Packing and Covering: New Paradigms and Algorithms

By

Abhishek Sahu

MATH10201504011

The Institute of Mathematical Sciences, Chennai

A thesis submitted to the

Board of Studies in Mathematical Sciences

In partial fulfillment of requirements

for the Degree of

DOCTOR OF PHILOSOPHY

of

HOMI BHABHA NATIONAL INSTITUTE

Dec, 2020

Homi Bhabha National Institute

Recommendations of the Viva Voce Committee

As members of the Viva Voce Committee, we certify that we have read the dissertation prepared by Abhishek Sahu entitled "Packing and Covering: New Paradigms and Algorithms" and recommend that it may be accepted as fulfilling the thesis requirement for the award of Degree of Doctor of Philosophy.

[illegible] Date: 27th Jan 2022

Chairman - Venkatesh Raman

[illegible] Date: 27 Jan 2022

Guide/Convenor - Saket Saurabh

[illegible] Date: 27 Jan 2022

Examiner - Rogers Mathew

V. Arvind Date: Jan 27 2022

Member 1 - V. Arvind

Sharma Date: 27 Jan 2022

Member 2 - Vikram Sharma

[illegible] Date: 27 Jan 2022

Member 3 - Geevarghese Philip

Final approval and acceptance of this thesis is contingent upon the candidate's submission of the final copies of the thesis to HBNI.

I hereby certify that I have read this thesis prepared under my direction and recommend that it may be accepted as fulfilling the thesis requirement.

Date: 27th Jan 2022

Place: Chennai

[illegible]

Saket Saurabh (Guide)

STATEMENT BY AUTHOR

Abhishek Sahu

Abhishek Sahu

DECLARATION

I hereby declare that the investigation presented in the thesis has been carried out by me. The work is original and has not been submitted earlier as a whole or in part for a degree / diploma at this or any other Institution / University.

Abhishek Sahu

Abhishek Sahu

LIST OF PUBLICATIONS ARISING FROM THE THESIS

Journal

1. **Dynamic Parameterized Problems**: R. Krithika, *Abhishek Sahu*, Prafullkumar Tale, in proceedings of the 11th International Symposium on Parameterized and Exact Computation *(ALGORITHMICA 2018)*.

2. **The Parameterized Complexity of Cycle Packing: Indifference is Not an Issue**: R. Krithika, *Abhishek Sahu*, Saket Saurabh, Meirav Zehavi, in proceeding the 13th Latin American Theoretical INformatics Symposium *(ALGORITHMICA 2019)*.

Conferences

1. **Dynamic Parameterized Problems**: R. Krithika, *Abhishek Sahu*, Prafullkumar Tale, in proceedings of the 11th International Symposium on Parameterized and Exact Computation *(IPEC 2016)*.

2. **Mixed Dominating Set**: A Parameterized Perspective: Pallavi Jain, M. Jayakrishnan, Fahad Panolan, Abhishek Sahu, *in proccedings of the 43rd International Workshop on Graph-Theoretic Concepts in Computer Science*, *(WG 2017)* .

3. **The Parameterized Complexity of Cycle Packing: Indifference is Not an Issue**: R. Krithika, Abhishek Sahu, Saket Saurabh, Meirav Zehavi, *in proceedings the 13th Latin American Theoretical INformatics Symposium*, *(LATIN 2018)*.

4. **Packing Arc-Disjoint Cycles in Tournaments**: Stephane Bessy,h Bougeret,R. Krithika, Abhishek Sahu, Saket Saurabh,Jocelyn Thiebaut, Meirav Zehavi, in proceedings of *44th International Symposium on Mathematical Foundations of Computer Science*, *(MFCS 2019)*.

5. **On the Complexity of Mixed Dominating Set**: Jayakrishnan Madathil, Fahad Panolan, Abhishek Sahu, Saket Saurabh, *in proceedings of the 14th International Computer Science Symposium in Russia*, *(CSR 2019)*.

6. **Graph Hamiltonicity Parameterized by Proper Interval Deletion Set**Petr A. Golovach, R. Krithika, Abhishek Sahu, Saket Saurabh, Meirav Zehavi, *in proceedings the 15th Latin American Theoretical INformatics Symposium*, *(LATIN 2020)*.

7. **Kernelization of Arc Disjoint Cycle Packing in α -bounded digraphs** Abhishek Sahu, Saket Saurabh,*in proceedings of the 15th International Computer Science Symposium in Russia*, *(CSR 2020)*.

Abhishek Sahu

Abhishek Sahu

DEDICATIONS

This page is optional.

ACKNOWLEDGEMENTS

First and foremost I would like to express my deepest appreciation to my advisor Prof. Saket Saurabh for his patience, motivation and knowledge. He has been immensely supportive and a positive presence through all the ups and downs. I am grateful for all the invaluable discussions and professional as well as personal advice that I have received from him. Without his guidance and persistent help this dissertation would not have been possible.

Sincere thanks to Meirav, Krithika, Fahad, Jayakrishnan, Pratik, Petr, Akanksha, Roohani, Pallavi, Diptapriyo, Pranabendu for all the technical discussions and to my colleagues-Lawqueen, Niranka, Jayakrishnan, Sanjukta, Yogesh for many wonderful memories. I remain deeply indebted to my parents and my brother for their unconditional support.

Contents

List of figures

Summary

In this section we briefly summarize the work done in my PhD thesis. We can divide our work into three parts. First we have shown polynomial kernels for ARC DISJOINT CYCLE PACKING in α-bounded digraphs and linear kernels in tournaments. CYCLE PACKING is polynomial time solvable on proper interval graphs. So we tried to analyze the complexity of the problem when extra k vertices are added which is CYCLE PACKING parameterized by the structural parameter- *proper interval deletion set*. We obtained $2^{\mathcal{O}(t\log t)}n^{\mathcal{O}(1)}$ time FPTalgorithm. We also designed $2^{\mathcal{O}(t\log t)}n^{\mathcal{O}(1)}$ time FPT algorithms for CYCLE/ PATH COVER parameterized by the same parameter.

In part 2, we looked at a special kind of domination problem known as MIXED DOMINATING SET and studied it from the parameterized perspective with respect to various parameters such as solution size, treewidth etc.. We also studied its complexity on different graph classes such as split graphs, proper interval graphs, $K_{d,d}$-free graphs and obtained various FPTalgorithms, kernelization results and some lower bounds.

In the third part, we have considered a special variation of dynamic problem where the input graph gets updated by a series of edge insertions and deletions and the goal is to efficiently maintain a solution. A dynamic version of a problem P is a quintuple (I,I',S,k,r) where I and I' are instances of P, and S is a solution (not necessarily optimal) to I. Further I' can be obtained from I by making at most k edits. The task is to determine whether there is a solution S' (not necessarily optimal) to I' that can be obtained from S by making at most r changes. We did solve various problems such as VERTEX COVER, FEEDBACK VERTEX

SET, CONNECTED VERTEX COVER, DOMINATING SET, CONNECTED DOMINATING SET. We studied these problems with respect to both parameters- *r* and *k*.

Chapter 1

Introduction

1.1 Preamble

Packing and Covering are some of the fundamental problems in graph theory. An H-PACKING problem is, given a graph G, what is the maximum number of disjoint graphs in H one can find in G. Similarly in H-COVERING problem we desire to find the minimum number of disjoint graphs in H that together constitute the graph G. Both these problems are extremely well studied and proved to be NP-hard. The COVERING problems that we study encompasses the very well known HAMILTONICITY problems. In part 1 of our thesis we study these problems where H is the class of cycles/paths. We study these problems with respect to the standard parameter (solution size) as well as some well known structural parameters.

Another problem that we consider that is not as well studied but still has significant importance is the TOTAL COVERING or MIXED DOMINATION problem. Given a graph G a set $S \subseteq V(G) \cup E(G)$ is a mixed dominating set iff every element $x \in (V(G) \cup E(G)) \setminus S$ is either adjacent to or incident with an element of S. Not only we consider MIXED DOMINATION with different parameters, but also we study its complexity on different graph classes.

The last set of problems that we focus on in our thesis is a specific kind of dynamic problem. Here the graph is being updated by a series of edge insertions and deletions. And the goal is to efficiently maintain a solution to the problem. Formally, a dynamic version of a problem P is a quintuple (I, I', S, k, r) where I and I' are instances of P and S is a solution (not necessarily optimal) to I. Further, I' can be obtained from I by making at most k edits. The task is to determine whether there is a solution S' (not necessarily optimal) to I' that can be obtained from S by making at most r changes.

We study the parameterized version of all the above mentioned problems. So first we reiterate the notations and definitions of parameterized complexity below.

Parameterized complexity: In this framework, each problem instance is associated with a non-negative integer k called parameter, and a problem is said to be fixed-parameter tractable if it can be solved in $f(k)n^{O(1)}$ time for some computable function f, where n is the input size. For convenience, the running time $f(k)n^{O(1)}$, where f grows super-polynomially with k is denoted as $O(f(k))$. A kernelization algorithm is a polynomial-time algorithm that transforms an arbitrary instance of the problem to an equivalent instance of the same problem whose size is bounded by some computable function g of the parameter of the original instance. The resulting instance is called a kernel and if g is a polynomial function, then it is called a polynomial kernel and we say that the problem admits a polynomial kernel. A decidable parameterized problem is fixed-parameter tractable if and only if it has a kernel (not necessarily of polynomial size). Kernelization typically involves applying a set of rules (called reduction rules) to the given instance to produce another instance. A reduction rule is said to be safe if it is sound and complete, i.e., applying it to the given instance produces an equivalent instance. In order to classify parameterized problems as being fixed-parameter tractable or not, the W-hierarchy is defined: $\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \ldots \subseteq \mathsf{XP}$ where $\mathsf{FPT}$is the set of all parameterized problems that are fixed-parameter tractable. It is believed that the subset relations in this sequence are all strict, and a parameterized problem that is hard for some complexity class above FPTin this hierarchy is said to be fixed-parameter intractable. As mentioned before, the set of parameterized problems that admit a polynomial kernel is contained in the class FPTand it is believed that this subset relation is also strict. For further details on parameterized algorithms, we refer to [25].

Structural parameters: In the early years of parameterized complexity and algorithms, problems were almost always parameterized by the solution size. Recent research has focused on other parameterizations based on structural parameters in the input [56], or above or below some guaranteed optimum values [47, 48, 72]. Such "non-standard" parameters are more likely to be small in practice. Also, once a problem is shown to be FPT or to have a polynomial sized kernel by a parameterization, it is natural to ask whether the problem is FPT(and admits a polynomial kernel) when parameterized by a provably smaller parameter. In the same vein, if we show that a problem is W-hard under a parameterization, it is natural to ask whether it is FPT when parameterized by a provably larger parameter.

Apart from solution size, *treewidth* is one of the most well studied parameters. However, in the context of CYCLE PACKING, our understanding of treewidth is *complete* in the following sense: while CYCLE PACKING is known to be solvable in time $\mathcal{O}^*(2^{\mathcal{O}(\mathsf{tw}\log\mathsf{tw})})$, it cannot be solved in time $\mathcal{O}^*(2^{o(\mathsf{tw}\log\mathsf{tw})})$ unless the Exponential Time Hypothesis fails [29]. Another parameter that has gained significant attention recently is the size of a *modulator* to a family of graphs. Let $\mathscr{F}$ be a family of graphs. Given a graph G and a set $S \subseteq V(G)$, we say that S is an *$\mathscr{F}$-modulator* if $G - S$ is in $\mathscr{F}$. For example, if $\mathscr{F}$ is the family of independent sets, forests, bipartite graphs, interval graphs and chordal graphs, then the modulator corresponds to a vertex cover, feedback vertex set, odd cycle transversal, interval deletion set and chordal deletion set, respectively. The size of S is also called the *vertex-deletion distance* to $\mathscr{F}$. One of the earliest studies in the realm of alternate parameterizations is by Cai [17]. Cai [17] studied COLORING problems parameterized by the vertex-deletion distance to various graph classes including bipartite graphs and split graphs. Fellows et al. [37] studied alternate parameterizations for problems that were proven to be intractable with respect to the standard parameterization. This led to a whole new ecology program and opened up a floodgate of new and exciting research. Structural parameterizations of the classical VERTEX COVER ([12, 56]) and FEEDBACK VERTEX SET [57] have also been explored. We refer to [56] for a detailed introduction to the whole program as well as the thesis of Jansen [55]. Focusing on structural parameters for CYCLE PACKING, one of the problems considered in our thesis, Bodlaender et al. [13] obtained polynomial kernels with respect to the size of vertex cover, the vertex-deletion distance to a cluster graph and the maximum leaf number (see [13] or [55] for definitions). There is also a kernel lower bound result known for parameterization with respect to solution size [14] leading to several other lower bounds.

1.2 Our contributions

We divide our work in the thesis into 3 parts. For each part we will briefly state our contributions. The detailed descriptions of previous work and our contributions can be found in the introductions of each chapter.

In part 1 of our thesis, we consider PACKING/COVERING problems. On tournaments VERTEX-DISJOINT CYCLE PACKING directly reduces to 3-SET PACKING and admits an $\mathcal{O}(k^2)$ kernel as a consequence (recently it was improved to a subquadratic kernel [68]). Hence instead we focus our attention on the arc variant of the problem i.e. ARC-DISJOINT CYCLE PACKING. We first study the problem on tournaments, where we give a linear bound ($\mathcal{O}(k)$) on *feedback vertex set* similar to the very well known Erdős-Pósa bound and design a linear kernel. Making use of this kernel we obtain an FPTalgorithm running in time $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$. The next question that interested us was, could we similarly design polynomial kernels on α-bounded digraphs which contain tournaments. We answer this question affirmatively by again obtaining an Erdős-Pósa like bound on *feedback vertex set* of size $\mathcal{O}(\alpha^2 k^2)$ and polynomial kernels for every constant value of α. We use one of the results obtained by Lochet et al. [70] that gives a small cut-preserving set on α-bounded digraphs. Combining their result with our theorems we obtain the desired polynomial kernel.

Next we consider VERTEX-DISJOINT CYCLE PACKING parameterized by some structural parameters. For the VERTEX-DISJOINT CYCLE PACKING problem, Bodlaender et al. [13] obtained polynomial kernels with respect to the size of vertex cover, the vertex-deletion distance to a cluster graph and the maximum leaf number. We try to contribute some results to this line of work. Since the problem is still open on interval graphs, we consider a natural subclass of interval graphs known as proper interval graphs. We study the problem parameterized by vertex deletion distance to proper interval graphs (t). Just as chordal graphs have clique-tree decomposition structure, proper interval graphs have clique-path decomposition structure [58]. We combine color coding, greedy strategy and multi layered dynamic programming to design an FPTalgorithm running in time $2^{\mathcal{O}(t \log t)} n^{\mathcal{O}(1)}$.

The next set of problems that we consider are- CYCLE/PATH COVERING, parameterized by the same structural parameter- proper interval deletion set. These problems are generalizations of HAMILTONIAN CYCLE/PATH problems. There has been an extensive study of structural parameters for problems related to PATH COVER and CYCLE COVER such as CYCLE PACKING, LONGEST PATH and LONGEST CYCLE [13, 64]. We first discovered some interesting properties (monotonicity properties) of paths in proper interval graphs. We explore these properties to design dynamic programming algorithms that solve the CYCLE/PATH COVERING in $2^{\mathcal{O}(t \log t)} n^{\mathcal{O}(1)}$. This also automatically solves the HAMILTONICITY problem parameterized by proper interval deletion set. We summarize all these results in Table 1.1.

In part 2 of our thesis we focus on the problem of MIXED DOMINATION. MDS (MIXED DOMINATING SET) parameterized by the solution size (k) can be solved in time

PACKING and COVERING Problems	Parameters	Our results
ARC DISJOINT CYCLE PACKING in tournaments	Solution size(k)	$\mathcal{O}(k)$ kernel, $\mathcal{O}^*(2^{\mathcal{O}(k\log k)})$ FPT
ARC DISJOINT CYCLE PACKING in α- bounded digraphs	Solution size(k)	$2\alpha^4 k^5(110\alpha^{35}k^{30})^{4^\alpha}$ kernel
VERTEX DISJOINT CYCLE PACKING	Proper interval deletion set (t)	$\mathcal{O}^*(2^{\mathcal{O}(t\log t)})$ FPT
VERTEX DISJOINT CYCLE COVERING	Proper Interval deletion set (t)	$\mathcal{O}^*(2^{\mathcal{O}(t\log t)})$ FPT
VERTEX DISJOINT PATH COVERING	Proper Interval deletion set (t)	$\mathcal{O}^*(2^{\mathcal{O}(t\log t)})$ FPT

Table 1.1: Our results on PACKING/COVERING problems

$7.465^k n^{\mathcal{O}(1)}$ on general graphs, and in time $2^{\mathcal{O}(\sqrt{k})}n^{\mathcal{O}(1)}$ on planar graphs.[*] We complement this result by showing that MDS on general graphs does not admit an algorithm with running time $2^{o(k)}n^{\mathcal{O}(1)}$ unless the Exponential Time Hypothesis (ETH) fails, and that it does not admit a polynomial kernel unless $coNP \subseteq NP/poly$. In addition, we provide an algorithm which, given a graph G together with a tree decomposition of width tw, solves MDS in time $6^{tw}n^{\mathcal{O}(1)}$. We also show that unless the Set Cover Conjecture fails, MDS does not admit an algorithm with running time $\mathcal{O}((2-\varepsilon)^{tw(G)}n^{\mathcal{O}(1)})$ for any $\varepsilon > 0$, where $tw(G)$ is the tree-width of G.

We also study the problem on various graph classes and obtain the following results.

- On split graphs, MDS does not admit a polynomial kernel unless $\mathsf{coNP} \subseteq \mathsf{NP/poly}$.
- On proper interval graphs, MDS is polynomial time solvable.
- On graphs that do not contain $K_{d,d}$ as a subgraph (biclique-free graphs), MDS admits a kernel of size $\mathcal{O}(k^d)$.

We use the standard branching technique to design an exact algorithm with running time $2^n n^{\mathcal{O}(1)}$ for MDS on general graphs.

[*]MDS parameterized by solution size has now an improved running time of $\mathcal{O}^*(3.510^k)$ [34].

In part 3 of our thesis we consider some well known parameterized problems in a dynamic framework where the input graph is being updated by a series of edge insertions and deletions. The goal is to efficiently maintain a solution. We recall the notations for the dynamic problem instance stated in the preamble section. In the context of parameterized algorithms, two relevant parameters are the edit distance k (number of updates) and the Hamming distance r (between the input solution and a solution to the updated instance). We revisit some classical parameterized problems in the dynamic setting, where the input graph gets updated. For a fixed collection of graphs Π, given a graph G and an integer l, the Π-DELETION problem is to determine if G has a set $S \subseteq V(G)$ of vertices with $|S| = l$ such that $G - S \in \Pi$. Π-DELETION is an abstraction of various classical problems in the graph theoretic framework. Examples include the classical VERTEX COVER and FEEDBACK VERTEX SET. Due to a generic result by Lewis and Yannakakis [69], it is known that finding a minimum solution to Π-DELETION is NP-hard in general for most choices of Π. Hence, it has been extensively studied in various algorithmic realms. We define the dynamic version of this problem referred to as DYNAMIC Π-DELETION and show NP-hardness, fixed-parameter tractability and kernelization results. Then, for the specific cases of Π-DELETION such as DYNAMIC VERTEX COVER and DYNAMIC FEEDBACK VERTEX SET, we give improved FPT algorithms with respect to k as the parameter and also obtain linear kernels. Then, for the same parameterization, we describe improved algorithms for DYNAMIC CONNECTED VERTEX COVER, DYNAMIC DOMINATING SET and DYNAMIC CONNECTED DOMINATING SET. For DYNAMIC DOMINATING SET and DYNAMIC CONNECTED DOMINATING SET, we show that these running times are optimal (up to polynomial factors) assuming the Set Cover Conjecture. The Table 1.2 summarizes these results along with the running time bounds known for these problems.

Dynamic Problem	Parameter k	Parameter r
VERTEX COVER	1.0822^k $\mathcal{O}(k)$ kernel	1.2738^r $\mathcal{O}(r^2)$ kernel
CONNECTED VERTEX COVER	4^k [2], 2^k ‡ No $k^{\mathcal{O}(1)}$ size kernel	W[2]-hard [2]
FEEDBACK VERTEX SET	1.6667^k (randomized) $\mathcal{O}(k)$ kernel	3.592^r, 3^r (randomized) $\mathcal{O}(r^2)$ kernel
CONNECTED FEEDBACK VERTEX SET	$2^{\mathcal{O}(k)}$ No $k^{\mathcal{O}(1)}$ size kernel	W[2]-hard
DOMINATING SET	$2^{\mathcal{O}(k^2)}$ [33], 2^k ‡ No $k^{\mathcal{O}(1)}$ size kernel	W[2]-hard [33]
CONNECTED DOMINATING SET	4^k [2], 2^k ‡ No $k^{\mathcal{O}(1)}$ size kernel	W[2]-hard [2]

Table 1.2: Summary of known and new results for different dynamic parameterized problems. All running time bounds are specified by ignoring polynomial factors. ‡ denotes that the running time is optimal under the Set Cover Conjecture. k and r respectively denote the parameters ”edit distance”and ”hamming distance”.

Chapter 2

Notations

In this chapter we state some notations and graph structural properties.

Undirected graphs: For an undirected graph G, $V(G)$ and $E(G)$ denote the set of vertices and edges, respectively. Two vertices u,v are said to be *adjacent* if there is an edge (denoted as uv or vu) between u and v. The (open) neighborhood of a vertex v, denoted by $N_G(v)$, is the set of vertices adjacent to v in G and its closed neighborhood $N_G[v]$ is the set $N_G(v) \cup \{v\}$. This notation is extended to subsets of vertices as $N_G[S] = \bigcup_{v \in S} N_G[v]$ and $N_G(S) = N_G[S] \setminus S$ where $S \subseteq V(G)$. The subscript in the notation for neighborhood is omitted if the graph under consideration is clear. For a set $S \subseteq V(G)$, $G[S]$ and $G - S$ denote the subgraphs of G induced on the set S and $V(G) \setminus S$, respectively. For a set of edges E', $V(E')$ denotes the union of the endpoints of the edges in E'. The contraction operation of an edge $e = uv$ in G adds a new vertex w adjacent to the vertices that are adjacent to either u or v and then deletes u and v. That is, the contraction of e in G results in a graph G' with $V(G') = V(G) \cup \{w\} \backslash \{u,v\}$ and $E(G') = \{xy | x,y \in V(G) \backslash \{u,v\}$ and $xy \in E(G)\} \cup \{wx | x \in N_G(u) \cup N_G(v)\}$. An independent set is a set of pairwise non-adjacent vertices. A forest is a graph with no cycles. A set $Q \subseteq V(G)$of pairwise adjacent vertices is called a clique. A clique of three vertices (which also forms a cycle on three vertices) is called a triangle. A graph is said to be *triangle-free* if it has no triangles.

Directed graphs: A *directed graph* (or *digraph*) is a pair consisting of a set of vertices and a set of arcs. An arc is specified as an ordered pair of vertices (called its endpoints). We consider only simple unweighted digraphs. For a digraph D, $V(D)$ and $A(D)$ denote the set of its vertices and the set of its arcs, respectively. Two vertices u, v are said to be *adjacent* in D if $uv \in A(D)$ or $vu \in A(D)$. For an arc $e = uv$, we define $\mathrm{h}(e) = v$ as the head of e and $\mathrm{t}(e) = u$ as the tail of e. For a vertex $v \in V(D)$, its *out-neighbourhood*, denoted by $N^+(v)$, is the set $\{u \in V(D)\colon vu \in A(D)\}$ and its *in-neighbourhood*, denoted by $N^-(v)$, is the set $\{u \in V(D)\colon uv \in A(D)\}$. For a set F of arcs, $V(F)$ denotes the union of the sets of endpoints of arcs in F. Given a digraph D and a subset X of vertices, we denote by $D[X]$ the digraph induced by the vertices in X. Moreover, we denote by $D \setminus X$ the digraph $D[V(D) \setminus X]$ and say that this digraph is obtained by *deleting X from D*. For a set $F \subseteq A(D)$, $D - F$ denotes the digraph obtained from D by deleting F.

Tournaments: A *tournament T* is a digraph in which for every pair u, v of distinct vertices either $uv \in A(T)$ or $vu \in A(T)$ but not both. In other words, a tournament T on n vertices is an orientation of the complete graph K_n. A tournament T can alternatively be defined by an ordering $\sigma(T) = (v_1, \dots, v_n)$ of its vertices and a set of *backward arcs* $\overleftarrow{A}_\sigma(T)$ (which will be denoted $\overleftarrow{A}(T)$ as the considered ordering is not ambiguous), where each arc $a \in \overleftarrow{A}(T)$ is of the form $v_{i_1}v_{i_2}$ with $i_2 < i_1$. Indeed, given $\sigma(T)$ and $\overleftarrow{A}(T)$, we define $V(T) = \{v_i : i \in [n]\}$ and $A(T) = \overleftarrow{A}(T) \cup \overrightarrow{A}(T)$ where $\overrightarrow{A}(T) = \{v_{i_1}v_{i_2} : (i_1 < i_2) \text{ and } v_{i_2}v_{i_1} \notin \overleftarrow{A}(T)\}$ is the set of *forward arcs* of T in the given ordering $\sigma(T)$. The pair $(\sigma(T), \overleftarrow{A}(T))$ is called a *linear representation* of the tournament T. A tournament is called *transitive* if it is a directed acyclic graph and a transitive tournament has a unique topological ordering. It is clear that for any linear representation $(\sigma(T), \overleftarrow{A}(T))$ of T the set $\overleftarrow{A}(T)$ is an FAS (feedback arc set) of T. Given a linear representation $(\sigma(T), \overleftarrow{A}(T))$ of a tournament T, a triangle C in T is a triple $(v_{i_1}, v_{i_2}, v_{i_3})$ with $i_l < i_{l+1}$ such that either $v_{i_3}v_{i_1} \in \overleftarrow{A}(T)$, $v_{i_3}v_{i_2} \notin \overleftarrow{A}(T)$ and $v_{i_2}v_{i_1} \notin \overleftarrow{A}(T)$ (in this case we call C a *triangle with backward arc $v_{i_3}v_{i_1}$*), or $v_{i_3}v_{i_1} \notin \overleftarrow{A}(T)$, $v_{i_3}v_{i_2} \in \overleftarrow{A}(T)$ and $v_{i_2}v_{i_1} \in \overleftarrow{A}(T)$ (in this case we call C a *triangle with two backward arcs*

$v_{i_3}v_{i_2}$ and $v_{i_2}v_{i_1}$). Given two tournaments T_1, T_2 defined by $\sigma(T_l)$ and $\overleftarrow{A}(T_l)$ with $l \in \{1,2\}$, we denote by $T = T_1T_2$ the tournament called the *concatenation of* T_1 *and* T_2, where $V(T) = V(T_2) \cup V(T_2)$, $\sigma(T) = \sigma(T_1)\sigma(T_2)$ is the concatenation of the two sequences, and $\overleftarrow{A}(T) = \overleftarrow{A}(T_1) \cup \overleftarrow{A}(T_2)$.

Proper Interval Graphs: An interval I, denoted as $[i, j]$ with $i \leq j$ and $i, j \in \mathbb{N}$, is the ordered set $\{i, \dots, j\}$ of consecutive integers. For an interval $I = [i, j]$, i is called the *left endpoint* and j is called the *right endpoint*. A graph is an interval graph if its vertices can be assigned to intervals such that there is an edge between two vertices if and only if their corresponding intervals have a non-empty intersection. A set of intervals assigned to the vertices of an interval graph satisfying this property is called an *interval representation*. An interval graph is called a *proper interval graph* if it has an interval representation in which no interval properly contains another interval. Such an interval representation is called a *proper interval representation*.

Let G be a proper interval graph and $\mathscr{I}_G$ be its proper interval representation. For each $v \in V(G)$ with interval $I(v)$, $l(v)$ denotes the left endpoint of $I(v)$ and $r(v)$ denotes the right endpoint of $I(v)$. Observe that for two distinct vertices u and v of G, if $l(u) < l(v)$, then $r(u) < r(v)$ too. This imposes a natural total order on the intervals which in turn defines a permutation $\pi : V(G) \to [|V(G)|]$ of the vertices obtained by listing the vertices in the ascending order of the left (or right) endpoints of their intervals. This ordering or permutation is called *proper interval ordering* and has the following property.

Proposition 1 ([75])**.** *Let G be a proper interval graph with proper interval ordering π. For every pair u, v of vertices with $\pi(u) < \pi(v)$, if $uv \in E(G)$, then $\{w \in V(G) : \pi(u) \leq \pi(w) \leq \pi(v)\}$ is a clique in G.*

Given a proper interval representation, the following result known from Ke et al. [58] states that the vertex set of the proper interval graph can be organized into a sequence of cliques satisfying certain properties.

