a thin spider with clique size equal to k_u and only 1 vertex in its R-set. An optimal solution

requires k_u+1 fill edges (including uv), a *tie* between the thin spider H and making u or v universal. (The optimality can be shown by contradiction. Let $G[C_u]$ be the thin spider $(\{s_1, s_2, \ldots, s_{k_u}\}, \{u, t_2, \ldots, t_{k_u}\}, \{b\})$ and let $G[C_v]$ be the P_2 av. If there were an optimal solution with at most k_u fill edges, then these would include the fill edge uv and at most $k_u - 1$ more fill edges; the latter $k_u - 1$ fill edges would be incident to the vertices $s_2, \ldots, s_{k_u}, t_2, \ldots, t_{k_u}$ for if there were a pair s_i, t_i $(2 \leq i \leq k_u)$ not incident to any fill edges then the vertices a, v, u, t_i, s_i would induce an F_5 or an F_2 depending on whether u, a are adjacent on not. Then, the vertices a, v, u, s_1, b would induce an F_3, a contradiction.)

C. $R = \emptyset$: Then, by Observation 3.2(i) and (ii), $G[C_u] = H[C_u]$ and $G[C_v] = H[C_v]$ and thus $deg_G(u) = k_u$ and $deg_G(v) = k_v$. The fill edges in H are precisely the fill edges with one endpoint in C_u and the other in C_v which are $k_u \cdot k_v$ in total.

Suppose without loss of generality that $k_u \geq k_v$. If we make u universal in G, the number of fill edges needed (including uv) is $k_u + 2\,k_v - 1$. The optimality of H implies that $k_u \cdot k_v \leq k_u + 2\,k_v - 1 \leq 3\,k_u - 1 < 3\,k_u$ and thus $k_v < 3$. We distinguish the following cases.

- $k_v = 1$: Then $G[C_v]$ is a P_2 and $G[C_u]$ is a thin spider $(\{s_1, s_2, \ldots, s_{k_u}\}$, $\{u, t_2, \ldots, t_{k_u}\}$, $\emptyset)$; an optimal solution requires k_u fill edges (including uv), which form the thin spider H (the solution H requires fewer fill edges than making u universal in G). (The optimality can be shown by contradiction. Let $G[C_v]$ be the P_2 av. If there were an optimal solution with at most $k_u - 1$ fill edges, then these would include the fill edge uv and at most $k_u - 2$ more fill edges; then, there would exist a pair s_i, t_i $(2 \leq i \leq k_u)$ not incident to any fill edges and the vertices a, v, u, t_i, s_i would induce an F_5 or an F_2 depending on whether u, a are adjacent on not, a contradiction.)
- $k_v = 2$: Then $G[C_v]$ is a P_4. In this case, the solution H requires $2\,k_u$ fill edges (including uv) whereas making u universal requires $k_u + 3$. The optimality of H implies that $k_u \cdot k_v = 2\,k_u \leq k_u + 3 \implies k_u \leq 3$. Since $k_u \geq k_v$, we have $2 \leq k_u \leq 3$.

  If $k_u = 2$ then $G[C_u]$ is also a P_4 and an optimal solution requires 4 fill edges (including uv), which form the thin spider H (the solution H requires fewer fill edges than making u or v universal which requires 5 fill edges).

If $k_u = 3$ then $G[C_u]$ is a headless thin spider with clique size equal to 3; an optimal solution requires 6 fill edges (including uv), a *tie* between the thin spider H and making u universal.

□

3.2.3 Case 3: The root node of the P_4-sparse tree of the solution H is a 2-node corresponding to a thick spider (S, K, R)

According to our convention, $|S| = |K| \geq 3$.

Case 3a. The vertices u, v belong to R. In this case, it is possible that $S \cup K \subset C_u$ or $S \cup K \subset C_v$ and in a fashion similar to the proof of Lemma 3.3, we can prove:

Lemma 3.7. *Suppose that an optimal solution H of the (P_4-sparse-2CC,+1)-MinEdge-Addition Problem for a P_4-sparse graph G and an added non-edge uv is a thick spider (S, K, R) with $u, v \in R$. If $S \cup K \subseteq C_u$ then $G[C_u]$ is a thick spider (S_G, K_G, R_G) and $K = K_G$ and $S = S_G$, i.e., $T_{u,1}(H) = T_{u,1}(G[C_u])$.*
A symmetric result holds if $S \cup K \subseteq C_v$.

Proof. We consider the following cases that cover all possibilities:

A. *The root node of the tree $T_{u,1}(G)$ is a 1-node.* We can prove that this case is *not possible*; the proof is identical to Case A in the proof of Lemma 3.3.

B. *The root node of the tree $T_{u,1}(G)$ is a 2-node corresponding to a thin spider (S_G, K_G, R_G).* We show that $K_G \subseteq K$. Suppose for contradiction that there existed a vertex $w \in K_G$, such that $w \notin K$. Moreover, since w is adjacent in G to u and so is in H, $w \notin S$. Then, w is not adjacent in H to the vertices in S, which implies that neither is in G and since $w \in K_G$, it implies that $S \subseteq S_G$ (note that $K_G \cup R_G \subset N_G[w]$). Moreover since $N_H(S) \subseteq K$, we have that $N_G(S) \subseteq K$, and since $|N_G(S)| = |S| = |K|$, it holds that $N_G(S) = K$. But then, if we replace in H the induced subgraph $H[S \cup K]$ by the induced subgraph $G[S \cup K]$, we get a solution for the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for G and the non-edge uv which requires fewer fill edges than H, in contradiction to the optimality of H. Therefore, $K_G \subseteq K$ which implies that $S_G \subseteq S$. But

again, if we replace in H the induced subgraph $H[S_G \cup K_G]$ by the induced subgraph $G[S_G \cup K_G]$ (note that each vertex in $(S \setminus S_G) \cup (K \setminus K_G)$ is adjacent to each vertex in K_G), we get a solution for the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for G and the non-edge uv which requires fewer fill edges than H, a contradiction. Therefore, such a case is *impossible*.

C. *The root node of the tree $T_{u,1}(G)$ is a 2-node corresponding to a thick spider $Q_G = (S_G, K_G, R_G)$.* Since Q_G is a thick spider, then for every vertex $w \in K_G$, it holds that $|N_G(w)| = |C_u| - 2$ which yields that $|N_H(w) \cap C_u| \geq |N_G(w) \cap C_u| = |C_u| - 2$. On the other hand, in H, for each vertex z in $V(H) \setminus K = V(G) \setminus K$, it holds that $N_H(z) \cap S = \emptyset$ and since $S \subset C_u$ and $|S| = |K| \geq 3$, $|N_H(z) \cap C_u| \leq |C_u - S| \leq |C_u| - 3$. Therefore, $K_G \subseteq K$. Since $S \cup K \subseteq C_u$ and since for each $p \in K_G$, p's only non-neighbor in $G[C_u]$ belongs to S_G, then p's non-neighbor in S is precisely p's non-neighbor in S_G; thus, $S_G \subseteq S$.

   Additionally, we show that $K = K_G$. Let $K_2 = K \setminus K_G = \emptyset$ and let S_2 be the set of non-neighbors in H of the vertices in K_2: $S_2 = \{\, w \mid \exists\, s \in K_2 :\ w \notin N_H(s) \,\}$. Let us consider the P_4-sparse graph H' consisting of the thick spider (S_G, K_G, R_G) where the induced subgraph $H'[R_G]$ coincides with the induced subgraph $H[V(G) \setminus (S_G \cup K_G)]$; note that each vertex in $S_2 \cup K_2$ is adjacent to each vertex in K_G. Clearly the graphs H and H' have the same fill edges with both endpoints in $V(G) \setminus (S_G \cup K_G)$. The number of fill edges in H with an endpoint in $S_G \cup K_G$ is $|K_G|\,|C_v| + |K_2|\,|S_G|$ whereas the number of fill edges in H' with an endpoint in $S_G \cup K_G$ is $|K_G|\,|C_v|$; the optimality of H immediately implies that $K_2 = \emptyset$.

□

However, unlike Case 2a, it turns out that this is the only possibility in this case.

Lemma 3.8. *Suppose that an optimal solution H of the (P_4-sparse-2CC,+1)-MinEdge-Addition Problem for a P_4-sparse graph G and a non-edge uv is a thick spider (S, K, R) with $u, v \in R$. Then, it is not possible that $S \cup K$ contains vertices from both C_u and C_v.*

Proof. Suppose for contradiction that $S \cup K$ contains a vertex in C_u and a vertex in C_v. Because $H[C_u]$ and $H[C_v]$ are connected (Observation 3.1(ii)), there exist vertices $u' \in K \cap C_u$ and $v' \in K \cap C_v$ and let u'', v'' be the non-neighbors in S of u', v' respectively. Then, if $u'' \in C_u$, u' is incident on $|(S \cup K \cup R) \cap C_v| = |C_v|$ fill edges in

H whereas if $u'' \in C_v$, u' is incident on $|(S \cup K \cup R) \cap C_v| - 1 = |C_v| - 1$ fill edges; a symmetric result holds for v' and v''. Before proceeding, we note that by making u universal in G, we would need at most $|V(G)| - 2$ fill edges since $deg_G(u) \geq 1$ because $|C_u| \geq 2$ and $G[C_u]$ is connected (Observation 3.1(i)). Next, we distinguish the following cases:

- $u'' \in C_u$ and $v'' \in C_v$: Then, in H, the number N of fill edges is

  $$N \geq |C_v| + |C_u| - 1 + 1$$

  where we subtract 1 for the double counted fill edge $u'v'$ and we add 1 for the fill edge uv, which implies that $N \geq |V(G)|$ in contradiction to the optimality of the solution H.

- $u'', v'' \in C_u$ or $u'', v'' \in C_v$: In either case, as in the previous item, in H, the number N of fill edges is

  $$N \geq (|C_u| + |C_v| - 1) - 1 + 1,$$

  which implies that $N \geq |V(G)| - 1$, again a contradiction to the optimality of H.

- $u'' \in C_v$ and $v'' \in C_u$: Then, in H, in addition to the fill edge uv and the $(|C_u|-1)+(|C_v|-1)-1 = |V(G)|-3$ fill edges incident on u', v', we note that any vertex in $K \setminus \{u', v'\}$ is adjacent to both u'', v'', thus being incident to at least 1 fill edge, for a total of at least $1+(|V(G)|-3)+(|K|-2) = |V(G)|+|K|-4 \geq |V(G)|-1$ fill edges, again a contradiction to the optimality of H.

□

Cases 3b and 3c. At least one of the vertices u, v belongs to $S \cup K$.

Lemma 3.9. *If there exists an optimal solution H of the (P_4-sparse-2CC,+1)-MinEdge-Addition Problem for the union of $G[C_u]$ and $G[C_v]$ and the non-edge uv such that the root node of the P_4-sparse tree corresponding to H is a 2-node corresponding to a thick spider (S, K, R) with at least one of u, v in $S \cup K$, then there exists an optimal solution of the same problem which results from making u or v universal in G.*

Proof. First, note that a vertex in the set K needs exactly 1 additional fill edge to become universal in H. The idea of the proof is to show that in each case at least one

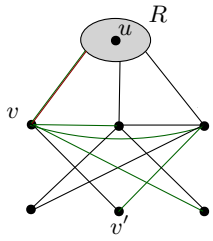

Figure 3.6: The fill edges (green edges) are $|R|+4$ including uv (red edge).

of u, v belongs to K and that by making it universal in H, we get an optimal solution that is no worse than H. Furthermore, recall that we consider that in a thick spider $|K| = |S| \geq 3$.

Case 3b: one of the vertices u, v belongs to R and the other belongs to $S \cup K$, which implies that in fact it belongs to K. Without loss of generality, we assume that $u \in R$ and $v \in K$. We show that $|C_v| \geq 2$. Otherwise, $C_v = \{v\}$, and we could get a solution H' with fewer fill edges than H by removing v and all incident edges from H (the resulting graph is still P_4-sparse) and by adding fill edges incident on u to all its non-neighbors including v, a contradiction; note that in H, v is incident on $|V(G)|-2$ fill edges (including uv) whereas u is adjacent to at least $|K|-1 \geq 2$ vertices other than v which implies that it has at most $|V(G)|-3$ non-neighbors including v. Thus, $|C_v| \geq 2$. Additionally, it holds that $K \cap C_v = \{v\}$ since otherwise, in addition to the fill edges incident on v in H, we would have at least 2 more fill edges whereas by making v universal in H, we get a solution that requires fewer fill edges than H, a contradiction; to see this, note that if $|K \cap C_v| \geq 3$ there exist at least 2 more fill edges connecting u to each of the vertices in $(K \cap C_v) \setminus \{v\}$, whereas if $|K \cap C_v| = 2$ there exist at least 2 more fill edges connecting the vertex in $(K \cap C_v) \setminus \{v\}$ to u and to the vertices in $K \cap C_u$ where $|K \cap C_u| = |K \setminus C_v| \geq 1$.

Therefore, $|C_v| \geq 2$ and $K \cap C_v = \{v\}$. In fact, $(C_v \setminus \{v\}) \subseteq S$; if there existed a vertex in $C_v \cap R$ then, in addition to the fill edges incident on v in H, we would have at least 2 more fill edges connecting that vertex to the vertices in $K \cap C_u$, again implying that making v universal in H would lead to a solution with fewer fill edges than H, a contradiction. Since $K \cap C_v = \{v\}$, each vertex in S is adjacent to at least 1 vertex in $K \cap C_u$ and thus the optimality of the solution H (versus the solution with v being universal in H) implies that $|C_v| = 2$, $|K| = 3$, and the only fill edges are those connecting the vertices in C_u to the vertices in C_v (Figure 3.6) for a total of $|R|+4$

Figure 3.7: The graph G with fill edges (green edges) including uv edge (red edge) where $C_u = \{u\}$ and $|K| \geq 4$, and its representation tree after addition of fill edges.

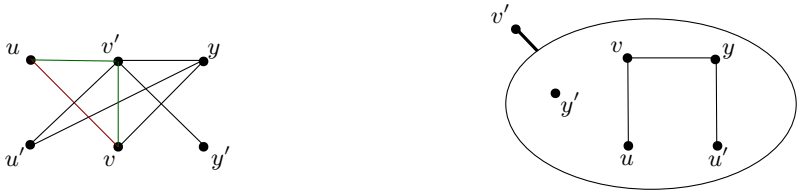

Figure 3.8: The graph G with fill edges (green edges) including uv edge (red edge) where $C_u = \{u\}$, $|K| = 4$, and $R = \emptyset$ and its representation tree after addition of fill edges.

fill edges (including uv) as in the case when v is universal in G.

Case 3c: the vertices u, v belong to $S \cup K$. Then, because the vertices in S form an independent set in H, at least one of u, v belongs to K; without loss of generality, let us assume that $u \in K$. We consider the following cases:

(i) $K \subseteq C_u$: Because $H[C_v]$ is connected 3.1, then $C_v = \{v\}$ which implies that v is incident on fill edges to $u \in K$ and to $|K| - 2 \geq 1$ more vertices in C_u; then, the optimality of the solution H (versus the solution with u being universal in G) implies that $|K| = 3$, and the fill edges are those connecting v to the vertices in $K \subseteq C_u$ (a total of 2 fill edges) matching the number of fill edges if u is universal in G.

(ii) $K \cap C_v \neq \emptyset$: We show that $v \notin K$. Otherwise, let $w \in K - \{u, v\}$ and $w' \in S$ be the non-neighbor of w. If $w, w' \in C_u$, then, in addition to the fill edges incident on u in H, H contains the 2 fill edges vw and vw', a contradiction to the optimality of H compared to the solution with u being universal in G; if $w, w' \in C_v$, the

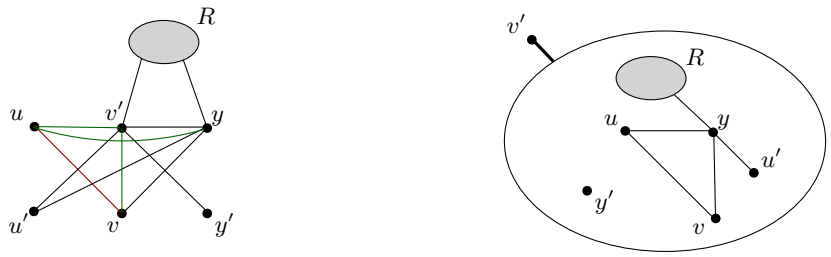

Figure 3.9: The graph G with fill edges (green edges) including uv edge (red edge) where $C_u = \{u\}$, $|K| = 3$, and $R \neq \emptyset$ and its representation tree after addition of fill edges.

case is symmetric considering v being universal in G. So consider that one of w, w' belongs to C_u and the other in C_v; due to symmetry, we can assume that $w \in C_u$ and $w' \in C_v$. Then H contains the fill edges vw and uw'. Now consider the non-neighbor x of u in S, which is adjacent to both v and w; if $x \in C_u$, then H also contains the fill edge vx and thus is not optimal compared to the solution with u being universal in G whereas if $x \in C_v$, H contains the fill edge wx and again H is not optimal compared to the solution with u being universal in G.

Thus $v \notin K$; since $u, v \in S \cup K$, then $v \in S$. Since $K \cap C_v \neq \emptyset$ and $H[C_v]$ is connected (Observation 3.1(i)), there exists $w \in K \cap C_v$ with w being adjacent to v. Then we can show that $K \setminus \{u\} \subseteq C_v$; otherwise, there would exist a vertex $x \in (K \setminus \{u\}) \cap C_u$ and if $x' \in S$ is a common neighbor of w, x, the graph H would include the fill edges wx and one of wx' or xx' (depending on whether x' belongs to C_u or to C_v, respectively) and thus H is not optimal compared to the solution with u being universal in G, a contradiction. In a similar fashion, $R \subseteq C_v$ for otherwise H would contain the at least 2 fill edges from any vertex in $R \cap C_u$ to all the vertices in $K \setminus \{u\}$ and would not be optimal compared to the solution with u being universal in G. A similar argument proves that there is at most 1 vertex in $S \cap C_u$ and that all the vertices in S that are adjacent to at least 2 vertices in $K \setminus \{u\}$ need also belong to C_v.

Then, either (i) $C_u = \{u\}$ or (ii) $C_u = \{u, z\}$ (where $z \in S$ is a neighbor of u) and $|K| = 3$ (otherwise z would be adjacent to at least 2 vertices in $K \setminus \{u\}$ and thus would need to belong to C_v, a contradiction). In the former case, H contains $|R| + 2|K| - 2$ fill edges incident on u, whereas in the latter, $|R| + 3|K| - 5$ fill

edges incident on u and z. However, we can show that in either case, we get a P_4-sparse graph by replacing these fill edges with fewer ones. In the following, let $u' \in S$ be the non-neighbor of u and $v', z' \in K$ be the non-neighbors of v, z respectively.