Proposition 2 ([58]). *Given a proper interval graph G with proper interval ordering π, there is a linear-time algorithm that outputs a partition of $V(G)$ into a sequence $Q_1,\dots,Q_q$ of (pairwise vertex-disjoint) cliques satisfying the following properties.*

- *For each pair of vertices $u \in Q_i$, $v \in Q_j$ with $1 \le i < j \le q$, $\pi(v) > \pi(u)$.*
- *For every edge $uv \in E(G)$, there exists $1 \le i \le q$ such that either $u,v \in Q_i$ or $u \in Q_i$ and $v \in Q_{i+1}$ or $u \in Q_{i+1}$ and $v \in Q_i$.*

Observe that this partition is different from the classical clique path decomposition of (proper) interval graphs. We refer to the ordered set of cliques $\mathscr{Q} = \{Q_1,\dots,Q_q\}$ as a *clique partition* of G. As a proper interval representation, a proper interval ordering π and a clique partition $\mathscr{Q}$ of a proper interval graph can be obtained in polynomial time [45, 58], we assume that $\mathscr{Q}$ and π are given as part of the input with the proper interval graph G. We remark that neither the proper interval representation (and hence the proper interval ordering resulting from it) nor the clique partition is unique.

Paths and cycles: A *path* $P = (v_1,\dots,v_\ell)$ is a sequence of distinct vertices where every consecutive pair of vertices are adjacent. We say that P *starts* at v_1 and *ends* at v_ℓ. The vertices (or vertex set) of P, denoted by $V(P)$, is the set $\{v_1,\dots,v_\ell\}$. The *endpoints* of P is the set $\{v_1,v_\ell\}$ and the *internal vertices* of P is the set $V(P)\setminus\{v_1,v_\ell\}$. The *length* of P is defined as $|V(P)|$. A *cycle* is a sequence $(v_1,\dots,v_\ell)$ of vertices such that $(v_1,\dots,v_\ell)$ is a path and $v_\ell v_1$ is an edge. A path $(v_1,\dots,v_\ell)$ is also represented as the ordered set $v_1 \to \dots \to v_\ell$. For a collection $\mathscr{P}$ of paths (or cycles), $V(\mathscr{P})$ denotes the set $\bigcup_{P\in\mathscr{P}} V(P)$. The concatenation of paths $P_1 = (v_1,\dots,v_{j-1},v_j)$ and $P_2 = (v_j,v_{j+1},\dots,v_\ell)$ such that $V(P_1)\cap V(P_2) = \{v_j\}$ is defined as the path $P_3 = (v_1,\dots,v_{j-1},v_j,v_{j+1},\dots,v_\ell)$.

A set of pairwise disjoint paths/cycles is called a *path/cycle packing* and a set of pairwise disjoint triangles is called a *triangle packing*. Similarly a set of disjoint paths/cycles that covers all the vertices of G is called a path/cycle cover of G. G is said to be *connected* if there is a path between every pair of its vertices and it is said to be *2-connected* if for each $v \in V(G)$, $G - v$ is connected.

All the above notations for paths and cycles are defined similarly for directed graphs. A digraph is called a *directed acyclic graph* if it has no cycles. A *feedback arc set* (FAS) is a set of arcs whose deletion results in an acyclic graph. For a digraph D, $\text{minfas}(D)$ denotes the size of a minimum FAS of D. Any directed acyclic graph D has an ordering $\sigma(D) = (v_1, \dots, v_n)$ called *topological ordering* of its vertices such that for each $v_i v_j \in A(D)$, $i < j$ holds. Given an ordering σ and two vertices u and v, we write $u <_\sigma v$ if u is before v in σ. For graph theoretic terms and definitions not stated explicitly here, we refer to Diestel [31].

Treewidth Let G be a graph. A *tree-decomposition* of a graph G is a pair $(\mathbb{T}, \mathscr{X} = \{X_t\}_{t \in V(\mathbb{T})})$ such that

- $\bigcup_{t \in V(\mathbb{T})} X_t = V(G)$,
- for all $xy \in E(G)$ there is a $t \in V(\mathbb{T})$ such that $\{x, y\} \subseteq X_t$, and
- for all $v \in V(G)$ the subgraph of $\mathbb{T}$ induced by $\{t \mid v \in X_t\}$ is connected.

The *width* of a tree decomposition is $\max_{t \in V(\mathbb{T})} |X_t| - 1$ and the *treewidth* of G is the minimum width over all tree decompositions of G and is denoted by $\text{tw}(G)$.

Functions and permutations: The set $\{1, \dots, n\}$ is denoted by $[n]$. Let $\Upsilon : A \to B$ be a function from a set A to a set B. For an element $a \in A$, $\Upsilon(a)$ is called the *image* of a. The *domain* of Υ, denoted by $\text{dom}(\Upsilon)$, is A and the codomain of Υ is B. The *image* (or *range*) of Υ, denoted by $\text{img}(\Upsilon)$, is the set $\{b \in B : \exists a \in A, \Upsilon(a) = b\}$. Υ is an *injective function* (or an injection) if for each $a_1, a_2 \in \text{dom}(\Upsilon)$, $\Upsilon(a_1) = \Upsilon(a_2)$ implies $a_1 = a_2$ holds. In that case, for an element $b \in \text{img}(\Upsilon)$, $\Upsilon^{-1}(b)$ denotes the element $a \in A$ with $\Upsilon(a) = b$. Υ is a *surjective function* (or a surjection) if $B = \text{img}(\Upsilon)$. Υ is a *bijection* if it is both an injection and a surjection. In that case, for each $b \in B$, $\Upsilon^{-1}(b)$ denotes the unique element $a \in A$ with $b = \Upsilon(a)$. An *empty function* is a function where the domain is the empty set. For a set $A' \subseteq \text{dom}(\Upsilon)$, $\Upsilon(A')$ denotes the set $\{\Upsilon(a) : a \in A'\}$. The function Υ

restricted to a subset $A' \subseteq A$ as the domain is defined as the function Υ' with $\mathrm{dom}(\Upsilon') = A'$ such that $\Upsilon'(a) = \Upsilon(a)$ for each $a \in A'$. A permutation σ of a set $A = \{a_1, \ldots, a_{|A|}\}$ is denoted by a sequence $a_{i_1}, a_{i_2}, \ldots, a_{i_{|A|}}$ where $\{i_1, i_2, \ldots, i_{|A|}\} = [|A|]$. Given a permutation $\sigma = a_{i_1}, a_{i_2}, \ldots, a_{i_{|A|}}$ of A, $\sigma^{-1}(j)$ denotes the element a_{i_j} for each $j \in [|A|]$. For a string β, $\beta(j)$ denotes the jth character in β.

Chapter 3

Cycle Packing and cycle/path covering

In this chapter first we look at ARC-DISJOINT CYCLE PACKING problem in tournaments parameterized by the solution size k. We prove a bound (on FVS) for tournaments similar to that of Erdős-Pósa using which we design a linear kernel for the above problem as well as give a $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ FPTalgorithm. Next, we extend these ideas to a superclass of graphs that contain tournaments, known as α-bounded digraphs and provide polynomial kernels for every constant value of α. We also study VERTEX-DISJOINT CYCLE PACKING problem from the perspective of a structural parameter - *proper interval deletion set*. We combine color-coding, greedy strategy and dynamic programming to design an FPTalgorithm running in time $2^{\mathcal{O}(k\log k)}n^{\mathcal{O}(1)}$. In the last section we focus on CYCLE/PATH COVERING parameterized by *proper interval deletion set* and obtain an FPTalgorithm running in time $2^{\mathcal{O}(k\log k)}n^{\mathcal{O}(1)}$.

3.1 Introduction

In this chapter our focus is on CYCLE PACKING and CYCLE/PATH COVERING problems. Since the publication of the classic Erdős-Pósa Theorem, CYCLE PACKING has received significant scientific attention in various algorithmic realms. In particular, VERTEX-

DISJOINT CYCLE PACKING in undirected graphs is one of the first problems studied in the framework of parameterized complexity. The Erdős-Pósa Theorem states that there exists a function $f(r) = \mathcal{O}(k \log k)$ such that for each non-negative integer k, every undirected graph either contains k vertex-disjoint cycles or has a feedback vertex set consisting of $f(k)$ vertices, using which one can easily design an FPTalgorithm for CYCLE PACKING problem. Although VERTEX-DISJOINT CYCLE PACKING in undirected graphs is FPTwith respect to the solution size [11, 71], it has no polynomial kernel unless NP $\subseteq$ coNP/poly [14]. In contrast, EDGE-DISJOINT CYCLE PACKING in undirected graphs admits a kernel with $\mathcal{O}(k \log k)$ vertices (and is therefore FPT). On directed graphs both these problems are equivalent and W[1]-hard [65, 85]. Therefore, studying these problems on a subclass of directed graphs is a natural direction of research. We focus on tournaments, which form a mathematically rich subclass of directed graphs with interesting structural and algorithmic properties. A tournament is a directed graph in which there is a single arc between every pair of distinct vertices. Any tournament that has a cycle also has a triangle [6]. Therefore, if a tournament has k vertex-disjoint cycles, then it also has k vertex-disjoint triangles. Thus, VERTEX-DISJOINT CYCLE PACKING in tournaments is just packing vertex-disjoint triangles. This problem is NP-hard [8]. A straight forward application of color coding [5] shows that this problem is in FPT. A kernel with $\mathcal{O}(k^2)$ vertices is an immediate consequence of the quadratic element kernel known for 3-SET PACKING [1]. Recently, a kernel with $\mathcal{O}(k^{1.5})$ vertices was shown for this problem using interesting variants and generalizations of the popular *expansion lemma* [67]. It is easy to verify that a tournament that has k arc-disjoint cycles need not necessarily have k arc-disjoint triangles. This observation hints that packing arc-disjoint cycles could be significantly harder than packing vertex-disjoint cycles. This is the starting point of our study. We give a bound analogous to Erdős-Pósa using which we design a linear kernel. This in turn gives us an FPTalgorithm.

Independence number plays a crucial role in solving the problem on tournaments. So a natural question one can ask is there a relationship between the independence number and a polynomial size kernel? Can we extend the polynomial kernel on tournaments to the

class of α-bounded digraphs that contains tournaments? Formally, for any integer $\alpha \geq 1$, the class of α-bounded digraphs, denoted by D_α, is defined as follows-
$D_\alpha = \{D : D$ is a digraph and the maximum size of an independent set in D is at most α $\}$.

Interested in this question we study the ARC DISJOINT CYCLE PACKING problem on D_α. Notice that an α-bounded digraph is a directed graph where the graph induced on any $\alpha + 1$ vertices has at least one arc. First, we prove a theorem analogous to the Erdős-Pósa Theorem to bound the feedback vertex set size of the input graph. Next we find an approximate feedback vertex set as well as state the notions and results of a *cut-preserving set* [70]. Working with the feedback vertex set, we give an algorithm to find the desired polynomial kernel.

The CYCLE PACKING problelms we have considered so far are with respect to the standard parameterization- number of disjoint cycles. The fixed-parameter tractability of CYCLE PACKING follows from the Robertson-Seymour theorem. In 1994, Bodlaender showed that CYCLE PACKING can be solved in $\mathcal{O}^*(2^{\mathcal{O}(k^2)})$ time [11]. The treewidth (tw) of a graph is not larger than the size of its feedback vertex set, and that a naive dynamic programming scheme solves CYCLE PACKING in $\mathcal{O}^*(2^{\mathcal{O}(tw \log tw)})$ time (see, e.g., [25]). Thus, the existence of an $\mathcal{O}^*(2^{\mathcal{O}(k \log^2 k)})$ time algorithm can be viewed as a direct consequence of the Erdös-Pósa Theorem. Recently, Lokshtanov et al. [71] obtained an algorithm with running time $\mathcal{O}^*(2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})})$ for CYCLE PACKING, improving upon the classical consequence of the Erdös-Pósa Theorem. However, in the next set of problems we focus on a *structural parameter* rather than the *solution size*.

A family of structural parameters that have gained significant attention recently are the size of modulators families of graphs. Let F be a family of graphs. Given a graph G and a set $S \subseteq V(G)$, we say that S is an F-modulator if $G \setminus S$ is in F. For example, if F is the family of independent sets, forests, bipartite graphs, interval graphs and chordal graphs, then the modulator corresponds to a vertex cover, feedback vertex set, odd cycle transversal, interval deletion set and chordal deletion set, respectively. The size of S is also called the vertex deletion distance to F. In CYCLE PACKING first Bodlaender et al. [13] obtained polynomial kernels with respect to the size of vertex cover, the vertex-deletion distance to a cluster graph and the maximum leaf number. Since CYCLE PACKING is solvable in $\mathcal{O}^*(2^{\mathcal{O}(tw \log tw)})$ time on graphs of treewidth tw, we have that CYCLE PACKING is FPTparameterized by vertex cover size, feedback vertex set size, pathwidth, and vertex-deletion distance to graphs of constant treewidth. We try to contribute some results to this line of work. However, the status of the problem when parameterized by the vertex-deletion distance to interval graphs or chordal graphs has not yet been studied. Cycle Packing is NP-complete on chordal graphs [46] and thus we cannot hope to have an algorithm with running time $n^{f(t)}$, where t is the size of the modulator to chordal graphs, unless $P = NP$. On the other hand, the classical complexity status of CYCLE PACKING on interval graphs is not known. That is, we do not know whether CYCLE PACKING admits a polynomial time algorithm on interval graphs. A natural graph class that is a subset of the class of interval graphs is the one of proper interval graphs (also known as indifference graphs and unit interval graphs in the literature). A graph is a proper interval graph if its vertices can be assigned to intervals such that there is an edge between two vertices if and only if their corresponding intervals have non-empty intersection. Further, this set of intervals should satisfy the property that no interval properly contains another. It is well known that *CyclePacking* can be solved in polynomial time on proper interval graphs [77]. This is the starting point of our work.

Just as chordal graphs have clique-tree decomposition structure, proper interval graphs have clique-path decomposition structure [58]. We combine color coding, greedy strategy and multi layered dynamic programming to obtain an FPT algorithm. We additionally assume that the proper interval deletion set T is part of the input. This assumption is reasonable as given a graph G and an integer t, there is an algorithm that, in $\mathcal{O}^*(6^t)$ time, outputs a proper interval deletion set of size at most t (if one exists) [18, 87].

The next set of problems (HAMILTONICITY) that we consider are parameterized by the same structural parameter- *proper interval deletion set*. In CYCLE/PATH COVERING, the task is to find minimum number of vertex disjoint cycles/paths that together cover all the vertices. These problems are generalizations of Hamiltonian Cycle/Path problems and are NP-hard. There has been an extensive study of structural parameters for problems related to PATH COVER and CYCLE COVER such as CYCLE PACKING, LONGEST PATH and LONGEST CYCLE [13, 64]. We explore the monotonicity properties of paths in proper interval graphs to design dynamic programming algorithms that solve the above mentioned problems. We show that PATH COVER and CYCLE COVER parameterized by the size of a proper interval deletion set are FPT. By parameterizing PATH COVER and CYCLE COVER with respect to the size of a proper interval deletion set as parameter, we attempt to understand the complexity of the problem on almost proper interval graphs. Recently, Chaplick et al. [19] obtained polynomial kernels and compression algorithms for PATH COVER and CYCLE COVER parameterized by a different measure of similarity with proper interval graphs. Our FPT algorithms also add to this study of structural parameterizations for these classical problems.

3.2 ARC-DISJOINT CYCLE PACKING **in tournaments (ACT)**

ARC-DISJOINT CYCLE PACKING **Parameter:** k

Input: A graph G, a proper interval deletion set T of G and a positive integer k.

Question: Does there exist k pairwise arc-disjoint cycles in G?

An interesting consequence of Erdős-Pósa Theorem is that it leads to an FPT algorithm for VERTEX-DISJOINT CYCLE PACKING running in time $\mathcal{O}^\star(2^{\mathcal{O}(k\log^2 k)})$ (see [71] for more details). Analogous to these results, we prove an Erdős-Pósa type theorem for tournaments and show that it leads to an $\mathcal{O}^\star(2^{\mathcal{O}(k\log k)})$ time algorithm and a linear vertex kernel for ARC-DISJOINT CYCLE PACKING problem in tournaments (ACT).

3.2.1 An Erdős-Pósa Type Theorem

In this subsection, we show certain interesting combinatorial results on arc-disjoint cycles in tournaments.

Theorem 1. *Let k and r be positive integers such that $r \leq k$. A tournament T contains a set of r arc-disjoint cycles if and only if T contains a set of r arc-disjoint cycles each of length at most* $2k+1$.

Proof. The reverse direction of the claim holds trivially. Let us now prove the forward direction. Let $\mathscr{C}$ be a set of r arc-disjoint cycles in T that minimizes $\sum_{C \in \mathscr{C}} |C|$. If every cycle in $\mathscr{C}$ is a triangle, then the claim trivially holds. Otherwise, let C be a longest cycle in $\mathscr{C}$ and let ℓ denote its length. Let v_i, v_j be a pair of non-consecutive vertices in C. Then, either $v_iv_j \in A(T)$ or $v_jv_i \in A(T)$. In any case, the arc e between v_i and v_j along with $A(C)$ forms a cycle C' of length less than ℓ with $A(C') \setminus \{e\} \subset A(C)$. By our choice of $\mathscr{C}$, this implies that e is an arc in some other cycle $\widehat{C} \in \mathscr{C}$. This property is true for the arc between any pair of non-consecutive vertices in C. C can have a maximum of $\binom{\ell}{2} - \ell$ many internal arcs and each of them must belong to some other cycle in in some other cycle $\widehat{C} \in \mathscr{C}$. Therefore, we have $\binom{\ell}{2} - \ell \leq \ell(k-1)$ leading to $\ell \leq 2k+1$. □

This result essentially shows that it suffices to determine the existence of k arc-disjoint cycles in T each of length at most $2k+1$ in order to determine if (T,k) is an yes-instance of ACT. This immediately leads to a quadratic Erdős-Pósa bound. That is, for every non-negative integer k, every tournament T either contains k arc-disjoint cycles or has an FAS of size $\mathscr{O}(k^2)$. Next, we strengthen this result to arrive at a linear bound.

We will use the following lemma known from [23] in the process*. For a digraph D, let $\Lambda(D)$ denote the number of non-adjacent pairs of vertices in D. That is, $\Lambda(D)$ is the number of pairs u, v of vertices of D such that neither $uv \in A(D)$ nor $vu \in A(D)$. Recall

*The authors would like to thank F. Havet for pointing out that Lemma 3 was a consequence of a result of [23], as well for an improvement of the constant in Theorem 2.

that for a digraph D, $\text{minfas}(D)$ denotes the size of a minimum FAS of D.

Lemma 3. [23] *Let D be a triangle-free digraph in which for every pair u, v of distinct vertices, at most one of uv or vu is in $A(D)$. Then, we can compute an FAS of size at most $\Lambda(D)$ in polynomial time.*

This leads to the following main result of this section.

Theorem 2. *For every non-negative integer k, every tournament T either contains k arc-disjoint triangles or has an FAS of size at most $5(k-1)$ that can be obtained in polynomial time.*

Proof. Let $\mathscr{C}$ be a maximal set of arc-disjoint triangles in T (that can be obtained greedily in polynomial time). If $|\mathscr{C}| \geq k$, then we have the required set of triangles. Otherwise, let D denote the digraph obtained from T by deleting the arcs that are in some triangle in $\mathscr{C}$. Clearly, D has no triangle and $\Lambda(D) \leq 3(k-1)$. Let F be an FAS of D obtained in polynomial time using Lemma 3. Then, we have $|F| \leq 3(k-1)$. Next, consider a topological ordering σ of $D-F$. Each triangle of $\mathscr{C}$ contains at most 2 arcs which are backward in this ordering. If we denote by F' the set of all the arcs of the triangles of $\mathscr{C}$ which are backward in σ, then we have $|F'| \leq 2(k-1)$ and $(D-F)-F'$ is acyclic. Thus $F^* = F \cup F'$ is an FAS of T satisfying $|F^*| \leq 5(k-1)$. □

3.2.2 A Linear Vertex Kernel

Next, we show that ACT has a linear vertex kernel. This kernel is inspired by the linear kernelization described in [9] for FAST and uses Theorem 2. Let T be a tournament on n vertices. First, we apply the following reduction rule.

Reduction Rule 3.2.1. *If a vertex v is not in any cycle, then delete v from T.*

This rule is clearly safe as our goal is to find k cycles and v cannot be in any of them.

To describe our next rule, we need to state a lemma known from [9]. An *interval* is a consecutive set of vertices in a linear representation $(\sigma(T), \overleftarrow{A}(T))$ of a tournament T.

Lemma 4 ([9]). [*] *Let $T = (\sigma(T), \overleftarrow{A}(T))$ be a tournament on which Reduction Rule 3.2.1 is not applicable. If $|V(T)| \geq 2|\overleftarrow{A}(T)|+1$, then there exists a partition $\mathscr{J}$ of $V(T)$ into intervals (that can be computed in polynomial time) such that there are $|\overleftarrow{A}(T) \cap E| > 0$ arc-disjoint cycles using only arcs in E where E denotes the set of arcs in T with endpoints in different intervals.*

Our reduction rule that is based on this lemma is as follows.

Reduction Rule 3.2.2. *Let $T = (\sigma(T), \overleftarrow{A}(T))$ be a tournament on which Reduction Rule 3.2.1 is not applicable. Let $\mathscr{J}$ be a partition of $V(T)$ into intervals satisfying the properties specified in Lemma 4. Reverse all arcs in $\overleftarrow{A}(T) \cap E$ and decrease k by $|\overleftarrow{A}(T) \cap E|$ where E denotes the set of arcs in T with endpoints in different intervals.*

Lemma 5. *Reduction Rule 3.2.2 is safe.*

Proof. Let T' be the tournament obtained from T by reversing all arcs in $\overleftarrow{A}(T) \cap E$. Suppose T' has $k - |\overleftarrow{A}(T) \cap E|$ arc-disjoint cycles. Then, it is guaranteed that each such cycle is completely contained in an interval. This is due to the fact that T' has no backward arc with endpoints in different intervals. Indeed, if a cycle in T' uses a forward (backward) arc with endpoints in different intervals, then it also uses a back (forward) arc with endpoints in different intervals. It follows that for each arc $uv \in E$, neither uv nor vu is used in these $k - |\overleftarrow{A}(T) \cap E|$ cycles. Hence, these $k - |\overleftarrow{A}(T) \cap E|$ cycles in T' are also cycles in T. Then, we can add a set of $|\overleftarrow{A}(T) \cap E|$ cycles obtained from the second property of Lemma 4 to these $k - |\overleftarrow{A}(T) \cap E|$ cycles to get k cycles in T. Conversely, consider a set of k cycles in T. As argued earlier, we know that the number of cycles that have an arc that is in E is at most $|\overleftarrow{A}(T) \cap E|$. The remaining cycles (at least $k - |\overleftarrow{A}(T) \cap E|$ of them) do not contain any

[*]Lemma 4 is Lemma 3.9 of [9] that has been rephrased to avoid the use of several definitions and terminology introduced in [9].

arc that is in E, in particular, they do not contain any arc from $\overleftarrow{A}(T) \cap E$. Therefore, these cycles are also cycles in T'. □

Thus, we have the following result.

Theorem 3. ACT *admits a kernel with $\mathcal{O}(k)$ vertices.*

Proof. Let (T,k) denote the instance obtained from the input instance by applying Reduction Rule 3.2.1 exhaustively. From Theorem 2, we know that either T has k arc-disjoint triangles or has an FAS of size at most $5(k-1)$ that can be obtained in polynomial time. In the first case, we return a trivial yes-instance of constant size as the kernel. In the second case, let F be the FAS of size at most $5(k-1)$ of T. Let $(\sigma(T), \overleftarrow{A}(T))$ be the linear representation of T where $\sigma(T)$ is a topological ordering of the vertices of the directed acyclic graph $T-F$. As $V(T-F) = V(T)$, $|\overleftarrow{A}(T)| \leq 5(k-1)$. If $|V(T)| \geq 10k-9$, then from Lemma 4, there is a partition of $V(T)$ into intervals with the specified properties. Therefore, Reduction Rule 3.2.2 is applicable (and the parameter drops by at least 1). When we obtain an instance where neither of the Reduction Rules 3.2.1 and 3.2.2 is applicable, it follows that the tournament in that instance has at most $10k$ vertices. □

3.2.3 An FPT Algorithm

Finally, we show that ACT can be solved in $\mathcal{O}^\star(2^{\mathcal{O}(k \log k)})$ time. The idea is to reduce the problem to the following ARC-DISJOINT PATHS problem in directed acyclic graphs: given a digraph D on n vertices and k ordered pairs $(s_1,t_1),\ldots,(s_k,t_k)$ of vertices of D, do there exist arc-disjoint paths $P_1,\ldots,P_k$ in D such that P_i is a path from s_i to t_i for each $i \in [k]$? On directed acyclic graphs, ARC-DISJOINT PATHS is known to be NP-complete [36], W[1]-hard [85] with respect to k as parameter and solvable in $n^{\mathcal{O}(k)}$ time [42]. Despite its fixed-parameter intractability, we will show that we can use the $n^{\mathcal{O}(k)}$ algorithm and Theorems 2 and 3 to describe an FPT algorithm for ACT.

Theorem 4. ACT *can be solved in* $\mathcal{O}^\star(2^{\mathcal{O}(k \log k)})$ *time.*

Proof. Consider an instance (T,k) of ACT. Using Theorem 3, we obtain a kernel $\mathscr{I} = (\widehat{T},\widehat{k})$ such that $\widehat{T}$ has $\mathcal{O}(k)$ vertices. Further, $\widehat{k} \leq k$. By definition, (T,k) is an yes-instance if and only if $(\widehat{T},\widehat{k})$ is an yes-instance. Using Theorem 2, we know that $\widehat{T}$ either contains $\widehat{k}$ arc-disjoint triangles or has an FAS of size at most $5(\widehat{k}-1)$ that can be obtained in polynomial time. If Theorem 2 returns a set of $\widehat{k}$ arc-disjoint triangles in $\widehat{T}$, then we declare that (T,k) is an yes-instance.

Otherwise, let $\widehat{F}$ be the FAS of size at most $5(\widehat{k}-1)$ returned by Theorem 2. Let D denote the (acyclic) digraph obtained from $\widehat{T}$ by deleting $\widehat{F}$. Observe that D has $\mathcal{O}(k)$ vertices. Suppose $\widehat{T}$ has a set $\mathscr{C} = \{C_1,\ldots,C_{\widehat{k}}\}$ of $\widehat{k}$ arc-disjoint cycles. For each $C \in \mathscr{C}$, we know that $A(C) \cap \widehat{F} \neq \emptyset$ as $\widehat{F}$ is an FAS of $\widehat{T}$. We can guess that subset F of $\widehat{F}$ such that $F = \widehat{F} \cap A(\mathscr{C})$. Then, for each cycle $C_i \in \mathscr{C}$, we can guess the arcs F_i from F that it contains and also the order π_i in which they appear. This information is captured as a partition $\mathscr{F}$ of F into $\widehat{k}$ sets, F_1 to $F_{\widehat{k}}$ and the set $\{\pi_1,\ldots,\pi_{\widehat{k}}\}$ of permutations where π_i is a permutation of F_i for each $i \in [\widehat{k}]$. Any cycle C_i that has $F_i \subseteq F$ contains a (v,x)-path between every pair (u,v), (x,y) of consecutive arcs of F_i with arcs from $A(D)$. That is, there is a path from $\mathrm{h}(\pi_i^{-1}(j))$ and $\mathrm{t}(\pi_i^{-1}((j+1) \mod |F_i|))$ with arcs from D for each $j \in [|F_i|]$. The total number of such paths in these $\widehat{k}$ cycles is $\mathcal{O}(|F|)$ and the arcs of these paths are contained in D which is a (simple) directed acyclic graph.