(i) $C_u = \{u\}$. If $|K| \geq 4$, we use $|K| + 1$ fill edges ($|K|$ fill edges connecting u to v and to the vertices in $K \setminus \{u\}$ and 1 more fill edge connecting v to v'); v' becomes universal in the resulting graph while the remaining vertices induce a thick spider with $S' = S \setminus \{u', v\}$, $K' = K \setminus \{u, v'\}$, and $R' = R \cup \{u, v, u'\}$ (Figure 3.7). If $|K| = 3$ and $R = \emptyset$, we use 3 fill edges to connect u to v and to v', and to connect v to v'; in the resulting graph (Figure 3.8), v' is universal and in the subgraph induced by the remaining vertices, the vertex in $S \setminus \{v, u'\}$ becomes isolated and the other vertices induce a P_4. If $|K| = 3$ and $R \neq \emptyset$, we use 4 fill edges by additionally using the fill edge uy where y is the vertex in $K \setminus \{u, v'\}$; in the resulting graph (Figure 3.9), the vertex v' and the vertex in $S \setminus \{v, u'\}$ are as in the case for $|K| = 3$ and $R = \emptyset$, vertex y is universal in the subgraph induced by the remaining vertices which in turn induce a disconnected graph with connected components R, $\{u'\}$, and $\{u, v\}$.

(ii) $C_u = \{u, z\}$ and $|K| = 3$. In this case, we use $3 = |K|$ fill edges to connect v to u and to connect u and z to z', and then z' becomes universal in the resulting graph (z' is universal in $H[C_v]$), in which the remaining vertices induce a disconnected subgraph with connected components $R \cup \{u', v'\}$ and $\{v, u, z\}$ (Figure 3.10).

In either case, we get a contradiction to the optimality of the solution H (note that for any $|K| \geq 4$ it holds that $|K| + 1 < 2|K| - 2 \leq |R| + 2|K| - 2$ whereas for $|K| = 3$ we have: $3 < 4 = 2|K| - 2 \leq |R| + 2|K| - 2$; for $R \neq \emptyset$, $4 < |R| + 4 = |R| + 2|K| - 2$; lastly, $3 < 4 = 3|K| - 5 \leq |R| + 3|K| - 5$).

□

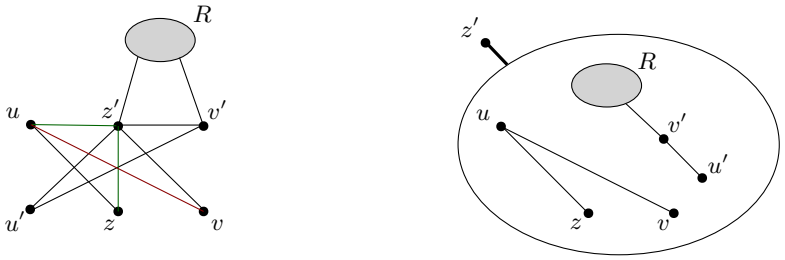

Figure 3.10: The graph G with fill edges (green edges) including uv edge (red edge) where $C_u = \{u, z\}$ and $|K| = 3$, and its representation tree after addition of fill edges.

3.3 Adding a Non-edge incident on a Vertex of the Clique or the Independent Set of a Spider

In this section, we consider the (P_4-sparse,+1)-MinEdgeAddition Problem for a spider $G = (S, K, R)$ and a non-edge e incident on a vertex in $S \cup K$. In the following, for simplicity, we assume that $S = \{s_1, s_2, \ldots, s_{|K|}\}$, $K = \{k_1, k_2, \ldots, k_{|K|}\}$, and $R = \{r_1, r_2, \ldots, r_{|R|}\}$ where $|K| \geq 2$ and $|R| \geq 0$.

3.3.1 Thin Spider

Suppose that the spider G is thin and that s_i is adjacent to k_i for each $i = 1, 2, \ldots, |K|$. The following lemmas address the cases of the addition of the non-edge e.

Lemma 3.10. *The (P_4-sparse,+1)-MinEdgeAddition Problem for the thin spider $G = (S, K, R)$ and a non-edge e incident on a vertex in S and a vertex in K admits an optimal solution that requires $|K| - 1$ fill edges (including e).*

Proof. Suppose, without loss of generality, that $e = k_1 s_2$. Then, we can get a P_4-sparse graph if, in addition to the fill edge e, we add the fill edges $k_2 s_j$ ($j = 3, \ldots, |K|$) or alternatively the fill edges $s_1 k_j$ ($j = 3, \ldots, |K|$) for a total of $|K| - 1$ fill edges.

To prove the optimality of this solution, assume for contradiction that there is an optimal solution of the (P_4-sparse,+1)-MinEdgeAddition Problem for the thin spider G and the non-edge $k_1 s_2$ with at most $|K| - 2$ fill edges, that is, for e and at most $|K| - 3$ additional fill edges. Because the number of pairs s_i, k_i ($3 \leq i \leq |K|$) is equal to $|K| - 2$, there exists a pair s_j, k_j among them such that neither s_j nor k_j is incident on any of the fill edges. Then, due to the addition of the non-edge $e = k_1 s_2$, the

vertices s_1, k_1, s_2, s_j, k_j induce a forbidden subgraph F_5 or F_3 (depending on whether s_1, s_2 have been made adjacent or not, respectively); a contradiction. □

Lemma 3.11. *The (P_4-sparse,+1)-MinEdgeAddition Problem for the thin spider $G = (S, K, R)$ and a non-edge e with both endpoints in S admits an optimal solution that requires λ fill edges (including the non-edge e) where*

$$\lambda = \begin{cases} 2|K| - 3, & \text{if } |R| = 0; \\ 2|K| - 2, & \text{if } |R| = 1; \\ 2|K| - 1, & \text{if } |R| \geq 2. \end{cases}$$

Proof. Suppose, without loss of generality, that $e = s_1 s_2$. Then, we can get a P_4-sparse graph if, in addition to the fill edge e, we add the following fill edges:

- if $R = \emptyset$, s_1k_3, ..., $s_1k_{|K|}$ and s_2k_3, ..., $s_2k_{|K|}$;
- if $R = \{r_1\}$, s_1k_3, ..., $s_1k_{|K|}$, s_2k_3, ..., $s_2k_{|K|}$, and 1 fill edge (among r_1s_1, r_1s_2, k_1s_2, k_2s_1) so that the forbidden subgraph F_1 induced by s_1, s_2, k_1, k_2, r_1 be comes a P_4-sparse graph;
- if $|R| \geq 2$, s_1k_2, s_1k_3, ..., $s_1k_{|K|}$ and s_2k_1, s_2k_3, ..., $s_2k_{|K|}$, s_1k_2 (then k_1, k_2 become universal);

for a total of λ fill edges as stated above.

To prove the optimality of this solution, suppose for contradiction that there exists an optimal solution G' that requires *fewer* than λ fill edges (including e). First, consider that $|K| = 2$. Then the values of λ imply that G' requires at most 0 fill edges if $|R| = 0$, at most 1 fill edge if $|R| = 1$, and at most 2 fill edges if $|R| \geq 2$ including e in each case. The number of fill edges if $|R| = 0$ leads to a contradiction since e is added. If $|R| = 1$, then the addition of e results in an F_1 (= house) and at least 1 additional fill edge needs to be added, a contradiction again. If $|R| \geq 2$, then each vertex $r \in R$ and the vertices s_1, s_2, k_1, k_2 induce an F_1, and additional fill edges are needed. If neither the fill edge k_1s_2 nor the fill edge k_2s_1 is added, then we need 1 fill edge incident on each $r \in R$; since G' requires at most 2 fill edges (including e) then $|R|+1 \leq 2 \iff |R| \leq 1$, in contradiction to the fact that $||R| \geq 2$. Since G' uses at most 2 fill edges including e, only one of k_1s_2 and k_2s_1 can be added; let that be the fill edge k_1s_2. But then the vertices s_1, s_2, k_2, r_1, r_2 induce a forbidden subgraph F_5 or F_3 (depending on whether r_1, r_2 are adjacent on not), a contradiction.

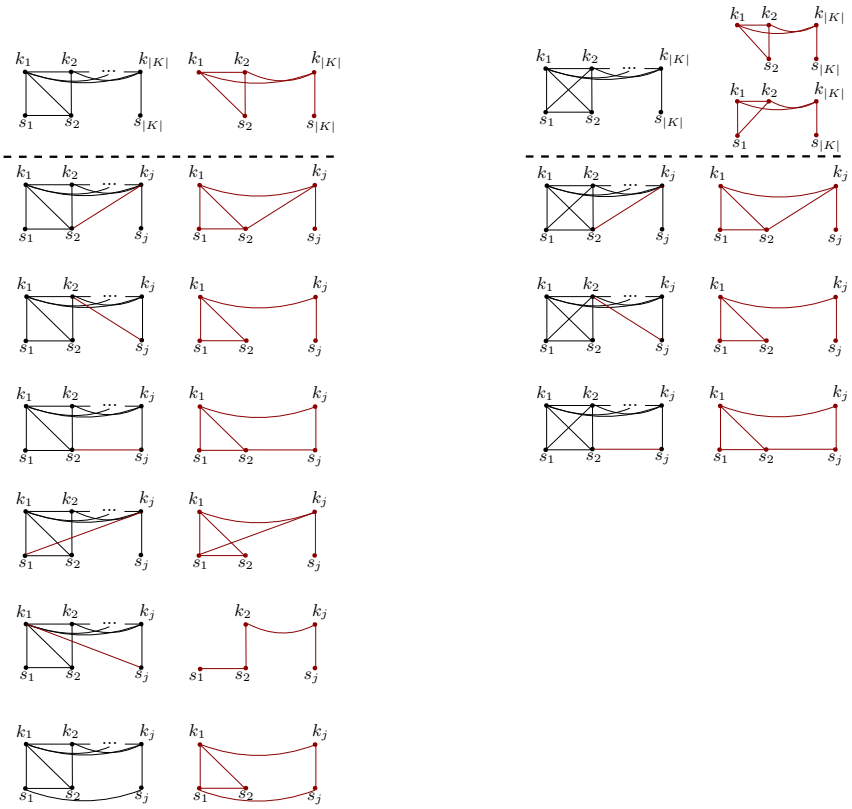

Figure 3.11: For the proof of Lemma 3.11: (left) at the top, the clique and stable set of the thin spider with the fill edges s_1s_2 and k_1s_2 (but not k_2s_1) and below the graph that results after the addition of 1 more fill edge; (right) at the top, the clique and stable set of the thin spider with the fill edges s_1s_2, k_1s_2, and k_2s_1 and below the graph that results after the addition of 1 more fill edge. The red graph next to each of the above graphs is an induced forbidden subgraph.

Now, consider that $|K| \geq 3$.

A. Suppose that neither the non-edge k_1s_2 nor the non-edge k_2s_1 is added. Then, the vertices k_1, k_2, s_1, s_2 induce a C_4. For each vertex k_j ($3 \leq j \leq |K|$), the vertices k_1, k_2, s_1, s_2, k_j induce a forbidden subgraph F_1 and thus for each such subgraph at least one fill edge needs to be added; since k_1s_2 and k_2s_1 cannot be added, this has to be adjacent to k_j (connecting it to s_1 or s_2). If only one of these two non-edges is added, say the edge k_js_2 but not the edge k_js_1, then an edge needs to be added adjacent to s_j, otherwise the vertices k_1, s_1, s_2, k_j, s_j induce a forbidden subgraph F_4. Thus, for each $j = 3, \ldots, |K|$, we need to add at least 2 fill edges, for a total of $2|K| - 4$ fill edges in addition to e. Moreover, if $|R| > 0$, for each vertex $r_i \in R$, the vertices k_1, k_2, s_1, s_2, r_i

induce a forbidden subgraph F_1 and thus at least 1 additional fill edge adjacent to r_i needs to be added. Then, the total number of fill edges is at least equal to $|R|+2\,|K|-3$ (including e), which is no less than the value of λ for all values of $|R|$.

B. Suppose that exactly one of the non-edges k_1s_2 and k_2s_1 is added. Without loss of generality, suppose that the non-edge k_1s_2 is added (and not the edge k_2s_1). Then, for each pair of vertices s_j, k_j $(3 \leq j \leq |K|)$, the vertices k_1, k_2, s_2, k_j, s_j induce a forbidden subgraph F_6. But a single fill edge is not enough (see Figure 3.11(left)). Thus at least $2 + 2\,(|K| - 2) = 2|K| - 2$ fill edges are needed (including e), which is no less than the value of λ for $|R| \leq 1$. If $|R| \geq 2$, the vertices s_1, s_2, k_2, r_1, r_2 induce a forbidden subgraph F_5 or F_3 (depending on whether r_1, r_2 are adjacent or not); hence, at least one more fill edge is needed, for a total of $2\,|K|-1$ fill edges (including e), which is no less than the value of λ for $|R| \geq 2$.

C. Suppose that both the edges k_1s_2 and k_2s_1 are added. Then, the vertices s_1, s_2, k_1, k_2 induce a K_4. For each pair of vertices k_j, s_j $(3 \leq j \leq |K|)$, the vertices s_1, k_1, k_2, k_j, s_j and k_1, k_2, s_2, k_j, s_j induce a forbidden subgraph F_5. But a single fill edge is not enough (as shown in Figure 3.11(right)). Thus, the total number of fill edges (including e) is at least $3 + 2\,(|K| - 2) = 2\,|K| - 1$, which is no less than the value of λ for all values of $|R|$. □

Lemma 3.12. *The (P_4-sparse,+1)-MinEdgeAddition Problem for the thin spider $G = (S, K, R)$ and a non-edge e incident on a vertex s in S and a vertex in R admits an optimal solution that requires $|K| - 1 + \mu$ fill edges (including e) where μ is the number of fill edges in an optimal solution of the (P_4-sparse,+1)-MinEdgeAddition Problem for the disconnected induced subgraph $G[\{s\} \cup R]$ and the non-edge e.*

Proof. Suppose, without loss of generality, that $s = s_1$ and $e = s_1r_1$ with $r_1 \in R$. Then, we can get a P_4-sparse graph if first we add the fill edges s_1k_j $(j = 2, 3, \ldots, |K|)$ which makes s_1 adjacent to all the vertices in K and then add the minimum number of fill edges so that the disconnected induced subgraph $G[\{s_1\} \cup R]$ with the non-edge s_1r_1 becomes P_4-sparse for a total of $|K| - 1 + \mu$ fill edges (including e); note that the only neighbor k_1 of s_1 in G is universal in $G[\{s_1\} \cup R]$.

To prove the optimality of this solution, we show that no optimal solution of the (P_4-sparse,+1)-MinEdgeAddition Problem for the thin spider G and the non-edge s_1r_1 has fewer than $|K| - 1$ fill edges incident on vertices in $(S \cup K) \setminus \{s_1, k_1\}$. Suppose, for contradiction, that there is a solution with at most $|K| - 2$ such fill edges.

Then, because the number of pairs k_i, s_i in $(S \cup K) \setminus \{s_1, k_1\}$ is equal to $|K| - 1$, there exists a pair k_j, s_j ($2 \leq j \leq |K|$) such that neither k_j nor s_j is incident to any of the fill edges. Then, due to the addition of the non-edge $e = s_1 r_1$, the vertices s_1, k_1, r_1, k_j, s_j induce a forbidden subgraph F_6; a contradiction. □

3.3.2 Thick Spider

Suppose that the spider G is thick and that s_i is non-adjacent to k_i for each $i = 1, 2, \ldots, |K|$. Additionally, according to our convention, we assume that $|K| \geq 3$.

Lemma 3.13. *The (P_4-sparse,+1)-MinEdgeAddition Problem for the thick spider $G = (S, K, R)$ and a non-edge e incident on a vertex in S and a vertex in K admits an optimal solution that requires only the fill-edge e.*

Proof. Suppose, without loss of generality, that $e = k_1 s_1$. Then, the addition of e makes k_1 universal, and no additional fill edges are needed, which is optimal. □

Lemma 3.14. *The (P_4-sparse,+1)-MinEdgeAddition Problem for the thick spider $G = (S, K, R)$ and a non-edge e with both endpoints in S admits an optimal solution that requires λ fill edges (including the non-edge e) where*

$$\lambda = \begin{cases} 2, & \text{if } |K| + |R| = 3; \\ 3, & \text{if } |K| + |R| \geq 4. \end{cases}$$

Proof. Suppose, without loss of generality, that $e = s_1 s_2$. Additionally, recall that we assume that $|K| \geq 3$. We can get a P_4-sparse graph if, in addition to the fill edge e, we add the fill edge $s_2 s_3$ if $|K| = 3$ and $|R| = 0$ (note that the complement of the resulting graph is the union of the P_2 $s_2 k_2$ and the P_4 $k_1 s_1 s_3 k_3$) and the fill edges $s_1 k_1$ and $s_2 k_2$ if $|K| + |R| \geq 4$ (note that k_1, k_2 are universal in the resulting graph).

To establish the optimality of this solution, we first observe that for $|K| = 3$ and $|R| = 0$, the vertices s_1, s_2, s_3, k_1, k_2 induce a forbidden subgraph F_1 and thus, at least 2 fill edges (including e) are needed. Next we show that for $|K| + |R| \geq 4$, no solution has fewer than 3 fill edges (including e). Suppose for contradiction that there is a solution with at most 2 fill edges. Due to e, the vertices s_1, s_2, s_3, k_1, k_2 induce a forbidden subgraph F_1, and thus at least 1 additional fill edge is needed.

A. This additional fill edge is $s_1 k_1$ or $s_2 k_2$. Due to symmetry, suppose without loss of generality that the fill edge $s_1 k_1$ is added. But then, the vertices s_1, s_2, s_3, k_2, k_3 induce a forbidden subgraph F_6, a contradiction.

B. None of the non-edges s_1k_1 and s_2k_2 is added. Then, the vertices s_1, s_2, k_1, k_2 induce a C_4 and for each $q \in \{s_3, \ldots, s_{|K|}\} \cup R$, the vertices s_1, s_2, k_1, k_2, q induce a forbidden subgraph F_1 and either the fill edge qs_1 or the fill edge qs_2 needs to be added (recall that none of s_1k_1, s_2k_2 is added). Since for the different possibilities of q, these fill edges are distinct and at most 1 fill edge is added in addition to e, then it must hold that $|K| + |R| - 2 = 1 \iff |K| + |R| = 3$, in contradiction to the fact that $|K| + |R| \geq 4$. □

Lemma 3.15. *The (P_4-sparse,+1)-MinEdgeAddition Problem for the thick spider $G = (S, K, R)$ and a non-edge e incident on a vertex s in S and a vertex in R admits an optimal solution that requires $1 + \mu$ fill edges (including e) where μ is the number of fill edges in an optimal solution of the (P_4-sparse,+1)-MinEdgeAddition Problem for the disconnected induced subgraph $G[\{s\} \cup R]$ and the non-edge e.*

Proof. Suppose, without loss of generality, that $s = s_1$ and $e = s_1r_1$ with $r_1 \in R$. Then, we can get a P_4-sparse graph if first we add the fill edge s_1k_1 which makes k_1 universal and s_1 adjacent to all the vertices in K, and then add the minimum number μ of fill edges (including e) so that the disconnected induced subgraph $G[\{s_1\} \cup R]$ with the non-edge e becomes P_4-sparse for a total of $1 + \mu$ fill edges.