The number of choices for F is $2^{|\widehat{F}|}$ and the number of choices for a partition $\mathscr{F} = \{F_1,\ldots,F_{\widehat{k}}\}$ of F and a set $X = \{\pi_1,\ldots,\pi_{\widehat{k}}\}$ of permutations is $2^{\mathcal{O}(|\widehat{F}|\log|\widehat{F}|)}$. Once such a choice is made, the problem of finding $\widehat{k}$ arc-disjoint cycles in $\widehat{T}$ reduces to the problem of finding $\widehat{k}$ arc-disjoint cycles $\mathscr{C} = \{C_1,\ldots,C_{\widehat{k}}\}$ in $\widehat{T}$ such that for each $1 \leq i \leq \widehat{k}$ and for each $1 \leq j \leq |F_i|$, C_i has a path P_{ij} between $\mathrm{h}(\pi_i^{-1}(j))$ and $\mathrm{t}(\pi_i^{-1}((j+1) \mod |F_i|))$ with arcs from $D = \widehat{T} - \widehat{F}$. This problem is essentially finding $r = \mathcal{O}(|\widehat{F}|)$ arc-disjoint paths in D and can be solved in $|V(D)|^{\mathcal{O}(r)}$ time using the algorithm in [42]. Therefore, the overall running time of the algorithm is $\mathcal{O}^\star(2^{\mathcal{O}(k \log k)})$ as $|V(D)| = \mathcal{O}(k)$ and $r = \mathcal{O}(k)$. $\square$

3.3 Arc Disjoint Cycle Packing in α-bounded digraphs

In this section we will design a polynomial size kernel for the Arc disjoint cycle packing problem in α-bounded digraphs. We make use of the independence of vertices in α-bounded digraphs to design a polynomial kernel. Towards that, in Subsection 3.3.1, we prove a theorem analogous to the Erdős-Pósa Theorem to bound the feedback vertex set size. In Subsection 3.3.2, we find an approximate feedback vertex set as well as state the notions and results of a *cut-preserving set* [70]. Working with the feedback vertex set, in Subsection 3.3.3 we give an algorithm to obtain the desired polynomial kernel.

3.3.1 An Erdős-Pósa type theorem for α-bounded digraphs

Here we show that there exists a function $f(r) = \mathscr{O}(2\alpha^2 r^2)$ such that for each non-negative integer r, every digraph $G \in \mathscr{D}_\alpha$ either contains r vertex disjoint cycles or has a directed feedback vertex set consisting of $f(r)$ vertices. We start by showing a density lemma about digraphs in $\mathscr{D}_\alpha$ and then use it to obtain the desired result.

Lemma 6. *Any α-bounded digraph G on n vertices has at least $n^2/2\alpha^2$ arcs.*

Proof. Let $S \subseteq V(G)$ be any set of size $\alpha+1$. Then by definition all the vertices in S can not be independent in G i.e. there must be at least one arc between some two vertices in S. Let's call this arc a witness for S (if there are more than one arcs, pick any arbitrary arc as a witness). The graph G has a total of $\binom{n}{\alpha+1}$ many vertex sets of size $\alpha+1$. And each of them must have a witness arc. Hence there are at least $\binom{n}{\alpha+1}$ witness arcs (not necessarily different arcs). But any arc xy can witness at most $\binom{n-2}{\alpha-1}$ sets of size $\alpha+1$, since x and y

are forced to be present in the set. This implies

$$
\begin{aligned}
|E(G)|\binom{n-2}{\alpha-1} &\geq \text{no. of witnesses} \geq \binom{n}{\alpha+1} \\
\implies |E(G)| &\geq \frac{\binom{n}{\alpha+1}}{\binom{n-2}{\alpha-1}} \geq (n^2/2\alpha^2)
\end{aligned} \tag{3.1}
$$

Hence G has at least $\frac{n^2}{2\alpha^2}$ many arcs. □ □

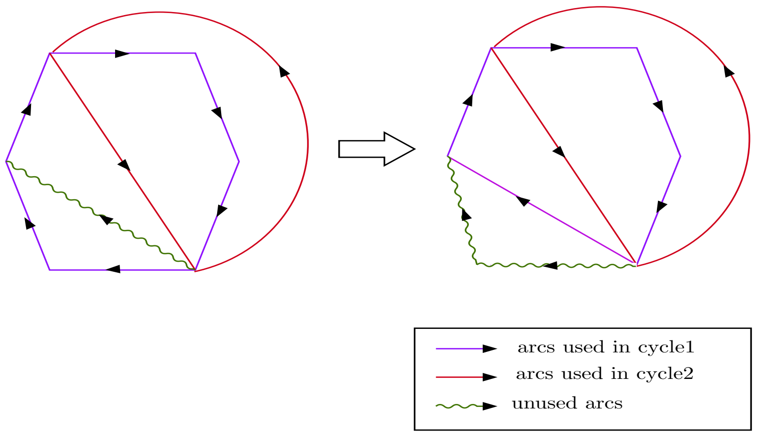


Fig 3.1: Replacement procedure to get a *nice* collection of cycles

Now we are ready to present our main result of this subsection.

Theorem 5. *Any α-bounded digraph G that does not have k arc disjoint cycles, has a feedback vertex set (FVS) of size at most $2\alpha^2k^2$. Furthermore, there exists a maximum size family of arc disjoint cycles, where each cycle has length at most $2\alpha^2k$.*

Proof. Suppose the graph G has a maximum of k' many arc disjoint cycles where $k' < k$. Let $\mathscr{C} = \{C_1, \ldots, C_{k'}\}$ be a *nice* collection of k' arc disjoint cycles i.e. there is no other set of k' arc disjoint cycles, which has less arcs than $\mathscr{C}$. Suppose C_i is the longest cycle in $\mathscr{C}$ with length l. We know that any induced subgraph of an α-bounded digraph is also α-bounded. Hence from Lemma 6, it follows that $G[V(C_i)]$ has at least $\frac{l^2}{2\alpha^2}$ edges. So cycle C_i has $(\frac{l^2}{2\alpha^2} - l)$ many internal chords as the only arcs that are not chords are the l edges used by the cycle.

Let the cycle be $C_i = (v_1 \to v_2 \to v_3 \ldots \to v_l \to v_1)$. We try to find a replacement cycle C^* in the following manner. Let $v_i v_j$ be any internal chord of C_i. If $i < j$, then $C^* = (v_1 \to \ldots \to v_i \to v_j \to v_{j+1}, \ldots \to v_l \to v_1)$, otherwise $C^* = (v_j \to v_{j+1} \ldots \to v_i \to v_j)$. Now clearly C^* has length strictly smaller than the length of C_i. But $\mathscr{C}$ was a nice solution. The only reason we can not replace the cycle C_i in $\mathscr{C}$ with C^* is because there must be some other cycle C_j such that $\{(v_i, v_j)\} \in E(C_j)$. Since the longest cycle in $\mathscr{C}$ has size l, the total number of edges used by all other cycles in $\mathscr{C}$ is at most $l(k'-1)$. But all the internal chords inside cycle C_i must be used by other cycles (otherwise $\mathscr{C}$ is not a nice collection of cycles). This implies

$$\begin{aligned} &\frac{l^2}{2\alpha^2} - l \leq l(k'-1) \\ \implies\ & l \leq 2\alpha^2 k' \leq 2\alpha^2 k \end{aligned} \tag{3.2}$$

So there is a feedback vertex set $F' = \bigcup_{i=1}^{k'} V(C_i)$ of size at most $2\alpha^2 k^2$ and in the nice collection $\mathscr{C}$ each cycle has length at most $2\alpha^2 k$. In Section 3.3.2, in fact we improve the bound on size of FVS to $\alpha^2 k^2$ when the graph does not have k arc disjoint cycles. □ □

3.3.2 Algorithm to find an FVS

Next we make the proof of Theorem 5 algorithmic. We will use this directed feedback vertex set to design our kernel. We first state the algorithm in the box.

```
Algorithm 1($G \in D_\alpha, k$)

1. Initialize $F = \emptyset$, $G' = G$, $i = 0$.
2. Run Breadth First Search on each vertex of $G'$. Find the shortest cycle $C_i$. If
   $G'$ is acyclic goto Step 5.
3. If the shortest cycle $C_i$ has length more than $2\alpha^2(k-i)$, return $G$ has $k$ arc
   disjoint cycles.
4. $F = F \cup V(C_i)$, $G' = G[V(G') \setminus V(C_i)]$, $i = i+1$.
5. If $i = k$, return $G$ has $k$ arc disjoint cycles.
6. If $G'$ is acyclic, return FVS $F$, otherwise goto Step 2.
```

From Theorem 5, any α-bounded digraph H which not does not have k arc disjoint cycles, has a cycle of size at most $2\alpha^2 k$. In other words any α-bounded digraph H, where the smallest cycle has length more than $2\alpha^2 k$, has at least k arc disjoint cycles. In the ith iteration of Step 2 of the above algorithm, we have already found i arc disjoint cycles $\{C_1, \ldots C_i\}$. If the shortest cycle in $G[V(G) \setminus \bigcup_{j=1}^{i} V(C_j)]$ has length more than $2\alpha^2(k-i)$, then it has at least $k-i$ arc disjoint cycles. But then the original graph G definitely has k arc disjoint cycles. This proves the correctness of the 3rd Step of the algorithm. In Step 5, if we can get k vertex disjoint cycles then of course G also has k arc disjoint cycles. If the graph G does not have k arc disjoint cycles, then in any ith iteration, the graph G' does not have more than $k-i$ arc disjoint cycles. So it also has a cycle of length at most $2\alpha^2(k-i)$ from Theorem 5. This, together with the fact that the algorithm runs at most k many iterations implies the FVS F that we get at the end of the algorithm has a maximum size of $2\alpha^2((k-1)+(k-2)+\ldots+1) \leq \alpha^2 k^2$. Since each step of the algorithm takes poly(n) time and each step is also executed at most poly(n) times, the entire algorithm runs in poly(n) time.

A cut-preserving set

Definition 7 ([70]). *For any digraph G, a positive integer k and $x,y \in V(G)$, we say that $\mathscr{Z} \subseteq V(G)$ is a k-cut-preserving set for (x,y) in G, if the following properties hold. Let $L = V(G) \setminus \mathscr{Z}$. For any path P from x to y in G, there exist paths $P_1, P_2, ..., P_e$ and a set $L_1, ..., L_e$ where each L_i is a list of k paths with the following properties:*

- *For every $i \in [e]$, P_i is a subpath of P from s_i to t_i.*
- *The P_is are internally disjoint and contain all vertices in $P \cap L$ as inner vertices.*
- *for every $i \in [e]$, L_i is a set of k vertex disjoint paths from s_i to t_i using only vertices of $\mathscr{Z}$.*
- *Replacing in P each P_i by one of the paths in L_i yields a path of $\mathscr{Z}$ from x to y.*

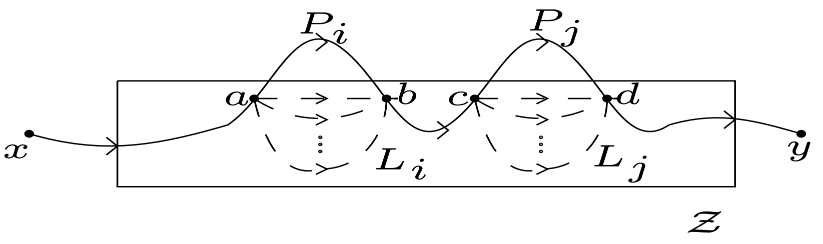


Fig 3.2: A cut-preserving set $\mathscr{Z}$ for (x,y)

Figure 2 gives an easy depiction of a cut-preserving set. Lochet et al. [70] have recently shown that in an α-bounded digraph G, from any x to y a k-cut-preserving set of size $f(k,\alpha)$ can be found in polynomial time where $f(k,\alpha) = (22k^5)^{4^\alpha}$.

Theorem 6 ([70]). *Let D be an α-bounded acyclic digraph and $x,y \in V(D)$ such that any (x,y)-vertex- cut in D has size at least $k+1$. Then one can, in polynomial time, compute a k-cut-preserving (x,y) in D of size at most $(22k^5)^{4^\alpha}$. Moreover in polynomial time one can obtain $k+1$ vertex disjoint paths from u to v where each path has length at most $2\alpha+1$.*

3.3.3 Algorithm to compute the kernel

In this subsection we gather everything and design our kernel. We start with the description and then prove its correctness and finally give the size bound.

Algorithm 2

1. Initialize TCL (total cycle length)=$2\alpha^2k^2$, $i = 1$, $Kernel = F$, ℓ (max cycle length)$= 2\alpha^2k$.
2. Let σ' be an ordered set on $F \times F$ such that $\sigma' = ((u_1,u_1)(u_1,u_2),(u_1,u_3),\dots(u_{|F|},u_{|F|}))$. Now fix an ordered set σ of size $k|F|^2$ such that, $\forall j \in [k|F|^2]$, $\sigma(j) = \sigma'(\lceil j/k \rceil)$. In σ each element from σ' is repeated k consecutive times i.e. σ=$((u_1,u_1),(u_1,u_1),\dots(u_1,u_2),(u_1,u_2),\dots(u_{|F|},u_{|F|}),(u_{|F|},u_{|F|}))$.
3. Intitialize $i = 1$.
4. Let $\sigma(i) = (x,y)$. Get a ($\mathscr{K} = TCL+\ell(2\alpha+1)$)-cut-preserving set $\mathscr{Z}$ from x to y of size $f(\mathscr{K},\alpha)$. $Kernel = Kernel \cup \mathscr{Z}$, $TCL = \mathscr{K}$.
5. if i=$k|F|^2$ (all elements of σ are exhausted) stop, else $i = i+1$, go to Step 4.
6. Return $Kernel$.

3.3.4 Running time and kernel size analysis

Step 1 and Step 2 takes $n^{\mathscr{O}(1)}$ time. Step 4 and 5 of the algorithm are executed $k|F|^2$ many times and in each iteration, we spend at most $n^{\mathscr{O}(1)}$ time to get the cut-preserving set $\mathscr{Z}$ [70]. Hence our algorithm runs in polynomial time.

Next we determine the final size of $Kernel$ set. Let $g(i)$ and $TCL(i)$ denote the size of $Kernel$ and TCL in the ith iteration. We get the following recurrence equations from the above algorithm and solve them to get a bound on the *Kernel* size:

1. $g(i) = g(i-1) + f(TCL(i-1) + \ell(2\alpha+1)), \alpha)$.
2. $TCL(i) = TCL(i-1) + \ell(2\alpha+1)$.
3. $g(0) = \alpha^2k^2, TCL(0) = 2\alpha^2k^2$.

Notice that TCL and g are strictly increasing functions. We compute their maximum values below.

$$\begin{aligned} TCL(i) &= 2\alpha^2k^2 + i(\ell(2\alpha+1)) \\ &= 2\alpha^2k^2 + i(2\alpha^2k(2\alpha+1)) \end{aligned} \tag{3.3}$$

$$\begin{aligned} TCL(k|F|^2) &= TCL(\alpha^4k^5) \\ &= 2\alpha^2k^2 + \alpha^4k^5(2\alpha^2k(2\alpha+1)) \\ &\leq 5\alpha^7k^6. \end{aligned} \tag{3.4}$$

Since g and f both are increasing functions, $g(k|F|^2)$ has the maximum value.

$$\begin{aligned} g(k|F|^2) &\leq g(0) + k|F|^2.f(TCL(k|F|^2), \alpha) \\ &\leq \alpha^2k^2 + \alpha^4k^5f(5\alpha^7k^6, \alpha) \\ &\leq 2\alpha^4k^5f(5\alpha^7k^6, \alpha) \\ &\leq 2\alpha^4k^5(110\alpha^{35}k^{30})^{4^\alpha} \\ &= P_1(k,\alpha) \end{aligned} \tag{3.5}$$

Hence g admits a maximum value (*Kernel* size) of $P_1(k,\alpha)$. Now if we can show that $G[Kernel]$ has k arc disjoint cycles iff G has k arc disjoint cycles, then the problem indeed admits a kernel of size $P_1(k,\alpha)$.

3.3.5 Correctness of the algorithm

In the forward direction, if $G[Kernel]$ has k arc disjoint cycles, then the graph G also has the same k arc disjoint cycles. We will use induction to prove the reverse direction. Suppose G has k arc disjoint cycles. Using the arguments in Theorem 5, we know there is a nice set of k arc disjoint cycles $\mathscr{C} = \{C_i\}_{i=1}^{k}$, where the total number of arcs in $\mathscr{C}$ is at most $2\alpha^2 k^2$ and any cycle C_i has length $l_i (\leq 2\alpha^2 k)$. Let us define the notions of segment and subsegment for our proof. Segments for $\mathscr{C}$ are the paths from x to y (where $x, y \in F$) and there is no other vertices of F in between. Subsegments are the maximal subpaths of segments that lie outside the *Kernel*. Refer to Fig.3.

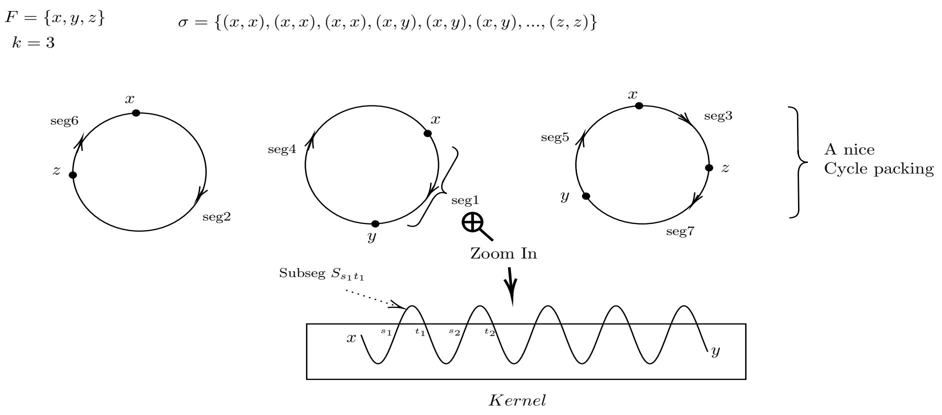


Fig 3.3: Segments and subsegments

Let $C_i = (u_{i_1} \rightarrow \mathscr{S}_{i_{12}} \rightarrow u_{i_2} \rightarrow \mathscr{S}_{i_{23}} \rightarrow u_{i_3} \dots u_{i_{k_i}} \rightarrow \mathscr{S}_{i_{k_i(k_i+1)}} \rightarrow u_{i_{(k_i+1)}})$, where u_{i_j} vertices are from the modulator F and $u_{i_1} = u_{i_{(k_i+1)}}$. $\mathscr{S}_{i_{j(j+1)}}$ denotes the segment of cycle C_i from the vertex u_{i_j} to $u_{i_{(j+1)}}$. In each induction step we will replace a segment between two modulator vertices with another segment that is completely contained inside *Kernel*, while maintaining the property that even after the replacement, the cycles in $\mathscr{C}$ are still arc disjoint. Notice each cycle in a nice collection can have at most $|F|$ many segments and hence there are at most $\alpha^2 k^3$ many segments in $\mathscr{C}$. If we are able to replace all the segments (a maximum of $\alpha^2 k^3$ many), then *Kernel* actually will have k arc disjoint cycles contained in it.

Let $S_i = \bigcup_{j=1}^{k_i} (u_{i_j}, u_{i_{(j+1)}})$ and $S = \uplus_{i=1}^{k} S_i$. Observe that a pair (x,y) can appear at most k many times in S (If any vertex appears more than once in a cycle, we can get another cycle that uses a strict subset of the arcs used by the original cycle). We use induction below to prove the correctness. Our induction properties will be as follows:

1. In qth step, we are able to replace the segments between the first q pairs of vertices of σ that appear in S, with segments that are completely contained inside *Kernel*.

2. There exists a collection of arc disjoint cycles with the replaced segments whose total length is not more than $TCL(q)$.

Induction step 1 (first segment replacement from *Kernel*)

Let $\mathscr{C}$ be a nice collection of arc disjoint cycles and $(x = u_{i_j}, y = u_{i_{(j+1)}})$ be the pair of vertices in S that appears first in σ. We will replace the segment from x to y with a segment completely contained inside *Kernel* if it already isn't, while still keeping it arc disjoint from all other segments. If the segment $\mathscr{S}_{i_{j(j+1)}}$ is completely inside the computed $\mathscr{X}$ in round 1, then it satisfies both the induction properties from Theorem 5. Otherwise for the pair (x,y), in *Kernel* we have stored enough vertices ($\mathscr{X}$) to get a $(TCL(0) + \ell(2\alpha+1))$-cut-preserving set.

From the cut-preserving set properties we have segment $\mathscr{S}_{i_{j(j+1)}} = P = (x = s_0 \rightarrow \mathscr{S}_{s_0 s_1} \rightarrow s_1 \rightarrow \mathscr{S}_{s_1 t_1} \rightarrow t_1 \rightarrow \mathscr{S}_{t_1 s_2} \ldots s_l \rightarrow \mathscr{S}_{s_l t_l} \rightarrow t_l \rightarrow \mathscr{S}_{t_l t_f} \rightarrow t_f = y)$, where $V(P) \setminus \mathscr{X} = \cup_{j=1}^{l} V(\mathscr{S}_{s_j t_j})$ is the set of vertices from P that are not in *Kernel*. Notice that $l \leq k_i \leq \ell$. And $\mathscr{X}$ is a $(TCL(0) + \ell(2\alpha+1))$-cut-preserving set. Then from Theorem 6 we can get $(TCL(0) + (\ell(2\alpha+1))$ many vertex disjoint paths each with length at most $2\alpha+1$ from s_j to t_j for any j that are completely inside *Kernel*.

First subsegment replacement

Hence from s_1 to t_1 there is a path that is vertex disjoint from $V(\mathscr{C})$, since $|V(\mathscr{C})| = 2\alpha^2 k^2 = TCL(0)$. Let this path be L_1 that has length at most $2\alpha+1$ and is completely inside $\mathscr{X}$. Now replace the subsegment $\mathscr{S}_{s_1t_1}$ with L_1 in P to get a new segment from s_0 to t_f, $P' = (s_0 \to \mathscr{S}_{s_0s_1} \to s_1 \to L_1 \to t_1 \to \mathscr{S}_{t_1s_2} \dots \to t_f)$ and get a new set of arc disjoint cycles by replacing the segment $\mathscr{S}_{i_{j(j+1)}}$ with P' i.e. $\mathscr{C} = \mathscr{C} \setminus \{C_i\} \cup \{C_i = (u_{i_1} \to \mathscr{S}_{i_{12}} \dots u_{i_j} \to P' \to u_{i_{(j+1)}} \dots \to \mathscr{S}_{i_{k_i(k_i+1)}} \to u_{i_{(k_i+1)}})\}$. Now the updated $\mathscr{C}$ has at most $2\alpha^2k^2 + 2\alpha + 1$ many vertices (or arcs) and L_1 is completely contained inside *Kernel*.

Second subsegment replacement

Similarly from s_2 to t_2 there are at least $(TCL(0) + \ell(2\alpha + 1))$ vertex disjoint paths in $\mathscr{X}$, each with length at most $2\alpha + 1$. In the udpated $\mathscr{C}$, there are at most $(TCL(0) + 2\alpha + 1)$ vertices, hence there is a path L_2 from s_2 to t_2 that is vertex disjoint from $V(\mathscr{C})$ and is completely inside $\mathscr{X}$. We replace the subsegment $\mathscr{S}_{s_2t_2}$ with L_2 and get a new path $P' = (s_0 \to \mathscr{S}_{s_0s_1} \to s_1 \to L_1 \to t_1 \to \mathscr{S}_{t_1s_2} \to s_2 \to L_2 \to t_2 \dots \to t_f)$. We also get a new set of arc disjoint cycles $\mathscr{C} = \mathscr{C} \setminus \{C_i\} \cup \{C_i = (u_{i_1} \to \mathscr{S}_{i_{12}} \dots u_{i_j} \to P' \to u_{i_{(j+1)}} \dots \to \mathscr{S}_{i_{k_i(k_i+1)}} \to u_{i_{(k_i+1)}})\}$. The new $\mathscr{C}$ has at most $2\alpha^2k^2 + 2(2\alpha + 1)$ many arcs. Moreover L_1 and L_2 are completely contained inside *Kernel*.

All l subsegments replacement

But $l \leq \ell$, as every cycle in the beginning had length at most $2\alpha^2k$. Hence we will be able to apply the above replacement procedure for all $\mathscr{S}_{s_it_i}$, where $i \leq l$. And get a new segment $P' = (s_0 \to \mathscr{S}_{s_0s_1} \to s_1 \to L_1 \to t_1 \dots \to s_l \to L_l \to t_l \to \mathscr{S}_{t_lt_f} \to t_f)$ which is completely contained inside *Kernel*. Now in C_i replacing the segment from u_{i_j} to $u_{i_{(j+1)}}$ by P' and updating the $\mathscr{C}$, we get a set of arc disjoint cycles where the segment from u_{i_j} to $u_{i_{(j+1)}}$ is completely contained inside *Kernel*. The updated set of arc disjoint cycles $\mathscr{C}$ uses at most $2\alpha^2k^2 + \ell(2\alpha + 1)$ many arcs. This proves the correctness for the first step of the induction.

Let the induction properties hold true for all $j < q$. So we are able to sucsessfully replace(or keep) the segments between the first $q - 1$ pairs of vertices in S that appear in σ with segments completely inside *Kernel*, such that the new collection of arc disjoint cycles with the replaced segments uses at most $TCL(q - 1)$ number of arcs.

Induction step q (qth segment replacement from *Kernel*)

Let $(x' = u_{i'_{j'}}, y' = u_{i'_{(j'+1)}})$ be the qth pair of vertices of σ that appear in S. The $(q-1)$ other pairs from S that appear in σ before (x', y'), their segments have already been replaced in $\mathscr{C}$ in the first $(q-1)$ steps of induction. Let $\mathscr{Z}$ be a $\mathscr{K}(= TCL(q-1) + \ell(2\alpha+1))$-cut-preserving set from x' to y' of size $f(\mathscr{K}, \alpha)$. Let the collection of cycles after the $(q-1)$ replacements be $\mathscr{C} = \{C_i\}_{i=1}^{k}$. Let $C_{i'} = (u_{i'_1} \to \mathscr{S}'_{i'_{12}} \to u_{i'_2} \ldots \to u_{i'_{k_{i'}}} \to \mathscr{S}'_{i'_{k_{i'}(k_{i'}+1)}} \to u_{i'_{(k_{i'}+1)}})$, where $u_{i'_{j'}}$ and $u_{i'_{(j'+1)}}$ vertices are from the modulator F and $u_{i'_1} = u_{i'_{(k_{i'}+1)}}$. The segment of cycle C_i from the vertex $u_{i'_{j'}}$ to $u_{i'_{(j'+1)}}$ is denoted by $\mathscr{S}'_{i'_{j'(j'+1)}}$. For the pair (x', y'), in *Kernel* we have stored enough vertices ($\mathscr{Z}$) to get a $\mathscr{K}$-cut-preserving set.

If the segment $\mathscr{S}'_{i'_{j'(j'+1)}}$ is completely inside $\mathscr{Z}$, we do not need to replace the segment at all and we can move onto the next segments. All the q segments are completely contained in *Kernel* and the total length of all cycles in $\mathscr{C}$ is at most $TCL(q-1) \leq TCL(q)$. This satisfies the induction properties for qth step.

If the segment is not completely contained in $\mathscr{Z}$ then from Theorem 6, we get the segment of the form $\mathscr{S}'_{i'_{j'(j'+1)}} = P = (x' = s'_0 \to \mathscr{S}'_{s'_0 s'_1} \to s'_1 \to \mathscr{S}'_{s'_1 t'_1} \ldots \to \mathscr{S}'_{s'_r t'_r} \to t'_r \to \mathscr{S}'_{t'_r t'_f} \to t'_f = y')$, where $\mathscr{S}'_{s'_i t'_i}$ are the subsegments that are not in $\mathscr{Z}$ i.e. $V(P) \setminus A = \cup_{i=1}^{r} V(\mathscr{S}'_{s'_i t'_i})$. Notice that $r \leq k_{i'} \leq \ell$. But since $\mathscr{Z}$ is a $\mathscr{K}$-cut-preserving set, we can get $\mathscr{K}$ many vertex disjoint paths each with length at most $2\alpha+1$ from s'_j to t'_j for all j.

First subsegment replacement

Hence from s'_1 to t'_1 there is a path that is vertex disjoint from $V(\mathscr{C})$ since $|V(\mathscr{C})| = TCL(q-1)$. Let this path be L'_1 that has length at most $2\alpha+1$ and is completely inside $\mathscr{Z}$. Now replace the subsegment $\mathscr{S}'_{s'_1 t'_1}$ with L'_1 in P to get a new path from s'_0 to t'_f. Let $P' = (s'_0 \to \mathscr{S}'_{s'_0 s'_1} \to s'_1 \to L'_1 \to t'_1 \ldots \to t'_f)$. We get a new set of arc disjoint cycles by replacing the segment $\mathscr{S}_{i'_{j'(j'+1)}}$ with P' i.e. $\mathscr{C} = \mathscr{C} \setminus C_i \cup \{C_i = (u_{i'_1} \to \mathscr{S}'_{i'_{12}} \ldots u_{i'_{j'}} \to P' \to u_{i'_{j'+1}} \ldots \to \mathscr{S}'_{i'_{k_{i'}(k_{i'}+1)}} \to u_{i'_{(k_{i'}+1)}})\}$. The updated $\mathscr{C}$ has at most $(TCL(q-1) + 2\alpha +$

1) many arcs and L'_1 is completely contained inside *Kernel*.