The optimality of this solution follows from the fact that, due to the addition of the non-edge $e = s_1r_1$, the vertices s_1, s_2, k_1, k_2, r_1 induce a forbidden subgraph F_6 and so at least 1 fill edge incident on a vertex in $S \cup K$ and other than e is needed. □

3.4 Adding an Edge to a General P_4-sparse Graph

It is not difficult to see that the following fact holds.

Observation 3.3. *Let G be a P_4-sparse graph, T be the P_4-sparse tree of G, and uv be a non-edge that we want to add. Suppose that the least common ancestor of the tree leaves corresponding to u, v in T is a 0-node and let C_u (C_v resp.) be the connected components containing u (v resp.) in G after having removed all of their common neighbors. Then an optimal solution of the (P_4-sparse,+1)-MinEdgeAddition Problem for the graph G and the non-edge uv can be obtained from G after we have replaced the induced subgraph $G[C_u \cup C_v]$ by an optimal solution of the (P_4-sparse-2CC,+1)-MinEdgeAddition Problem for the union of $G[C_u]$ and $G[C_v]$ and the non-edge uv.*

In light of the lemmas in Section 3.3 and Observation 3.3, Algorithm P_4-sparse-Edge-Addition for solving the (P_4-sparse,$+1$)-MinEdgeAddition Problem for a P_4-sparse graph G and a non-edge uv computes the least common ancestor of the leaves corresponding to u and v, and if it is a 2-node, it applies the results in Lemmas 3.10-3.15 calling Algorithm (P_4-sparse-2CC)-Edge-Addition for the problem on a 2-component graph in the S-R case whereas if it is a 0-node, we apply Observation 3.3, compute the connected components that include u and v and call Algorithm (P_4-sparse-2CC)-Edge-Addition.

Algorithm (P_4-sparse-2CC)-Edge-Addition relies on the lemmas of Section 3.2; it has as input the connected components C_u and C_v containing u and v respectively and the P_4-sparse trees $T(G[C_u])$ and $T(G[C_v])$ of the induced subgraphs $G[C_u]$ and $G[C_v]$. It first checks if $C_u = \{u\}$ or $C_v = \{v\}$ in which case it calls Algorithm P_4-sparse-Tail-Addition. Otherwise it checks for the special cases of Lemmas 3.5 and 3.6 and if they apply, it computes the number of fill edges as suggested in the lemmas. Next, it ignores $T_{u,1}(G[C_u])$ if its root node is a 0-node and similarly for $T_{v,1}(G[C_v])$. Otherwise, it computes the fill edges of a P_4-sparse graph H on the vertex set $C_u \cup C_v$ having an edge set that is a superset of $E(G[C_u \cup C_v]) \cup \{uv\}$

- which results from making u universal in $G[C_u \cup C_v]$,
- which results from making v universal in $G[C_u \cup C_v]$,
- in which $T_{u,1}(H) = T_{u,1}(G[C_u])$,
- in which $T_{u,1}(H) = T_{v,1}(G[C_v])$, and
- as in the special case of Lemma 3.1

making recursive calls in the last 3 cases.

The algorithms can be easily augmented to return a minimum cardinality set of fill edges (including uv).

Time and space Complexity. Let the given graph G have n vertices and m edges. The P_4-sparse tree of a given P_4-sparse graph G can be constructed in $O(n+m)$ time and its number of nodes and height is $O(n)$. Then the time to compute the number of fill edges (excluding the call to Algorithm (2CC-P_4-sparse)-Edge-Addition) is $O(n)$.

Theorem 3.1. *Let G be a P_4-sparse graph on n vertices and m edges and u, v be two non-adjacent vertices of G. Then for the (P_4-sparse,$+1$)-MinEdgeAddition Problem for the*

graph G and the non-edge uv, we can compute the minimum number of fill edges needed (including uv) in $O(n^2)$ time and $O(n^2)$ space.

3.5 Concluding Remarks

In this chapter, we study the minimum completion problem of a P_4-sparse graph G with an added edge, namely given a P_4-sparse graph G and a non-edge xy (i.e., a pair of non-adjacent vertices x and y) of G, find the minimum number of non-edges of G that need to be added to G so that the resulting graph is also a P_4-sparse graph and contains xy as an edge and it is called as (P_4-sparse,$+1$)-MinEdgeAddition Problem.

For any optimal solution of the problem, we prove that there is an optimal solution whose form is of one of a small number of possibilities. This along with the solution of the problem when the added edge connects two non-adjacent vertices of a spider or connects two vertices in different connected components of the graph enables us to present a linear-time algorithm for the problem. Specifically, for optimal solution H and its P_4-sparse tree $T(H)$, there exist two cases: the root node of T to be 1-node, and the root node of T to be 2-node with two options, thin or thick spider. Also, we present the cases, according to what node is the least common ancestor of the leaves corresponding to u, v in $T(G)$: if it is a 0-node, then the graph G is a disconnected graph, which consists of two connected components each containing one of the endpoints of the added non-edge uv ((P_4-sparse-2CC,$+1$)-MinEdgeAddition Problem), and if it is a 2-node with at least one of the leaves corresponding to u, v being a child of the 2-node, the vertices u, v belong to the same spider subgraph.

As a consequence of lemmas and theorems about P_4-sparse graphs, we prove the efficiency and correctness of proposed Algorithm (P_4-sparse-2CC)-Edge-Addition according all the properties and the structure of P_4-sparse graph and its tree representation.

Chapter 4

Edge Modification: Application to Watermarking

4.1 Introduction

In graph theory, graph modification problems are fundamental. Garey and Johnson mentioned 18 different types of vertex and edge modification problems in 1979 [7]. Edge modification problems in graphs have a lot of applications in different areas, and many polynomial-time algorithms and NP-completeness proofs for this kind of problems are known. Edge modification challenges require making minor changes to an input graph's edge set in order to produce a graph with the required attribute, property and structure. These issues are significant in computer science and have application in several field, including molecular biology, numerical algebra and hiding information. Adding an edge corrects a false negative mistake, whereas deleting an edge corrects a false positive error. A graph is frequently used to model experimental

data, and edge modifications correlate to correcting errors in the data. We summarize below some of these applications.

Firstly, interval modification problems have important applications in physical mapping of DNA (see [9, 9]). Large DNA molecules are first divided into smaller fragments since direct sequencing of such large DNA molecules is currently not feasible. The order of the fragments is lost during this process, and its reconstructing is extremely difficult. Testing for overlap between any two fragments and using the result to deduce the order of the fragments is one method of reconstructing the order. This problem can be modeled as follows: Create a graph G where the edges connecting two vertices are defined by whether or not the respective segments of those vertices overlap. If G were an interval graph, the reconstruction issue would be solved by finding a realization of G. However, experimental data is error-prone and, hence, G is only close to being an interval graph. Completion, deletion, and editing problems can arise for both interval and unit interval graphs depending on the technology used and the kind of experimental errors.

Furthermore, the minimum fill-in problem, also known as the chordal completion problem, arises when a sparse symmetric positive-definite matrix is subjected to a numerical Gaussian elimination [8]. Finding an elimination sequence that introduces the fewest number of additional non-zero elements into the matrix is preferable because the computation's time and storage requirements depend on how sparse the matrix is. This problem is equivalent to the minimum fill-in problem, as shown by Rose [33].

Hiding information is another area where edge modification problems could apply. Software watermarking is a defense technique used to prevent or discourage software piracy by embedding a signature in the code, i.e., an identifier or equivalently a watermark representing the owner [64, 65]. When an illegal copy is made, the ownership can be claimed by extracting this identifier or watermark. In [66], a software watermarking system is presented which encodes an integer number w (i.e., a watermark) as a reducible permutation flow-graph $F[\pi^*]$ embeddable in the code through the use of a self-inverting permutation π^*. In this section, we theoretically investigate this watermarking system and exploit structural properties of the self-inverting permutation π^* encoding the watermark in order to prove its resilience to edge-modification attacks on the flow-graph $F[\pi^*]$. Based on the minimum number of edge modifications needed to be applied on $F[\pi^*]$ so that a different watermark can

be extracted from the resulting graph, we give a characterization of the watermarks as strong, intermediate or weak and provide good recommendations for the choices of watermark.

More precisely, the software watermarking problem can be described as the problem of *embedding* a structure w into a program P such that w can be reliably located and *extracted* from P even after P has been subjected to code transformations such as translation, optimization and obfuscation [37, 67]. In recent years, software watermarking has received considerable attention [68, 69, 70, 71, 72, 73]. The patent by Davidson and Myhrvold [70] presented the first published software watermarking algorithm. The preliminary concepts of software watermarking also appeared in [74] and patents [72, 73]. Collberg *et al*. [68, 69] presented detailed definitions for software watermarking. Authors of papers [75, 76] have given brief surveys of software watermarking research. Furthermore, Craver *et al*.[77, 78] present the capability of invisible watermarking schemes to resolve copyright ownership and generate a set of sufficient conditions that watermarks must satisfy to provide unambiguous proof of ownership.

Several software watermarking algorithms have appeared in the literature that encode watermarks as graph structures [79, 80, 70, 81], among which the graph-based methods that encode a watermark number w as a reducible flow-graph structure F capturing properties which make it resilient to attacks.

Such a graph-based codec algorithm was presented by Chroni *et al*. [82, 66]; the algorithm uses self-inverting permutations for encoding watermark numbers as reducible flow-graphs, which do not differ from the graph data structures built by real programs. Subsequently, Mpanti *et al*. [83] experimentally investigated the resilience of the used reducible permutation graph under edge-modifications. Based on the above ideas and watermarking scheme, Bento et al. [84, 85] introduced a linear-time algorithm which succeeds in retrieving deterministically the n-bit identifiers encoded by such graphs (with $n > 2$) even if 2 edges are missing. In addition, they defined a formal characterization of the class of graphs generated by Chroni *et al*.'s codec algorithms [82, 66], which are called canonical reducible permutation graphs and they give a linear-time recognition algorithm for them. Their results reinforce the effectiveness of Chroni *et al*.'s scheme as a possible software watermarking solution. Finally, Mpanti and Nikololopoulos [86] proposed two different reducible permutation flow-graphs incorporating important structural properties which are derived from the

bitonic subsequences forming the self-inverting permutation.

Chionis *et al*. [87] implemented the W-RPG (named WaterRpg) watermarking model on several Java application programs and evaluated it under various criteria in order to gain information about its practical behavior. They proposed call-graphs as key-objects in their watermarking model for embedding the graph into an application program and discussed the properties of dynamic call-graphs. In such a setting, the edges of the embedded watermark (as mentioned earlier, this is a reducible flow-graph) are inserted as additional calls to specific functions of the original application program. Then, an edge modification attack (an edge modification, insertion or deletion) is a corresponding modification of function calls whereas a node modification attack (a node insertion or deletion) affects the set of called functions. More recently, Novac *et al*. [88] proposed LLWM, an LLVM-based watermarking framework automating the embedding of watermarks, which can incorporate a codec watermarking system based on an existing watermarking technique as an LLVM pass. Moreover, they presented an experimental evaluation of known watermarking systems (including the W-RPG (named WaterRPG) codec watermarking system) in real-world applications and investigated them with respect to their stealth, credibility, capacity, overhead, and resilience against additive, subtractive, and distorting attacks.

Our Contribution. In light of the results in [88] and since the behavior of a watermarked program against attacks may differ based on the watermark used, it becomes apparent that the choice of watermark is of critical importance. To this effect and in the framework of the codec system presented in [82, 66], which encodes an integer watermark number w as a reducible permutation flow-graph $F[\pi^*]$ embeddable in the code through the use of a self-inverting permutation π^*, in this work, we present the types of changes in the self-inverting permutation π^* and corresponding edge modifications of the resulting reducible permutation graph $F[\pi^*]$ that maintain the structure and properties of these two components. These operations are Swap(), Move-in() and Move-out().

We next compute the minimum number of edge modifications on $F[\pi^*]$, namely $\texttt{minVM}(w)$, so that a different watermark is extracted from the modified reducible graph, and by means of that, we characterize the watermark as W-RPG-strong, W-RPG-intermediate or W-RPG-weak. Finally, we mention here that, this characterization enables us to provide good recommendations for the choice of watermarks.

Road Map. The chapter is organized as follows: In Section 4.2 we establish the notation and briefly present the W-RPG codec watermarking system. In Section we prove the resilience of any watermark $w \in R_n = [2^{n-1}, 2^n - 1]$ and show the main results of our work. In Section 4.4 we provide the characterizations of the watermark numbers. Finally, in Section we conclude the paper with possible future extensions.

4.2 The W-RPG Codec System

In this section we briefly present the W-RPG codec watermarking system proposed by Chroni *et al*. [82, 37, 66]. The system embeds an integer watermark w in code as a reducible permutation graph constructed from a self-inverting permutation (SiP) obtained from w. We next introduce some definitions that are key to describe algorithms for encoding numbers as self-inverting permutations in order to encoded as reducible permutation graph.

Self-inverting Permutation. A permutation π over a set A is an arrangement of the elements of the set A into some sequence or order, or if the set A is already ordered, π is a rearrangement of the elements of A into a one-to-one correspondence with itself. In this thesis, we consider permutations π over the set $N_n = \{1, 2, \ldots, n\}$. Let $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ be such a permutation. By π_i we denote the ith element of π, while by π_i^{-1} we denote the position in π of the element $\pi_i \in N_n$ [26]. The *length* of a permutation π is the number of elements in π. The *reverse* of π, denoted π^R, is the permutation $\pi^R = (\pi_n, \pi_{n-1}, \ldots, \pi_1)$. The inverse of π is the permutation $\tau = (\tau_1, \tau_2, \ldots, \tau_n)$ with $\tau_{\pi_i} = \pi_{\tau_i} = i$. It is obvious that every permutation has a unique inverse, and the inverse of the inverse is the original permutation. Also, a *subsequence* of a permutation $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ is a sequence $\alpha = (\pi_{i_1}, \pi_{i_2}, \ldots, \pi_{i_k})$ such that $i_1 < i_2 < \ldots < i_k$. If, in addition,$\pi_{i_1} < \pi_{i_2} < \ldots < \pi_{i_k}$, then we say that α is an *increasing subsequence* of π, while if $\pi_{i_1} > \pi_{i_2} > \ldots > \pi_{i_k}$ we say that α is a *decreasing subsequence* of π; the length of a subsequence α is the number of elements in α.

A cycle of π is a sequence $c = (\pi_{i_1}, \pi_{i_2}, \ldots, \pi_{i_p})$ such that ${\pi_{i_1}}^{-1} = \pi_{i_2}, {\pi_{i_2}}^{-1} = \pi_{i_3}, \ldots, \pi_{i_k})^{-1} = \pi_{i_1}$. An element i of π forms a *1-cycle* if $i = \pi_i^{-1}$; two elements i, j form a *2-cycle* if $i = \pi_j^{-1}$ and $j = \pi_i^{-1}$. The definition of the inverse of a permutation

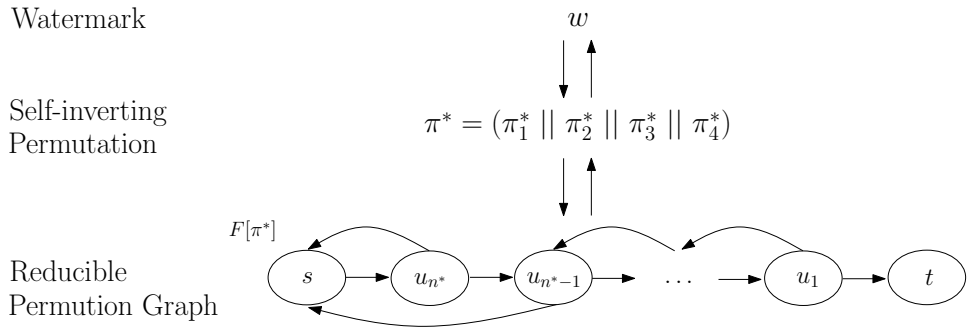

Figure 4.1: The main data components of the codec system for a watermark number $w \in R_n = [2^{n-1}, 2^n - 1]$ where $n^* = 2n + 1$.

implies that a permutation is a self-inverting permutation if and only if all its cycles are of length 1 or 2; hereafter, we shall denote a 2-cycle by $(x; y)$ with $x > y$ and a 1-cycle by (x) or, equivalently, $(x; x)$.

Definition 4.1. Let $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ be a permutation over the set N_n, $n > 1$. The inverse of the permutation π is the permutation $q = (q_1, q_2, \ldots, q_n)$ with $q_{\pi_i} = \pi_{q_i} = i$. A *self-inverting permutation* (or SiP) of length n is a permutation that is its own inverse: for each $1 \leq i \leq n$, $\pi_{\pi_i} = i$.

For a watermark whose binary representation has length n, the produced SiP (denoted π^*) has length $n^* = 2n + 1$. For example, for the watermark $w = 12$, the SiP produced is $\pi^* = (5, 6, 9, 8, 1, 2, 7, 4, 3)$, where $n^* = 9$ and $n = 4$ because it exists in range R_4. The reverse of π^* is $\pi^R = (3, 4, 7, 2, 1, 8, 9, 6, 5)$ and an increasing subsequence of π^* is $\alpha_1 = (5, 6, 9, 8)$.

Reducible Permutation Graph. We consider finite graphs with no multiple edges. A *flow-graph* is a directed graph with an initial node s from which all other nodes are reachable. A directed graph G is *strongly connected* when there is a directed path $x \to y$ for all nodes x, y in $V(G)$. A node $u \in V(G)$ is an *entry* for a subgraph H of the graph G when there is a path $p = (y_1, y_2, \ldots, y_k, u)$ such that $p \cap H = \{u\}$ (see [89, 90]).

Definition 4.2. A flow-graph is *reducible* when it does not have a strongly connected subgraph with two (or more) entries.

There are some other equivalent definitions of the reducible flow-graphs which use a few more graph-theoretic concepts. A depth first search (DFS) of a flow-graph

partitions its edges into *tree edges* (making up a spanning tree known as a DFS tree), *forward edges* (pointing to a successor in the spanning tree), *back edges* (pointing to a predecessor in the spanning tree, plus the cycle-edges), and *cross edges* (the remaining edges). It is well known that tree, forward, and cross edges form a dag known as a DFS dag. Hecht and Ullman show that a flow-graph F is reducible if and only if F has a unique DFS dag or equivalently if and only if the graph F can be transformed into a single node by repeated application of the transformations $\mathcal{T}_1$ and $\mathcal{T}_2$, where $\mathcal{T}_1$ removes a cycle-edge, and $\mathcal{T}_2$ picks a non-initial node y that has only one incoming edge $(x; y)$ and glue nodes x and y [89, 90].

Encoding and Decoding Process. The codec system W-RPG (see Figure 4.1) consists of the algorithms `Encode_W.to.SiP` and `Encode_SiP.to.RPG`, which enable us to encode a watermark w into a self-inverting permutation π^* of length $n^* = 2n + 1$ and the latter into a reducible permutation graph $F[\pi^*]$ on $n^* + 2 = 2n + 3$ nodes, respectively, where n is the length of the binary representation of w, as well as the corresponding decoding algorithms `Decode_SiP.to.W` and `Decode_RPG.to.SiP`.

The system's encoding strategy allows it to encode any integer w as a self-inverting permutation π^* of length $n^* = 2n + 1$, where $n^* = 2\lceil \log_2 w \rceil + 1$. Additionally, the produced RPG $F[\pi^*]$ has nodes u_{n^*+1}, u_{n^*}, ..., u_i, ..., u_0, which include a unique root node $s = u_{n^*+1}$, namely header node, with only 1 outgoing edge (from which all other nodes of $F[\pi^*]$ are reached). Also it exists the footer node which is a unique sink node $t = u_0$ with only 1 incoming edge and no outgoing edges, and all nodes that form the body of the produced reducible permutation graph $F[\pi^*]$, namely n^* nodes with 2 outgoing edges: from node u_i we have a forward edge to node u_{i-1} and a back-edge to node u_m, where $m > i$. See Figure 4.1. For example, for the watermark $w = 12$, the produced SiP is $(5, 6, 9, 8, 1, 2, 7, 4, 3)$, and the RPG is as shown in Figure 4.2. This codec system uses a construction method that captures crucial structural characteristics and properties into π^* to encode an integer w as a self-inverting permutation π^*. These characteristics and properties allow an attack-detection system to recognize changes (node and edge modifications) that an attacker has made.

Moreover, we note that the forward edges of $F[\pi^*]$ form a forward Hamilton path whereas for the back-edges we have [66]:

Note 4.1. In the reducible permutation graph $F[\pi^*]$ produced from SiP π^*, for each $i = 1, 2, \ldots, n^*$, the back-edge from node u_i points to node s if there is no larger

number to the left of $i \in \pi^*$, otherwise it points to node u_m where m is the rightmost number to the left of i in π^* that is larger than i.

For example, in the graph $F[\pi^*]$ which corresponds to the SiP $(7, 8, 11, 12, 13, 10, 1, 2, 9, 6, 3, 4, 5)$ encoding the watermark $w = 51 = (110011)_2$, the back-edge from nodes $u_7, u_8, u_{11}, u_{12}, u_{13}$ points to s, from u_{10} points to u_{13}, from u_1, u_2, u_9 points to u_{10}, from u_6 points to u_9, and from u_3, u_4, u_5 points to u_6.

The permutation π^* can be encoded as a reducible permutation flow-graph $F[\pi^*]$ and correctly decoded from $F[\pi^*]$ in $O(n^*)$ time and space, where n^* is the length of π^*. Recall that the SiP π^* can be encoded into a reducible permutation graph $F[\pi^*]$ in $O(n)$ time and space using algorithm `Encode_SiP.to.RPG` and decoded from $F[\pi^*]$ in $O(n)$ time and space using algorithm `Decode_RPG.to.SiP`; see [82].

Properties of the W-RPG Codec System. To be effective, a graph watermark codec system needs to provide several key properties of its structural components. In proposed algorithms by Chroni and Nikolopoulos, the suggested watermarking technique has properties that make it robust to multiple code transformations. Based on the structure of a self-inverting permutation π^* produced by Algorithm `Encode_W.to.SiP`, which takes as input an integer w, and the type of reducible permutation graphs $F[\pi^*]$, which encoded a self-inverting permutation π^* by Algorithm `Encode_SiP.to.RPG`, four important properties (Odd-One Property, Bitonic Property, Block Property and Range Property) of π^* or $4-$Chain Property are incorporated into the codec watermark graph $F[\pi^*]$ in order to make it resilient against attacks [66]. In particular, the Odd-One Property describes the property of SiP (self-inverting permutation) that all its cycles are of length 1 or 2, and the Bitonic Property refers to construction of self-inverting permutation π^* from the bitonic sequence $\pi^b = X||Y^R$, where X and Y are increasing subsequences and thus the bitonic property of π^b is encapsulated in the cycles of π^*. Moreover, the Block Property is mentioned that the first part of $B' = 00\ldots0||B||0$, where B be the binary representation of the integer w, contains the leftmost n bits, each equal to 0, where n is the length of the binary representation of the integer w, and the Range Property is that the graph $F[\pi^*]$ produced by W-RPG codec system consists of $|V(F[\pi^*])| = n^* + 1 = 2n + 3$ nodes, where $n^* = 2n + 1$. Whenever a flow-graph $F[\pi^*]$, encoding the watermark w_i, is attacked having k edges modified, then the decoding algorithm returns a true-incorrect watermark $w_j \neq w_i$, if all the four properties are satisfied during the decoding process, i.e., the process of getting the

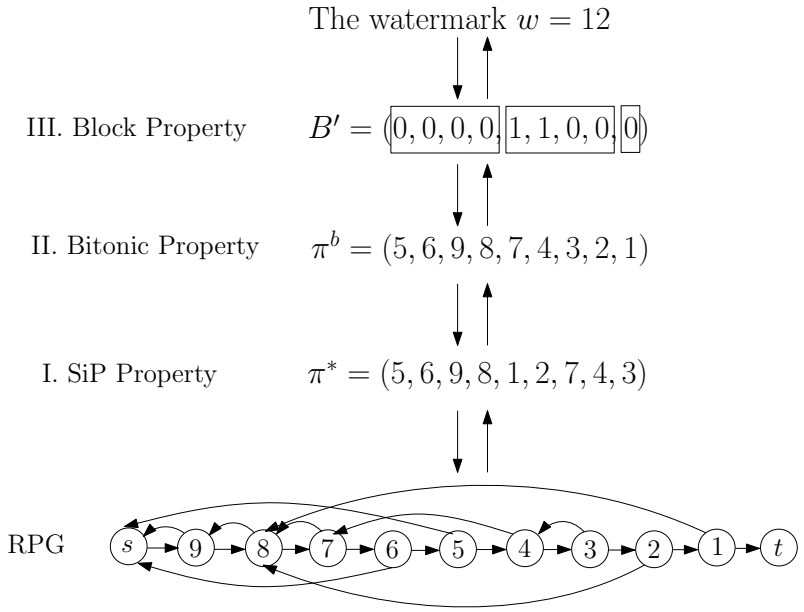

Figure 4.2: The reducible permutation graph produced for the watermark $w = 12$ by using the self-inverting permutation $\pi^* = (5, 6, 9, 8, 1, 2, 7, 4, 3)$ with its system code properties.

watermark w_i from $F'[\pi^*]$.

Structure of the SiPs Used in the W-RPG Codec System. Consider a self-inverting permutation π^* encoding an integer $w \in R_n = [2^{n-1}, 2^n - 1]$, where n is the length of the binary representation of w; we distinguish the following two cases [66]:

Zero-and-One case: $w \in [2^{n-1}, 2^n - 2]$. In this case, the structure of π^* consists of four subsequences, that is, $\pi^* = \pi_1^* \mid\mid \pi_2^* \mid\mid \pi_3^* \mid\mid \pi_4^*$, having the following forms:

$$\pi^* = (n+1, n+2, \ldots, n+k) \mid\mid (p_1, p_2, \ldots, \beta) \mid\mid (1, 2, \ldots, k, \alpha) \mid\mid (q_1, q_2, \ldots, \gamma),$$

where

- $\pi_1^* = (n+1, n+2, \ldots, n+k)$ is an *increasing* sequence of length k consisting of k consecutive integers starting always with $n+1$, where $k \geq 1$,
- $\pi_2^* = (p_1, p_2, \ldots, \beta)$ is a *bitonic* sequence of length $n-k$ with elements of the set $\{n+k+2, n+k+3, \ldots, 2n+1\}$, where $max = 2n+1$,
- $\pi_3^* = (1, 2, \ldots, k, \alpha)$ is an *increasing* sequence of length $k+1$ consisting of k consecutive integers starting always with 1 followed by the integer $\alpha = n+k+1$ (note that the integer α forms the 1-cycle of π^*), and

- $\pi_4^* = (q_1, q_2, \ldots, \gamma)$ is a sequence of length $n-k$ with elements of the set $\{k+1, k+2, \ldots, k+i, \ldots, n\}$, where $k+i$ is the index of the i-th smallest element of π_2^*. Thus, the integer γ is the index of the $max = 2n+1$ element of π_2^*.

From the structure of subsequences π_1^*, π_2^*, π_3^* and π_4^* of the SiP π^*, it follows that (i) the subsequences π_1^* and π_2^* contain all the elements of the set $(\{n+1, \ldots, 2n+1\} - \{\alpha\})$, (ii) all the elements of π_2^* are larger than those of π_1^*, (iii) the last element β of π_2^* is greater than any element of $\pi_3^*||\pi_4^*$, and (iv) the last element α of π_3^* is greater that any element of π_4^*.

All-One case: $w = 2^n - 1$. In this case, the sequences π_2^* and π_4^* have no elements and, thus, π^* consists of two subsequences, that is, $\pi^* = \pi_1^* \mid\mid \pi_3^*$, having the following forms:

$$\pi^* = (n+1, n+2, \ldots, 2n) \mid\mid (1, 2, \ldots, n, 2n+1),$$

where

- $\pi_1^* = (n+1, n+2, \ldots, 2n)$ is an *increasing* sequence of length n consisting of n consecutive integers starting with $n+1$, and
- $\pi_3^* = (1, 2, \ldots, n, 2n+1)$ is an *increasing* sequence of length $n+1$ consisting of n consecutive integers starting always with 1 followed by the $max = 2n+1$ element of π^*. In this case, the max element $2n+1$ forms the 1-cycle of π^*.

Since the subsequences π_3^* and π_4^* contain the element α located at position α (1-cycle) and at any other position t the location of t in π_1^* or π_2^*, it holds:

Note 4.2. For a self-inverting permutation $\pi^* = \pi_1^* \mid\mid \pi_2^* || \pi_3^* \mid\mid \pi_4^*$ structured as above, the contents of $\pi_1^* \mid\mid \pi_2^*$ uniquely determine the entire π^*.

For example, if we know that $\pi_1^* = (7, 8)$ and $\pi_2^* = (11, 12, 13, 10)$, then we know that 9 is in position 9 (1-cycle), and positions $7, 8, 10, 11, 12, 13$ contain $1, 2, 6, 3, 4, 5$, respectively.

Finally, the decoding process of the W-RPG system ensures that a watermark number gets extracted from every SiP structured as described above.

4.3 Characterizing Watermark Numbers

In this section, we classify the watermark numbers encoded as a reducible permutation graph $F[\pi^*]$ regarding the valid edge-modification attacks on $F[\pi^*]$ and we present the main theorem for the characterization of them. Let $F[\pi^*]$ be a flow-graph which encodes the integer w and let F' be the graph resulting from $F[\pi^*]$ after an edge modification. Then, we say that F' is either a *false-incorrect* or a *true-incorrect* graph:

- F' is *false-incorrect* if the decoding process of the W-RPG codec system fails to return a watermark from the graph $F'[*]$, whereas
- F' is *true-incorrect* if the decoding process extracts from F' and returns an integer $w' \neq w$.

4.3.1 Valid Modification Operations on a SiP

We concentrate on the edge modifications that turn an RPG $F[\pi^*]$ into a true-incorrect graph F'. Then, clearly, (i) F' has to have the structure of RPGs produced by the encoding process of the W-RPG system and (ii) the SiP resulting from it in the decoding process has to match the structure of SiPs presented in Section 4.2. Since condition (i) can be easily checked, we concentrate on condition (ii) and we investigate which changes in the structure of SiPs used in the W-RPG system maintain this structure. We consider the necessary SiP operations to transform a SiP structured as described in Section 4.2 into another such SiP. In particular, in light of Remark 4.2, it suffices to concentrate on the elements of π_1^* and π_2^*. We have the following operations:

- `Swap()`: the operation swaps two elements in either π_1^* or π_2^* also making corresponding changes in the rest of π^*. Since π_1^* is increasing, this can only happen in π_2^* and only in one of the following two ways:
  - The element max is swapped with $max - 1$ (which is next to it) or with $max - 2$ only if π_2^* contains the subsequence $(max - 2, max - 1, max)$ or $(max, max - 1, max - 2)$. This follows easily from the fact that π_2^* is bitonic and hence it should contain the subsequence $(max - 2, max - 1, max)$ or $(max - 2, max, max - 1)$ or their right-to-left counterparts.

– Two elements $i, j \neq max$ are swapped. Then, it is easy to see that i, j cannot both belong either to the increasing or to the decreasing subsequence of π_2^* and that i, j should be consecutive numbers (To see the latter, consider that i belongs to the increasing subsequence and j to the decreasing subsequence of π_2^* and assume that $j = i + k > i + 1$; the case $j < i - 1$ is symmetric. Since j can be swapped with i without violating the SiP structure, the element $i + \ell$ following i should be larger than j. But then, all the elements $i + 1, i + 2, \ldots, i + \ell - 1$ (which include j) should belong to the decreasing subsequence, and thus the element following j is $j - 1 > i$. But then placing i before $j - 1$ would violate the fact that they belong to a decreasing subsequence.) As an example consider the operation `Swap()` on the elements 12 and 13 of the SiP $(7, 8)||(11, 12, 13, 10)||(1, 2, 9)||(6, 3, 4, 5)$. Then, for the resulting permutation to have the described structure, the elements 4 and 5 in π_4^* get also swapped and the resulting SiP is $(7, 8)||(11, 13, 12, 10)||(1, 2, 9)||(6, 3, 5, 4)$ which can be produced from $F[\pi^*]$ if the 3 back-edges $(11, s)$, $(10, 13)$, $(3, 6)$, and $(4, 6)$ get changed to $(11, 13)$, $(10, 11)$, $(3, 5)$, and $(4, 5)$, respectively.

- `Move-in()`: the operation takes elements of either π_1^* or π_2^* and moves them in other positions in the same subsequence also making corresponding changes in the rest of π^*. As with the `Swap()` operation, this operation cannot be performed on elements of π_1^*. As an example consider the operation `Move-in()` on the element 11 of the SiP $(7, 8)||(11, 12, 13, 10)||(1, 2, 9)||(6, 3, 4, 5)$. Then, for the resulting permutation to have the described structure, element 11 is inserted between 13 and 10 and the element 5 in π_4^* gets inserted between 6 and 3. The resulting SiP is$(7, 8, 12, 13, 11, 10, 1, 2, 9, 6, 5, 3, 4)$ which can be produced from $F[\pi^*]$ if the 4 back-edges $(11, s)$, $(10, 13)$, $(3, 6)$, and $(4, 6)$ get changed to $(11, 13)$, $(10, 11)$, $(3, 5)$, and $(4, 5)$, respectively.

- `Move-out()`: the operation takes elements of either π_1^* or π_2^* and moves them in other subsequences also making corresponding changes in the rest of π^*. As this operation can be applied to both π_1^* and π_2^*, we have two cases.

  (a) a number of elements of π_1^* are moved to π_2^* and π_3^*. In particular, the SiP structure implies that these elements must be the *largest* consecutive

numbers in $\pi^* - 1$; the smallest of them is moved to π_3^* becoming the 1-cycle whereas the remaining elements and the element that used to be the 1-cycle are placed in π_2^*. As an example consider the operation `Move-out()` on 1 element of π_1^* of the SiP $(7,8)||(11,12,13,10)||(1,2,9)||(6,3,4,5)$. Then, this element has to be the largest element 8 of π_1^* which becomes the 1-cycle whereas the element 9 is inserted in π_2^*. If it is placed before 11 then we get the SiP $(7,9,11,12,13,10,1,8,2,6,3,4,5)$ which can be produced from $F[\pi^*]$ if the 4 back-edges $(9,10)$, $(8,s)$, $(2,10)$, and $(6,9)$ get changed to $(9,s)$, $(8,10)$, $(2,8)$, and $(6,8)$, respectively, whereas if it is placed after 10, we get the SiP $(7,11,12,13,10,9,1,8,6,5,2,3,4)$ which can be produced from $F[\pi^*]$ if the 5 back-edges $(1,10)$, $(8,s)$, $(2,10)$, $(3,6)$, and $(4,6)$ get changed to $(1,9)$, $(8,9)$, $(2,5)$, $(3,5)$, and $(4,5)$, respectively.

(b) a number of elements of π_2^* are moved to π_1^* and π_3^*. In particular, the SiP structure implies that these elements must be the *smallest* consecutive numbers in $\pi^* - 1$; the largest of them is moved to π_3^* becoming the 1-cycle whereas the remaining elements and the element that used to be the 1-cycle are placed at the end of π_1^*. As an example consider the operation `Move-out()` on an element of π_1^* of the SiP $(7,8)$ || $(11,12,13,10)$ || $(1,2,9)$ || $(6,3,4,5)$. Then, this element has to be the smallest element 10 of π_2^* which becomes the 1-cycle whereas the element 9 is inserted in π_1^*. If it is placed before 11 then we get the SiP $(7,8,9,11,12,13,1,2,3,10,4,5,6)$ which can be produced from $F[\pi^*]$ if the 7 back-edges $(9,10)$, $(1,10)$, $(2,10)$, $(3,6)$, $(4,6)$, $(5,6)$, and $(6,9)$ are changed to $(9,s)$, $(1,13)$, $(2,13)$, $(3,13)$, $(4,10)$, $(5,10)$, and $(6,10)$, whereas if it is placed after 13, we get the SiP $(7,8,11,12,13,9,1,2,6,10,3,4,5)$ which can be produced from $F[\pi^*]$ if the 6 back-edges $(1,10)$, $(2,10)$, $(9,10)$, $(3,6)$, $(4,6)$, and $(5,6)$ get changed to $(1,9)$, $(2,9)$, $(9,13)$, $(3,10)$, $(4,10)$, and $(5,10)$, respectively.

These three operations are called as SiP operations where, given a SiP π_1^*, we can create another SiPs π_i^* through the `Swap()`, `Move-in()` and/or `Move-out()` operations.

Definition 4.3. Let w be a watermark number encoded as a reducible permutation graph $F[\pi^*]$ through the self-inverting permutation π^*. The minimum number of valid edge-modification (or `minVM`(w)) of the graph $F[\pi^*]$ is the minimum SiP operations in π^* in order to create another SiPs π_i^*.

Specifically, an edge-modification on graph $F[\pi^*]$ which preserves the SiP operations, i.e., the decoding algorithm on graph $F[\pi^*]$ returns a true-incorrect watermark, is called valid edge-modification, otherwise it is called invalid edge-modification, i.e., the graph $F[\pi^*]$ does not satisfy the SiP operations and thus the decoding algorithm returns nothing.

4.3.2 Main Results

The above description implies that these operations cover all possible cases to generate a SiP of the described structure. Then, we can prove the main result of this work.