Second subsegment replacement

Similarly from s'_2 to t'_2, there are at least $\mathscr{K}$ vertex disjoint paths in $\mathscr{Z}$ each with length $(2\alpha+1)$. In the updated $\mathscr{C}$, there are at most $TCL(q-1)+(2\alpha+1)$ arcs. Hence there is a subsegment(path) L'_2 from s'_2 to t'_2 that is vertex disjoint from $V(\mathscr{C})$ and completely contained inside $\mathscr{Z}$. We replace the subsegment $\mathscr{S}'_{s'_2t'_2}$ with L'_2 and get a new path from s'_0 to t'_f, $P' = (s'_0 \to \mathscr{S}'_{s'_0s'_1} \to s'_1 \to L'_1 \to t'_1 \to \mathscr{S}'_{t'_1s'_2} \to s'_2 \to L'_2 \to t'_2 \ldots \to t'_f)$. We get a new set of arc disjoint cycles by replacing the segment $\mathscr{C} = \mathscr{C} \setminus \{C_i\} \cup \{C_i = (u_{i'_1} \to \mathscr{S}'_{i'_{12}} \ldots u_{i'_{j'}} \to P' \to u_{i'_{(j'+1)}} \ldots \to u_{i'_{(k_{i'}+1)}})\}$. The new $\mathscr{C}$ has at most $TCL(q-1)+2(2\alpha+1)$ many arcs. Moreover L'_1 and L'_2 are completely contained inside *Kernel*.

All r subsegments replacement

But $r \leq \ell$, as every cycle in the beginning had length at most $2\alpha^2k$. Hence we will be able to apply the above replacement procedure for all $\mathscr{S}'_{s'_it'_i}$, where $i \leq r$. And get a new segment $P' = (s'_0 \to \mathscr{S}'_{s'_0s'_1} \to s'_1 \to L'_1 \to t'_1 \ldots \to s'_r \to L'_r \to t'_r \to \mathscr{S}'_{t'_rt'_f} \to t'_f)$ which is entirely contained inside *Kernel*. Now in $C_{i'}$ replacing the segment from x' to y' by P' and updating $\mathscr{C}$, we get the set of arc disjoint cycles where the segment from $u_{i'_{j'}}$ to $u_{i'_{(j'+1)}}$ is completely contained inside *Kernel*. The updated set of arc disjoint cycles $\mathscr{C}$ uses at most $TCL(q-1)+\ell(2\alpha+1) = TCL(q)$ many arcs. This proves the correctness of the induction.

Theorem 7. ARC DISJOINT CYCLE PACKING *in α-bounded digraphs, when parameterized by the number of cycles k, admits a kernel of size $P_1(k,\alpha)$ where $P_1(k,\alpha) = 2\alpha^4k^5(110\alpha^{35}k^{30})^{4^\alpha}$.*

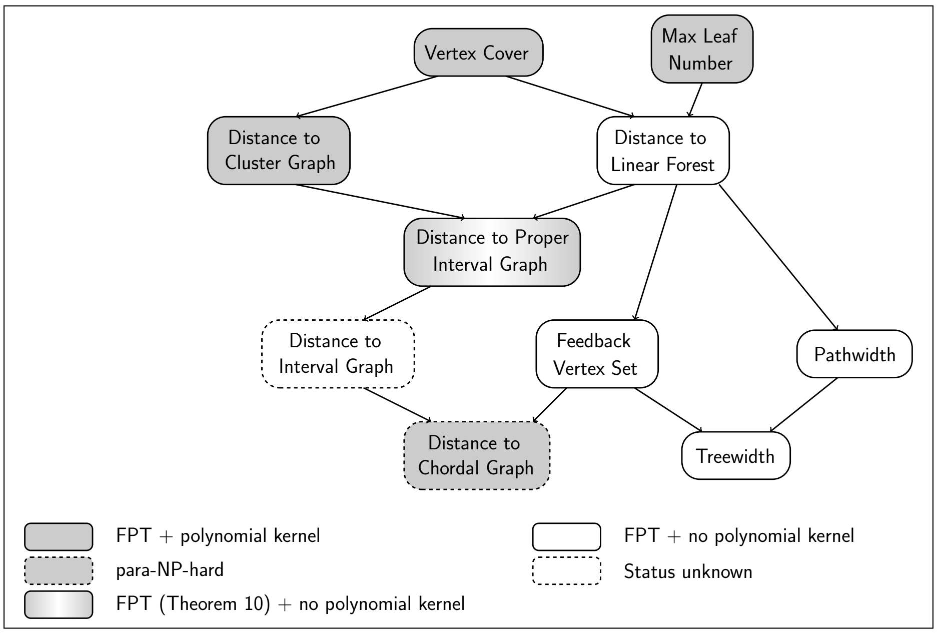

Vertex Cover
Max Leaf Number
Distance to Cluster Graph
Distance to Linear Forest
Distance to Proper Interval Graph
Distance to Interval Graph
Feedback Vertex Set
Pathwidth
Distance to Chordal Graph
Treewidth
FPT + polynomial kernel
para-NP-hard
FPT (Theorem 10) + no polynomial kernel
FPT + no polynomial kernel
Status unknown

Fig 3.4: Structural parameterizations of Cycle Packing. For deletion distance parameters, assume that a modulator is part of the input. The parameter values are the minimum possible for a given graph. An arrow from parameter x to parameter y means that necessarily x $\geq$ y.

3.4 Vertex-Disjoint Cycle Packing parameterized by proper interval deletion set

In this subsection we will focus on VERTEX-DISJOINT CYCLE PACKING parameterized by the structural parameter *proper interval deletion set* and will design an FPTalgorithm running in time $\mathcal{O}^*(2^{\mathcal{O}(t\log t)})$ where t is the size of the deletion set. We state the currently known results on VERTEX-DISJOINT CYCLE PACKING parameterized by different structural parameters in the diagram below. And for this section we will interchangeably use the terms CYCLE PACKING and VERTEX-DISJOINT CYCLE PACKING . We will use r to denote size of the packing and t to denote the proper interval deletion set though out this subsection.

CYCLE PACKING	**Parameter:** $\|T\|$
Input: A graph G, a proper interval deletion set T of G and a positive integer r.	
Question: Does there exist r pairwise vertex-disjoint cycles in G?	

We show that this problem is FPT and this is the main result of this section.

Theorem 8. CYCLE PACKING *parameterized by the size t of a proper interval deletion set can be solved in* $\mathcal{O}^*(2^{\mathcal{O}(t\log t)})$ *time.*

We assume that the proper interval deletion set T is part of the input. This assumption is reasonable as given a graph G and an integer t, there is an algorithm that, in $\mathcal{O}^*(6^t)$ time, outputs a proper interval deletion set of size at most t (if one exists) [18, 87].

Overview of our Algorithm and Techniques. The FPT algorithm combines various ingredients like color coding, greedy strategy and a multi-layered dynamic programming routine. One of the most important properties of proper interval graphs that makes this graph class amenable to elegant polynomial-time algorithms for several classical problems is the existence of *proper interval orderings* [75]. A proper interval ordering is an ordering $v_1,\dots,v_n$ of the vertices of a proper interval graph such that for any two adjacent vertices v_i and v_j, the set $\{v_i,v_{i+1},\dots,v_j\}$ is a clique (a set of pairwise adjacent vertices). This ordering also leads to a partition (called *clique partition*) of the vertices into a sequence of disjoint cliques such that the endpoints of any edge are in the same clique or in consecutive cliques. Our algorithm crucially uses the properties of such partitions and proper interval orderings. We essentially reduce the problem of finding cycles in G to finding appropriate paths in $G-T$ (which is a proper interval graph). Our approach consists of (i) a guessing phase, where we determine important relations between the vertices of T and the rest of the graph $G-T$. This allows us to replace the vertices in T by variables that capture precisely the roles of those vertices; (ii) a coloring phase, which allows us to separate the tasks associated with individual variables, that later allows us to employ a greedy strategy; (iii) a dynamic programming routine over a clique partition that incorporates a greedy strategy using the properties of a proper interval ordering to find an assignment to these variables. An illustrative overview of the algorithm is given in Figure 3.5.

Now, we explain each of the phases in detail. In the first phase, we reduce CYCLE PACKING to multiple instances of a constraint satisfaction problem which we call CON-

STRAINED PATH ASSIGNMENT. For this purpose, we show the existence of a solution with "nice" properties. The properties of such a solution allow us to reduce CYCLE PACKING to an auxiliary problem which is just a constrained variant of CYCLE PACKING. The idea behind this reduction is the following. Any solution consists of cycles of two types; those cycles that are entirely contained in $G-T$ and those cycles that have a vertex from T. The number of cycles that contain a vertex from T is at most $|T|$. We first guess this number ℓ. Then, for each of the ℓ cycles, we guess the vertices from T that it contains and also the order in which they appear. This information is captured as a partition of T into ℓ ordered sets, T_1 to T_ℓ. Any cycle C_i with $T_i \subseteq T$ contains a path between every pair of consecutive vertices of T_i with internal vertices from $G-T$. The total number of such paths in these ℓ cycles is $|T|$. We guess the number of internal vertices of each such path as being zero, one, two or at least three. We further observe that paths with at least three internal vertices can be assumed to satisfy a certain condition regarding their intersections, which we explicitly encode as a constraint. This observation specifically uses the properties of proper interval graphs that distinguishes this graph class from the superclass of interval graphs. Then, the problem boils down to finding a collection of vertex-disjoint paths in $G-T$ satisfying certain constraints.

As these paths are all completely contained in $G-T$, we can delete T from G once the constraints to be satisfied are encoded. For this encoding, we introduce four sets of variables. Type 1 and Type 2 variables together are placeholders for paths of length two. Type 3 variables correspond to paths of length one. These three types of variables should simply be assigned to vertices. Finally, Type 4 variables are placeholders for paths of length at least three, where in practice we only demand such paths to be of length at least two, but still retain the intersections-related constraint. Apart from the length constraints (specified as variable types), the endpoints of the solution paths need to satisfy the adjacency relationship with respect to T. These constraints are captured using appropriate functions from the variables to a collection of subsets of $V(G-T)$. Furthermore, we have to make additional guesses concerning orientations of paths with respect to the clique partition. The task is then to find an assignment to the variables satisfying these constraints that maximizes the maximum number of vertex-disjoint triangles in the remaining (proper interval) graph. This completes the first phase of the algorithm.

In the second phase, we reduce CONSTRAINED PATH ASSIGNMENT to a "colored" variant called COLORFUL CONSTRAINED PATH ASSIGNMENT using color coding. Let S denote the set of variables of Type 1, 2 and 3. Let W denote the set of variables of Type 4. We color the graph with $|S|$ colors and find a solution (which is an assignment of paths to variables in $S \cup W$) satisfying certain color constraints. Such a solution (called *colorful solution*) is one in which any pair of vertices assigned to distinct variables of S have distinct colors. This property is independent of the assignment to variables in W. By looking for only colorful solutions, we not only reduce the search space of solutions but also make the assignment to two variables in S independent of each other. This independence allows us to resort to a greedy strategy in the next phase where we only look for a *canonical solution*. Informally, a canonical solution is a colorful solution that is *aligned* to the left or to the right with respect to the proper interval ordering where we interpret leftmost as "first" and rightmost as "last". The third and final phase is an algorithm to find a canonical solution, if it exists, thereby solving COLORFUL CONSTRAINED PATH ASSIGNMENT. This is indubitably the most technical part of the FPT algorithm that employs a dynamic programming routine incorporating a greedy strategy over the clique partition of $G - T$ to find a suitable assignment to variables in $S \cup W$. We use dynamic programming primarily to assign paths to W and resort to a greedy choice when it comes to assigning paths to S. This greedy strategy is of the flavour "choose the first/last vertex from the vertices belonging to some specific restriction of a color set". This choice naturally follows from the definition of a canonical solution and from the properties of a proper interval ordering (and clique partition).

Kernelization Complexity. There is a kernel lower bound result known for CYCLE PACKING with respect to the solution size r by a reduction from DISJOINT FACTORS [14].

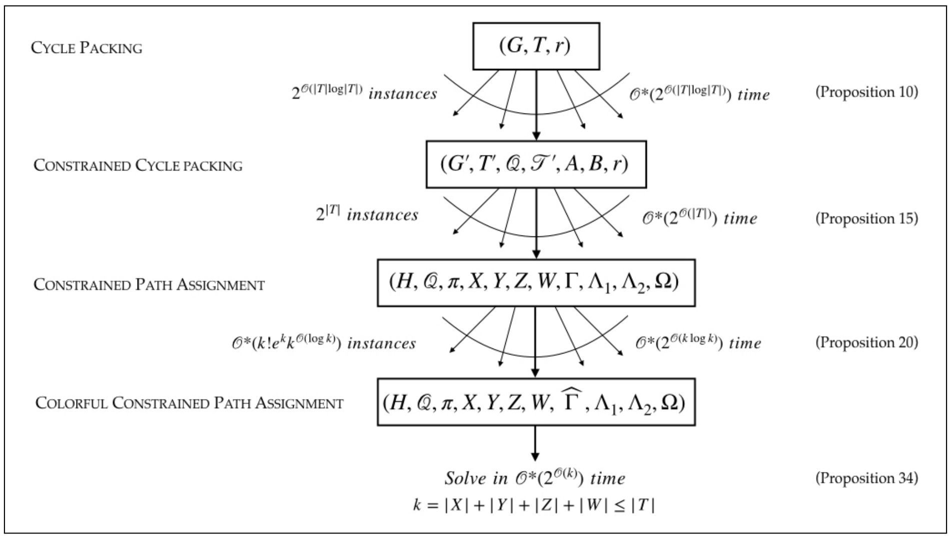

Cycle Packing
(G, T, r)
$2^{\mathcal{O}(|T|\log|T|)}$ instances
$\mathcal{O}^*(2^{\mathcal{O}(|T|\log|T|)})$ time
(Proposition 10)
Constrained Cycle packing
$(G', T', \mathcal{Q}, \mathcal{T}', A, B, r)$
$2^{|T|}$ instances
$\mathcal{O}^*(2^{\mathcal{O}(|T|)})$ time
(Proposition 15)
Constrained Path Assignment
$(H, \mathcal{Q}, \pi, X, Y, Z, W, \Gamma, \Lambda_1, \Lambda_2, \Omega)$
$\mathcal{O}^*(k!e^k k^{\mathcal{O}(\log k)})$ instances
$\mathcal{O}^*(2^{\mathcal{O}(k\log k)})$ time
(Proposition 20)
Colorful Constrained Path Assignment
$(H, \mathcal{Q}, \pi, X, Y, Z, W, \widehat{\Gamma}, \Lambda_1, \Lambda_2, \Omega)$
Solve in $\mathcal{O}^*(2^{\mathcal{O}(k)})$ time
$k = |X| + |Y| + |Z| + |W| \leq |T|$
(Proposition 34)

Fig 3.5: Illustrative Overview of the Algorithm

This reduction constructs a graph that is obtained by adding r vertices adjacent to some vertices of a path. As a path is a proper interval graph, it follows that the graph has a proper interval deletion set of size r. Therefore, it follows that Cycle Packing parameterized by the size of a proper interval deletion set does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly.

Next we will list some properties of proper interval graphs which we are going to use throughout this section.

3.4.1 Set of cycles in proper interval graphs

Throughout the discussion, for a proper interval graph G, we fix an interval representation along with the proper interval ordering π and the clique partition $\mathcal{Q}$ obtained from it. Next, we observe some properties of a collection of cycles in a proper interval graph. As every induced cycle in a proper interval graph is a triangle, the following property holds.

Proposition 8. *Let $\mathcal{C}$ be a set of vertex-disjoint cycles in a proper interval graph G. Then, G has a set $\mathcal{C}'$ of vertex-disjoint triangles with $|\mathcal{C}| = |\mathcal{C}'|$.*

The next property follows from Proposition 111.

Proposition 9. *Let C be a triangle in a proper interval graph G with clique partition $\{Q_1,\dots,Q_q\}$. Then, there is an integer $i \in [q-1]$ such that $C \subseteq Q_i \cup Q_{i+1}$.*

Now, we define the notion of a *nice set of triangles*.

Definition 10. **(Nice set of triangles)** *Let G be a proper interval graph with clique partition $\{Q_1,\dots,Q_q\}$ and proper interval ordering π. A set $\mathscr{C} = \{C_1,\dots,C_r\}$ of vertex-disjoint triangles in G is a nice set of triangles if it satisfies the following properties.*

- **(NT.1)** *For each $i \in [q]$, at most two vertices in Q_i are not present in any triangle of $\mathscr{C}'$.*
- **(NT.2)** *For each $i \in [q-1]$, at most two vertices in Q_i are present in triangles in $\mathscr{C}'$ that have vertices from Q_{i+1}.*
- **(NT.3)** *For each $i \in [q-1]$, at most two vertices in Q_{i+1} are present in triangles in $\mathscr{C}'$ that have vertices from Q_i.*

Lemma 11. *Let G be a proper interval graph with clique partition $\{Q_1,\dots,Q_q\}$ and proper interval ordering π. If G has a maximal set $\mathscr{C}$ of vertex-disjoint triangles, then G has a nice maximal set $\mathscr{C}'$ of triangles with $|\mathscr{C}| \leq |\mathscr{C}'|$ that can be obtained in polynomial time.*

Proof. First, if there is a clique Q_i such that $|Q_i \setminus V(\mathscr{C})| > 2$, then $\mathscr{C}$ is not maximal because any set of three distinct vertices from $Q_i \setminus V(\mathscr{C})$ can be added to $\mathscr{C}$ to get a larger set of vertex-disjoint triangles containing $\mathscr{C}$. Next, suppose there is a clique Q_i that has three vertices a,b,c that are present in triangles of $\mathscr{C}$ that have vertices from Q_{i+1}. Consider the following cases. In this analysis, vertex identifiers with subscripts correspond to vertices in Q_{i+1} and the others correspond to vertices in Q_i. In each of the cases, we identify a set $\mathscr{C}'$ of vertex-disjoint triangles that is at least as large as $\mathscr{C}$ and therefore replaces $\mathscr{C}$. Note that in each of these cases, if $\mathscr{C}$ satisfies (NT.1), then $\mathscr{C}'$ too satisfies (NT.1). Let C_1, C_2 and C_3 be the triangles in $\mathscr{C}$ that contain a, b and c, respectively.

- C_1, C_2 and C_3 are all distinct. From Proposition 9, each of these triangles is of one of the following types.
  - $C_1 = \{a, a_1, a_2\}, C_2 = \{b, b_1, b_2\}$ and $C_3 = \{c, c_1, c_2\}$. Then, $\mathscr{C}' = (\mathscr{C} \setminus \{C_1, C_2, C_3\}) \cup (\{\{a,b,c\}, \{a_1, b_1, c_1\}, \{a_2, b_2, c_2\}\})$.
  - $C_1 = \{a, d, a_1\}, C_2 = \{b, b_1, b_2\}$, and $C_3 = \{c, c_1, c_2\}$. Now, $\mathscr{C}' = (\mathscr{C} \setminus \{C_1, C_2\}) \cup (\{\{a,b,d\}, \{a_1, b_1, b_2\}\})$.
  - $C_1 = \{a, d, a_1\}, C_2 = \{b, e, b_1\}$, and $C_3 = \{c, c_1, c_2\}$. Here, $\mathscr{C}' = (\mathscr{C} \setminus \{C_2, C_3\}) \cup (\{\{b,e,c\}, \{b_1, c_1, c_2\}\})$.
  - $C_1 = \{a, d, a_1\}, C_2 = \{b, e, b_1\}$, and $C_3 = \{c, f, c_1\}$. Now, $\mathscr{C}' = (\mathscr{C} \setminus \{C_1, C_2, C_3\}) \cup (\{\{a,b,c\}, \{a_1, b_1, c_1\}, \{d, e, f\}\})$.
- $C_1 = C_2$. Once again from Proposition 9, each of C_1, C_2 and C_3 is of one of the following types.
  - $C_1 = C_2 = \{a, b, x_1\}$ and $C_3 = \{c, y_1, z_1\}$. Then, $\mathscr{C}' = (\mathscr{C} \setminus \{C_1, C_3\}) \cup (\{\{a,b,c\}, \{x_1, y_1, z_1\}\})$.
  - $C_1 = C_2 = \{a, b, x_1\}$ and $C_3 = \{c, d, y_1\}$. Let z denote the vertex from $X = \{a, b, c, d\}$ that maximizes $\pi(z)$. As a is adjacent to x_1 and $\pi(x_1) > \pi(z) \geq \pi(a)$, it follows that z is adjacent to x_1 as well from Proposition 110. Similarly, as c is adjacent to y_1 and $\pi(y_1) \geq \pi(z) \geq \pi(c)$, z too is adjacent to y_1. Then, $\mathscr{C}' = (\mathscr{C} \setminus \{C_1, C_3\}) \cup (\{X \setminus \{z\}, \{z, x_1, y_1\}\})$.

Repeating this replacement procedure, we obtain a set $\mathscr{C}$ of triangles that satisfies (NT.1) and (NT.2). Finally, suppose there is a clique Q_{i+1} that has more than two vertices that are present in triangles of $\mathscr{C}$ that have vertices from Q_i. Then, there are four vertices $v, x, y, z \in Q_{i+1}$ that are in triangles $C_1 = \{a, x, v\}$ and $C_2 = \{b, y, z\}$ where $a, b \in Q_i$ by (NT.2). Without loss of generality let $\pi(a) < \pi(b)$. Let u denote the vertex from $X = \{v, x, y, z\}$ that minimizes $\pi(u)$. As a is adjacent to x and $\pi(a) < \pi(u) \leq \pi(x)$, it follows that a is adjacent to u as well from Proposition 110. Similarly, since a is adjacent to u and

$\pi(a) < \pi(b) < \pi(u)$, b is adjacent to u too. Then, $\mathscr{C}' = \mathscr{C} \setminus \{C_1, C_2\} \cup (\{X \setminus \{u\}, \{u, a, b\}\})$. Further, $\mathscr{C}'$ satisfies (NT.1) and (NT.2) as $\mathscr{C}$ satisfies them. Repeating this as long as possible ensures that (NT.3) is satisfied. □

3.4.2 Reduction to Constrained Path Assignment

In this section, we show how to reduce an instance $\mathscr{I} = (G, T, r)$ of CYCLE PACKING to multiple instances of CONSTRAINED PATH ASSIGNMENT. We first define the notion of a *nice set of cycles*.

Definition 12. (Nice set of cycles) *Let G be a graph and $T \subseteq V(G)$ be a proper interval deletion set. Let H be the proper interval graph $G - T$ with clique partition $\mathscr{Q} = \{Q_1, \ldots, Q_q\}$. A set $\mathscr{C} = \{C_1, \ldots, C_r\}$ of vertex-disjoint cycles in G is a nice set of cycles if it satisfies the following properties.*

- **(NC.1)** *For each $i \in [r]$, if $V(C_i) \subseteq V(H)$, then C_i is a triangle.*
- **(NC.2)** *For each $i \in [r]$ with $V(C_i) \cap T \neq \emptyset$, if C_i has a path P with $V(P) \subseteq V(H)$, then for each $j \in [q]$, $|V(P) \cap Q_j| \leq 2$.*
- **(NC.3)** *For each $j \in [q]$, let ℓ_j denote the number of maximal paths $P_1, \ldots, P_{\ell_j}$, each of length at least three, such that for each $i \in [\ell_j]$ there exists a cycle $C_{k_i} \in \mathscr{C}$ such that $V(P_i) \subseteq V(C_{k_i}) \cap V(H)$, and $V(P_i) \cap Q_j \neq \emptyset$. Then, $\ell_j \leq 14$.*

Lemma 13. *Let $T \subseteq V(G)$ be a proper interval deletion set of a graph G. Let H be the proper interval graph $G - T$ with clique partition $\mathscr{Q} = \{Q_1, \ldots, Q_q\}$. Given a set $\mathscr{C}$ of r vertex-disjoint cycles, a nice set $\mathscr{C}^*$ of r vertex-disjoint cycles can be obtained in polynomial time.*

Proof. From Proposition 8, $\mathscr{C}$ satisfies (NC.1) as H is a proper interval graph. Suppose there is a cycle $C \in \mathscr{C}$ and a clique $Q \in \mathscr{Q}$ with $|V(C) \cap Q| \geq 3$. Let a, b, c be three distinct vertices in $V(C) \cap Q$. Then, $\mathscr{C}' = (\mathscr{C} \setminus \{C\}) \cup \{a,b,c\}$ is another set of r vertex-disjoint cycles in G. By applying this procedure as long as possible, we obtain a set $\mathscr{C}'$ of r vertex-disjoint cycles in G that satisfies (NC.2). Observe that if $\mathscr{C}$ satisfies (NC.1), then $\mathscr{C}'$ too satisfies (NC.1). Consider a clique Q_j where $j \in [q]$. Let ℓ_j denote the number of maximal paths $P_1, \ldots, P_{\ell_j}$, each of length at least three, such that for all $i \in [\ell_j]$ there exists a cycle $C_i \in \mathscr{C}$ such that $V(P_i) \subseteq V(C_i) \cap V(H)$, and $V(P_i) \cap Q_j \neq \emptyset$. Let $\mathscr{P} = \{P_1, \ldots, P_{\ell_j}\}$. From Proposition 111, each path in $\mathscr{P}$ is in at least one of the following subsets.

- $\mathscr{P}_1 = \{P \in \mathscr{P} : |V(P) \cap Q_j| = 1, |V(P) \cap Q_{j-1}| \geq 1, |V(P) \cap Q_{j+1}| \geq 1\}$.
- $\mathscr{P}_2 = \{P \in \mathscr{P} : |V(P) \cap Q_j| = 1, |V(P) \cap Q_{j-1}| \geq 1, |V(P) \cap Q_{j-2}| \geq 1\}$.
- $\mathscr{P}_3 = \{P \in \mathscr{P} : |V(P) \cap Q_j| = 1, |V(P) \cap Q_{j+1}| \geq 1, |V(P) \cap Q_{j+2}| \geq 1\}$.
- $\mathscr{P}_4 = \{P \in \mathscr{P} : |V(P) \cap Q_j| = 2, |V(P) \cap Q_{j-1}| \geq 1\}$.
- $\mathscr{P}_5 = \{P \in \mathscr{P} : |V(P) \cap Q_j| = 2, |V(P) \cap Q_{j+1}| \geq 1\}$.
- $\mathscr{P}_6 = \{P \in \mathscr{P} : |V(P) \cap Q_j| = 1, |V(P) \cap Q_{j-1}| = 2\}$.
- $\mathscr{P}_7 = \{P \in \mathscr{P} : |V(P) \cap Q_j| = 1, |V(P) \cap Q_{j+1}| = 2\}$.

By the pigeonhole principle, if $|\mathscr{P}| > 14$, then there is an index $i \in [7]$ such that $|\mathscr{P}_i| \geq 3$. Let P_1, P_2 and P_3 be three distinct paths in $\mathscr{P}_i$. Let C_1, C_2 and C_3 be the cycles (not necessarily distinct) in $\mathscr{C}'$ that contain P_1, P_2 and P_3, respectively. We claim that C_1, C_2 and C_3 can be replaced by three triangles in $\mathscr{C}'$. Consider the case when $i = 1$. Let $\{a_1, b_1, c_1\} \subseteq V(C_1)$, $\{a_2, b_2, c_2\} \subseteq V(C_2)$ and $\{a_3, b_3, c_3\} \subseteq V(C_3)$ such that $A = \{a_1, a_2, a_3\} \subseteq Q_j$, $B = \{b_1, b_2, b_3\} \subseteq Q_{j-1}$ and $C = \{c_1, c_2, c_3\} \subseteq Q_{j+1}$. Then, $(\mathscr{C}' \setminus \{C_1, C_2, C_3\}) \cup \{A, B, C\}$ is also a set of at least r vertex-disjoint cycles in G. A similar replacement works for other values of i. An illustration of the same is shown in Figure 3.6.