Theorem 4.1. *Let w be a watermark number encoded as a reducible permutation graph $F[\pi^*]$ through the self-inverting permutation π^* and let $b(w) = b_1 b_2 \dots b_n$ be its binary representation and $B = b_2 b_3 \dots b_{n-1}$ be the internal block of w. For the minimum number of valid edge-modification $\texttt{minVM}(w)$ of the graph $F[\pi^*]$ we distinguish the following cases:*

1. *The internal block B of w contains at least two 0s. Then, $\texttt{minVM}(w) = 3$.*
2. *The internal block B of w contains exactly one 0 (i.e., the watermark number has the form $w = 11^{\ell}01^{r}b_n$ with $\ell, r \geq 0$ and $\ell + r = n - 3$). Then,*

$$\texttt{minVM}(w) = \begin{cases} 4 + min\{\ell, r-1\}, & \text{if } b_n = 0 \text{ and } r > 0 \\ 4, & \text{if } b_n = 0 \text{ and } r = 0 \\ 4 + min\{\ell, r\}, & \text{if } b_n = 1 \text{ and } r \geq 0 \end{cases}$$

   where ℓ, r are the numbers of consecutive 1s before and after the unique 0 in B, respectively.

3. *The internal block B of w contains no 0s (i.e., the watermark number has the form $w = 11 \dots 1b_n$). Then, $\texttt{minVM}(w) = 4$.*

Proof. We distinguish the following three cases depending on the number of 0s in the internal block $B = b_2 \cdots b_{n-1}$ of the watermark $w = b_1 b_2 \cdots b_{n-1} b_n$.

Case 1. The internal block of the watermark w contains at least two 0s. In this case, by construction (see Algorithm `Encode_W.to.SiP` [82]), the $max = 2n + 1$ and the $max - 1 = 2n$ elements of π^* are not located in the last position of π_2^*, that is, $\beta \neq max - 1$ and $\beta \neq max$.

Let $\gamma = \pi^{*-1}_{2n+1}$ be the index of the max element in π^*. Since π^*_2 is a bitonic sequence, it follows that the elements max and $max-1$ are in consecutive positions in π^*_2 and thus the index π^{*-1}_{2n} of the $max-1=2n$ element is either $\gamma-1$ or $\gamma+1$.

We assume that $\gamma - 1 = \pi^{*-1}_{2n}$ (the case where $\gamma+1=\pi^{*-1}_{2n}$ is handled in a similar manner). In this case, the watermark number w encodes a SiP $\pi^* = \pi^*_1 \,||\, \pi^*_2 \,||\, \pi^*_3 \,||\, \pi^*_4$ having the following structure:

$$\begin{aligned}\pi^* = {} & (n+1, n+2, \ldots, n+k) \,||\, (p_1, p_2, \ldots, p_i, max-1, max, p_j, \ldots, \beta) \,||\\ & (1, 2, \ldots, k, \alpha) \,||\, (q_1, q_2, \ldots, q_m, \ldots, \gamma-1, \gamma),\end{aligned} \tag{4.1}$$

Now we perform valid modifications on the elements of the SiP π^*. In fact, we apply the process `Swap()` on the elements $max-1$ and max of the SiP π^* resulting in the new SiP ϕ^* having the following structure:

$$\begin{aligned}\phi^* = {} & (n+1, n+2, \ldots, n+k) \,||\, (p_1, p_2, \ldots, p_i, max, max-1, p_j, \ldots, \beta) \,||\\ & (1, 2, \ldots, k, \alpha) \,||\, (q_1, q_2, \ldots, q_m, \ldots, \gamma, \gamma-1),\end{aligned} \tag{4.2}$$

where along with the swapping of max and $max-1$ the elements at positions max and $max-1$ get also swapped.
Let $F[\phi^*]$ be the true-incorrect reducible permutation graph that results from ϕ^*. The graph $F[\phi^*]$ meets the structural properties of the RPGs used in the W-RPG system. On the other hand, it is not difficult to see that the only back-edges that need to be changed are those of the nodes corresponding to $max-1$, p_j, and $\gamma-1$; the back-edges $(max-1, s)$, (p_j, max), and $(\gamma-1, q_m)$ where $1 \le m \le n-k-2$ need to be changed to $(max-1, max)$, $(p_j, max-1)$, and $(\gamma-1, \gamma)$, respectively.

From the above, we conclude that the graph $F[\phi^*]$ is a true-incorrect reducible permutation graph, encoding a watermark number $w' \neq w$, which results from $F[\pi^*]$ after performing 3 edge-modifications on its back-edges. Thus, $\texttt{minVM}(w) = 3$.

Case 2. The internal block B of the watermark w contains exactly one 0 and thus w has the form:

$$w = 1\underbrace{1\ldots1}_{\ell}0\underbrace{1\ldots1}_{r}b_n$$

where $\ell, r \ge 0$ and $\ell + r = n-3$.

Subcase 2.1: $\mathbf{b_n = 0}$ and $\mathbf{r > 0}$. In this case, the watermark number w is encoded by

a SiP $\pi^* = \pi_1^*||\pi_2^*||\pi_3^*||\pi_4^*$ having the following structure:

$$\pi^* = (n+1, n+2, \ldots, n+\ell+1) \;||\; (n+\ell+3, \ldots, n+\ell+r+2, max, max-1) \;||\; (1, 2, \ldots, \ell+1, \alpha) \;||\; (\ell+2, \ldots, \ell+r+1, n, n-1), \tag{4.3}$$

where $max = 2n+1$, $max - 1 = 2n$ and $\alpha = n + \ell + 2$. Now, the watermark w is

$$w = 1\underbrace{1\ldots1}_{\ell}0\underbrace{1\ldots1}_{r}0$$

and, thus, $max - 1 = 2n$ is the last element of the sequence π_2^*; in fact, $\pi_{2n}^{-1} = n$ and $\pi_{2n+1}^{-1} = n - 1$.

Let ϕ^* be the SiP resulting from the permutation π^* after performing some valid modifications on its elements and let $F[\phi^*]$ be the true-incorrect reducible permutation graph encoding a watermark number $w' \neq w$; the valid modifications belong to the following three categories:

(i) `Swap()`. Based on the description of the `Swap()` operation, the operation can be applied only on the pairs $(max-1, max)$ and $(max-1, max-2)$, where $max - 2 = n + \ell + r + 2$.

The application of `Swap()` on the pair $(max, max-1)$ results in the SiP ϕ^*:

$$\phi^* = (n+1, n+2, \ldots, n+\ell+1) \;||\; (n+\ell+3, \ldots, n+\ell+r+2, max-1, max) \;||\; (1, 2, \ldots, \ell+1, \alpha) \;||\; (\ell+2, \ldots, \ell+r+1, n-1, n). \tag{4.4}$$

Since ϕ^* is a valid SiP and $\phi^* \neq \pi^*$, the graph $F[\phi^*]$ is a true-incorrect reducible permutation graph encoding the watermark number $w' \neq w$, where

$$w' = 1\underbrace{1\ldots1}_{\ell}0\underbrace{1\ldots1}_{r}1$$

i.e., the last bit changed from 0 to 1. The back-edges that need to be changed so that we get the graph $F[\phi^*]$ are those emanating from the nodes corresponding to $max - 1, 1, 2, \ldots, \ell + 1, \alpha, n - 1$ and thus, in this case, the number of edge modifications needed is $4 + \ell$.

In turn, the application of `Swap()` on the pair $(max-2, max-1)$ results in a SiP ϕ^* such that the corresponding graph $F[\phi^*]$ can be produced from $F[\pi^*]$

after $5+\ell$ edge modifications. Indeed, in this case the structure of ϕ^* is:

$$\phi^* = (n+1, n+2, \ldots, n+\ell+1) \;||\; (n+\ell+3, \ldots, max-1, max, n+\ell+r+2) \;||\; (1, 2, \ldots, \ell+1, \alpha) \;||\; (\ell+2, \ldots, \ell+r, n, n-2, n-1) \tag{4.5}$$

and, thus, the back-edges that need to be changed so that we get the graph $F[\phi^*]$ are those emanating from the nodes corresponding to $n+\boldsymbol{\ell}+r+2, max-1, 1, 2, \ldots, \ell+1, \alpha, n-2$.

(ii) `Move-in()`. The `Move-in()` operation is performed on sequence π_2^* which, in this case, is a bitonic sequence of the form:

$$\pi_2^* = (n+\ell+3, \ldots, n+\ell+i-1, n+\ell+i, n+\ell+i+1, \ldots, n+\ell+r+2, max, max-1)$$

Let $n+\ell+i$ $(3 \leq i \leq r+2)$ be an element of the increasing subsequence of π_2^*. Since $n+\ell+i < max-1$, a `Move-in()` operation on the element $n+\ell+i$ results into moving the element in the last position of π_2^*. Thus, the resulting sequence ϕ_2^* is:

$$\phi_2^* = (n+\ell+3, \ldots, n+\ell+i-1, n+\ell+i+1, \ldots, n+\ell+r+2, max, max-1, n+\ell+i).$$

Then, $\pi_1^* = \phi_1^*$ and $\pi_3^* = \phi_3^* = (1, 2, \ldots, \ell+1, \alpha)$, while the sequences π_4^* and ϕ_4^* are:

$$\pi_4^* = (\ell+2, \ldots, \ell+i-2, \ell+i-1, \ell+i, \ldots, \ell+r+1, n, n-1)$$

and

$$\phi_4^* = (\ell+2, \ldots, \ell+i-2, n, \ell+i-1, \ldots, \ell+r, n-1, n-2).$$

In this case, the resulting true-incorrect graph $F[\phi^*]$ contains $1+|\phi_3^*|+(r-i+3)$ nodes whose back-edge needs to be updated: these nodes correspond to the element $n+\ell+i$ of sequence ϕ_2^*, to all the elements $1, 2, \ldots, \ell+1, \alpha$ of sequence ϕ_3^*, due to element $n+\ell+i$, and to the elements $\ell+i-1, \ldots, \ell+r, n-2$ of sequence ϕ_4^*. Thus, the total number of such nodes is $6+\ell+r-i$ where $3 \leq i \leq r+2$ and $r > 0$. It follows that the graph $F[\phi^*]$ is produced after at least $4+\ell$ edge modifications. In fact, there is a true-incorrect graph $F[\phi^*]$ that results from $F[\pi^*]$ after exactly $4+\ell$ edge modifications: it suffices to apply a `Move-in()`

operation on the element $n+\ell+r+2 = max-2$. This graph $F[\phi^*]$ encodes the watermark number $w' \neq w$, where

$$w' = 1\underbrace{1\ldots1}_{\ell}0\underbrace{1\ldots10}_{r}0.$$

Now, consider the case where the `Move-in()` operation is applied on either the element max or $max-1$ of π_2^*. Both these cases are reduced to Case 2.1(i), where the graph $F[\phi^*]$ is produced after $4+\ell$ edge modifications as well.

(iii) `Move-out()`. Recall that the sequences of the SiP π^* are:

$$\begin{aligned}
\pi_1^* &= (\underbrace{n+1, n+2, \ldots, n+\ell+1}_{\ell+1}) \\
\pi_2^* &= (\underbrace{n+\ell+3, \ldots, n+\ell+r+2, max, max-1}_{r+2}) \\
\pi_3^* &= (\underbrace{1, 2, \ldots, \ell+1, \alpha}_{\ell+2}) \quad \pi_4^* = (\underbrace{\ell+2, \ldots, \ell+r+1, n, n-1}_{r+2})
\end{aligned} \tag{4.6}$$

where $max = 2n+1$ and $\alpha = n+\ell+2$.

(iii.a): We consider first the case where i elements are moved from π_1^* to π_3^* and π_2^*, and let ϕ^* be the resulting SiP. As we mentioned, such an operation moves the i largest elements from π_1^* and produces the SiP ϕ^* with the following structure:

$$\begin{aligned}
\phi_1^* &= (\underbrace{n+1, \ldots, n+\ell+1-i}_{\ell+1-i}) \\
\phi_2^* &= (\underbrace{n+\ell+3-i, \ldots, n+\ell+1, \alpha, n+\ell+3, \ldots, max-1}_{r+2+i}) \\
\phi_3^* &= (\underbrace{1, 2, \ldots, \ell+1-i, \alpha'}_{\ell+2-i}) \quad \phi_4^* = (\underbrace{\ell+2-i, \ldots, \ell+1, \ell+2, \ldots, n-1}_{r+2+i})
\end{aligned} \tag{4.7}$$

where $\alpha = n+\ell+2$ and $\alpha' = n+\ell+2-i$.

The resulting true-incorrect graph $F[\phi^*]$ contains $3+i+r$ ($1 \leq i \leq \ell$) nodes whose back-edge needs to be updated: these nodes correspond to the element $\alpha = n+\ell+2$ of the sequence ϕ_2^*, the element $\alpha' = n+\ell+2-i$ of the sequence ϕ_3^*, and the i elements $\ell+2-i, \ldots, \ell+1$ plus the $r+1$ nodes $\ell+2, \ldots, \ell+r+1, n$ of the sequence ϕ_4^*, due to element α'. It follows that the graph $F[\phi^*]$ is produced after at least $4+r$ edge modifications. In fact, there is a true-incorrect graph $F[\phi^*]$ that results from $F[\pi^*]$ after exactly $4+r$ edge modifications: it suffices to apply

a `Move-out()` operation on the element $n+\ell+1$ of π_1^*. This graph $F[\phi^*]$ encodes the watermark $w' \neq w$ where

$$w' = 1\underbrace{1\ldots1}_{\ell-i}0\underbrace{1\ldots1}_{r+i}0$$

(iii.b): Consider now the case where j elements are moved from π_2^* to π_3^* and π_1^*, where $1 \leq j \leq r$ (the cases where $j = r+1$ and $j = r+2$ will be considered separately); recall that these are the j smallest elements $n+\ell+3, n+\ell+4, \ldots, n+\ell+2+j$ of π_2^* and the resulting SiP ϕ^* has the following structure:

$$\begin{aligned}
\phi_1^* &= (\underbrace{n+1,\ldots,n+\ell+1,\alpha,n+\ell+3,\ldots,n+\ell+1+j}_{\ell+1+j}) \\
\phi_2^* &= (\underbrace{n+\ell+3+j,\ldots,max-1}_{r+2-j}) \\
\phi_3^* &= (\underbrace{1,2,\ldots,\ell+1+j,\alpha'}_{\ell+2+j}) \quad \phi_4^* = (\underbrace{\ell+2+j,\ldots,n,n-1}_{r+2-j})
\end{aligned} \tag{4.8}$$

where $\alpha = n+\ell+2$ and $\alpha' = n+\ell+2+j$.

The resulting true-incorrect graph $F[\phi^*]$ contains the following nodes whose back-edge needs to be updated: the node $\alpha = n+\ell+2$ of the sequence ϕ_1^*, the node $\alpha' = n+\ell+2+j$ of the sequence ϕ_3^*, and the j nodes $\ell+2, \ell+3, \ldots, \ell+1+j$ plus the $r+1-j$ nodes $\ell+2+j, \ldots, n$ of the sequence ϕ_4^* due to element α'. Thus, $F[\phi^*]$ results from $F[\pi^*]$ after $3+r$ edge modifications and encodes the watermark $w' \neq w$ where

$$w' = 1\underbrace{1\ldots1}_{\ell+j}0\underbrace{1\ldots1}_{r-j}0$$

for $j = 1, 2, \ldots, r$.

In the case where the $j = r+1$ smallest elements of π_2^* are moved to π_3^* and π_1^*, the resulting SiP ϕ^* has the structure

$$\begin{aligned}
&\phi_1^* = (n+1, n+2, \ldots, n+\ell+1, \alpha, \ldots, max-2 = 2n-1) \quad \phi_2^* = (max) \\
&\phi_3^* = (1, 2, \ldots, \ell+1, \ell+2, \ldots, n-1, \alpha') \quad \phi_4^* = (n)
\end{aligned} \tag{4.9}$$

where $\alpha' = max - 1$, and thus the corresponding true-incorrect graph $F[\phi^*]$ contains the following nodes whose back-edge needs to be updated: the node α in ϕ_1^*, the $n-1$ nodes $1, 2, \ldots, n-1$) in ϕ_3^*, and the node n in ϕ_4^*. Thus, $F[\phi^*]$

results from $F[\pi^*]$ after $n+1$ edge modifications where $n \geq 4$, and encodes the watermark $w' \neq w$ where

$$w' = 1\underbrace{11\ldots1}_{n-2}0.$$

In the case where the $j = r+2$ smallest elements of π_2^* are moved to π_3^* and π_1^* (i.e., all the elements of π_2^* are moved), the sequences ϕ_2^* and ϕ_4^* of the resulting SiP ϕ^* are empty. Thus, ϕ^* actually consists of two increasing sequences $\phi^* = \phi_1^*||\phi_3^*$ and has the following structure:

$$\begin{aligned} &\phi_1^* = (n+1, n+2, \ldots, n+\ell+1, \alpha, \ldots, max-1 = 2n) \quad \phi_2^* = () \\ &\phi_3^* = (1, 2, \ldots, \ell+1, \ell+2, \ldots, n, \alpha') \quad \phi_4^* = () \end{aligned} \tag{4.10}$$

where $\alpha' = max = 2n+1$. The resulting graph $F[\phi^*]$ contains the following nodes whose back-edge needs to be updated: the nodes $\alpha = n+\ell+2$ and $max-1 = 2n$ in ϕ_1^* and the $r+2 = |\phi_2^*|$ nodes $\ell+2, \ell+3, \ldots, n$ due to element $max-1$. in ϕ_2^*, i.e., the Thus, $F[\phi^*]$ results from $F[\pi^*]$ after $4+r$ edge modifications and encodes the watermark $w' \neq w$ where

$$w' = 1\underbrace{11\ldots1}_{n-2}1.$$

Summing up the case where $b_n = 0$ and $r > 0$, we conclude that a true-incorrect reducible permutation graph $F[\phi^*]$ encoding a watermark number $w' \neq w$ can result from $F[\pi^*]$ after performing either $4+\ell$ or $3+r$ edge modifications on its back-edges. Thus, $\texttt{minVM}(w) = 4 + min\{\ell, r-1\}$.

Subcase 2.2: $\mathbf{b_n = 0}$ and $\mathbf{r = 0}$. In this case, the watermark number w is encoded by the SiP $\pi^* = \pi_1^*||\pi_2^*||\pi_3^*||\pi_4^*$ having the following structure:

$$\begin{aligned} &\pi_1^* = (\underbrace{n+1, n+2, \ldots, n+\ell+1}_{\ell+1}) \quad \pi_2^* = (\underbrace{max, max-1}_{2}) \\ &\pi_3^* = (\underbrace{1, 2, \ldots, \ell+1, \alpha = n+\ell+2}_{\ell+2}) \quad \pi_4^* = (\underbrace{n, n-1}_{2}). \end{aligned} \tag{4.11}$$

where $max = 2n+1$, $max-1 = 2n$, $\alpha = n+\ell+2$ and $\ell = n-3$. The watermark number w is

$$w = 1\underbrace{111\ldots1}_{\ell = n-3}00.$$

Let ϕ^* be the SiP resulting from π^* after performing some valid modifications on its elements. We consider:

(i) `Swap()`. The `Swap()` operation can only be applied on the pair $(max, max-1)$ of the bitonic sequence π_2^* and the structure of the resulting SiP ϕ^* is

$$\begin{aligned}\phi^* = (n+1, n+2, \ldots, n+\ell+1) \;||\; (max-1, max) \;||\\ (1, 2, \ldots, \ell+1, \alpha) \;||\; (n-1, n).\end{aligned} \tag{4.12}$$

The true-incorrect graph $F[\phi^*]$ contains the following nodes whose back-edge needs to be updated: the node $max-1$ in ϕ_2^*, the node $n-1$ in ϕ_4^*, and the $\ell+2$ nodes $1, 2, \ldots, \ell+1, \alpha$ in ϕ_3^*. Thus, $F[\phi^*]$ results from $F[\pi^*]$ after $4+\ell = n+1$ edge modifications where $n \geq 4$.

(ii) `Move-in()`. It is easy to see that the SiP ϕ^* which results from π^* by applying one `Move-in()` operation on sequence π_2^* has the same structure with that resulting in the previous Case 2.2(i). Thus, the graph $F[\phi^*]$ results from $F[\pi^*]$ after $4+\ell$ edge modifications.

(iii) `Move-out()`. We consider the two variants of the `Move-out()` operation:

(iii.a): We consider first the case where i elements are moved from π_1^* to π_3^* and π_2^* where $1 \leq i \leq \ell$, and let ϕ^* be the resulting SiP. This case results from the Case 2.1(iii) by setting $r = 0$. It follows that there exists a true-incorrect graph $F[\phi^*]$ resulting from $F[\pi^*]$ after exactly 4 edge modifications; the watermark number $w' \neq w$ encoded by $F[\phi^*]$ is

$$w' = 1\underbrace{1\ldots1}_{\ell-1}010.$$

(iii.b): The cases where the $max-1$ element or both the $max-1$ and max elements are moved from π_2^* result to a true-incorrect graph $F[\phi^*]$ after more than 4 edge modifications.

Summarizing, for the case where $b_n = 0$ and $r = 0$, we conclude that a true-incorrect reducible permutation graph $F[\phi^*]$ encoding a watermark number $w' \neq w$ can result from $F[\pi^*]$ after performing at least 4 edge modifications on its back-edges. Thus, $\texttt{minVM}(w) = 4$.

Subcase 2.3: $\mathbf{b_n = 1}$ and $\mathbf{r \geq 0}$. In this case, the watermark number w is encoded by

a SiP $\pi^* = \pi_1^* \mid\mid \pi_2^* \mid\mid \pi_3^* \mid\mid \pi_4^*$ where

$$\pi_1^* = (\underbrace{n+1, n+2, \ldots, n+\ell+1}_{\ell+1}) \quad \pi_2^* = (\underbrace{n+\ell+3, \ldots, n+\ell+r+2, max-1, max}_{r+2})$$
$$\pi_3^* = (\underbrace{1, 2, \ldots, \ell+1, \alpha}_{\ell+2}) \quad \pi_4^* = (\underbrace{\ell+2, \ldots, \ell+r+1, n-1, n}_{r+2}).$$

(4.13)

where $max = 2n+1$ and $\alpha = n+\ell+2$. The max element is located at the last position of the sequence π_2^* and, thus, π_2^* is an increasing sequence of length $r+2$. In the case under consideration, the watermark number w encoded by π^* is

$$w = 1\underbrace{1\ldots1}_{\ell}0\underbrace{1\ldots1}_{r}1.$$

Let $F[\phi^*]$ be the true-incorrect reducible permutation graph resulting from $F[\pi^*]$ after performing some valid modifications on the back-edges and let ϕ^* be the corresponding SiP encoding a watermark number $w' \neq w$.

(i) `Swap()`. The pair $(max-1, max)$ is the only pair of elements of π_2^* on which we can apply a `Swap()` operation after which the structure of the resulting SiP ϕ^* is

$$\phi^* = (n+1, n+2, \ldots, n+\ell+1) \mid\mid (n+\ell+3, \ldots, n+\ell+r+2, max, max-1) \mid\mid (1, 2, \ldots, \ell+1, \alpha) \mid\mid (\ell+2, \ldots, \ell+r+1, n, n-1).$$

(4.14)

The graph $F[\phi^*]$ results from $F[\pi^*]$ after having appropriately updated the back-edges of the nodes $max-1,\ 1,\ 2, \ldots,\ \ell+1,\ \alpha,\ n-1$ and the encoded watermark number $w' \neq w$ is

$$w' = 1\underbrace{1\ldots1}_{\ell}0\underbrace{1\ldots1}_{r}0.$$

Thus, in this case, the number of valid edge modifications of the graph $F[\pi^*]$ is $4+\ell$.

(ii) `Move-in()`. Let $p_1, p_2, \ldots, p_i$ be i elements of π_2^* (see Equation 4.13) such that $p_1 < p_2 < \cdots < p_i$ and $p_1 = n+\ell+m$ where $3 \leq m \leq r+3$. We perform `Move-in()` operations on the elements $p_1, p_2, \ldots, p_i$ and let ϕ_2^* be the resulting bitonic sequence of the resulting SiP ϕ^*. Then, ϕ_2^* is

$$\phi_2^* = (\mathsf{P}, max, p_i, p_{i-1}, \ldots, p_1)$$

where P is an increasing sequence of length $r+1-i$ consisting of the remaining elements of π_2^* lying to the left of max after having applied the operations. Let $q_1, q_2, \ldots, q_j$ be the elements $\pi_{p_1}^{*-1}, \pi_{p_1}^{*-1}+1, \ldots, \pi_{max-1}^{*-1}$. Then, the sequence ϕ_4^* has the form:

$$\phi_4^* = (\mathtt{Q}, n, q_1, q_2, \ldots, q_j)$$

where Q is an increasing sequence of length $m-3$ consisting of the indices of the elements $n+\ell+3, n+\ell+4, \ldots, n+\ell+m-1$. Moreover, $\pi_1^* = \phi_1^* = (n+1, n+2, \ldots, n+\ell+1)$ and $\pi_3^* = \phi_3^* = (1, 2, \ldots, \ell+1, \alpha)$.

Thus, the corresponding true-incorrect graph $F[\phi^*]$ results from $F[\pi^*]$ after having updated the back-edge of the elements $p_i, p_{i-1}, \ldots, p_1$ of the sequence ϕ_2^*, the elements $1, 2, \ldots, \ell+1, \alpha$ of the sequence ϕ_3^*, and the elements $q_1, q_2, \ldots, q_j$ of the sequence ϕ_4^*. In total, the graph $F[\phi^*]$ results after $i+\ell+2+(\pi_{max}^{*-1} - \pi_{p_1}^{*-1}) = i+\ell+2+(n-m)$ edge modifications. By setting $i=1$ and $(n-m)=1$, we conclude that we can obtain a true-incorrect graph after having performed $4+\ell$ edge modifications on graph $F[\pi^*]$.

(iii) Move-out(). We perform a Move-out() operation either by moving i elements from π_1^* to π_3^* and π_2^* ($1 \leq i \leq \ell$) or by moving j elements from π_2^* to π_3^* and π_1^* ($1 \leq j \leq r+2$), where

(iii.a): We consider first the case where the i largest elements of π_1^* are moved to π_3^* and π_2^*, which produces the SiP ϕ^* with

$$\begin{aligned}
\phi_1^* &= (\underbrace{n+1, \ldots, n+\ell+1-i}_{\ell+1-i}) \\
\phi_2^* &= (\underbrace{n+\ell+3-i, \ldots, n+\ell+1, \alpha, n+\ell+3, \ldots, max}_{r+2+i}) \\
\phi_3^* &= (\underbrace{1, 2, \ldots, \ell+1-i, \alpha'}_{\ell+2-i}) \quad \phi_4^* = (\underbrace{\ell+2-i, \ldots, \ell+1, \ell+2, \ldots, n}_{r+2+i})
\end{aligned} \tag{4.15}$$

where $\alpha' = n+\ell+2-i$. The resulting true-incorrect graph $F[\phi^*]$ results from $F[\pi^*]$ after having updated the back-edge of the nodes corresponding to the element $\alpha = n+\ell+2$ of the sequence ϕ_2^*, the element α' of the sequence ϕ_3^*, and the i elements $\ell+2-i, \ldots, \ell+1$ plus the $r+2$ nodes $\ell+2, \ldots, \ell+r+1, n-1, n$ of the sequence ϕ_4^*. Thus, the graph $F[\phi^*]$ results after $4+i+r$ edge modifications, where $1 \leq i \leq \ell$. It follows that there exists a true-incorrect graph $F[\phi^*]$ resulting after $5+r$ edge modifications.

(iii.b): We follow a similar approach as in Case 2.1(iii.b). Let j elements are moved from π_2^* to π_3^* and π_1^*, where $1 \leq j \leq r$ (the cases where $j = r+1$ and $j = r+2$ will be considered separately). The moved elements are the j smallest elements $n+\ell+3, n+\ell+4, \ldots, n+\ell+2+j$ of π_2^* and the resulting SiP ϕ^* is

$$\begin{aligned}
\phi_1^* &= (\underbrace{n+1, \ldots, n+\ell+1, \alpha, n+\ell+3, \ldots, n+\ell+1+j}_{\ell+1+j}) \\
\phi_2^* &= (\underbrace{n+\ell+3+j, \ldots, max}_{r+2-j}) \\
\phi_3^* &= (\underbrace{1, 2, \ldots, \ell+1+j, \alpha'}_{\ell+2+j}) \quad \phi_4^* = (\underbrace{\ell+2+j, \ldots, n-1, n}_{r+2-j})
\end{aligned} \tag{4.16}$$

where $\alpha' = n+\ell+2+j$.

The true-incorrect graph $F[\phi^*]$ results from $F[\pi^*]$ after back-edge changes of the nodes corresponding to the element $\alpha = n+\ell+2$ of the sequence ϕ_1^*, the element $\alpha' = n+\ell+2+j$ of the sequence ϕ_3^*, and the j elements $\ell+2, \ell+3, \ldots, \ell+1+j$ plus the $r+2-j$ elements $\ell+2+j, \ldots, n-1, n$ of the sequence ϕ_4^* due to element α'. Thus, $F[\phi^*]$ requires $4+r$ edge modifications and encodes the watermark number $w' \neq w$ of the form:

$$w' = 1\underbrace{1\ldots1}_{\ell+j}0\underbrace{1\ldots1}_{r-j}1$$

where, $j = 1, 2, \ldots, r$.

The cases where the $j = r+1$ and $j = r+2$ are exactly the same as the corresponding cases in Case 2.1(iii.b).

Concluding, in the case that $b_n = 1$ and $r \geq 0$, it holds that a true-incorrect reducible permutation graph $F[\phi^*]$ encoding a watermark number $w' \neq w$ can result from $F[\pi^*]$ after performing either $4+\ell$ or $4+r$ edge-modifications on its back-edges. Thus, $\texttt{minVM}(w) = 4 + min\{\ell, r\}$.

Case 3. In this case, the internal block B of the watermark number w contains no 0s and, thus, the binary representation of the number w has one of the following two forms:

$$1\underbrace{1\ldots11}_{n-2}0 \qquad \text{or} \qquad 1\underbrace{1\ldots11}_{n-2}1.$$

(3.1) In the former case where $b_n = 0$, the watermark number w is encoded by a SiP $\pi^* = \pi_1^* \mid\mid \pi_2^* \mid\mid \pi_3^* \mid\mid \pi_4^*$ having the following structure:

$$\begin{aligned}\pi^* = (n+1, n+2, \ldots, 2n-i-1, 2n-i, \ldots, 2n-1) \mid\mid (max) \mid\mid \\ (1, 2, \ldots, n-i-1, n-i, \ldots, n-1, \alpha) \mid\mid (n)\end{aligned} \tag{4.17}$$

where $max = 2n+1$ and $\alpha = 2n$.

From the structures of π_1^* and π_2^*, it follows that the only operation we can apply is the `Move-out()` on the elements of both these sequences.

Let us first consider the case where the i largest elements $2n-i, 2n-i+1, \ldots, 2n-1$ from π_1^* are moved to π_3^* and π_2^*. Then, the structure of the resulting SiP ϕ^* is

$$\begin{aligned}\phi^* = (n+1, n+2, \ldots, 2n-i-1) \mid\mid (2n-i+1, \ldots, 2n-1, \alpha, max) \mid\mid \\ (1, 2, \ldots, n-i-1, \alpha') \mid\mid (n-i, \ldots, n-1, n),\end{aligned} \tag{4.18}$$

where $\alpha' = 2n-i$. The true-incorrect graph $F[\phi^*]$ results from $F[\pi^*]$ after modifying the back-edge of the nodes corresponding to the element $\alpha = 2n$ in sequence ϕ_2^*, the element $\alpha' = 2n-i$ in sequence ϕ_3^*, and the $i+1$ elements $n-i, \ldots, n-1, n$ in sequence ϕ_4^*. Thus, in total $2+i+1$ edge modifications are needed. Since $1 \leq i \leq n-1$, it follows that a true-incorrect graph $F[\phi^*]$ can be obtained from $F[\pi^*]$ by modifying at least 4 edges. In fact, we can do exactly 4 edge modifications and encode the watermark number $w' \neq w$ where

$$w' = 1\underbrace{1\ldots10}_{n-2}1.$$

Let us now consider the case where the element max is moved out of π_2^*. Then, both sequences ϕ_2^* and ϕ_4^* become empty and the sequences ϕ_1^* and ϕ_3^* have the following structure:

$$\begin{aligned}\phi^* = (n+1, n+2, \ldots, 2n-i-1, 2n-i, \ldots, 2n-1, \alpha = 2n) \mid\mid () \mid\mid \\ (1, 2, \ldots, n-i-1, n-i, \ldots, n-1, \alpha' = max) \mid\mid ().\end{aligned} \tag{4.19}$$

The nodes of the graph $F[\pi^*]$ whose back-edge needs to point to another node in order to create the true-incorrect graph $F[\phi^*]$ are those corresponding to the elements $\alpha' = max$, $\alpha = 2n$ and $1, 2, \ldots, n-i-1, n-i, \ldots, n-1$ of ϕ_3^* due to the last element $\alpha = 2n$ of ϕ_1^*. Thus, the graph $F[\phi^*]$ can be obtained from $F[\pi^*]$ by modifying $n+1$ edges, where $n \geq 4$.

(3.2) In the latter case where $b_n = 1$, the watermark number w is encoded by a SiP $\pi^* = \pi_1^* \,||\, \pi_2^* \,||\, \pi_3^* \,||\, \pi_4^*$ having the following structure:

$$\begin{aligned}\pi^* = (n+1, n+2, \ldots, 2n-i, 2n-i+1, \ldots, 2n) \,||\, () \,||\\ (1, 2, \ldots, n-i-1, n-i, \ldots, n-1, n, \alpha) \,||\, (),\end{aligned} \tag{4.20}$$

where $\alpha = max = 2n+1$. The only operation that we can apply on π_1^* is the `Move-out()` operation on π_1^* and suppose that the i largest elements of π_1^*, i.e., $2n-i+1, \ldots, 2n-1, 2n$, are moved to π_2^* and π_3^*. Then, by choosing the last element $2n$ of π_1^* to be the last element of π_2^*, the structure of the resulting SiP ϕ^* becomes the following:

$$\begin{aligned}\phi^* = (n+1, n+2, \ldots, 2n-i) \,||\, (2n-i+2, \ldots, \alpha = max, 2n) \,||\\ (1, 2, \ldots, n-i, \alpha' = 2n-i+1) \,||\, (n-i+1, \ldots, n, n-1),\end{aligned} \tag{4.21}$$

In this case, the true-incorrect graph $F[\phi^*]$ results from $F[\pi^*]$ after changing the back-edge of the nodes corresponding to the elements $2n$ and $\alpha' = 2n-i+1$ of ϕ_2^* and ϕ_3^*, respectively, the elements $n-i+1, \ldots, n$ due to the last element $\alpha' = 2n-i+1$ of ϕ_3^*, and the element $n-1$ due to the element n of ϕ_4^*. Thus, the graph $F[\phi^*]$ can be obtained from $F[\pi^*]$ by modifying $2+i$ edges. Since we require the element $2n$ to be in the last position of π_2^*, we have that $i \geq 2$. Thus, the graph $F[\phi^*]$ can be obtained from $F[\pi^*]$ by modifying 4 edges and the watermark number $w' \neq w$ encoded by $F[\phi^*]$ is

$$w' = 1\underbrace{1 \ldots 10}_{n-2}0.$$

It is easy to see that in the case where the element $2n$ is not located in the last position of π_2^*, the number of edge modifications exceeds 4.

Summarizing, in Case 3, we conclude that a true-incorrect reducible permutation graph $F[\phi^*]$ can result from $F[\pi^*]$ after having performed at least 4 edge modifications on its back-edges. Thus, $\texttt{minVM}(w) = 4$. □

4.4 Strong and Weak Watermark Numbers

In the previous section, for each watermark w, we computed the minimum number $\texttt{minVM}(w)$ of edge modifications of the constructed reducible permutation graph

which are needed so that a watermark different from w may be extracted from the modified reducible permutation graph. Since the greater the value of $\texttt{minVM}(w)$ is, the more difficult it is to alter the watermark embedded, we obviously are interested in watermarks that maximize the value of $\texttt{minVM}(w)$. Therefore, we characterize the watermarks as follows:

Definition 4.4. Let w be an integer watermark in $R_n = [2^{n-1}, 2^n - 1]$. We say that:

- w is *W-RPG-strong* if the value $\texttt{minVM}(w)$ is maximum in the range R_n;
- w is *W-RPG-weak* if the value $\texttt{minVM}(w)$ is minimum in the range R_n;
- w is *W-RPG-intermediate* in the remaining cases.

Then, Theorem 4.1 directly implies that all the watermarks whose internal block B contains at least two 0s are W-RPG-weak watermarks with $\texttt{minVM}(w) = 3$; moreover, for each of these W-RPG-weak watermarks, it holds that after 3 edge modifications, exactly 1 different watermark may be extracted (see proof of Case 1 of Theorem 4.1). For the W-RPG-strong watermarks, we have the following corollary.

Corollary 4.1. *Let w be an integer watermark in $R_n = [2^{n-1}, 2^n - 1]$.*

(i) If n is odd, then there is a unique W-RPG-strong watermark w, which has the form $w = 11^{\ell}01^{\ell}1$ ($\ell \geq 0$) with $\texttt{minVM}(w) = n + \ell = \frac{n-1}{2} + 3$.

(ii) If n is even, then there are 3 W-RPG-strong watermarks w of the forms $w = 11^{\ell}01^{\ell+1}0$, $w = 11^{\ell}01^{\ell+1}1$, $w = 11^{\ell+1}01^{\ell}1$ ($\ell \geq 0$) with $\texttt{minVM}(w) = \frac{n}{2} + 2$.