This procedure can be applied for each $j \in [q]$. When this replacement can no longer be made, we have a set $\mathscr{C}^*$ of r vertex-disjoint cycles that satisfies (NC.3). Further, if $\mathscr{C}'$

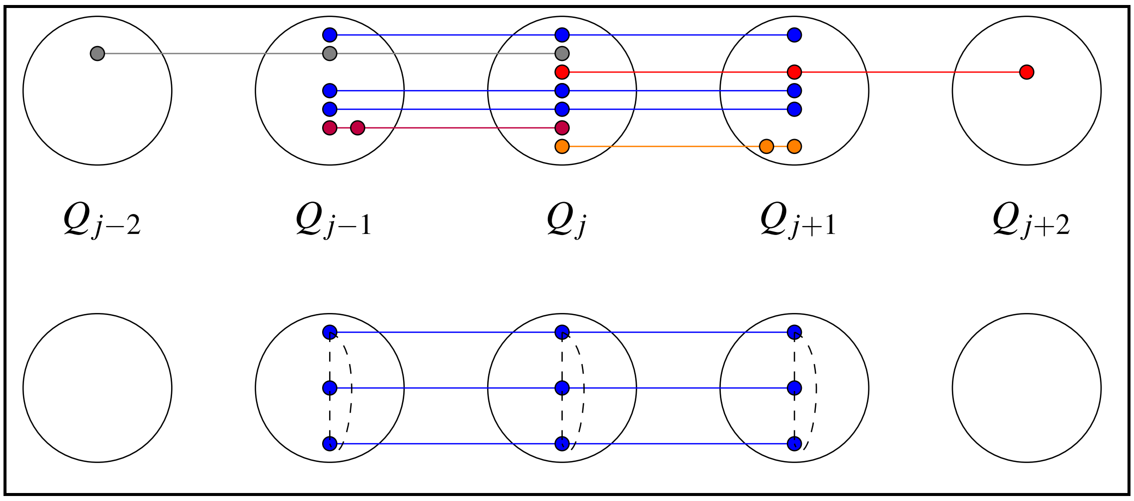


Fig 3.6: Illustration of the replacement of the cycles containing the paths highlighted in blue by the set of three triangles highlighted using dashed edges.

satisfies (NC.1) and (NC.2), then $\mathscr{C}^*$ also satisfies them. In other words, $\mathscr{C}^*$ is a nice set of cycles. $\square$

Next, using Lemma 13, we reduce $\mathscr{I}$ to multiple instances of CONSTRAINED CYCLE PACKING defined as follows.

CONSTRAINED CYCLE PACKING **Parameter:** $|T|$

Input: A graph G, a proper interval deletion set T of G, a clique partition $\{Q_1,\dots,Q_q\}$ of $H = G - T$, a partition $\mathscr{T}$ of T into sets $T_1,\dots,T_\ell$, a set $\{\sigma_1,\dots,\sigma_\ell\}$ of permutations where σ_i is a permutation of T_i for each $i \in [\ell]$, a set $\{\gamma_i \in \{0,1,2,3\}^{|T_i|} : i \in [\ell]\}$ of strings and a positive integer r.

Question: Do there exist r vertex-disjoint cycles $\mathscr{C}$ in G such that the following properties hold?

- There are ℓ cycles $C_1, C_2, \dots, C_\ell$ in $\mathscr{C}$ such that for each $i \in [\ell]$, $V(C_i) \cap T = T_i$.
- For each $i \in [\ell]$ and $j \in [|T_i|]$ with $\gamma_i(j) \in \{0,1,2\}$, C_i has a path P_{ij} between $\sigma_i^{-1}(j)$ and $\sigma_i^{-1}(1 + (j \bmod |T_i|))$ with $\gamma_i(j)$ internal vertices all of which are in H.
- For each $i \in [\ell]$ and $j \in [|T_i|]$ with $\gamma_i(j) = 3$, C_i has a path P_{ij} between $\sigma_i^{-1}(j)$ and $\sigma_i^{-1}(1 + (j \bmod |T_i|))$ with at least 2 internal vertices all of which are in H. Further, $|V(P_{ij}) \cap Q_p| \leq 2$ for each $p \in [q]$.
- For each $p \in [q]$, the number of P_{ij} paths such that $\gamma_i(j) = 3$, $V(P_{ij}) \cap Q_p \neq \emptyset$, $i \in [\ell]$ and $j \in [|T_i|]$ is at most 14.

The idea behind this reduction is the following. Any solution for $\mathscr{I}$ consists of cycles of two types; those cycles that are entirely contained in $G - T$ and those cycles that have a vertex from T. The total number of cycles is r and the number of cycles that contain a vertex from T is $\ell \leq |T|$. We first guess this number ℓ. Then, for each of the ℓ cycles C_i with $i \in [\ell]$, we guess the vertices T_i from T that it contains and also the order σ_i in which they appear. This information is captured as a partition of T into ℓ sets, T_1 to T_ℓ and the set $\{\sigma_1,\dots,\sigma_\ell\}$ of permutations where σ_i is a permutation of T_i for each $i \in [\ell]$. Any cycle C_i with $T_i \subseteq T$ contains a path between every pair of consecutive vertices of T_i with internal vertices from $G - T$. Note that, in this context, $\sigma_i^{-1}(|T_i|)$ and $\sigma_i^{-1}(1)$ are treated as consecutive vertices. Therefore, there is a path from $\sigma_i^{-1}(j)$ and $\sigma_i^{-1}(1 + (j \bmod |T_i|))$

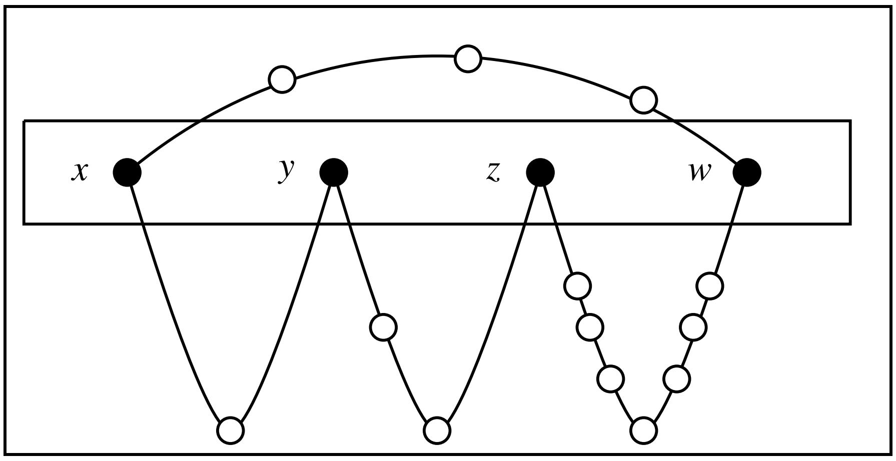


Fig 3.7: An illustration of the cycle C_i containing $T_i \subseteq T$ where $T_i = \{x,y,z,w\}$ in the order mentioned. In a particular guess, the number of internal vertices between x and y is one and between y and z is two. Also, the number of internal vertices between z and w and between w and x is at least three.

with internal vertices from $G - T$ for each $j \in [|T_i|]$. The total number of such paths in these ℓ cycles is $|T|$. We guess the number of internal vertices of each such path as being zero, one, two or at least three. This information is encoded in the set $\{\gamma_i : i \in [\ell]\}$. An example is illustrated in Figure 3.7.

Paths with zero internal vertices are trivial to handle. Paths of length at least three can be assumed to satisfy a certain condition (given by Lemma 13) regarding their intersections. However, in practice, we only demand such paths to be of length at least two, but still retain the intersections-related constraint.

Proposition 14. *There is an algorithm that, given an instance $\mathscr{I} = (G,T,r)$ of* CYCLE PACKING, *runs in $\mathcal{O}^*(2^{\mathcal{O}(|T|\log|T|)})$ time and returns a set of $2^{\mathcal{O}(|T|\log|T|)}$ instances of* CONSTRAINED CYCLE PACKING *such that $\mathscr{I}$ is a yes-instance if and only if at least one of the returned instances is a yes-instance.*

Proof. Let $\mathscr{Q}$ denote the clique partition $\{Q_1,\ldots,Q_q\}$ of $G - T$. Let $\mathscr{F}$ be the set of all subsets of T. For each $S \in \mathscr{F}$, let $\mathscr{P}_S$ be the set of all partitions of S into non-empty sets. Let α denote a tuple $(S,\mathscr{T},A,B)$ with the following interpretation.

- $S \subseteq T$ and $\mathscr{T} = \{T_1,\ldots,T_\ell\}$ is an element of $\mathscr{P}_S$.

- $A = \{\sigma_1, \dots, \sigma_\ell\}$ where σ_i is a permutation of T_i for each $i \in [\ell]$.
- $B = \{\gamma_i \in \{0,1,2,3\}^{|T_i|} : i \in [\ell]\}$ where γ_i satisfies the following properties for each $i \in [\ell]$.
  - if $|T_i| = 1$, then $\gamma_i \in \{2,3\}$.
  - if $|T_i| = 2$, then either $\gamma_i(1) \in \{1,2,3\}$ or $\gamma_i(2) \in \{1,2,3\}$.
- For each $i \in [\ell]$ and for each $j \in [|T_i|]$, if $\gamma_i(j) = 0$, then $\sigma_i^{-1}(j)$ and $\sigma_i^{-1}(1 + (j \bmod |T_i|))$ are adjacent.

For each such tuple $\alpha = (S, \mathscr{T}, A, B)$, we create an instance $\mathscr{I}_\alpha = (G - T', S, \mathscr{Q}, \mathscr{T}, A, B, r)$ of CONSTRAINED CYCLE PACKING where $T' = T \setminus S$. Observe that we create $2^{\mathcal{O}(|T| \log |T|)}$ instances and each of these instances can be obtained in polynomial time. We claim that $\mathscr{I}$ is a yes-instance if and only if there is at least one tuple $\alpha = (S, \mathscr{T}, A, B)$ such that $\mathscr{I}_\alpha$ is a yes-instance. The backward direction of the claim follows immediately as any solution for $\mathscr{I}_\alpha$ is also a solution for $\mathscr{I}$ by the definitions of the corresponding problems. For the forward direction, consider a set $\mathscr{C} = \{C_1, \dots, C_r\}$ of r vertex-disjoint cycles of G. From Lemma 13, we can assume $\mathscr{C}$ to be a nice set of cycles. Let S denote the subset of vertices of T that are present in some cycle of $\mathscr{C}$. Let ℓ denote the number of cycles of $\mathscr{C}$ that contain a vertex from T (or equivalently from S). Without loss of generality, let $C_1, \dots, C_\ell$ be these cycles. Define $T_i = V(C_i) \cap S$ for each $i \in [\ell]$. Then, $\{T_1, \dots, T_\ell\}$ is a partition of S.

For each $i \in [\ell]$, let σ_i denote the order of occurrence of vertices (which is unique up to cyclic shifts) of T_i in C_i. For each $i \in [\ell]$, let $\eta_i(j)$ denote the number of internal vertices of the path between $\sigma_i^{-1}(j)$ and $\sigma_i^{-1}(1 + (j \bmod |T_i|))$ in C_i where $j \in [|T_i|]$. For each $i \in [\ell]$, define the string γ_i character-wise as follows: for each $j \in [|T_i|]$, $\gamma_i(j) = \eta_i(j)$ if $\eta_i(j) \in \{0,1,2\}$; otherwise $\gamma_i(j) = 3$. Observe that if for some $i \in [\ell]$ we have $|T_i| = 1$, then $\gamma_i(1) \in \{2,3\}$. Similarly, if for some $i \in [\ell]$ we have $|T_i| = 2$, then either $\gamma_i(1) \in \{1,2,3\}$ or $\gamma_i(2) \in \{1,2,3\}$. By setting $A = \{\sigma_i : i \in [\ell]\}$, $B = \{\gamma_i : i \in [\ell]\}$ and $\alpha = (S, \mathscr{T}, A, B)$, it follows that $\mathscr{C}$ is a solution for $\mathscr{I}_\alpha$. This completes the proof of the claim. □

Finally, we reduce an instance $\mathscr{I}$ of CONSTRAINED CYCLE PACKING to multiple instances of CONSTRAINED PATH ASSIGNMENT. We need to state some preliminaries before defining this problem. Observe that in CONSTRAINED CYCLE PACKING, the task is to find a collection of vertex-disjoint paths satisfying certain constraints in a proper interval graph. Let H be this proper interval graph with proper interval ordering π and clique partition $\mathscr{Q} = \{Q_1, \ldots, Q_q\}$. We use sets, W, X, Y and Z, of variables corresponding to placeholders for paths that we wish to find in H. For each path P_{ij} with $\gamma_i(j) = 1$, there is a variable in Z, and for each path P_{ij} with $\gamma_i(j) = 2$, there is a variable in X and a variable in Y. Finally, for each path P_{ij} with $\gamma_i(j) = 3$, there is a variable in W. The variables in $X \cup Y \cup Z$ have to be assigned to vertices (paths of length one) and the variables in W have to be assigned to paths of length at least two. Apart from the length constraints, the solution paths of $\mathscr{I}$ need to satisfy the adjacency relationship with respect to the proper interval deletion set T. These constraints are captured using functions Γ, Λ_1, Λ_2, Ω. For example, if we need to find a path between $t_1 \in T$ and $t_2 \in T$ with exactly two internal vertices both of which are from H, then we have a variable $x \in X$ and a variable $y \in Y$ corresponding to this constraint. Further, $\Gamma(x)$ is $N(t_1) \cap V(H)$ and $\Gamma(y)$ is $N(t_2) \cap V(H)$ with the interpretation that in any valid assignment g of vertices to $X \cup Y \cup Z$, $g(x) \in \Gamma(x)$ and $g(y) \in \Gamma(y)$. Moreover, we require $g(x)$ and $g(\Omega(x))$ to be adjacent. To encode this constraint, we set $\Omega(x) = y$. Similarly, if we need to find a path between $t_1 \in T$ and $t_2 \in T$ with exactly 1 internal vertex which is from H, then we have a variable $z \in Z$ and set $\Gamma(z)$ to be $N(t_1) \cap N(t_2) \cap V(H)$ with the interpretation that we require $g(z) \in \Gamma(z)$. Finally, if we need to find a path between $t_1 \in T$ and $t_2 \in T$ with at least two internal vertices all of which are from H, then we have a variable $w \in W$ corresponding to this constraint. Further, $\Lambda_1(w)$ is $N(t_1) \cap V(H)$ and $\Lambda_2(w)$ is $N(t_2) \cap V(H)$. The interpretation is that the path assigned to w has to *start* at a vertex in $\Lambda_1(w)$ and *end* at a vertex in $\Lambda_2(w)$. Observe that the notions of starting vertex and ending vertex of a path are derived from the proper interval ordering π of H. It might very well be the case that w can only be assigned to a path that starts at a vertex in $\Lambda_2(w)$ and ends at a vertex in $\Lambda_1(w)$. Therefore, such possibilities have to be handled while creating multiple instances of CONSTRAINED PATH ASSIGNMENT corresponding to $\mathscr{I}$. Observe that from Lemma 13, it suffices to assign each variable w in W to a path P of length at least two such that $|V(P) \cap Q_i| \leq 2$ for each $i \in [q]$. Using functions Γ, Ω, Λ_1 and Λ_2, we ensure that the vertices/paths assigned to $X \cup Y \cup Z \cup W$ indeed form cycles (when combined with T) with the required properties specified by $\mathscr{I}$.

CONSTRAINED PATH ASSIGNMENT is an optimization problem with an objective to find an assignment (of paths) to the placeholders W, X, Y and Z such that the maximum number of vertex disjoint triangles in a subgraph of H induced by the remaining vertices is maximized. First, we state what an instance to this problem looks like.

Definition 15. (Instance of CONSTRAINED PATH ASSIGNMENT) *An instance* $\mathscr{J} = (H, \mathscr{Q}, \pi, X, Y, Z, W, \Gamma, \Lambda_1, \Lambda_2, \Omega)$ *of* CONSTRAINED PATH ASSIGNMENT *consists of the following components.*

- *A proper interval graph H with clique partition $\mathscr{Q} = \{Q_1, Q_2, \ldots Q_q\}$ and proper interval ordering π.*
- *Sets X, Y, Z, W of variables.*
- *Functions $\Gamma : X \cup Y \cup Z \to \mathscr{R}$, $\Lambda_1 : W \to \mathscr{S}$, $\Lambda_2 : W \to \mathscr{S}'$ where $\mathscr{R}$, $\mathscr{S}$, $\mathscr{S}'$ are collections of subsets of $V(H)$.*
- *A bijection $\Omega : X \to Y$.*

As we would use CONSTRAINED PATH ASSIGNMENT only to solve CONSTRAINED CYCLE PACKING, we will require the paths that are a part of the solution to have a certain structure. For this purpose, we define the set paths(H) to denote the set of paths $P = v_1, \ldots, v_j$ in H that have the following properties.

- $|V(P)| \geq 2$.
- $|V(P) \cap Q_i| \leq 2$ for each $i \in [q]$.

- Let Q_{q_i} be the clique in $\mathscr{Q}$ containing v_i and Q_{q_j} be the clique in $\mathscr{Q}$ containing v_j. Then, $q_i \leq q_j$.

Next, we will define the desired properties of a solution to an instance $\mathscr{J}$ of CONSTRAINED PATH ASSIGNMENT. Given two sets A and B, functions $h : A \to \text{paths}(H)$ and $g : B \to V(H)$ are said to have *disjoint images* if for each pair of elements $a \in A$, $b \in B$, we have $g(b) \notin V(h(a))$.

Definition 16. (Feasible solution) *Given an instance* $\mathscr{J} = (H, \mathscr{Q}, \pi, X, Y, Z, W, \Gamma, \Lambda_1, \Lambda_2, \Omega)$ *of* CONSTRAINED PATH ASSIGNMENT*, a pair* (h, g) *of injective functions* $h : W \to \text{paths}(H)$ *and* $g : X \cup Y \cup Z \to V(H)$ *with disjoint images is said to be a feasible solution of* $\mathscr{J}$ *if the following properties hold.*

(i) For each $x \in X$*,* $y \in Y$ *and* $z \in Z$*,* $g(x) \in \Gamma(x)$*,* $g(y) \in \Gamma(y)$ *and* $g(z) \in \Gamma(z)$*.*

(ii) For each $x \in X$*,* $g(x)g(\Omega(x)) \in E(H)$*.*

(iii) For each $w \in W$*,* $h(w)$ *is a path starting at* $u \in Q_i \cap \Lambda_1(w)$ *and ending at* $v \in Q_j \cap \Lambda_2(w)$ *for some* $i, j \in [q]$ *with* $i \leq j$*.*

(iv) For each pair w, w' *of distinct variables in* W*,* $V(h(w)) \cap V(h(w')) = \emptyset$*.*

(v) For each $i \in [q]$*,* $|\{w \in W : V(h(w)) \cap Q_i \neq \emptyset\}| \leq 14$*.*

Informally, a feasible solution (h, g) has the following properties.

- g assigns a vertex of H to each variable in $X \cup Y \cup Z$ such that the assignment restricted to certain pairs $x \in X$ and $y \in Y$ is a path of length two.

- h assigns a path from $\text{paths}(H)$ to each variable in W satisfying certain intersection constraints propagated from the CONSTRAINED CYCLE PACKING instance $\mathscr{I}$.

Further, we do require h and g to be injective as our interest is in finding vertex-disjoint paths. We also require the images of g and h to be disjoint in the previously mentioned sense. Now, we are ready to formally define the CONSTRAINED PATH ASSIGNMENT problem.

CONSTRAINED PATH ASSIGNMENT **Parameter:** $|X|+|Y|+|Z|+|W|$

Input: A proper interval graph H with clique partition $\mathscr{Q} = \{Q_1, Q_2, \dots Q_q\}$ and proper interval ordering π, sets X, Y, Z, W of variables, functions $\Gamma : X \cup Y \cup Z \to \mathscr{R}$, $\Lambda_1 : W \to \mathscr{S}$, $\Lambda_2 : W \to \mathscr{S}'$ where $\mathscr{R}$, $\mathscr{S}$, $\mathscr{S}'$ are collections of subsets of $V(H)$ and a bijection $\Omega : X \to Y$.

Output: A feasible solution (h,g) that maximizes the number of vertex-disjoint triangles in $H - (V(\text{img}(h)) \cup \text{img}(g))$.

Definition 17. (Value of a feasible solution) *The value of a feasible solution (h,g) of an instance $\mathscr{J}$ of* CONSTRAINED PATH ASSIGNMENT*, denoted by $val_{\mathscr{J}}((h,g))$, is the maximum number of vertex-disjoint triangles in $H - (V(\text{img}(h)) \cup \text{img}(g))$.*

Definition 18. (Optimum solution) *An optimum solution of an instance $\mathscr{J}$ of* CONSTRAINED PATH ASSIGNMENT *is a feasible solution (h,g) that maximizes $val_{\mathscr{J}}((h,g))$ over all feasible solutions of $\mathscr{J}$ and its value $val_{\mathscr{J}}((h,g))$ is denoted by $opt(\mathscr{J})$.*

We omit the subscript in the notation for value if the instance under consideration is implicit. Now, we show the following reduction.

Proposition 19. *There is an algorithm that, given an instance $\mathscr{I} = (G, T, \mathscr{Q}, \{T_1, \dots, T_\ell\}, A, B, r)$ of* CONSTRAINED CYCLE PACKING*, runs in $\mathscr{O}^*(2^{\mathscr{O}(|T|)})$ time and returns a set of $2^{|T|}$ instances of* CONSTRAINED PATH ASSIGNMENT *such that $\mathscr{I}$ is a yes-instance if and only if at least one of the returned instances $\mathscr{J}$ satisfies $opt(\mathscr{J}) \geq r - \ell$. Further, the parameter of each of the returned instances is linearly upper-bounded by the parameter of $\mathscr{I}$.*

Proof. Let H denote the proper interval graph $G - T$ with clique partition $\mathscr{Q} = \{Q_1, \dots, Q_q\}$ and proper interval ordering π. Let A be the set $\{\sigma_1, \dots, \sigma_\ell\}$ of permutations where σ_i is a permutation of T_i for each $i \in [\ell]$. Let B be the set $\{\gamma_i \in \{0,1,2,3\}^{|T_i|} : i \in [\ell]\}$. Without loss of generality, we assume that $\mathscr{I}$ satisfies the following properties for each $i \in [\ell]$. Otherwise, we can declare that $\mathscr{I}$ is a no-instance.

- If $|T_i| = 1$, then $\gamma_i(1) \in \{2,3\}$.
- if $|T_i| = 2$, then either $\gamma_i(1) \in \{1,2,3\}$ or $\gamma_i(2) \in \{1,2,3\}$.
- For each $j \in [|T_i|]$, if $\gamma_i(j) = 0$, then $\sigma_i^{-1}(j)$ and $\sigma_i^{-1}(1 + (j \bmod |T_i|))$ are adjacent.

Initialize X, Y, Z and W to be empty sets and Γ, Λ_1, Λ_2, Ω and Υ to be empty functions. For each $i \in [\ell]$, let S_i denote the set of all strings in $\{1,2\}^{|T_i|}$. For each $\lambda \in \{\beta_1 \ldots \beta_\ell : \beta_i \in S_i, \forall i \in [\ell]\}$, create an instance $\mathscr{J}_\lambda = (H, \mathscr{Q}, \pi, X, Y, Z, W, \Gamma, \Lambda_1, \Lambda_2, \Omega)$ of CONSTRAINED PATH ASSIGNMENT as follows: for each $i \in [\ell]$ with $T_i = \{t_1, \ldots, t_c\}$ in the order specified by σ_i, perform the following steps for each $j \in [c]$. We remark that the function Υ with $\mathrm{dom}(\Upsilon) = \bigcup_{i \in [\ell]} \{(i,j) : j \in [|T_i|]\}$ and $\mathrm{img}(\Upsilon) = W \cup X \cup Z$ would be used in the proof below, and is not part of the construction of the new instance.

1. Let j' denote $1 + (j \bmod c)$. Let $b = \beta_i(j)$ and $d = \begin{cases} 1 & \text{if } b = 2, \\ 2 & \text{if } b = 1. \end{cases}$
2. If $\gamma_i(j) = 3$, add a variable w to W. Add w to $\mathrm{dom}(\Lambda_1)$ and to $\mathrm{dom}(\Lambda_2)$. Add (i,j) to $\mathrm{dom}(\Upsilon)$. Set $\Lambda_b(w) = N(t_j) \cap V(H)$, $\Lambda_d(w) = N(t_{j'}) \cap V(H)$, $\Upsilon((i,j)) = w$.
3. If $\gamma_i(j) = 2$, add variables x to X, y to Y. Add x to $\mathrm{dom}(\Omega)$ and (i,j) to $\mathrm{dom}(\Upsilon)$. Add x and y to $\mathrm{dom}(\Gamma)$. Set $\Gamma(x) = N(t_j) \cap V(H), \Gamma(y) = N(t_{j'}) \cap V(H)$, $\Omega(x) = y$ and $\Upsilon((i,j)) = x$.
4. If $\gamma_i(j) = 1$, add a variable z to Z. Add z to $\mathrm{dom}(\Gamma)$ and (i,j) to $\mathrm{dom}(\Upsilon)$. Set $\Gamma(z) = N(t_j) \cap N(t_{j'}) \cap V(H)$, $\Upsilon((i,j)) = z$.

As each of these steps can be executed in polynomial time, it follows that $\mathscr{J}_\lambda$ can be computed in polynomial time for a particular choice of λ. Note that the number of choices for λ is $2^{|T_1|}2^{|T_2|}\dots 2^{|T_\ell|} = 2^{|T|}$ since λ is a string in $\{1,2\}^{|T|}$. Therefore, the total running time of this reduction is $\mathcal{O}^*(2^{\mathcal{O}(|T|)})$. Further, observe that $|X|+|Y|+|Z|$ is $|T|$. That is, the parameter of $\mathscr{J}_\lambda$ is linearly upper-bounded by the parameter of $\mathscr{I}$.

We claim that $\mathscr{I}$ is a yes-instance if and only if $opt(\mathscr{J}_\lambda) \geq r - \ell$ for some λ. Let ϖ denote an arbitrary permutation of vertices of T. Consider the forward direction of the claim. Let $\mathscr{C}' = \{C_1, \dots, C_r\}$ be a solution to $\mathscr{I}$. Let $\mathscr{C} = \{C_1, \dots, C_{\widehat{r}}\}$ be a maximal set of vertex-disjoint cycles of G containing $\mathscr{C}'$ where $\widehat{r} \geq r$. From Lemma 13, we can assume $\mathscr{C}$ to be a nice set of cycles. We will show the existence of a feasible solution (h,g) to $\mathscr{J}_\lambda$ with $val((h,g)) = \widehat{r} - \ell$ where $\lambda \in \{1,2\}^{|T|}$. Without loss of generality, let $\{C_1, \dots, C_\ell\}$ be the set of cycles in $\mathscr{C}$ that have a vertex from T. For each $i \in [\ell]$ with $T_i = \{t_1, \dots, t_c\}$ in the order specified by C_i, perform the following steps for each $j \in [c]$. Let σ_i denote the permutation $t_1 \dots t_c$ of T_i. Let $\beta_1, \dots, \beta_\ell$ be initialized to empty strings.

1. Let j' denote $1 + (j \bmod c)$.

2. Suppose $\gamma_i(j) = 3$. Set $h(\Upsilon((i,j)))$ to be the maximal subpath P_{ij} (in H) of the path in C_i between $\sigma_i^{-1}(j)$ and $\sigma_i^{-1}(j')$. Let a_{ij} denote the starting vertex of P_{ij} and b_{ij} denote the ending vertex of P_{ij}. Let Q_a be the clique in $\mathscr{Q}$ containing a_{ij} and Q_b be the clique in $\mathscr{Q}$ containing b_{ij}. If $a \leq b$, then set $\beta_i(j) = 1$; otherwise set $\beta_i(j) = 2$.

3. Suppose $\gamma_i(j) = 2$. Set $g(\Upsilon((i,j)))$ to be the vertex u that succeeds $\sigma_i^{-1}(j)$ in C_i. Set $g(\Omega(\Upsilon((i,j))))$ to be the vertex v that succeeds u in C_i.

4. Suppose $\gamma_i(j) = 1$. Set $g(\Upsilon((i,j)))$ to be the vertex v that succeeds $\sigma_i^{-1}(j)$ in C_i.

Then, by construction, $H - (V(\text{img}(h)) \cup \text{img}(g))$ has $\widehat{r} - \ell \geq r - \ell$ vertex-disjoint triangles. Further, the first three conditions in the definition of a feasible solution to an instance of CONSTRAINED PATH ASSIGNMENT are satisfied by the construction of Γ, Λ_1, Λ_2 and Ω. The fourth condition requiring the vertex-disjointness of the paths assigned to variables in W is satisfied as these paths come from pairwise vertex-disjoint cycles. Finally, the last condition holds as these paths come from a nice set of cycles. Thus, (h,g) is a feasible solution for $\mathscr{J}_\lambda$ with $val((h,g)) = \widehat{r} - \ell$ where $\lambda = \beta_1 \dots \beta_\ell$. Hence, $opt(\mathscr{J}_\lambda) \geq r - \ell$.

Conversely, suppose $opt(\mathscr{J}_\lambda) \geq r - \ell$ for some $\lambda \in \{\beta_1, \dots, \beta_\ell : \beta_i \in S_i, \ \forall i \in [\ell]\}$. Let (h,g) be a feasible solution to $\mathscr{J}_\lambda$ with $val((h,g)) = r' \geq r - \ell$. Let $\mathscr{C}_\triangle$ be a set of r' vertex-disjoint triangles in $H - (V(\text{img}(h)) \cup \text{img}(g))$. Clearly, each triangle in this set is also a triangle in G. We will construct a set $\mathscr{C}$ of ℓ cycles in G such that any two distinct cycles in $\mathscr{C} \cup \mathscr{C}_\triangle$ are disjoint. For each $i \in [\ell]$ with $T_i = \{t_1, \dots, t_c\}$ in the order specified by σ_i, perform the following steps for each $j \in [c]$.