Proof. Theorem 4.1 implies that the maximum value of $\texttt{minVM}(w)$ of valid edge modifications is $4 + min\{\ell, r-1\}$ if $b_n = 0$ and $r > 0$ or $4 + min\{\ell, r\}$ if $b_n = 1$ and $r \geq 0$; the maximum values are obtained if $|\ell - (r-1)| \leq 1$ in the former case and $|\ell - r| \leq 1$ in the latter case.

(i) If $n = 2\kappa + 1, \kappa \in \mathbb{Z}$ then the $max\{\texttt{minVM}(w)\}$ for $w \in R_n$ is obtained for a unique w of the form $w = 11^{\ell}01^{r}1$, where $\ell = r = \frac{n-3}{2} = k - 1$ and $b_n = 1$; then, $max\{\texttt{minVM}(w)\} = 4 + \ell = \frac{n-1}{2} + 3 = k + 3$.

(ii) If $n = 2\kappa, \kappa \in \mathbb{Z}$ then the binary representation of watermark w is of the form $w = 11^{\ell}01^{r}0$, where $r - \ell = 1$ and $b_n = 0$, with $max\{\texttt{minVM}(w)\} = 4 + \ell$ or $|r - \ell| = 1$

and $b_n = 1$, with $max\{\texttt{minVM}(w)\} = 4 + \ell$ or $max\{\texttt{minVM}(w)\} = 4 + r$. This means that there are 3 integers w in the range R_n with maximum value of $\texttt{minVM}(w)$, as follows.

- $w = 11^{\ell}01^{r}0$, where $r - \ell = 1$ and $b_n = 0$ (top subcase in Case 2 of Theorem 4.1): then, $\texttt{minVM}(w) = 4+\ell$ where $\ell+r = \ell+\ell+1 = n-3 \longrightarrow \ell = \frac{n}{2}-2$, and hence $\texttt{minVM}(w) = 4 + \ell = \frac{n}{2} + 2$.
- if the watermark w is of the form $w = 11^{\ell}01^{r}1$, where $|r - \ell| = 1$ and $b_n = 1$ (bottom subcase in Case 2 of Theorem 4.1): then, $\texttt{minVM}(w) = 4 + \min\{\ell, r\}$ where $\ell+r = 2\min\{\ell, r\}+1 = n-3 \longrightarrow \min\{\ell, r\} = \frac{n}{2}-2$; again, $\texttt{minVM}(w) = 4 + \min\{\ell, r\} = \frac{n}{2} + 2$.

□

We note that if the minimum number of edge modifications is applied to the constructed reducible permutation graph, it may be the case that more than one different watermarks may be produced. Obviously, a good recommendation for a watermark would be a W-RPG-strong watermark such that from the modified reducible permutation graph after exactly $\max\{\texttt{minVM}(w)\}$ edge modifications the minimum number of different watermarks may be extracted. Then, Corollary 4.1 and the proof of Case 2 of Theorem 4.1 imply:

Theorem 4.2. *The W-RPG-strongest watermark $w \in R_n = [2^{n-1}, 2^n - 1]$ is of the form $w = 11^{\ell}01^{\ell+1}1$.*

Proof. Let us consider the W-RPG-strong watermarks exhibited in Corollary 4.1.

(i) If n is odd, then the unique W-RPG-strong watermark is $w = 11^{\ell}01^{\ell}1$, and there are just $\ell + 2$ watermarks different from w that may be extracted from the modified reducible permutation graph after $4+\ell$ edges have been modified. By Subcase 2.3 of Theorem 4.1, there is just 1 other watermark w' if we apply `Swap()`, and another $\ell+1$ watermarks w' if we apply `Move-out()` with $j \in \{1, 2, \ldots, r+1\}$ where $r = \ell$.

(ii) If n is even, then we have the 3 W-RPG-strong watermarks $w = 11^{\ell}01^{\ell+1}0$, $w = 11^{\ell}01^{\ell+1}1$, and $w = 11^{\ell+1}01^{\ell}1$.

- if $w = 11^{\ell}01^{\ell+1}0$, the number of watermarks $w' \neq w$ extracted from the modified reducible permutation graph is $\ell + 3$. By Subcase 2.2 of Theorem 4.1, there is just 1 other watermark w' if we apply `Swap()`, another 1 if we apply `Move-in()` and minimize the value of $6 + \ell + r - i$, and another $\ell + 1$ watermarks w' if we apply `Move-out()` with $j \in [\{1, 2, \ldots, r\}$ where $r = \ell + 1$.
- if $w = 11^{\ell}01^{\ell+1}1$, there is only 1 watermark $w'' \neq w$ extracted if we apply `Swap()` (see Subcase 2.3 of Theorem 4.1).
- if $w = 11^{\ell+1}01^{\ell}1$, the number of different watermarks $w'' \neq w$ extracted is $\ell + 3$. By Subcase 2.3 of Theorem 4.1, there are ℓ different watermarks w' extracted if we apply `Move-out()` with $j \in \{1, 2, \ldots, r + 1\}$ where $r = \ell + 1$.

Thus, the theorem follows. □

Example. According to Corollary 4.1, if $n = 7$ (odd), the unique *W-RPG-strong* watermark in range $R_7 = [2^6, 2^7 - 1] = [64, 127]$ is 1110111 (i.e. watermark number $w = 119$) with $\texttt{minVM}(w) = \frac{7-1}{2} + 3 = 6$. From the same Corollary, if $n = 8$ (even), there are three *W-RPG-strong* watermarks in range $R_8 = [2^7, 2^8 - 1] = [128, 255]$, namely 11101110, 11101111 and 11110111 with $\texttt{minVM}(w) = \frac{8}{2} + 3 = 9$ and from Theorem 4.2, the *W-RPG-strongest* watermark in range R_8 is $w = 239$ (its binary representation $b(w) = 11101111$). Finally, one of *W-RPG-weak* watermarks in the same range R_8 is the number 227 because its binary representation (11100011) contains three 0s and the minimum number of valid edge-modification is 3 and one of *W-RPG-intermediate* watermarks in the same range R_8 is the number 253 with binary representation 11111101 and $\texttt{minVM}(w) = 4$.

4.5 Concluding Remarks

Following up on the software watermarking method presented in the W-RPG codec system by Chroni *et al*. [82, 37, 66], in this section, we theoretically studied the resilience of the created reducible permutation graphs under edge-modification attacks. For a specific range $R_n = [2^{n-1}, 2^n - 1]$, we classify each watermark number $w \in R_n$ into one of three categories. Firstly, a watermark can be characterized as W-RPG-strong watermark, in the case where it has $max\{\texttt{minVM}(w)\}$, secondly it can be clas-

sified as W-RPG-weak watermark, in the case where $\texttt{minVM}(w) = 3$ and and lastly if $3 < \texttt{minVM}(w) < max\{\texttt{minVM}(w)\}$, it is a W-RPG-intermediate watermark. By also minimizing the number of watermarks different from w extracted from the modified reducible permutation graph after $\texttt{minVM}(w)$ edge modifications, we conclude that the best choice of watermark in R_n is a W-RPG-strong watermark of the form $11^{\ell}01^{\ell+1}1$ ($\ell \geq 0$) where n is even and $\ell = \frac{n}{2} + 2$.

Chapter 5

Conclusions and Future Work

5.1 Edge Modification Problems

In this chapter, we conclude the results of edge modification problems such as adding a tail and adding an edge in some classes of perfect graphs and its application to watermarking.

5.1.1 Adding a Tail

According to the graph modification problems, we focused on the minimum $\mathcal{C}$-completion problem on a graph G and edge modification problem particularly. We assume a graph class $\mathcal{C}$ to which a graph G belongs on it. In Chapter 2, we studied the problem of connecting a node $w \notin V(G)$ to G by a single edge uw where $u \in V(G)$. The study of these problems is crucial for real world problems because a lot of times we have to deal with the addition of new data as a node and how the initial structure of graph will not be affected or which is the appropriate modification/changes in order to maintain its properties and structure.

We call as *tail*, the addition of a node $w \notin V(G)$ in a node $u \in V(G)$ through a non edge uw. Thus, our goal was to determine the minimum number of fill edges (including the tail) that must be added in order to the resulting graph to belong to

the class $\mathcal{C}$ since the graph G', which results from G after the addition of the tail, is not necessarily a graph of class $\mathcal{C}$. We focused on the classes of perfect graphs, which include many important families of graphs, i.e a graph in which the chromatic number of every induced subgraph equals the order of the largest clique of that subgraph, namely clique number.

The first class of perfect graphs where we studied this completion problem, is the split graphs. Split graph is a graph in which the vertices can be partitioned into a clique and an independent set and it may have more than one partition into a clique and an independent set. Let us consider a split graph G, and we add a tail uw, where $u \in V(G)$ and $w \notin V(G)$. Depending on which vertex set (independent set S or clique set K) the node u belongs to, it is configured with the minimum number of fill edges. If $u \in K$, the minimum number of fill edges needed (in addition to the tail uw) is 0 and if $u \in S$ and $degree(u) = k_u$, the minimum number of fill edges needed (in addition to the tail uw) is $|K| - k_u$.

Secondly, we studied the same problem (adding a tail in a graph G) in classes of threshold and quasi-threshold graphs. According to the literature, the threshold graphs are precisely those A-free graphs (i.e it contains no induced subgraph isomorphic to P_4 or C_4) containing no induced subgraph isomorphic to $2K_2$ and quasi-threshold contains no induced subgraph isomorphic to P_4 or C_4 (i.e., they are A-free graphs). The results of threshold graphs are that the minimum number of edges (in addition to the tail uw) that need to be added is $\min_{0 \leq l \leq i} A(l)$ where $A(l) = \sum_{j=0}^{l-1} |V_{j,1}| + \sum_{w=l+1}^{i} \sum_{j=2}^{k_w} |V_{w,j}|$. if $u \in V_{i,1} \subseteq K$, or $A(l) = \sum_{j=0}^{l-1} |V_{j,1}| + \sum_{w=l}^{h} \sum_{j=1}^{k_w} |V_{w,j}| + \sum_{j=i}^{l-1} |V_{j,1}|$ if $u \in V_{i,j} \subseteq S$, where $2 \leq j \leq k_i$.

Furthermore, we investigated the properties and the structure of P_4-sparse graph, as well as its tree representation. P_4-sparse graph G is a graph where no set of five vertices in G induces more than one chordless path of length three. Let G be a P_4-sparse graph and T_G be its P_4-sparse tree. Considering the addition of a tail uw incident on a node u of G, we proved that there exists a minimum P_4-sparse completion of the graph $G + uw$ such that for the P_4-sparse tree $T_{G'}$ of the resulting graph G'. As a special case, we consider that the given graph is a spider H (thin or thick) and we add a tail uw to a vertex u of H such that the parent-node of u in the P_4-sparse tree T_H of H is the 2-node corresponding to H. Finally, for this purpose, we described the **Algorithm P_4-sparse-Tail-Addition** in Chapter 2, which computes the minimum number of edges to be added to $G + uw$ so that the resulting graph is P_4-sparse in

$O(n)$ time.

An interesting open question is the minimum $\mathcal{C}$-completion of other classes of perfect graphs. The problem of adding a tail in graph G, where G belongs to class of interval or permutation graph, would be a first step for solution of general problem of adding an edge in these classes of graphs; we leave it as an open problem for future investigation.

5.1.2 Adding an Edge

In the scope of this thesis, we studied the problem of adding an edge in a graph G, which belongs to class of P_4-sparse graphs, namely given a P_4-sparse graph G and a non-edge xy of G, find the minimum number of non-edges of G that need to be added to G so that the resulting graph is a P_4-sparse graph and contains xy as an edge.

Firstly, we investigated the special case in which the given P_4-sparse graph G consists of 2 connected components each containing one of the endpoints of the added non-edge uv, namely (P_4-sparse-2CC,$+1$)-MinEdgeAddition, i.e., the least common ancestor of the leaves corresponding to u, v in P_4-sparse tree T is a 0-node. For any optimal solution H, we distinguished the cases for what root node (1-node, 2-node thin, or 2-node thick) of P_4-sparse tree is. In the following, there are presented the approach of solution of adding a non-edge incident on a vertex of the clique or the independent set of a spider graph (thin or thick spider).

According to the lemmas and theorems we previously described and the combination of cases (included the case of addition a tail to P_4-sparse graph, which was described in the previous chapter), we presented the general algorithm for computation of (P_4-sparse graph,$+1$)-MinEdgeAddition. In addition, we proved the efficiency and correctness of general algorithm, showing its complexity based on the lemmas and theorems that have been given in Chapter 3.

As a prime further research target there has left the study of (P_4-sparse graph, -1)-MinEdgeAddition problem, namely given a graph G and deletion of an edge $uv \in E(G)$, find the minimum number of non-edges (without uv) to be added to $G - uv$ so that the resulting graph belongs to $P4$-sparse graphs. This problem is included in general problem of $\mathcal{C}$-completion problem. Moreover, on the same goal is embedded the investigation of ($\mathcal{C}, \pm k$)-MinEdgeAdditionproblem to other classes

of perfect graphs. Additionally, on the same concept several algorithms could also be deployed as a result of the properties of perfect graphs and their specific structure.

5.2 Applications

In the last decade, a wide range of software watermarking techniques has been proposed among which the graph-based methods that encode watermark numbers as graphs whose structure resembles that of real program graphs. Recently, Chroni *et al*. [82, 37, 66] proposed a codec system algorithm for multiple encoding a watermark into a graph structure: an integer (i.e., a watermark) is encoded first into a self-inverting permutation π^* and then into a reducible permutation graph. In this watermark coded system, which is based on graphs and structure properties, it is considered necessary the study of the resilience of these graphs against edge modification attacks.

The aim of this thesis is a theoretical study about the resilience of the created reducible permutation graphs under edge-modification attacks. These reducible permutation graphs are extracted by W-RPG codec system presented by Chroni *et al*. [82, 37, 66], which is a software watermarking method based on the structural properties of the self-inverting permutation π^* encoding the watermark in order to prove its resilience to edge-modification attacks on the flow-graph $F[\pi^*]$. In this thesis and specifically Chapter 4, we arranged the watermark numbers in classes according to shared qualities or characteristics.

Thus, the three categories where we classify each watermark number are W-RPG-strong watermark, W-RPG-weak watermark and W-RPG-intermediate watermark. In the first category, W-RPG-strong watermark, is when the watermark number w has $max\{\texttt{minVM}(w)\}$, where $minVM(w)$ is the minimum number of valid edge-modification of the graph $F[\pi^*]$, i.e. the minimum SiP operations in π^* in order to create other self-inverting permutations (SiPs) π_i^*. In the second category, W-RPG-weak watermark, then the watermark number w has $\texttt{minVM}(w) = 3$ which is the less number of valid edge-modification that can be made. In the last category, W-RPG-intermediate watermark is the watermark number which has $3 < \texttt{minVM}(w) < max\{\texttt{minVM}(w)\}$, where the most watermark numbers belong. All above classification is done on a specific range $R_n = [2^{n-1}, 2^n - 1]$, with $n > 2$. The next question, which we solved in

the thesis, is the best choice of a watermark in R_n and if it exists or if it is unique. We conclude that the best choice of watermark in R_n is a W-RPG-strong watermark of the form $11^{\ell}01^{\ell+1}1$ ($\ell \geq 0$) where n is even and $\ell = \frac{n}{2} + 2$, by minimizing the number of watermarks different from w extracted from the modified reducible permutation graph after $\texttt{minVM}(w)$ edge modifications.

It would be very interesting to come up with new efficient codec algorithms and structures having "better" properties with respect to resilience to attacks. Another interesting question with practical value is whether the class of reducible permutation graphs can be extended so that it includes other classes of graphs with structural properties capable to efficiently encode watermark numbers. Furthermore, the evaluation of the W-RPG system codec algorithms and structures under other watermarking quality measurements is another interesting topic of study which will provide detailed information about their practical behavior.

Finally, in light of graph modification problems, it would be very interesting to investigate theoretically the resilience of others watermarking codec systems that are based on graphs. Through theory of completion, deletion, and editing problems, it would be effective to find the possibility of altering other components of the watermark structure and how we can handle the possible additive, subtractive, and distorting attacks on watermark structure, i.e., edge-modification, edge-insertion or deletion and node-insertion or deletion attacks on watermark graph in order to the embedded watermark represents the real owner; we leave it as a direction for future work.

Bibliography

[1] B. Jamison and S. Olariu, "A tree representation for p_4-sparse graphs," *Discrete Applied Mathematics*, vol. 35, pp. 115—129, 1992.

[2] J. M. Lewis and M. Yannakakis, "The node-deletion problem for hereditary properties is np-complete," *Journal of Computer and System Sciences*, vol. 20, no. 2, pp. 219–230, 1980.

[3] C. Lund and M. Yannakakis, "The approximation of maximum subgraph problems," in *International Colloquium on Automata, Languages, and Programming*. Springer, 1993, pp. 40–51.

[4] T. Fujito, "Approximating node-deletion problems for matroidal properties," *Journal of Algorithms*, vol. 31, no. 1, pp. 211–227, 1999.

[5] M. Okun and A. Barak, "A new approach for approximating node deletion problems," *Information processing letters*, vol. 88, no. 5, pp. 231–236, 2003.

[6] M. Kumar, S. Mishra, N. S. Devi, and S. Saurabh, "Approximation algorithms for node deletion problems on bipartite graphs with finite forbidden subgraph characterization," *Theoretical Computer Science*, vol. 526, pp. 90–96, 2014.

[7] M. R. Garey and D. S. Johnson, "Computers and intractability," *A guide to the theory of NP-completeness*, 1979.

[8] M. Yannakakis, "Computing the minimum fill-in is np-complete," *SIAM Journal on Algebraic Discrete Methods*, vol. 2, no. 1, pp. 77–79, 1981.

[9] P. Goldberg, M. Golumbic, H. Kaplan, and R. Shamir, "Four strikes against physical mapping of DNA," *J. Comput. Bio.*, vol. 2(1), pp. 139–152, 1985.

[10] F. Margot, "Some complexity results about threshold graphs (1994) discrete appl," *Math*, vol. 49, pp. 229–308.

[11] E. El-Mallah and C. Colbourn, "The complexity of some edge deletion problems," *IEEE transactions on circuits and systems*, pp. 354–362, 1988.

[12] M. Yannakakis, "Edge-deletion problems," *SIAM Journal on Computing*, vol. 10, no. 2, pp. 297–309, 1981.

[13] A. Natanzon, R. Shamir, and R. Sharan, "Complexity classification of some edge modification problems," *Discrete Applied Mathematics*, vol. 113, no. 1, pp. 109–128, 2001.

[14] K. Cirino, S. Muthukrishnan, N. Narayanaswamy, and H. Ramesh, "Graph editing to bipartite interval graphs: exact and asymptotic bounds," in *International Conference on Foundations of Software Technology and Theoretical Computer Science*. Springer, 1997, pp. 37–53.