(i) Let s denote the variable $\Upsilon((i,j))$.

(ii) If $\gamma_i(j) = 3$ and $\beta_i(j) = 1$, let P_{ij} be the path $h(s)$.

(iii) If $\gamma_i(j) = 3$ and $\beta_i(j) = 2$, let P_{ij} be the path obtained from $h(s)$ by reversing the order of its vertices.

(iv) If $\gamma_i(j) = 2$, let P_{ij} be the path $g(s), g(\Omega(s))$.

(v) If $\gamma_i(j) = 1$, let P_{ij} be the path $g(s)$.

(vi) If $\gamma_i(j) = 0$, let P_{ij} be the empty path on zero vertices.

Define C_i to be the cycle $\sigma_i^{-1}(1), P_{i1}, \sigma_i^{-1}(2), P_{i2}, \dots, \sigma_i^{-1}(c), P_{ic}$. Now, $\mathscr{C}'_\triangle = \{C_1, \dots, C_\ell\}$ is a set of vertex-disjoint cycles in $G - V(\mathscr{C})$. Therefore, G has a set $\mathscr{C}_\triangle \cup \mathscr{C}'_\triangle$ of r vertex-disjoint cycles. Moreover, this set is a nice set of cycles. Therefore, the last condition in the definition of CONSTRAINED CYCLE PACKING holds. The other conditions hold by the construction of W, X, Y and Z. This completes the proof of the claim. □

3.4.3 Reduction to Colorful Constrained Path Assignment

At this point, we have reduced an instance of CYCLE PACKING to an instance of CONSTRAINED PATH ASSIGNMENT. In this section, we describe a reduction from CON-

STRAINED PATH ASSIGNMENT to its colorful variant using color coding. Consider an instance $\mathscr{J} = (G, \mathscr{Q}, \pi, X, Y, Z, W, \Gamma, \Lambda_1, \Lambda_2, \Omega)$ of CONSTRAINED PATH ASSIGNMENT. Let $X = \{x_1, \ldots, x_{r_X}\}$, $Y = \{y_1, \ldots, y_{r_Y}\}$, $Z = \{z_1, \ldots, z_{r_Z}\}$ and $W = \{w_1, \ldots, w_{r_W}\}$. Recall that each variable in $X \cup Y \cup Z$ has to be assigned to a vertex of G and each variable in W has to be assigned to a path of length at least two in G.

We color the vertices of G uniformly at random from the color set $[\hat{r}]$ where $\hat{r} = r_X + r_Y + r_Z$. Let $\chi : V(G) \to [\hat{r}]$ denote this coloring.

Proposition 20 ([5])**.** *If U is a subset of $V(G)$ of size $\hat{r}$, then the probability that the vertices of U are colored with pairwise distinct colors is at least $e^{-\hat{r}}$.*

Next, we define the notion of a colorful solution for our problem.

Definition 21. (Colorful solution) *A feasible solution (h, g) of $\mathscr{J}$ that satisfies the property that for any two distinct variables $s, t \in X \cup Y \cup Z$, $\chi(g(s)) \neq \chi(g(t))$ is said to be a colorful solution.*

Observe that the characteristic of a solution being colorful does not depend on the assignment to the variables in W. Now, we define the notion of an *optimum colorful solution*.

Definition 22. (Optimum colorful solution) *An optimum colorful solution of $\mathscr{J}$ is a colorful solution that maximizes $val_{\mathscr{J}}((h, g))$ over all colorful solutions (h, g).*

Rephrasing Proposition 20 in the context of CONSTRAINED PATH ASSIGNMENT, we have the following observation.

Observation 23. *If (h, g) is a feasible solution of $\mathscr{J}$, then (h, g) is a colorful solution of $\mathscr{J}$ with probability at least $e^{-\hat{r}}$.*

Armed with the guarantee that an optimum solution of $\mathscr{J}$ is colorful with sufficiently high probability, we focus on finding an optimum colorful solution of $\mathscr{J}$. Now that we have reduced our search space from the set of all feasible solutions to the set of all colorful solutions, we will simplify the instance accordingly. For each $i \in [\hat{r}]$, let V_i denote the set $\{v \in V(G) : \chi(v) = i\}$ of vertices of G that have been colored i by χ. Let δ be a permutation of $[\hat{r}]$. We use δ to specify the exact color of the vertex that is to be assigned to a variable in $X \cup Y \cup Z$. Define the function $\widehat{\Gamma} : X \cup Y \cup Z \to 2^{V(G)}$ as follows.

- For each $i \in [r_X]$, $\widehat{\Gamma}(x_i) = \Gamma(x_i) \cap V_{\delta(i)}$.
- For each $i \in [r_Y]$, $\widehat{\Gamma}(y_i) = \Gamma(y_i) \cap V_{\delta(r_X+i)}$.
- For each $i \in [r_Z]$, $\widehat{\Gamma}(z_i) = \Gamma(z_i) \cap V_{\delta(r_X+r_Y+i)}$.

For example, if δ specifies that a variable s is to be assigned to a vertex that has color i, then we restrict the set of vertices that can possibly be assigned to s to those that are colored i. Let $\mathscr{J}(\chi, \delta)$ denote the instance $(G, \mathscr{Q}, \pi, X, Y, Z, W, \widehat{\Gamma}, \Lambda_1, \Lambda_2, \Omega)$ of CONSTRAINED PATH ASSIGNMENT. Since for each pair of variables $s, t \in X \cup Y \cup Z$, we have $\widehat{\Gamma}(s) \cap \widehat{\Gamma}(t) = \emptyset$, $\mathscr{J}(\chi, \delta)$ has the following property.

Observation 24. *Any feasible solution (h, g) of $\mathscr{J}(\chi, \delta)$ is also a colorful solution.*

For the sake of clarity, we subsequently call the CONSTRAINED PATH ASSIGNMENT problem in which, for each pair of variables $s, t \in X \cup Y \cup Z$, $\Gamma(s) \cap \Gamma(t) = \emptyset$ holds, as the COLORFUL CONSTRAINED PATH ASSIGNMENT problem.

Observation 25. *If (h, g) is a colorful solution of $\mathscr{J}$, then there exists a permutation δ of $[\hat{r}]$ such that (h, g) is a feasible solution of $\mathscr{J}(\chi, \delta)$.*

Using the standard technique of derandomization of algorithms based on color coding [5, 25, 82], we have the following result by taking $n = |V(G)|$.

Proposition 26 ([5, 25, 82]). *Given integers $n, \hat{r} \geq 1$, there is a family $\mathscr{F}_{n,\hat{r}}$ of coloring functions $\chi : V(G) \to [\hat{r}]$ of size $e^{\hat{r}} \hat{r}^{\mathscr{O}(\log \hat{r})} \log n$ that can be constructed in $e^{\hat{r}} \hat{r}^{\mathscr{O}(\log \hat{r})} n \log n$*

time satisfying the following property: for every set $U \subseteq V(G)$ of size $\hat{r}$, there is a function $\chi \in \mathscr{F}_{n,\hat{r}}$ such that $\chi(u) \neq \chi(v)$ for any two distinct vertices $u, v \in U$.

Then, we have the following result.

Proposition 27. *There is an algorithm that, given an instance $\mathscr{J}$ of* CONSTRAINED PATH ASSIGNMENT, *runs in $\mathscr{O}^*(2^{\mathscr{O}(\hat{r}\log\hat{r})})$ time where $\hat{r} = |X|+|Y|+|Z|$ and returns a set of at most $\mathscr{O}^*(\hat{r}!\, e^{\hat{r}}\hat{r}^{\mathscr{O}(\log\hat{r})})$ instances of* COLORFUL CONSTRAINED PATH ASSIGNMENT *such that at least one of the returned instances $\widehat{\mathscr{J}}$ satisfies $opt(\mathscr{J}) = opt(\widehat{\mathscr{J}})$.*

Proof. Given $\mathscr{J}$, we compute the family $\mathscr{F}_{n,\hat{r}}$ of $e^{\hat{r}}\hat{r}^{\mathscr{O}(\log\hat{r})}\log n$ coloring functions using Proposition 26 where $\hat{r} = |X|+|Y|+|Z|$ and n is the number of vertices in the graph of the instance $\mathscr{J}$. For each coloring function $\chi \in \mathscr{F}_{n,\hat{r}}$ and for each permutation δ of $[\hat{r}]$, we create the instance $\mathscr{J}(\chi,\delta)$ of COLORFUL CONSTRAINED PATH ASSIGNMENT as described earlier. Overall, we create $\hat{r}!\, e^{\hat{r}}\hat{r}^{\mathscr{O}(\log\hat{r})}\log n$ instances of COLORFUL CONSTRAINED PATH ASSIGNMENT. From Observation 25 and Proposition 26, there is a coloring function $\chi \in \mathscr{F}_{n,\hat{r}}$ and a permutation δ of $[\hat{r}]$ such that (h,g) is an optimum solution of $\mathscr{J}$ if and only if (h,g) is an optimum solution of the instance $\mathscr{J}(\chi,\delta)$. Further, the running time of the reduction is $\mathscr{O}^*(2^{\mathscr{O}(\hat{r}\log\hat{r})})$. Thus the claim holds. $\square$

3.4.4 An FPT Algorithm for Colorful Constrained Path Assignment

In this section, we describe an algorithm to solve COLORFUL CONSTRAINED PATH ASSIGNMENT in $\mathscr{O}^*(2^{\mathscr{O}(k\log k)})$ time where $k = |X|+|Y|+|Z|+|W|$. The algorithm uses a dynamic programming routine and a greedy strategy. Consider an instance $\mathscr{J} = (G,\mathscr{Q},\pi,X,Y,Z,W,\Gamma,\Lambda_1,\Lambda_2,\Omega)$. Let $\mathscr{Q}$ be the given clique partition $\{Q_1,\ldots,Q_q\}$ of G.

Properties of a Feasible Solution

First, we observe some properties of a feasible solution which follow from the structure of $\mathscr{Q}$.

Observation 28. *If (h,g) is a feasible solution of $\mathscr{J}$, then for each $s \in X \cup Y$ with $g(s) \in Q_i$ for some $i \in [q]$, we have $g(\Omega(s)) \in Q_j$ with $|i-j| \leq 1$.*

This observation follows from the requirement that $g(s)g(\Omega(s)) \in E(G)$ enforced by COLORFUL CONSTRAINED PATH ASSIGNMENT and from the property of $\mathscr{Q}$ given by Proposition 111.

Lemma 29. *If (h,g) is a feasible solution of $\mathscr{J}$, then there is a feasible solution (h',g) such that for each variable $w \in W$ and for each $i, j \in [q]$ with $i < j$, every vertex from $V(h'(w)) \cap Q_i$ occurs before any vertex from $V(h'(w)) \cap Q_j$ in $h'(w)$. Further, $val((h,g)) \leq val((h',g))$.*

Proof. Let w be a variable in W such that $h(w)$ is a path starting from $u \in Q_{\hat{i}}$ and ending at $v \in Q_{\hat{j}}$. Note that $\hat{i} \leq \hat{j}$ and $|V(h(w))| \geq 2$. For every variable $w' \in W \setminus \{w\}$, let $h'(w') = h(w')$. For each $i, j \in [q]$ with $i < j$, if every vertex from $V(h(w)) \cap Q_i$ occurs before any vertex from $V(h(w)) \cap Q_j$ in $h(w)$, then $h'(w) = h(w)$. Otherwise, let $u' \in Q_i$ be the first vertex in $h(w)$ such that the vertex v' succeeding u' is in a clique Q_j with $j < i$. Observe that $\hat{i} \leq i$. From Proposition 111, we have $j = i-1$. Consider the following cases.
Case ($\hat{i} = \hat{j}$ or $|V(h(w))| = 2$): Let $h'(w) = u, v$. Then, $h'(w)$ and $h(w)$ have the same starting and ending vertices. Further, $|V(h'(w))| = 2$ and $V(h'(w)) \subseteq V(h(w))$ implying that $val((h,g)) \leq val((h',g))$. That is, (h',g) is a feasible solution of $\mathscr{J}$ with the required property.
Case ($\hat{i} < \hat{j}$ and $\hat{i} = i$): As $\hat{i} < \hat{j}$, there is a vertex $v'' \in Q_{\hat{i}}$ distinct from u such that $h(w)$ contains a subpath from v' to v''. Let $h'(w)$ be the path obtained from $h(w)$ by replacing the subpath from u' to v'' (via v') by the subpath u', v''. Then, (h',g) is a feasible solution of $\mathscr{J}$ as $|V(h'(w))| \geq 2$ and $h'(w)$ and $h(w)$ have the same starting and ending vertices.

Further, as $V(h'(w)) \subset V(h(w))$, it follows that $val((h,g)) \leq val((h',g))$.

Case ($\hat{i} < \hat{j}$, $\hat{i} < i$ and $\hat{j} \leq i-1$): Now, there is a vertex $v'' \in Q_{\hat{j}}$ distinct from v such that $h(w)$ contains a subpath from u to v'' followed by a subpath from v'' to u'. Let $h'(w)$ be the path obtained from $h(w)$ by replacing the subpath from v'' to u' by the subpath v'', v. Then, once again (h',g) is a feasible solution of $\mathscr{J}$ with $V(h'(w)) \subset V(h(w))$.

Case ($\hat{i} < \hat{j}$, $\hat{i} < i$ and $\hat{j} > i-1$): Here, there is a vertex $u'' \in Q_i$ distinct from u' such that $h(w)$ has a subpath from u to u' followed by a subpath from u' to v' which is followed by a subpath from v' to u'' that is followed by a subpath from u'' to v. Let $h'(w)$ be the path obtained from $h(w)$ by replacing the subpath from u' to u'' (via v') by the subpath u', u''. Then, once again (h',g) is a feasible solution of $\mathscr{J}$ with $V(h'(w)) \subset V(h(w))$.

By repeating this process (at most $|V(G)|$ times), we obtain the solution (h',g) with the desired property. □

Subsequently, we will only consider feasible solutions with the property given by Lemma 29.

Canonical Solutions

Next, we will define (in Definition 32) the notion of a feasible solution with more special properties. Let us identify what nice properties a feasible solution can have. Let (h^*, g^*) be a feasible solution of $\mathscr{J}$. Let $\mathscr{C}$ be a maximum size set of nice vertex-disjoint triangles in $G - (V(\text{img}(h^*)) \cup \text{img}(g^*))$. Consider the clique Q_i for some $i \in [q]$.

Definition 30. (Partition of Q_i with respect to (h^*, g^*)) *Given a feasible solution (h^*, g^*) of $\mathscr{J}$ and an integer $i \in [q]$, the clique Q_i can be partitioned into the following subsets.*

- $D^i_1 = \{v \in Q_i : v \notin V(\text{img}(h^*)) \cup \text{img}(g^*) \cup V(\mathscr{C})\}$.
- $D^i_2 = \{v \in Q_i : v \in V(\text{img}(h^*))\}$.
- $D^i_3 = \{v \in Q_i : \exists C \in \mathscr{C} \text{ with } v \in C \text{ and } C \cap Q_{i+1} \neq \emptyset\}$.

- $D_4^i = \{v \in Q_i : \exists C \in \mathscr{C} \textit{ with } v \in C \textit{ and } C \cap Q_{i-1} \neq \emptyset\}$.
- $D_5^i = \{v \in Q_i : v \in \operatorname{img}(g^*)\}$.
- $D_6^i = \{v \in Q_i : \exists C \in \mathscr{C} \textit{ with } v \in C \textit{ and } C \subseteq Q_i\}$.

We now observe certain properties of the sets D_j^i given by Definition 30.

Observation 31. *The partition of Q_i with respect to (h^*, g^*) given by Definition 30 satisfies the following properties.*

- *D_1^i is the set of vertices of Q_i that are neither in $V(\mathscr{C})$ nor in a path assigned to some variable in W nor assigned to any variable in $X \cup Y \cup Z$. By Lemma 11, $|D_1^i| \leq 2$.*
- *D_2^i is the set of vertices of Q_i that are in a path assigned to some variable in W. As there can be at most fourteen paths that have a vertex from Q_i (by definition of* COLORFUL CONSTRAINED PATH ASSIGNMENT*), it follows that $|D_2^i| \leq 28$.*
- *D_3^i is the set of vertices of Q_i that are present in triangles that have vertices from both Q_i and Q_{i+1}. By Lemma 11, $|D_3^i| \leq 2$.*
- *D_4^i is the set of vertices of Q_i that are present in triangles that have vertices from both Q_i and Q_{i-1}. By Lemma 11, $|D_4^i| \leq 2$.*
- *D_5^i is the set of vertices of Q_i that are assigned to variables in $X \cup Y \cup Z$.*
- *D_6^i is the set of vertices of Q_i that are present in triangles of $\mathscr{C}$ contained in Q_i.*

An example is given in Figure 3.8. Consider a pair u, v of vertices such that $u \in D_5^i$ and $v \in D_6^i$. Let $s \in X \cup Y \cup Z$ be the variable such that $g^*(s) = u$. Then, $u \in \Gamma(s)$. Suppose $v \in \Gamma(s)$. Let (h^*, g^{**}) denote the pair of functions obtained from (h^*, g^*) by setting $g^{**}(s) = v$. Consider the following cases.

- (Case $s \in Z$): Then, (h^*, g^{**}) is also a feasible solution of $\mathscr{J}$.

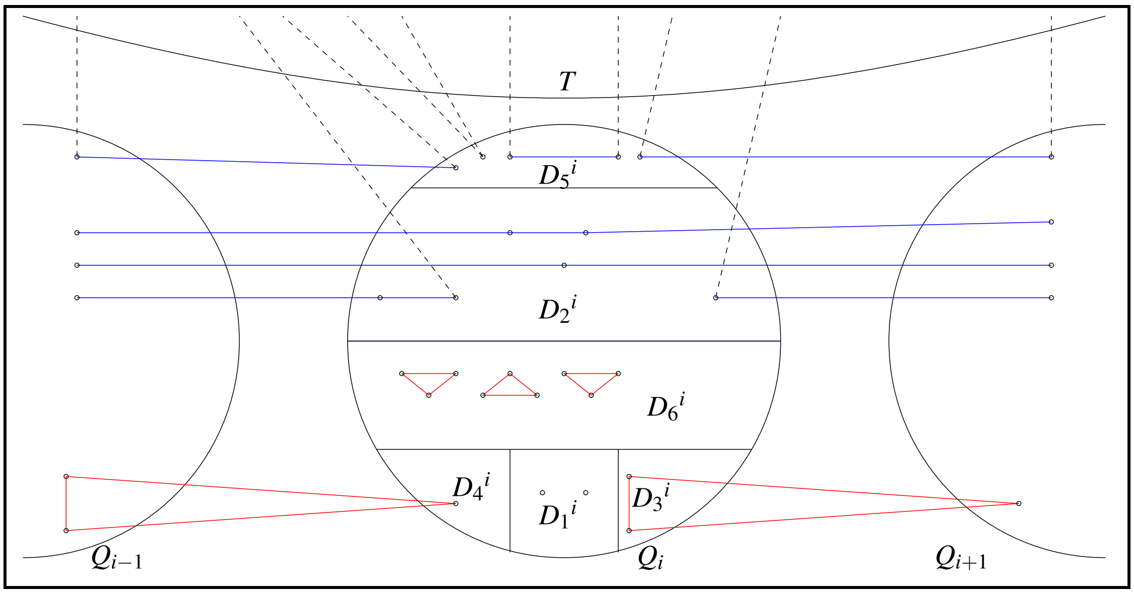


Fig 3.8: Illustration of the partition of Q_i into sets D^i_j for each $j \in [6]$. The blue lines indicate the edges that are obtained from (h^*, g^*) and the red triangles are elements of $\mathscr{C}$.

- (Case $s \in X \cup Y$ and $g^*(\Omega(s)) \in Q_i \cup Q_{i+1}$): If $\pi(v) > \pi(u)$, then (h^*, g^{**}) is also a feasible solution of $\mathscr{J}$ since v and $g^{**}(\Omega(s))$ are adjacent by Proposition 110.

- (Case $s \in X \cup Y$ and $g^*(\Omega(s)) \in Q_{i-1}$): If $\pi(v) < \pi(u)$, then (h^*, g^{**}) is also a feasible solution of $\mathscr{J}$ since v and $g^{**}(\Omega(s))$ are adjacent by Proposition 110.

Informally, in each of the cases, we obtain an *aligned* (to the left or to the right with respect to π) solution (h^*, g^{**}) with $val((h^*, g^*)) = val((h^*, g^{**}))$. This leads us to the notion of a *canonical solution* of $\mathscr{J}$ that allows us to resort to a greedy strategy for assigning vertices to variables in $X \cup Y \cup Z$.

We emphasize the role played by color coding here in the context of solving CONSTRAINED PATH ASSIGNMENT. Observe that there is no feasible solution (h', g') of $\mathscr{J}$ such that there is a variable $t \in X \cup Y \cup Z$ distinct from s with $g'(t) = v$. This crucially uses the fact that for each pair of distinct variables $s, t \in X \cup Y \cup Z$, we have $\Gamma(s) \cap \Gamma(t) = \emptyset$. Therefore, if $v \in \Gamma(s)$, then $v \notin \Gamma(t)$ for every $t \neq s$. In particular, the sets in $\{D^i_6 \cap \Gamma(s) : s \in X \cup Y \cup Z\}$ form a partition of D^i_6 into $|X|+|Y|+|Z|$ (not necessarily non-empty) sets. This property is achieved using our application of color coding to the instance of CONSTRAINED PATH

ASSIGNMENT from which $\mathscr{J}$ may be obtained (see Section 3.4.3 for details).

Definition 32. (Canonical solution) *Let (h,g) be a feasible solution of $\mathscr{J}$. Let $\mathscr{C}$ be a maximum size set of vertex-disjoint triangles in $G-(V(\text{img}(h))\cup\text{img}(g))$. Let $\mathscr{C}'$ denote the set of triangles in $\mathscr{C}$ that are contained in Q_i for some $i\in[q]$. Let $V'=V(\mathscr{C}')\cup\text{img}(g)$ and $D_i=V'\cap Q_i$. Then, (h,g) is a canonical solution if the following properties are satisfied.*

- *For each $i\in[q]$ and for each $s\in X\cup Y$ with $g(s)\in Q_i$ and $g(\Omega(s))\in Q_{i+1}\cup Q_i$, $g(s)$ is the vertex that maximizes $\pi(g(s))$ over all vertices in $D_i\cap\Gamma(s)$.*
- *For each $i\in[q]$ and for each $s\in X\cup Y$ with $g(s)\in Q_i$ and $g(\Omega(s))\in Q_{i-1}$, $g(s)$ is the vertex that minimizes $\pi(g(s))$ over all vertices in $D_i\cap\Gamma(s)$.*
- *For each $i\in[q]$ and for each $s\in Z$ with $g(s)\in Q_i$, $g(s)$ is the vertex that maximizes $\pi(g(s))$ over all vertices in $D_i\cap\Gamma(s)$.*

Note that the function h in the feasible solution (h,g) plays no role in deciding if (h,g) is indeed a canonical solution or not. Next, we prove the existence of canonical solutions.

Lemma 33. *If (h,g) is a feasible solution of $\mathscr{J}$, then there is a canonical solution (h,g^*) of $\mathscr{J}$ with $val((h,g))=val((h,g^*))$.*

Proof. Let $Q_0=\emptyset$. For an integer $i\in[q]\cup\{0\}$, we say that (h,g) is an *i-canonical solution* if it satisfies the properties in Definition 32 for cliques $Q_0,Q_1,\ldots,Q_i$. In this context, a q-canonical solution is a canonical solution. We show the existence of a q-canonical solution by induction on q. For $q=0$, observe that (h,g) is a q-canonical solution. Suppose (h,g) is an $(i-1)$-canonical solution for some $i\geq 1$. We show that there is another feasible solution (h,g^*) that is an i-canonical solution. Let $\mathscr{C}$ be a maximum size set of vertex-disjoint triangles in $G-(V(\text{img}(h))\cup\text{img}(g))$. Let $\mathscr{C}'$ be the set $\{C\in\mathscr{C}:\exists j\in[q],C\subseteq Q_j\}$ and $V'=V(\mathscr{C}')$. Initialize (h,g^*) to (h,g). Observe that V' is the set of vertices in $V(\mathscr{C})$ that are present in triangles of $\mathscr{C}$ that are completely contained in some clique Q_j, $j \in [q]$. Let $D_i = V' \cap Q_i$.

Let s be a variable in $X \cup Y$ such that $g(s) \in Q_i$. Then, from Observation 28, either $g(\Omega(s)) \in Q_{i-1}$ or $g(\Omega(s)) \in Q_{i+1} \cup Q_i$. In the former case, update $g^*(s)$ to the vertex v in $D_i \cap \Gamma(s)$ that minimizes $\pi(v)$. From Proposition 111, as $\pi(g^*(s)) \leq \pi(g(s))$, $\pi(g(\Omega(s))) < \pi(g^*(s))$ and $(g(s), g(\Omega(s))) \in E(G)$, we have $(g^*(s), g^*(\Omega(s))) \in E(G)$. In the latter case, update $g^*(s)$ to the vertex v in $D_i \cap \Gamma(s)$ that maximizes $\pi(v)$. From Proposition 111, as $\pi(g^*(s)) \geq \pi(g(s))$, $\pi(g(\Omega(s))) > \pi(g^*(s))$ and $(g(s), g(\Omega(s))) \in E(G)$, we have $(g^*(s), g^*(\Omega(s))) \in E(G)$. In any case, v is either $g(s)$ or v is in a triangle C of $\mathscr{C}$ with vertices only from Q_i. Therefore, the set $\mathscr{C}^*$ of triangles obtained from $\mathscr{C}$ from replacing v by $g(s)$ is a set of same size as $\mathscr{C}$. Execute this replacement procedure for all variables in $X \cup Y$ that are assigned to vertices in Q_i by g. At the end of this reassignment, we have the desired i-canonical solution. $\square$

Recall that by the definition of COLORFUL CONSTRAINED PATH ASSIGNMENT, for a pair s, t of distinct variables in $X \cup Y \cup Z$, we have $\Gamma(s) \cap \Gamma(t) = \emptyset$ and hence $(D_i \cap \Gamma(s)) \cap (D_i \cap \Gamma(t)) = \emptyset$. Therefore, once a variable $s \in \text{dom}(g)$ is reassigned to a vertex by g^*, its assignment does not change subsequently during reassignment of other variables.

Finding Canonical Solutions

Lemma 33 implies that it suffices to find an optimum solution of $\mathscr{J}$ that is canonical. We will describe a dynamic programming algorithm that finds such a solution (if one exists) by processing the cliques in $\mathscr{Q} = \{Q_1, \ldots, Q_q\}$ in the increasing order of their indices. Let $Q_{q+1} = \emptyset$. For each $i \in [q+1]$, let G_i denote the subgraph of G induced by $\bigcup_{j=1}^{i} Q_j$.

Setting up the Table

For each $i \in [q+1]$, we maintain a table $\mathbb{T}_i$. Let $\mathbb{T}$ denote the table that contains all entries of $\mathbb{T}_i$ for each $i \in [q+1]$. Before describing the entries in $\mathbb{T}_i$, we give an overview of what we would ideally like to store in these entries.

Let (h^*, g^*) be an optimum solution of $\mathscr{J}$. By Lemma 33, we can assume (h^*, g^*) to be a canonical solution. Let $\mathscr{C}$ be a maximum size set of vertex-disjoint triangles in $G - (V(\text{img}(h^*)) \cup \text{img}(g^*))$ satisfying the properties listed in Lemma 11. As each of the graphs in the sequence $G_1, G_2, \ldots, G_q, G_{q+1}$ is a subgraph of the graph succeeding it, we like to process the cliques $Q_1, \ldots, Q_q, Q_{q+1}$ from "left-to-right", that is, from Q_1 to Q_{q+1}, in order to compute (h^*, g^*) (or a feasible solution (h^{**}, g^{**}) with $val((h^{**}, g^{**})) = val((h^*, g^*))$). We would first like to understand how (h^*, g^*) and $\mathscr{C}$ look like when they are restricted to G_i for a fixed $i \in [q+1]$. This could shed insight into the subproblem that we want to solve on G_i and the (partial) solution that we want to store for G_i.