[15] P. L. Hammer and B. Simeone, "The splittance of a graph," *Combinatorica*, vol. 1, no. 3, pp. 275–284, 1981.

[16] R. Shamir, R. Sharan, and D. Tsur, "Cluster graph modification problems," *Discrete Applied Mathematics*, vol. 144, no. 1-2, pp. 173–182, 2004.

[17] L. Cai, "Fixed-parameter tractability of graph modification problems for hereditary properties," *Information Processing Letters*, vol. 58, no. 4, pp. 171–176, 1996.

[18] J. Casas-Roma, J. Herrera-Joancomartí, and V. Torra, "A survey of graph-modification techniques for privacy-preserving on networks," *Artificial Intelligence Review*, vol. 47, no. 3, pp. 341–366, 2017.

[19] A. Beygelzimer, G. Grinstein, R. Linsker, and I. Rish, "Improving network robustness by edge modification," *Physica A: Statistical Mechanics and its Applications*, vol. 357, no. 3-4, pp. 593–612, 2005.

[20] F. V. Fomin, S. Saurabh, and N. Misra, "Graph modification problems: A modern perspective," in *International Workshop on Frontiers in Algorithmics*. Springer, 2015, pp. 3–6.

[21] S. Bruckner, F. Hüffner, and C. Komusiewicz, "A graph modification approach for finding core–periphery structures in protein interaction networks," *Algorithms for Molecular Biology*, vol. 10, no. 1, pp. 1–13, 2015.

[22] C. Komusiewicz, A. Nichterlein, and R. Niedermeier, "Parameterized algorithmics for graph modification problems: on interactions with heuristics," in *International Workshop on Graph-Theoretic Concepts in Computer Science*. Springer, 2015, pp. 3–15.

[23] S. Rajabzadeh, P. Shahsafi, and M. Khoramnejadi, "A graph modification approach for k-anonymity in social networks using the genetic algorithm," *Social Network Analysis and Mining*, vol. 10, no. 1, pp. 1–17, 2020.

[24] M. Kiabod, M. N. Dehkordi, and B. Barekatain, "A fast graph modification method for social network anonymization," *Expert Systems with Applications*, vol. 180, p. 115148, 2021.

[25] C. Hoáng, "Perfect graphs," PhD Thesis, McGill University, Montreal, Canada, 1985.

[26] M. C. Golumbic, *Algorithmic graph theory and perfect graphs*. Elsevier, 2004.

[27] ——, *Algorithmic Graph Theory and Perfect Graphs*, 2nd ed., ser. Annals of Discrete Mathematics. Elsevier, 2004, vol. 57.

[28] R. Merris, "Split graphs," *European Journal of Combinatorics*, vol. 24, no. 4, pp. 413–430, 2003.

[29] V. Chvátal and P. Hammer, "Set-packing and threshold graphs," *Research Report CORR 73-21*, 1973.

[30] P. B. Henderson and Y. Zalcstein, "A graph-theoretic characterization of the pv_chunk class of synchronizing primitives," *SIAM Journal on Computing*, vol. 6, no. 1, pp. 88–108, 1977.

[31] S. Ma, W. Wallis, and J. Wu, "Optimization problems on quasi-threshold graphs," *J. Comb. Inf. Syst. Sci*, vol. 14, pp. 105–110, 1989.

[32] B. Jamison and S. Olariu, "Linear time optimization algorithms for p4-sparse graphs," *Discrete Applied Mathematics*, vol. 61, no. 2, pp. 155–175, 1995.

[33] D. J. Rose, "A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations," in *Graph theory and computing*. Elsevier, 1972, pp. 183–217.

[34] S. D. Nikolopoulos, L. Palios, and C. Papadopoulos, "A fully dynamic algorithm for the recognition of p4-sparse graphs," *Theoretical Computer Science*, vol. 439, pp. 41–57, 2012.

[35] B. Jamison and S. Olariu, "Recognizing p_4-sparse graphs in linear time," *SIAM J. Comput.*, vol. 21, pp. 381—406, 1992.

[36] S. D. Nikolopoulos and L. Palios, "Adding an edge in a cograph," in *International Workshop on Graph-Theoretic Concepts in Computer Science*. Springer, 2005, pp. 214–226.

[37] M. Chroni and S. D. Nikolopoulos, "An efficient graph codec system for software watermarking," in *2012 IEEE 36th Annual Computer Software and Applications Conference Workshops*. IEEE, 2012, pp. 595–600.

[38] M. Yannakakis, "Node-and edge-deletion np-complete problems," in *Proceedings of the tenth annual ACM symposium on Theory of computing*, 1978, pp. 253–264.

[39] L. Ibarra, "Fully dynamic algorithms for chordal graphs and split graphs." *ACM Trans. Algorithms*, vol. 4, pp. 1–20, 2008.

[40] P. Hell, R. Shamir, and R. Sharan, "A fully dynamic algorithm for recognizing and representing proper interval graphs," *SIAM J. Comput.*, vol. 31, pp. 289–305, 2002.

[41] R. Shamir and R. Sharan, "A fully dynamic algorithm for modular decomposition and recognition of cographs," *Discrete Applied Mathematics*, vol. 136, no. 2-3, pp. 329–340, 2004.

[42] P. Heggernes and F. Mancini, "Dynamically maintaining split graphs," *Discrete Applied Mathematics*, vol. 157, pp. 2047–2069, 2009.

[43] K. Toyonaga, C. R. Johnson, and R. Uhrig, "Multiplicities: Adding a vertex to a graph," in *International Conference on Matrix Analysis and its Applications*. Springer, 2015, pp. 117–126.

[44] J. H. Bevis, K. K. Blount, G. J. Davis, G. S. Domke, and V. A. Miller, "The rank of a graph after vertex addition," *Linear algebra and its applications*, vol. 265, no. 1-3, pp. 55–69, 1997.

[45] I. C. Lopes and J. V. de Carvalho, "An integer programming model for the minimum interval graph completion problem," *Electronic Notes in Discrete Mathematics*, vol. 36, pp. 583–590, 2010.

[46] M. Chimani, C. Gutwenger, P. Mutzel, and C. Wolf, "Inserting a vertex into a planar graph," in *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 2009, pp. 375–383.

[47] U. Brandes, M. Hamann, B. Strasser, and D. Wagner, "Fast quasi-threshold editing," in *Algorithms-ESA 2015*. Springer, 2015, pp. 251–262.

[48] A. Brandstädt, V. B. Le, and J. Spinrad, "Graph classes - a survey," *SIAM Monographs in Discrete Mathematics and Applications*, pp. 188–202, 1999.

[49] S. Foldes and P. L. Hammer, "Split graphs having dilworth number two," *Canadian Journal of Mathematics*, vol. 29, no. 3, pp. 666–672, 1977.

[50] M. C. Golumbic, "Split graphs," in *Annals of Discrete Mathematics*. Elsevier, 2004, vol. 57, pp. 149–156.

[51] S. D. Nikolopoulos, "Recognizing cographs and threshold graphs through a classification of their edges," *Information Processing Letters*, vol. 74, no. 3-4, pp. 129–139, 2000.

[52] M. Kano and S. Nikolopoulos, "On the structure of a-free graphs: Part ii," Tech. Report TR-25-99, Department of Computer Science, University of Ioannina, Tech. Rep., 1999.

[53] M. C. Golumbic, "Trivially perfect graphs," *Discrete Mathematics*, vol. 24, pp. 105–107, 1978.

[54] E. S. Wolk, "The comparability graph of a tree," *Proceedings of the American Mathematical Society*, vol. 13, no. 5, pp. 789–795, 1962.

[55] ——, "A note on" the comparability graph of a tree"," *Proceedings of the American Mathematical Society*, vol. 16, no. 1, pp. 17–20, 1965.

[56] D. Corneil, H. Lerches, and L. Burlingham, "Complement reducible graphs," *Discrete Applied Mathematics*, vol. 3, pp. 163–174, 1981.

[57] D. Corneil, Y. Perl, and L. Stewart, "A linear recognition algorithm for cographs," *SIAM J. Comput*, vol. 14, pp. 926–934, 1985.

[58] H. J. Veldman, "A result on hamiltonian line graphs involving restrictions on induced subgraphs," *Journal of graph theory*, vol. 12, no. 3, pp. 413–420, 1988.

[59] S. D. Nikolopoulos and C. Papadopoulos, "The number of spanning trees in k n-complements of quasi-threshold graphs," *Graphs and Combinatorics*, vol. 20, no. 3, pp. 383–397, 2004.

[60] S. D. Nikolopoulos, "Parallel algorithms for hamiltonian problems on quasi-threshold graphs," *Journal of Parallel and Distributed Computing*, vol. 64, no. 1, pp. 48–67, 2004.

[61] S. Kirkland, "Completion of laplacian integral graphs via edge addition," *Discrete mathematics*, vol. 295, no. 1-3, pp. 75–90, 2005.

[62] S. Akbari, E. Ghorbani, and M. R. Oboudi, "Edge addition, singular values, and energy of graphs and matrices," *Linear Algebra and its Applications*, vol. 430, no. 8-9, pp. 2192–2199, 2009.

[63] M. Papagelis, "Refining social graph connectivity via shortcut edge addition," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 10, no. 2, pp. 1–35, 2015.

[64] C. Collberg and J. Nagra, *Surreptitious software: obfuscation, watermarking, and tamperproofing for software protection*. Pearson Education, 2009.

[65] D. Grover, "The protection of computer software: Its technology and applications," *The British Computer Society Monographs in Informatics*, 1992.

[66] M. Chroni, S. D. Nikolopoulos, and L. Palios, "Encoding watermark numbers as reducible permutation graphs using self-inverting permutations," *Discrete Applied Mathematics*, vol. 250, pp. 145–164, 2018.

[67] G. Myles and C. Collberg, "Software watermarking via opaque predicates: Implementation, analysis, and attacks," *Electronic Commerce Research*, vol. 6, no. 2, pp. 155–171, 2006.

[68] C. Collberg and C. Thomborson, "On the limits of software watermarking," Department of Computer Science, The University of Auckland, New Zealand, Tech. Rep., 1998.

[69] ——, "Software watermarking: Models and dynamic embeddings," in *Proceedings of the 26th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. ACM, 1999, pp. 311–324.

[70] R. I. Davidson and N. Myhrvold, "Method and system for generating and auditing a signature for a computer program," Sep. 24 1996, uS Patent 5,559,884.

[71] D. Eppstein, M. T. Goodrich, J. Lam, N. Mamano, M. Mitzenmacher, and M. Torres, "Models and algorithms for graph watermarking," in *International Conference on Information Security*. Springer, 2016, pp. 283–301.

[72] S. A. Moskowitz and M. Cooperman, "Method for stega-cipher protection of computer code," Apr. 28 1998, uS Patent 5, 745, 569.

[73] P. R. Samson, "Apparatus and method for serializing and validating copies of computer software," Feb. 15 1994, uS Patent 5, 287, 408.

[74] R. Ghiya and L. J. Hendren, "Is it a tree, a dag, or a cyclic graph? a shape analysis for heap-directed pointers in c," in *Proceedings of the 23rd ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. ACM, 1996, pp. 1–15.

[75] L. Zhang, Y. Yang, X. Niu, and S. Niu, "A survey on software watermarking," *Journal of software*, vol. 14, no. 2, pp. 268–277, 2003.

[76] W. Zhu, C. Thomborson, and F. Y. Wang, "A survey of software watermarking," in *International Conference on Intelligence and Security Informatics*. Springer, 2005, pp. 454–458.

[77] S. Craver, N. Memon, B. L. Yeo, and M. Yeung, "Resolving rightful ownerships with invisible watermarking techniques: Limitations, attacks, and implications," *IEEE Journal on Selected areas in Communications*, vol. 16(4), pp. 573–586, 1998.

[78] ——, "Can invisible watermarks resolve rightful ownerships?" in *Proceedings of the IS&T/SPIE Conference on Storage and Retrieval for Image and Video Databases V*, vol. 3022. ACM, 1997, pp. 310–321.

[79] C. Collberg, S. Kobourov, E. Carter, and C. Thomborson, "Error-correcting graphs for software watermarking," in *Proceedings of the 29th Workshop on Graph Theoretic Concepts in Computer Science*. Springer, 2003, pp. 156–167.

[80] C. Collberg, A. Huntwork, E. Carter, G. Townsend, and M. Stepp, "More on graph theoretic software watermarks: Implementation, analysis, and attacks," *Information and Software Technology*, vol. 51, no. 1, pp. 56–67, 2009.

[81] R. Venkatesan, V. Vazirani, and S. Sinha, "A graph theoretic approach to software watermarking," in *International Workshop on Information Hiding*. Springer, 2001, pp. 157–168.

[82] M. Chroni and S. D. Nikolopoulos, "Encoding watermark integers as self-inverting permutations," in *Proceedings of the 11th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing on International Conference on Computer Systems and Technologies*. ACM, 2010, pp. 125–130.

[83] A. Mpanti, S. D. Nikolopoulos, and M. Rini, "Experimental study of the resilience of a graph-based watermarking system under edge modifications," in *Proceedings of the 21st Pan-Hellenic Conference on Informatics*. ACM, 2017, p. 34.

[84] L. M. Bento, D. R. Boccardo, R. C. Machado, V. G. P. de Sá, and J. L. Szwarcfiter, "On the resilience of canonical reducible permutation graphs," *Discrete Applied Mathematics*, vol. 234, pp. 32–46, 2018.

[85] ——, "Full characterization of a class of graphs tailored for software watermarking," *Algorithmica*, vol. 81(7), pp. 2899–2916, 2019.

[86] A. Mpanti and S. D. Nikolopoulos, "Graph-structured watermarking using bitonic sequences of self-inverting permutations," in *Proceedings of the 20th Pan-Hellenic Conference on Informatics*. ACM, 2016, p. 13.

[87] I. Chionis, M. Chroni, and S. D. Nikolopoulos, "Evaluating the waterrpg software watermarking model on java application programs," in *Proceedings of the 17th Panhellenic Conference on Informatics*, 2013, pp. 144–151.

[88] D. Novac, C. Eichler, and M. Philippsen, "Llwm & ir-mark: Integrating software watermarks into an llvm-based framework," in *Proceedings of the 2021 Research*

on Offensive and Defensive Techniques in the Context of Man at the End (MATE) Attacks, 2021, pp. 35–41.

[89] M. S. Hecht and J. D. Ullman, "Flow graph reducibility," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 188–202, 1972.

[90] ——, "Characterizations of reducible flow graphs," *Journal of the ACM (JACM)*, vol. 21, no. 3, pp. 367–375, 1974.

[91] M. E. J. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, no. 2, pp. 167–256, 2003.

[92] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's highly available key-value store," in *Proceedings of 21st ACM SIGOPS Symposium on Operating Systems Principles (SOSP)*, 2007, pp. 205–220.

[93] R. K. Jain, D.-M. W. Chiu, and W. R. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," Digital Equipment Corporation, Tech. Rep. DEC-TR-301, 1984. [Online]. Available: http://www.cse.wustl.edu/~jain/papers/ftp/fairness.pdf

[94] B. Beckmann, "Managing Wire Delay in Chip Multiprocessor Caches," PhD Thesis, Department of Computer Sciences, University of Wisconsin-Madison, Aug. 2006.

[95] C. Lai and S.-L. Lu, "Efficient Victim Mechanism on Sector Cache Organization," in *Advances in Computer Systems Architecture*, 2004, pp. 16–29. [Online]. Available: http://www.springerlink.com/content/aklx8nk7vch3tr4u

[96] D. Rubino. IE9 for Windows Phone 7: Adobe Flash, demos and development. [Online]. Available: http://www.wpcentral.com/ie9-windows-phone-7-adobe-flash-demos-and-development-video

Author's Publications

- Mpanti, A., Nikolopoulos, S. D., and Palios, L.: Adding an Edge in a P_4-Sparse Graph (submitted)
- Mpanti, A., Nikolopoulos, S. D., and Palios, L.: Adding a Tail in Classes of Perfect Graphs (submitted)
- Georgiadis, L., Kefallinos, D., Mpanti, A., and Nikolopoulos, S. D.: An Experimental Study of Algorithms for Packing Arborescences. In 20th International Symposium on Experimental Algorithms, 2022.
- Mpanti, A., Nikolopoulos, S. D., and Palios, L.: Strong watermark numbers encoded as reducible permutation graphs against edge modification attacks. Journal of Computer Security, (Preprint), 1-22, 2022.
- Dounavi, H. M., Mpanti, A., Nikolopoulos, S. D., and Polenakis, I.: Detection and Classification of Malicious Software based on Regional Matching of Temporal Graphs. In International Conference on Computer Systems and Technologies' 21, pp. 28-33, 2021.
- Dounavi, H. M., Mpanti, A., Nikolopoulos, S. D., and Polenakis, I.: A graph-based framework for malicious software detection and classification utilizing temporal-graphs. Journal of Computer Security, (Preprint), 1-38, 2021.
- Mpanti, A., Nikolopoulos, S. D., and Polenakis, I.: A graph-based model for malicious software detection exploiting domination relations between system-call groups. In Proceedings of the 19th International Conference on Computer Systems and Technologies, pp. 20-26, 2018.
- Mpanti, A., Nikolopoulos, S. D. and Polenakis, I.: "Defending Hardware-based Attacks on Trusted Computing using a Hardware-Integrity Attestation Proto-

col". In: Proceedings of the 18 th International Conference on Computer Systems and Technologies 2017 ACM.

- Mpanti, A., Nikolopoulos, S. D. and Rini, M.: "Experimental Study of the Resilience of a Graph-based Watermarking System under Edge Modifications". PCI 2017: 34:1-34:6

- Mpanti, A. and Nikolopoulos, S. D.: "Graph-structured Watermarking using Bitonic Sequences of Self-inverting Permutations". PCI 2016: 13 [i1]

- Mpanti, A. and Nikolopoulos, S. D.: "Two RPG Flow-graphs for Software Watermarking using Bitonic Sequences of Self-inverting Permutations". CoRR abs/1607.02281, 2016.

Short Biography

Anna Mpanti received her B.Sc. degree in Mathematics (2015) from the Department of Mathematics of the University of Ioannina, Greece. In 2017 she received his postgraduate degree in the Department of Computer Science and Engineering of the Polytechnic School of the University of Ioannina, acquiring the postgraduate degree (MSc) in Theory of Computer Science. Her Master Thesis, entitled "Graph-based Algorithmic Techniques for Watermarking using Self-inverting Permutations and Bitonic Sequences", was conducted under the supervision of Professor Stavros D. Nikolopoulos.

Her research interests are focused on the design and analysis of algorithms, algorithms engineering, approximation algorithms, algorithmic graph theory, and information hiding. Anna has been an assistant in the laboratories of the undergraduate course on Introduction to Programming and has also been an assistant in the laboratories of the undergraduate course on Design and Analysis of Algorithms in the Department of Computer Science & Engineering, University of Ioannina.