Ideally, we want to store the number (or set) of triangles in $\mathscr{C}$ that are contained in G_i. Let us call this set $\mathscr{C}'$. Any triangle C in $\mathscr{C} \setminus \mathscr{C}'$ has a vertex from Q_j for some $j \geq i+1$. We cannot *see* this triangle until we look at G_j. Our subproblem on G_i can afford to forget such triangles provided it remembers the vertices from Q_i that are present in these triangles. That is, we need to remember the set $Next^*$ of vertices of Q_i that are present in triangles in $\mathscr{C}$ that have a vertex from both Q_i and Q_{i+1}. Note that no vertex in Q_i can be in a triangle that has a vertex from Q_{i+2} by Proposition 111. Let Not^* be the set of vertices of Q_i that are not in the set $V(\mathscr{C}) \cup V(\text{img}(h^*)) \cup \text{img}(g^*)$. Let us partition $\mathscr{C}'$ into $\mathscr{C}_1$ and $\mathscr{C}_2$ such that $\mathscr{C}_1$ is the set of triangles contained in G_{i-1} and $\mathscr{C}_2 = \mathscr{C}' \setminus \mathscr{C}_1$. Suppose we have computed $|\mathscr{C}_1|$ by solving the subproblem on G_{i-1}. Then, $|\mathscr{C}_1|$ can be used to compute $|\mathscr{C}'|$ provided we know the set $Q_i \cap V(\mathscr{C}_2)$. That is, we need to know the set $Prev^*$ of vertices of Q_i that are present in triangles in $\mathscr{C}'$ that have a verti from both Q_i and Q_{i-1}. So far, we have identified two types of vertices (those in $Prev^*$ and those in $Next^*$) in Q_i with respect to (h^*, g^*). In order to understand the role of other vertices in Q_i, we partition Q_i

into 6 sets $D^i_1, D^i_2, D^i_3, D^i_4, D^i_5, D^i_6$ given by Definition 30. From Observation 31, we have $|D^i_1|, |D^i_3|, |D^i_4| \leq 2$ and $|D^i_2| \leq 28$.

Observe that D^i_1 is *Not**, D^i_3 is *Next** and D^i_4 is *Prev**. When we are processing Q_i (to solve the problem on G_i), we can afford to store all possible choices for D^i_j for each $j \in [4]$. The number of such choices is upper bounded by a polynomial (in $|Q_i|$) function. Though, the number of choices for D^i_5 is huge, the size of D^i_5 is at most k where $k = |X| + |Y| + |Z| + |W|$. Instead of guessing D^i_5, we can guess the set $S^*_i = \{v \in X \cup Y \cup Z : g^*(v) \in Q_i\}$ of variables that are assigned to vertices from Q_i by g^*. There are at most 2^k such choices. Similarly, we can guess the set $L^*_i = \{w \in W : V(h^*(w)) \cap Q_i \neq \emptyset\}$ of variables that are assigned to paths that contain a vertex from Q_i by h^*. We can also map these variables to vertices in D^i_2. Let us now handle the set D^i_6. Each vertex in D^i_6 is in a triangle that is contained in Q_i. Further, the set of vertices of Q_i that are a part of some triangle contained completely in Q_i is D^i_6. Thus, $|D^i_6|$ is a multiple of three and though the set of choices for such triangles is huge, the number of such triangles is easy to estimate. This number is simply $|D^i_6|/3$ which is equal to $(|Q_i| - \sum_{j=1}^{5} |D^i_j|)/3$. As Q_i is a clique, once D^i_5 is determined, any arbitrary set of $(|Q_i| - \sum_{j=1}^{5} |D^i_j|)/3$ triangles consisting of the remaining vertices is good enough for our solution.

As mentioned earlier, we would like to solve the subproblem on G_i using the previously computed solution on G_{i-1}. For this purpose, we need some information regarding the solution contained in G_{i-1}. Let us define the following sets.

- $S^* = \{v \in X \cup Y \cup Z : g^*(v) \in V(G_{i-1})\}$ is the set of variables in $X \cup Y \cup Z$ that are assigned to vertices in G_{i-1} by g^*. Observe that $S^* \cap S^*_i = \emptyset$.
- $L^* = \{w \in W : V(h^*(w)) \subseteq V(G_{i-1})\}$ is the set of variables in W that are assigned to paths in G_{i-1} by h^*. Note that $L^* \cap L^*_i = \emptyset$.

Once again the number of choices for S^* and L^* is $\mathcal{O}(2^k)$. Therefore, we can also guess S^* and L^* while processing Q_i. For technical convenience, we guess another special subset

L^*_{new} of L^*_i which denotes the set of variables in W that are assigned to paths that start at a vertex in Q_i by h^*. That is, $L^*_{new} = \{w \in L^*_i : V(h^*(w)) \cap V(G_i) \subseteq Q_i\}$. Now, we are ready to describe the entries of $\mathbb{T}$.

Index of an Entry. We identify each entry in $\mathbb{T}$ by an index.

Definition 34. (Index of an entry) *Let $i \in [q+1]$. Each entry in $\mathbb{T}_i$ is indexed by a tuple $\tau = (i, S_i, L_i, L_{new}, Prev, Next, Not, S, L, h_1, h_2)$ with the following interpretation.*

- *$S_i \subseteq X \cup Y \cup Z$, $L_i \subseteq W$ and $L_{new} \subseteq L_i$.*
- *Prev, Next and Not are pairwise disjoint subsets of Q_i.*
- *$S \subseteq X \cup Y \cup Z$ and $L \subseteq W$ with $S \cap S_i = \emptyset$ and $L \cap L_i = \emptyset$.*
- *$h_1 : L_i \to Q_i$ is an injective function that maps each variable in L_i to a vertex in Q_i.*
- *$h_2 : L'_i \to Q_i$ is an injective function that maps each variable in a subset L'_i of L_i to a vertex in Q_i.*

We remark that not all indices with an interpretation as given in Definition 34 would represent entries in the table. Clearly, if this was the case, the total number of entries would be too large for our purposes. Intuitively, $\mathbb{T}_i(\tau)$ indicates the possibility of the existence of an optimum solution (h^*, g^*) to $\mathscr{J}$ with the following properties. An illustration of the same is given in Figure 3.9.

- S is the set of variables that are assigned to vertices in G_{i-1} by g^*.
- L is the set of variables that are assigned to paths in G_{i-1} by h^*.
- S_i is the set of variables that are assigned to vertices in Q_i by g^*.
- L_i is the set of variables that are assigned to paths that have a vertex in Q_i by h^*. Clearly, $|L_i| \leq 14$.

- For each $w \in L_i$, $h_1(w)$ is the first vertex in $h^*(w)$ that is from Q_i and $h_2(w)$ is the second vertex (if $w \in \text{dom}(h_2)$) in $h^*(w)$ that is from Q_i. Specifically, for each $w \in L_i$, $h_1(w) \neq h_2(w)$. Further, $\text{img}(h_1) \cap \text{img}(h_2) = \emptyset$ as $\{h^*(w) : w \in W\}$ is a set of pairwise vertex-disjoint paths.

- L_{new} is the set of variables that are assigned to paths that start at a vertex in Q_i by h^*. In particular, for each $w \in L_{new}$, $V(h^*(w)) \cap V(G_{i-1}) = \emptyset$ and $h_1(w) \in \Lambda_1(w)$.

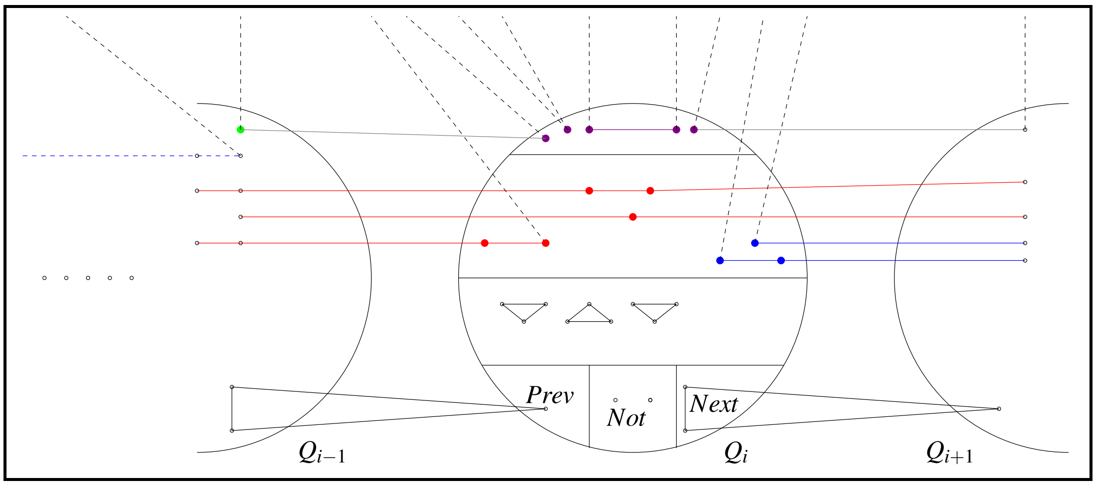


Fig 3.9: Components of an index $\tau = (i, S_i, L_i, L_{new}, Prev, Next, Not, S, L, h_1, h_2)$. Vertices assigned to S are the filled green circles and vertices assigned to S_i are the filled violet circles. The paths assigned to L are highlighted using blue dashed lines. The paths assigned to $L_i \setminus L_{new}$ are highlighted using red lines and filled red circles. The paths assigned to L_{new} are highlighted using filled blue circles and blue lines.

Eventually, $\mathbb{T}_i(\tau)$ would store the information relating to a maximum size set $\mathscr{C}$ of nice triangles in $G - (V(\text{img}(h^*)) \cup \text{img}(g^*))$ with the following properties.

- $(Not \cap V(\text{img}(h^*)) \cup \text{img}(g^*) = \emptyset$ and no triangle in $\mathscr{C}$ has a vertex from Not. From Lemma 11, $|Not| \leq 2$.

- For each $v \in Prev$, there is a triangle T in $\mathscr{C}$ that contains v and a vertex from Q_{i-1}. By Lemma 11, $|Prev| \leq 2$.

- For each $v \in Next$, there is a triangle T in $\mathscr{C}$ that contains v and a vertex from Q_{i+1}. By Lemma 11, $|Next| \leq 2$.

- $|Q_i| - (|Prev| + |Next| + |Not| + |\text{img}(h_1)| + |\text{img}(h_2)| + |S_i|)/3$ is the number of triangles in $\mathscr{C}$ that are contained in Q_i.

We next define the notion of a valid index. We only store entries with valid indices in the table.

Valid Indices. The set of conditions that determine the validity of an index is based on our requirement of what we want to store in the entry corresponding to it.

Definition 35. (Valid index) *An index $\tau = (i, S_i, L_i, L_{new}, Prev, Next, Not, S, L, h_1, h_2)$ in $\mathbb{T}_i$ is said to be valid if the following conditions are satisfied.*

- **(VI.1)** $|L_i| \leq 14$ *and* $|Prev|, |Next|, |Not| \leq 2$.
- **(VI.2)** $|Q_i| - (|Prev| + |Next| + |Not| + |\text{img}(h_1)| + |\text{img}(h_2)| + |S_i|)$ *is a non-negative integer divisible by three.*
- **(VI.3)** $\text{img}(h_1) \cap \text{img}(h_2) = \emptyset$.
- **(VI.4)** *For each $w \in L_{new}$, $h_1(w) \in \Lambda_1(w)$.*
- **(VI.5)** *If $i = 1$, then $Prev = S = L = \emptyset$ and $L_i = L_{new}$.*
- **(VI.6)** *If $i = q + 1$, then $S_i, L_i, L_{new}, Prev, Next, Not, \text{dom}(h_1)$ and $\text{dom}(h_2)$ are all empty sets. Further, $S = X \cup Y \cup Z$ and $L = W$.*
- **(VI.7)** *If $i = q$, then $Next = \emptyset$, $S_i \cup S = X \cup Y \cup Z$ and $L_i \cup L = W$.*
- **(VI.8)** *If $i = q$, then for each $w \in \text{dom}(h_1) \setminus \text{dom}(h_2)$, $h_1(w) \in \Lambda_2(w)$.*
- **(VI.9)** *If $i = q$, then for each $w \in \text{dom}(h_2)$, $h_2(w) \in \Lambda_2(w)$.*

(VI.1)–(VI.3) are justified by the interpretation of the sets that define an index. (VI.4) encodes the requirement that for each $w \in L_{new}$, the path $h^*(w)$ starts at a vertex in $\Lambda_1(w)$.

(VI.5)–(VI.9) consider the special cases when an index identifies an entry in $\mathbb{T}_1$ or $\mathbb{T}_q$ or $\mathbb{T}_{q+1}$. Among these, (VI.5)–(VI.7) are again justified by the interpretation of the sets that define an index. (VI.8) and (VI.9) encode the requirement that for each $w \in L_q$, the path $h^*(w)$ ends at a vertex in $\Lambda_2(w)$. This is due to the fact that any such path should end at Q_q since $Q_{q+1} = \emptyset$.

Graph H_τ and Instance J_τ. Let us again consider our canonical solution (h^*, g^*) of $\mathscr{J}$ and see how it looks like when restricted to G_i. The sets L^*, S^*, L_i^*, S_i^* and L_{new}^* are as defined earlier (see the explanation preceding Definition 34). Let L_{old}^* be the set $\{w \in L_i^* : V(h^*(w)) \cap Q_{i-1} \neq \emptyset\}$. That is, L_{old}^* is the set of variables in W that are assigned to paths that not only have a vertex in Q_i but also have a vertex from Q_{i-1}. Let us try to understand what is the precise task to be performed on G_i. We would like to capture this task with a "specialized" instance of CONSTRAINED PATH ASSIGNMENT, so we would not need to introduce yet another problem definition. Hence, we would next like to proceed by formally defining how to restrict the original problem instance to fit our local task. For the sake of readability, let us first separately explain how to restrict the graph itself, dismissing some irrelevant information. The formal definition corresponding to this explanation is captured by Definition 36 below.

Let h' denote h^* with domain restricted to $L^* \cup L_{old}^*$ and g' denote g^* with domain restricted to $S^* \cup S_i^*$. We would ideally like to store the partial solution (h', g') in the table. However, $V(\mathrm{img}(h'))$ is not necessarily a subset of $V(G_i)$. Recall that by the definition of CONSTRAINED PATH ASSIGNMENT, we need to ensure that a path whose construction has already started, corresponding to some variable $w \in W$, would terminate at a vertex from $\Lambda_2(w)$. However, when we are considering Q_i, the path has only reached a vertex in Q_i and its end vertex could be located at some future clique (outside the current graph G_i into which we would like to zoom in). To obtain a valid instance of CONSTRAINED PATH ASSIGNMENT, we need to terminate all paths. It is guaranteed that, for every variable

$w \in L^*_{old}$, the subpath of $h'(w)$ contained in G_i is of length at least two. This property is not necessarily guaranteed for a variable in L^*_{new}. For every variable $w \in L^*_{old}$, if we set $h'(w)$ to be this path, then (h', g') becomes a feasible solution for an appropriate subproblem in G_i. Clearly, we also need to update Λ_2 accordingly, which is done later in Definition 37. Then, when we restrict (h^*, g^*) to the graph G_i, we see that each subpath that has not terminated at Q_i or earlier can be forced to terminate at Q_i. The handling of paths of length exactly two (assigned to variables in X and Y) which have now turned to paths of length one (with a "to be determined" second vertex) will be discussed later in Definition 37. Here, we address paths that are assigned to variables in W which have not terminated at Q_i or earlier.

At this point, it is worth mentioning that, at an entry indexed by $\tau = (i, S_i, L_i, L_{new}, Prev, Next$ $, Not, S, L, h_1, h_2)$, we have information on whether a path to be assigned to a variable in $L_i \setminus L_{new}$ can be terminated at Q_i or not. If for such a variable w with $w \in \mathrm{dom}(h_2)$, we have $h_2(w) \in \Lambda_2(w)$, then this path can be terminated at $h_2(w)$. Similarly, if $w \in L_i \setminus \mathrm{dom}(h_2)$, then also this path can be terminated at $h_1(w)$ provided $h_1(w) \in \Lambda_2(w)$ holds. In both cases, the path will be of length at least two. Towards the formulation of the restriction of the graph, we first define Q' as the set $(h_2(L_{new}) \cup h_1(L_{new}) \cup Next \cup Not)$. Indeed, the vertices in Not and $Next$ clearly do not affect the task to be done up and including the point where we process clique Q_i, it is only necessary to ensure that they are not yet used, for which purpose we may simply remove them. Now, recall that $h_1(L_{new})$ is the set of vertices that start new paths (assigned to variables in L_{new}). These are also only relevant when we consider a clique Q_j with $j \geq i+1$, hence, we can simply remove these vertices while processing Q_i. Similarly, $h_2(L_{new})$ can also be deleted (even if they end the new paths).

Definition 36. **(Graph H_τ)** *Let $\tau = (i, S_i, L_i, L_{new}, Prev, Next, Not, S, L, h_1, h_2)$ be the index of an entry in $\mathbb{T}_i$. Let $Q' = (h_2(L_{new}) \cup h_1(L_{new}) \cup Next \cup Not)$. Then, H_τ denotes the subgraph of G_i induced by $V(G_i) \setminus Q'$.*

Clearly, H_τ is a proper interval graph with clique partition $\mathscr{Q}' = \{Q_1, \ldots, Q_{i-1}, Q_i \setminus$

$Q'\}$ and proper interval ordering π' which is π restricted to $V(H_\tau)$. Next, we define the subproblem and an instance of it associated with τ. Recall that in a partial solution (h', g') defined earlier, it is possible that there is a vertex $s \in X \cup Y$ such that $g'(s) \in V(G_i)$ but $g'(\Omega(s)) \notin V(G_i)$. In such a case, we treat s as a variable in Z. Observe that the only property that distinguishes a variable $s \in X \cup Y$ from a variable $t \in Z$ is that the assignment of a vertex to s depends on the assignment of a vertex to $\Omega(s)$ while the assignment of a vertex to t is independent of all other assignments. However, when we say that we forget the dependence of a variable s and treat it as being independent, it appears as if we are forgetting crucial information. Note that our interest is only in canonical solutions (see Lemma 33) and in a canonical solution, a variable in Z is always assigned to the "rightmost" permissible vertex (determined by Γ). Therefore, the possibility that s and $\Omega(s)$ can indeed be assigned to a pair of adjacent vertices is not affected by treating s as a variable in Z in the local instance. We remark that Proposition 110 is crucially used for this property to be guaranteed.

Definition 37. **(Instance $\mathscr{J}_\tau$)** *Let $\tau = (i, S_i, L_i, L_{new}, Prev,$ $Next, Not, S, L, h_1, h_2)$ be the index of an entry in $\mathbb{T}_i$. Let $X' = (S \cup S_i) \cap X$, $Y' = (S_i \cup S) \cap Y$ and $Z' = (S_i \cup S) \cap Z$. Let W' be $L \cup (L_i \setminus L_{new})$. Let $\widehat{\Gamma}$ denote Γ restricted to $X' \cup Y' \cup Z'$ as the domain defined as follows.*

- *For each $s \in S$, set $\widehat{\Gamma}(s) = \Gamma(s) \cap V(G_{i-1})$.*
- *For each $s \in S_i$, set $\widehat{\Gamma}(s) = \Gamma(s) \cap Q_i$.*

Similarly, let $\widehat{\Lambda}_1$ denote Λ_1 restricted to W' as the domain defined as follows.

- *For each $w \in L \cup (L_i \setminus L_{new})$, $\widehat{\Lambda}_1(w) = \Lambda_1(w) \cap V(G_{i-1})$.*

Let $\widehat{\Lambda}_2$ denote the function with $\mathrm{dom}(\widehat{\Lambda}_2) = W'$ *defined as follows.*

- *For each $w \in L$, $\widehat{\Lambda}_2(w) = \Lambda_2(w) \cap V(G_{i-1})$.*

- *For each $w \in L_i \setminus L_{new}$ with $w \in \mathrm{dom}(h_2)$, $\Lambda_2(w) = \{h_2(w)\}$.*
- *For each $w \in L_i \setminus L_{new}$ with $w \notin \mathrm{dom}(h_2)$, $\Lambda_2(w) = \{h_1(w)\}$.*

Let $\widehat{X} = \{x \in S_i \cap X : \Omega(x) \notin S_i \cup S\}$ and $\widehat{Y} = \{y \in S_i \cap Y : \Omega(y) \notin S_i \cup S\}$. Let Ω' be the bijection Ω restricted to $X' \setminus \widehat{X}$ as the domain. Then, $(H_\tau, \mathscr{Q}', \pi', X' \setminus \widehat{X}, Y' \setminus \widehat{Y}, Z' \cup \widehat{X} \cup \widehat{Y}, W', \widehat{\Gamma}, \widehat{\Lambda}_1, \widehat{\Lambda}_2, \Omega')$ is the instance $\mathscr{J}_\tau$ of COLORFUL CONSTRAINED PATH ASSIGNMENT *corresponding to τ.*

The arguments to the instance $\mathscr{J}_\tau$ are simply the arguments to the original instance restricted to H_τ with additional restrictions which are justified by the interpretation of τ. We need to take into account the meaning of τ, as we would like to consider a solution space more restricted than simply the set of all canonical solutions (in order to be able to use the stored solution in the future). For this purpose, we ensure that S_i and S can be interpreted as we intended, meaning g_τ (of a feasible solution (h_τ, g_τ)) maps variables belonging to S_i to vertices in the current clique Q_i, and variables belonging to S to the vertices in previous cliques. We also ensure that any variable w in L (that is associated with a path that has already been terminated) is indeed assigned to a path fully contained in G_{i-1}. In particular, such a path should have started in G_{i-1}. Further, we need that any variable in $L_i \setminus L_{new}$ is associated with a path that has started before Q_i.

So far, we have defined the input of an instance associated with the local task performed when processing a clique Q_i, but we have not yet defined the objective of the task itself. This objective is precisely what we aim to capture in the following Definition 38. We will first informally explain the intuition guiding us here. We are interested in combining the following ingredients.

(i) triangles that are fully contained in G_{i-1} (which were already identified in earlier computations).

(ii) triangles that "lie between" Q_{i-1} and Q_i, that is, triangles that contain at least one vertex from Q_{i-1} and at least one vertex from Q_i.

(iii) local triangles fully contained in Q_i.

When combining these ingredients, we clearly need to ensure that the result is a feasible solution to the instance we have just created in Definition 37. We first require that *Prev* is "meaningful" in the sense that all of its vertices indeed belong to triangles of ingredient (ii). Next, we ensure that each variable $w \in W$ that belongs to $L_i \setminus L_{new}$, and hence is associated with a path that is still "under construction", is assigned to a path whose intersection with Q_i is precisely $\{h_1(w)\} \cup \{h_2(w)\}$.

Value of the Entry $\mathbb{T}_i(\tau)$. We now formally define the information stored in an entry of $\mathbb{T}_i$.

Definition 38. (Value of $\mathbb{T}_i(\tau)$) *Let $\tau = (i, S_i, L_i, L_{new}, Prev, Next, Not, S, L, h_1, h_2)$ be the index of an entry in $\mathbb{T}_i$. Then, $\mathbb{T}_i(\tau)$ stores the number Δ_τ which is the maximum size of a set $\mathscr{C}$ of vertex-disjoint triangles in $H_\tau - (V(\text{img}(h_\tau)) \cup \text{img}(g_\tau))$ over all canonical solutions (h_τ, g_τ) of $\mathscr{J}_\tau$ satisfying the following properties.*

- *Every vertex in Prev is in a triangle C in $\mathscr{C}$ that has a vertex from Q_{i-1}.*
- *For each $w \in L_i \setminus L_{new}$, $V(h_\tau(w)) \cap Q_i = \{h_1(w)\} \cup \{h_2(w)\}$.*

If no such (h_τ, g_τ) exists, then $\Delta_\tau = -\infty$.

We call a canonical solution (h_τ, g_τ) of $\mathscr{J}_\tau$ satisfying the above two properties a *feasible solution* for $\mathscr{J}_\tau$ and $val((h_\tau, g_\tau))$ denotes $|\mathscr{C}|$. In this context, a feasible solution (h_τ, g_τ) with maximum $val((h_\tau, g_\tau))$ is called an *optimum* solution. Observe that any feasible solution (h_τ, g_τ) of $\mathscr{J}_\tau$ also satisfies the following properties.

- For each $s \in S_i$, $g_\tau(s) \in Q_i$ and for each $s \in S$, $g_\tau(s) \in V(G_{i-1})$.
- For each $w \in L$, $V(h_\tau(w)) \subseteq V(G_{i-1})$.

Note that for the index $\tau = (q+1, S_{q+1}, L_{q+1}, L_{new}, Prev, Next, Not, S, L, h_1, h_2)$, H_τ is G and $\mathscr{J}_\tau$ is $\mathscr{J}$. More importantly, Δ_τ is $opt(\mathscr{J})$ as an optimum solution for $\mathscr{J}_\tau$ is an optimum solution for $\mathscr{J}$. This completes the description of $\mathbb{T}$.

Filling the Table

Now, we move on to filling up the entries in $\mathbb{T}$. The following lemma states how to fill the entries in $\mathbb{T}_1$.

Lemma 39. *Let $\tau = (1, S_1, L_1, L_{new}, Prev, Next, Not, S, L, h_1, h_2)$ be the index of an entry in $\mathbb{T}_1$. Let $loc = (|Q_1| - (|Prev| + |Next| + |Not| + |\text{img}(h_1)| + |\text{img}(h_2)| + |S_1|))/3$. Then, $\Delta_\tau = loc$.*

Proof. Observe that $Prev = \emptyset$, $L_1 = L_{new}$, $S = \emptyset$, $L = \emptyset$ and loc is an integer by Definition 35. Also, H_τ is an induced subgraph of the complete graph $G[Q_1]$. Any optimum solution (h_τ, g_τ) of $\mathscr{J}_\tau$ assigns S_1 to a set of $|S_1|$ vertices in $Q_1 \setminus (Prev \cup Next \cup Not \cup \text{img}(h_1) \cup \text{img}(h_2))$. The remaining vertices in Q_1 are partitioned into loc triangles in $H_\tau - (V(\text{img}(h_\tau)) \cup \text{img}(g_\tau))$. Therefore, Δ_τ is loc. □

Next, we characterize the value of an entry in $\mathbb{T}_i$ (where $i > 1$) as a function of the value of an entry in $\mathbb{T}_{i-1}$.

Function Φ_τ. We first associate an auxiliary information with each index $\tau = (i, S_i, L_i, L_{new}, Prev, Next, Not, S, L, h_1, h_2)$. This information is an injective function $\Phi_\tau : S_i \rightarrow Q_i$. Consider an optimum solution (h_τ, g_τ) of $\mathscr{J}_\tau$. Let $U_\tau = Q_i - (Prev \cup Next \cup Not \cup \text{img}(h_1) \cup \text{img}(h_2))$. As (h_τ, g_τ) is a canonical solution, the following holds.

- For $s \in S_i \cap (X \cup Y)$ with $g_\tau(\Omega(s)) \in Q_{i+1} \cup Q_i$, $g_\tau(s)$ is the vertex in Q_i that maximizes $\pi(g_\tau(s))$ over all vertices in $U_\tau \cap \Gamma(s)$.

- For each $s \in S_i \cap (X \cup Y)$ with $g_\tau(\Omega(s)) \in Q_{i-1}$, $g_\tau(s)$ is the vertex in Q_i that minimizes $\pi(g_\tau(s))$ over all vertices in $U_\tau \cap \Gamma(s)$.

- For each $s \in Z \cap S_i$, $g_\tau(s)$ is the vertex in Q_i that maximizes $\pi(g_\tau(s))$ over all vertices in $U_\tau \cap \Gamma(s)$.

We store precisely the function g_τ with domain restricted to S_i as the function Φ_τ. Recall that, in the above list, for a pair s,t of distinct variables in $X \cup Y \cup Z$, we have $(U_\tau \cap \Gamma(s)) \cap (U_\tau \cap \Gamma(t)) = \emptyset$ since $\Gamma(s) \cap \Gamma(t) = \emptyset$. As a consequence, Φ_τ can be obtained by just examining the components of τ.

Definition 40. (Function Φ_τ) *Let $\tau = (i, S_i, L_i, L_{new}, Prev, Next, Not, S,\ L,\ h_1, h_2)$. We compute Φ_τ using the following procedure.*

1. *Let U_τ be $Q_i - (Prev \cup Next \cup Not \cup \mathrm{img}(h_1) \cup \mathrm{img}(h_2))$ and Var be S_i.*

2. *Initialize Φ to be the empty function.*

3. *For each variable $s \in Var$, execute the following steps.*

   (a) If $s \in Z$, $s \in X \cup Y$ with $\Omega(s) \in S_i$, or $s \in X \cup Y$ with $\Omega(s) \notin S \cup S_i$, let v be the vertex in $U_\tau \cap \Gamma(s)$ that maximizes $\pi(v)$ (if one exists).

   (b) If $s \in X \cup Y$ with $\Omega(s) \in S \setminus S_i$, let v be the vertex in $U_\tau \cap \Gamma(s)$ that minimizes $\pi(v)$ (if one exists).

   (c) If v is well-defined, then add s to $\mathrm{dom}(\Phi)$, set $\Phi(s) = v$ and delete s from Var.

4. *If $\mathrm{dom}(\Phi) = S_i$, then set Φ_τ to Φ. Otherwise, set Φ_τ to the empty function.*

We now explain the importance of storing this information. Once again, let us consider the canonical optimum solution (h^*, g^*) of $\mathscr{J}$. Let $\mathscr{C}$ be a maximum size set of vertex-disjoint triangles in $G - (V(\mathrm{img}(h^*)) \cup \mathrm{img}(g^*))$. The sets $Prev^*$, $Next^*$, Not^*, L^*, S^*, L_i^*, S_i^* and L_{new}^* are as defined earlier (see the explanation preceding Definition 34). Suppose (h^*, g^*) and $\mathscr{C}$ satisfy the following properties: $L = L^*$, $S = S^*$, $S_i =$

S_i^*, $L_i = L_i^*$, $L_{new} = L_{new}^*$, $Prev = Prev^*$, $Next = Next^*$, $Not = Not^*$. Further, suppose for each $w \in L_i$, $V(h^*(w)) \cap Q_i = \{h_1(w)\} \cup \{h_2(w)\}$. Let $U_\tau = Q_i - (Prev \cup Next \cup Not \cup \text{img}(h_1) \cup \text{img}(h_2))$. Then, g^* with domain restricted to S_i is the function Φ_τ. Recall that to compute Δ_τ, we will look up entries in $\mathbb{T}_{i-1}$. Suppose we look at the index ν of an entry (and the value Δ_ν) that may be used to compute Δ_τ. Let $\nu = (i-1, S_{i-1}, L_{i-1}, L'_{new}, Prev', Next', Not', S', L', h'_1, h'_2)$. Note that we would have only stored the value Δ_ν and not a canonical solution (h_ν, g_ν) of $\mathscr{J}_\nu$ that achieves it. However, if for some $s \in S_i \cap (X \cup Y)$ such that $\Omega(s) \in \text{dom}(\Phi_\nu)$, $\Phi_\tau(s)$ and $\Phi_\nu(\Omega(s))$ are not adjacent, then there is no canonical solution (or rather, no feasible solution) (h^*, g^*) of $\mathscr{J}$ such that g^* with domain restricted to S_i is Φ_τ and g^* with domain restricted to S_{i-1} is Φ_ν. In other words, we cannot use Δ_ν to compute Δ_τ. This justifies the importance of Φ_τ.

Computing Δ_τ. Finally, we deal with computing Δ_τ. Let $\mathscr{C}'$ be the set of triangles in $\mathscr{C}$ that contain both a vertex from Q_i and a vertex from Q_{i-1}. We define the following sets of variables.

- $S_{i-1} = \{v \in X \cup Y \cup Z : g^*(v) \in Q_{i-1}\}$, the set of variables in $X \cup Y \cup Z$ that are assigned to vertices in Q_{i-1} by g^*.
- $L_{i-1} = \{w \in W : V(h^*(w)) \cap Q_{i-1} \neq \emptyset\}$, the set of variables in W that are assigned to paths that have a vertex from Q_{i-1} by h^*.
- $L'_{new} = \{w \in L_{i-1} : V(h^*(w)) \cap V(G_{i-1}) \subseteq Q_{i-1}\}$, the set of variables in W that are assigned to paths that start at a vertex in Q_{i-1} by h^*.
- $S' = \{v \in X \cup Y \cup Z : g^*(v) \in V(G_{i-2})\}$, the set of variables in $X \cup Y \cup Z$ that are assigned to vertices in G_{i-2} by g^*.
- $L' = \{w \in W : V(h^*(w)) \subseteq V(G_{i-2})\}$, the set of variables in W that are assigned to paths in G_{i-2} by h^*.

Next, we define the following sets of vertices.

- *$Prev'$* is the set of vertices of Q_{i-1} that are present in triangles in $\mathscr{C}$ that have a vertex from both Q_{i-2} and Q_{i-1}.
- *$Next'$* is the set of vertices of Q_{i-1} that are present in triangles in $\mathscr{C}$ that have a vertex from both Q_i and Q_{i-1}.
- *Not'* is the set of vertices of Q_{i-1} that are not in the set $V(\mathscr{C}) \cup V(\mathrm{img}(h^*)) \cup \mathrm{img}(g^*)$.

Note that these sets together form the index $\nu = (i-1, S_{i-1}, L_{i-1}, L'_{new}, Prev', Next', Not', S', L', h'_1, h'_2)$ where for each $w \in L_{i-1}$, $V(h^*(w)) \cap Q_{i-1} = \{h'_1(w)\} \cup \{h'_2(w)\}$. Moreover, either $L_{i-1} \subseteq L_i$ or for each $w \in L_{i-1} \setminus L_i$, $|V(h^*(w))| \geq 2$. Clearly, the components of τ and ν are related. This relation is stated below in Definition 41 using the notion of compatible entries.

Definition 41. (Compatible indices)
Let $\tau = (i, S_i, L_i, L_{new}, Prev, Next, Not, S, L, h_1, h_2)$ be the index of an entry in $\mathbb{T}_i$. If $i = q+1$, then the index ν of any entry in $\mathbb{T}_q$ is compatible with τ. Suppose $1 < i \leq q$. Let $\nu = (i-1, S_{i-1}, L_{i-1}, L'_{new}, Prev', Next', Not', S', L', h'_1, h'_2)$ be the index of an entry in $\mathbb{T}_{i-1}$. We say that ν and τ are compatible if the following properties hold.

- **(CI.1)** *$S' \subseteq S$, $L' \subseteq L$, $S_{i-1} \cup S' = S$ and $L_{new} \cap L_{i-1} = \emptyset$.*
- **(CI.2)** $|Prev| = \begin{cases} 0 & \textit{if } |Next'| = 0, \\ 2 & \textit{if } |Next'| = 1, \\ 1 & \textit{if } |Next'| = 2. \end{cases}$
- **(CI.3)** *There is a set $\mathscr{C}'$ of vertex-disjoint triangles in $G[Prev \cup Next']$ such that every vertex in $Prev \cup Next'$ is in some triangle of $\mathscr{C}'$.*
- **(CI.4)** *$G[Not \cup Not']$ has no triangle.*

- **(CI.5)** *For each* $w \in \text{dom}(h'_1) \setminus (\text{dom}(h'_2) \cup L_i)$, $h'_1(w) \in \Lambda_2(w)$.

- **(CI.6)** *For each* $w \in \text{dom}(h'_2) \setminus L_i$, $h'_2(w) \in \Lambda_2(w)$.

- **(CI.7)** *For each* $w \in \text{dom}(h'_2) \cap \text{dom}(h_1)$, $h'_2(w)$ *and* $h_1(w)$ *are adjacent.*

- **(CI.8)** *For each* $w \in (\text{dom}(h'_1) \cap \text{dom}(h_1)) \setminus \text{dom}(h'_2)$, $h'_1(w)$ *and* $h_1(w)$ *are adjacent.*

- **(CI.9)** *For each* $w \in L_{i-1} \setminus L_i$, *one of the following holds.*

  – *(Case* $w \notin L'_{new}$*): If* $w \in \text{dom}(h'_2(w))$, *then* $h'_2(w) \in \Lambda_2(w)$. *Otherwise,* $h'_1(w) \in \Lambda_2(w)$.

  – *(Case* $w \in L'_{new}$*):* $w \in \text{dom}(h'_1) \cap \text{dom}(h'_2)$ *and* $h'_2(w) \in \Lambda_2(w)$.

- **(CI.10)** *For each* $s \in S_i \cap (X \cup Y)$ *such that* $\Omega(s) \in \text{dom}(\Phi_v)$, $\Phi_\tau(s)$ *and* $\Phi_v(\Omega(s))$ *are adjacent.*

(CI.1) follows from the definition of the corresponding sets. (CI.2) and (CI.3) specify that the triangles in $\mathscr{C}'$ are formed exactly by the vertices in $Prev \cup Next'$. (CI.4) encodes the maximality of $\mathscr{C}$. Suppose there is a variable $w \in W$ that is in $L \cap L_{i-1}$. That is, $h^*(w)$ has ended at Q_{i-1}. We could have gathered this information while processing Q_{i-1} as explained earlier (see the paragraph preceding Definition 36). If we are going to compute Δ_τ using Δ_v, it is necessary that we should be able to forget the paths assigned to such variables as they have ended before Q_i. Then, such paths should have ended at legal vertices specified by Λ_2. This check is made using (CI.5) and (CI.6). Suppose there is a variable $w \in W$ that is in $L_i \cap L_{i-1}$. That is, $V(h^*(w))$ has a vertex (specified by h'_1 and h'_2) from Q_{i-1} and a vertex (specified by h_1 and h_2) from Q_i. (CI.7) and (CI.8) ensure that $h'_1(w), h'_2(w), h_1(w), h_2(w)$ is indeed a path. (CI.10) is justified by the fact that Φ_τ is g^* with domain restricted to S_i and Φ_v is g^* with domain restricted to S_{i-1}. (CI.9) ensures that each path assigned to a variable $w \in L_{i-1} \setminus L_i$ that has terminated before Q_i (in particular, at Q_{i-1}) is of length at least two and has its end vertex in $\Lambda_2(w)$.

Now, we have set up the framework to characterize each entry in $\mathbb{T}_i$ for $i > 1$ as a function of an appropriate previous entry. This characterization is given in Lemma 42. Before proceeding to its statement and proof, we will give an outline of the same. Let $\tau = (i, S_i, L_i, L_{new}, Prev, Next, Not, S, L, h_1, h_2)$ be the index of an entry in $\mathbb{T}_i$ for which we wish to compute Δ_τ. Recall that Δ_τ is the value of an optimum solution for $\mathscr{J}_\tau$. That is, Δ_τ is the maximum size of a set $\mathscr{C}$ of vertex-disjoint triangles in $H_\tau - (V(\text{img}(h_\tau)) \cup \text{img}(g_\tau))$ satisfying properties mentioned in Definition 38. Observe that there are three types of triangles in $\mathscr{C}$.

- Triangles that are completely contained in Q_i. Let c_1 denote the number of such triangles.
- Triangles that have a vertex from Q_{i-1} and a vertex from Q_i. Let c_2 denote the number of such triangles.
- Triangles that are contained in G_{i-1}. Let c_3 denote the number of such triangles.

Then, Δ_τ is the sum of three values – c_1, c_2 and c_3. The first two values can be computed by just examining τ while the third value can be obtained by examining entries in $\mathbb{T}_{i-1}$. This observation is formalized in the following lemma.

Lemma 42. *Let $\tau = (i, S_i, L_i, L_{new}, Prev, Next, Not, S, L, h_1, h_2)$ be the index of an entry in $\mathbb{T}_i$ for some $i > 1$. Then, $\Delta_\tau = -\infty$ if and only if there is no entry in $\mathbb{T}_{i-1}$ whose index is compatible with τ. Further, if $\Delta_\tau \neq -\infty$, then $\Delta_\tau = \Delta_\nu + \rho + loc$ where the terms in the summation have the following definition.*

- $loc = (|Q_i| - (|Prev| + |Next| + |Not| + |\text{img}(h_1)| + |\text{img}(h_2)| + |S_i|))/3$.
- *$\rho \in \{0, 1\}$ is the maximum number of vertex-disjoint triangles in $G[Prev \cup Next']$.*
- *ν is the index of an entry in $\mathbb{T}_{i-1}$ that maximizes Δ_ν among all indices compatible with τ.*

Proof. Recall from Definition 36 that H_τ is the subgraph of G induced by $V(G_i) \setminus Q'$ where $Q' = (h_2(L_{new}) \cup h_1(L_{new}) \cup Next \cup Not)$. Let (h_ν, g_ν) be an optimum solution of $\mathscr{J}_\nu$. Let $\mathscr{C}_\nu$ be a maximum size set of vertex-disjoint triangles of $H_\nu - (V(\text{img}(h_\nu)) \cup \text{img}(g_\nu))$ with the required properties given by Definition 38. Then, we obtain a feasible solution (h, g) of $\mathscr{J}_\tau$ by extending (h_ν, g_ν) as follows.

- For each $w \in L \setminus L'_{new}$, $h(w) = h_\nu(w)$ and for each $s \in S$, $g(s) = g_\nu(s)$.
- For each $w \in L \cap L'_{new}$, $h(w)$ is the path $h'_1(w)h'_2(w)$.
- For each $w \in L_i \setminus L_{new}$, $h(w)$ is the path $h_\nu(w)h_1(w)h_2(w)$.
- For each $s \in S_i$, $g(s) = \Phi_\tau(s)$.

The feasibility of (h, g) follows from the definition of $\mathscr{J}_\tau$. Let $Loc = Q_i \setminus (Prev \cup Next \cup Not \cup \text{img}(\Phi_\tau) \cup \text{img}(h_1) \cup \text{img}(h_2))$. Let $\mathscr{C}_1$ be a maximum size set of triangles in $G[Loc]$. Let $\mathscr{C}_2$ be a maximum size set of triangles in $G[Prev \cup Next']$. Then, $\mathscr{C} = \mathscr{C}_\nu \cup \mathscr{C}_1 \cup \mathscr{C}_2$ is a set of vertex-disjoint triangles in $H_\tau - (V(\text{img}(h)) \cup \text{img}(g))$ such that $|\mathscr{C}| = |\mathscr{C}_\nu| + |\mathscr{C}_1| + |\mathscr{C}_2| = \Delta_\nu + loc + \rho$. Therefore, $\Delta_\tau \geq \Delta_\beta + \rho + loc$ where β is the index of an entry in $\mathbb{T}_{i-1}$ that maximizes Δ_β among all indices compatible with τ.

Let (h_τ, g_τ) be an optimum solution to $\mathscr{J}_\tau$. Let $\mathscr{C}_\tau$ be a maximum size set of vertex-disjoint triangles of $H_\tau - (V(\text{img}(h_\tau)) \cup \text{img}(g_\tau))$ with the required properties given by Definition 38. Without loss of generality, assume that $\mathscr{C}_\tau$ satisfies the properties mentioned in Lemma 11. As (h_τ, g_τ) is a canonical solution, g_τ restricted to the domain as S_i is Φ_τ. Let $\mathscr{C}'_1$ be the set of triangles in $\mathscr{C}_\tau$ that are completely contained in Q_i. Let $\mathscr{C}'_2$ be the set of triangles in $\mathscr{C}_\tau$ that have a vertex from Q_{i-1} and a vertex from Q_i. Let $\mathscr{C}'_3$ be the set of triangles in $\mathscr{C}_\tau$ that are contained in G_{i-1}. Then, $\Delta_\tau = |\mathscr{C}_\tau| = |\mathscr{C}'_1| + |\mathscr{C}'_2| + |\mathscr{C}'_3|$ where $|\mathscr{C}'_1| = loc$. If $|Prev| \neq 0$, then $|\mathscr{C}'_2| = 1$. Otherwise, $|\mathscr{C}'_2| = 0$. Let $\rho' = |\mathscr{C}'_2|$.

Let $Next''$ be the vertices of Q_{i-1} that are present in triangles in $\mathscr{C}'_2$. Note that $V(\mathscr{C}'_2) = Prev \cup Next''$. Let S'_{i-1} denote the set of variables from $X \cup Y \cup Z$ that are assigned to vertices from Q_{i-1} by g_τ. Clearly, $S_i \cap S'_{i-1} = \emptyset$ and $S'_{i-1} \subseteq S$. Let L'_{i-1} denote the set of variables that are assigned to paths that intersect with Q_{i-1} by h_τ. Let L''_{new} denote the set of variables that are assigned to paths that start from Q_{i-1} by h_τ. Let $Prev''$ denote the set of vertices of Q_{i-1} that are present in triangles in $\mathscr{C}_\tau$ that have a vertex in Q_{i-2}. Let Not'' be the vertices of $Q_{i-1} \setminus (V(\text{img}(h_\tau)) \cup \text{img}(g_\tau))$ that are not in any triangle in $\mathscr{C}_\tau$. Let S'' and L'' denote the sets of variables that have been assigned to vertices and paths, respectively, in G_{i-2} by h_τ. Define the functions $h''_1 : L_{i-1} \to Q_{i-1}$ and $h''_2 : L_{i-1} \to Q_{i-1}$ as follows. For each $w \in L_{i-1}$, $h''_1(w)$ is the first vertex from Q_{i-1} in the path $h_\tau(w)$ and $h''_2(w)$ is the second vertex (if it exists) from Q_{i-1} in the path $h_\tau(w)$.

Let β denote the index $(i-1, S'_{i-1}, L'_{i-1}, L''_{new}, Prev'', Next'', Not'', S'', L'', h''_1, h''_2)$ obtained from (h_τ, g_τ). Let (h', g') be the pair of functions obtained from (h_τ, g_τ) by setting the domain of h' to $L \cup (L'_{i-1} \setminus L''_{new})$ and reassigning $h'(w)$ to be the subpath of $h_\tau(w)$ contained in G_{i-1} for each $w \in L''_{i-1} \setminus L''_{new}$. Observe that each such path is of length at least two. Hence, by the definition of $\mathscr{J}_\beta$, (h', g') is a feasible solution of $\mathscr{J}_\beta$. Thus, $\Delta_\beta \geq |\mathscr{C}'_3|$. Therefore, $\Delta_\tau \leq \Delta_\beta + \rho' + loc$. Note that $\rho' = \rho$ as this number depends only on the size of $Prev$. Finally, by the definition of compatible entries, τ and β are compatible. This completes the proof of the claim. $\square$

This completes the description of the computation of entries of $\mathbb{T}$.

Pseudo Code for Filling $\mathbb{T}$. The procedure for computing $\mathbb{T}$ using Lemmas 39 and 42 is described in Algorithm 1. We obtain $opt(\mathscr{J})$ as the value Δ_τ where τ is the index of an entry in $\mathbb{T}_{q+1}$. Recall that there is only one index τ in $\mathbb{T}_{q+1}$ by Definitions 34 and 35.

Now, we are ready to formally prove the main result of this section.

Proposition 43. *There is an algorithm that solves* COLORFUL CONSTRAINED PATH ASSIGNMENT *in* $\mathscr{O}^*(2^{\mathscr{O}(|X|+|Y|+|Z|+|W|)})$ *time.*

Algorithm 1: Pseudocode for Computing $\mathbb{T}$

```
    Input: $(G, \{Q_1, \dots, Q_q\}, \pi, X, Y, Z, W, \Gamma, \Lambda_1, \Lambda_2, \Omega)$
    Output: Tables $\mathbb{T}_1, \dots, \mathbb{T}_{q+1}$
 1  Let $Q_{q+1} = \emptyset$.
 2  for each $i \in [q+1]$ do
 3      for each index $\tau = (i, S_i, L_i, L_{new}, Prev, Next, Not, S, L, h_1, h_2)$ of an entry in $\mathbb{T}_i$ do
 4          Initialize $\Delta_\tau = -\infty$.
 5          Let $loc = (|Q_i| - (|Prev| + |Next| + |Not| + |\mathrm{img}(h_1)| + |\mathrm{img}(h_2)| + |S_i|))/3$.
 6          if $\mathrm{dom}(\Phi_\tau) = S_i$ then
 7              if $i = 1$ then
 8                  Set $\Delta_\tau = loc$.
 9              else
10                  if there is no index $\nu$ in $\mathbb{T}_{i-1}$ compatible with $\tau$ then
11                      Set $\Delta_\tau = -\infty$.
12                  else
13                      for each index $\nu$ in $\mathbb{T}_{i-1}$ compatible with $\tau$ do
14                          if $|Prev| \neq 0$ then
15                              $\rho = 1$.
16                          else
17                              $\rho = 0$.
18                          if $\Delta_\tau < \Delta_\nu + \rho + loc$ then
19                              Set $\Delta_\tau = \Delta_\nu + \rho + loc$.
20          else
21              Set $\Delta_\tau = -\infty$.
22  return $\{\mathbb{T}_i : i \in [q+1]\}$.
```

Proof. Consider an instance $\mathscr{J} = (G, \mathscr{Q}, \pi, X, Y, Z, W, \Gamma, \Lambda_1, \Lambda_2, \Omega)$ of COLORFUL CONSTRAINED PATH ASSIGNMENT. Let n denote the number of vertices of G and $\{Q_1, \ldots, Q_q\}$ be the clique partition $\mathscr{Q}$ of G. Let k denote $|X|+|Y|+|Z|+|W|$. We compute tables $\mathbb{T}_i$ for each $i \in [q+1]$ where $q \leq n$ using Algorithm 1. Let s denote $|W|$ and r denote $|X|+|Y|+|Z|$. An entry in $\mathbb{T}_i$ is uniquely identified by the index $\tau = (i, S_i, L_i, L_{new}, Prev, Next, Not, S, L, h_1, h_2)$. We bound the number of choices for such indices in order to bound the size of $\mathbb{T}_i$. As S_i and S are disjoint subsets of $X \cup Y \cup Z$, the number of choices for pairs (S_i, S) is therefore at most 4^r. Similarly, the number of choices for triples (L_i, L_{new}, L) is at most 8^s. Since *Prev*, *Next* and *Not* are pairwise disjoint subsets of sizes at most two of Q_i (which has size at most n), there are at most $\mathscr{O}(n^6)$ choices for triples $(Prev, Next, Not)$. Finally, as $|\mathrm{dom}(L_i)| \leq 14$, it follows that there are at most $\mathscr{O}(n^{28})$ choices for functions (h_1, h_2). Therefore, the size of $\mathbb{T}_i$ is $\mathscr{O}^*(2^{\mathscr{O}(k)})$. Consequently, the size of $\mathbb{T}$ is $\mathscr{O}^*(2^{\mathscr{O}(k)})$. To fill an entry in $\mathbb{T}_i$, it is necessary to look at all entries in $\mathbb{T}_{i-1}$ and the rest is a polynomial time computation. Hence, the time taken to compute entries of $\mathbb{T}$ is also $\mathscr{O}^*(2^{\mathscr{O}(k)})$. Then, by examining the (unique) index τ of $\mathbb{T}_{q+1}$, the final answer Δ_τ can be obtained. If Δ_τ is $-\infty$, then $\mathscr{J}$ has no feasible solution. Otherwise, $opt(\mathscr{J}) = \Delta_\tau$. Therefore, the overall running time of the algorithm is $\mathscr{O}^*(2^{\mathscr{O}(k)})$. □

3.4.5 Putting it all Together: Proof of Theorem 1

The main result (Theorem 10) now follows from Propositions 14, 19, 27 and 43.

Theorem 9. *10* **(restated)** CYCLE PACKING *parameterized by the size t of a proper interval deletion set can be solved in $\mathscr{O}^*(2^{\mathscr{O}(t \log t)})$ time.*

Proof. Consider an instance $\mathscr{I} = (G, T, r)$ of CYCLE PACKING with parameter $|T|$. Using Proposition 14, we create a set of $2^{\mathscr{O}(|T| \log |T|)}$ instances of CONSTRAINED CYCLE PACKING such that $\mathscr{I}$ is a yes-instance if and only if at least one of the instances is a yes-instance. This step takes $\mathcal{O}^*(2^{\mathcal{O}(|T|\log|T|)})$ time. Each such instance $\mathscr{I}'$ is of the form $(G', T', \mathscr{Q}, \mathscr{T}', A, B, r)$ with parameter $|T'|$. The components of $\mathscr{I}'$ have the following interpretation.

- G' is a subgraph of G induced by a subset of vertices that contains $V(G-T)$ and $T' \subseteq T$.
- T' is a proper interval deletion set of G' and $\mathscr{Q}$ is a clique partition of $G'-T'$.
- $\mathscr{T}' = \{T_1, \ldots, T_\ell\}$ is a partition of T' into ℓ non-empty subsets for some $\ell \leq |T'|$.
- A is a set $\{\sigma_1, \ldots, \sigma_\ell\}$ where σ_i is a permutation of T_i for each $i \in [\ell]$.
- B is a set $\{\gamma_i \in \{0,1,2,3\}^{|T_i|} : i \in [\ell]\}$.

For each instance $\mathscr{I}' = (G', T', \mathscr{Q}, \mathscr{T}', A, B, r)$ (with parameter $|T'|$) created in the previous step, we use Proposition 19 to construct (in $\mathcal{O}^*(2^{\mathcal{O}(|T'|)})$ time) a set of $2^{|T|}$ instances of Constrained Path Assignment such that $\mathscr{I}'$ is a yes-instance if and only if at least one of the instances $\mathscr{J}$ satisfies $opt(\mathscr{J}) \geq r - \ell$. Each such instance $\mathscr{J}$ is of the form $(H, \mathscr{Q}, \pi, X, Y, Z, W, \Gamma, \Lambda_1, \Lambda_2, \Omega)$ whose components have the following interpretation. The parameter of $\mathscr{J}$ is $k = |X|+|Y|+|Z|+|W|$.

- H is the graph $G'-T'$ with clique partition $\mathscr{Q} = \{Q_1, \ldots, Q_q\}$ and proper interval ordering π.
- Each of X, Y, Z and W is a set of $\mathcal{O}(|T'|)$ variables.
- Γ is a function from $X \cup Y \cup Z$ to a collection of subsets of $V(H)$.
- Λ_1 and Λ_2 are functions from W to collections of subsets of $V(H)$.
- Ω is a bijection from X to Y.

For each instance $\mathscr{J} = (H, \mathscr{Q}, \pi, X, Y, Z, W, \Gamma, \Lambda_1, \Lambda_2, \Omega)$ (with parameter $k = |X|+|Y| +|Z|+|W|$) created in this step, we use Proposition 27 to create $\mathcal{O}^*(k!\,e^k k^{\mathcal{O}(\log k)})$ instances of COLORFUL CONSTRAINED PATH ASSIGNMENT such that at least one of these instances $\widehat{\mathscr{J}}$ satisfies $opt(\mathscr{J}) = opt(\widehat{\mathscr{J}})$. Each such instance $\widehat{\mathscr{J}}$ (with parameter k) is an instance $(H, \mathscr{Q}, \pi, X, Y, Z, W, \widehat{\Gamma}, \Lambda_1, \Lambda_2, \Omega)$ of CONSTRAINED PATH ASSIGNMENT where $\widehat{\Gamma}$ is a function from $X \cup Y \cup Z$ to a collection of subsets of $V(H)$. Also, for each variables $s \in X \cup Y \cup Z$, $\widehat{\Gamma}(s) \subseteq \Gamma(s)$. Moreover, for each pair s,t of distinct variables in $X \cup Y \cup Z$, we have $\widehat{\Gamma}(s) \cap \widehat{\Gamma}(t) = \emptyset$.

Finally, we solve each such instance $\widehat{\mathscr{J}}$ of CONSTRAINED PATH ASSIGNMENT with parameter k using Proposition 43 in $\mathscr{O}^*(2^{\mathscr{O}(k)})$ time. Therefore, the overall running time of the algorithm for solving the instance $\mathscr{I}$ of CYCLE PACKING is $\mathscr{O}^*(2^{\mathscr{O}(|T|\log|T|)})$. □

3.5 CYCLE COVER and PATH COVER parameterized by proper interval deletion set

In the previous subsections we looked at the CYCLE PACKING problem. In this subsection we shift our attention to the CYCLE/PATH COVER problem. We study these problems parameterized by the same structural parameter, *proper interval deletion set*. In particular, we show that PATH COVER and CYCLE COVER are fixed-parameter tractable (FPT) when parameterized by k, the size of a proper interval deletion set (a set of vertices whose deletion results in a proper interval graph). For this purpose, we design an algorithm with $2^{\mathscr{O}(k\log k)}n^{\mathscr{O}(1)}$ running time for each of these problems. Notice that we denote the size of proper interval deletion set by k for this particular subsection. Our algorithms use several interesting properties paths/cycles of proper interval graphs and a dynamic programming procedure over clique partitions to solve these problems in the mentioned FPTtime. As a consequence, we get the same fixed-parameter tractability results for HAMILTONIAN CYCLE and HAMILTONIAN PATH problems.

PATH COVER **Parameter:** $|T|$

Input: A graph H, a proper interval deletion set T of H and an integer r.

Question: Does H have a path cover of size at most r?

CYCLE COVER **Parameter:** $|T|$

Input: A graph H, a proper interval deletion set T of H and an integer r.

Question: Does H have a cycle cover of size at most r?

We show that these problems are FPT and this is the main result of this section. By parameterizing PATH COVER and CYCLE COVER with respect to the size of a proper interval deletion set as parameter, we attempt to understand the complexity of the problem on *almost proper interval graphs*. Recently, Chaplick et al. [19] obtained polynomial kernels and compression algorithms for PATH COVER and CYCLE COVER parameterized by a different measure of similarity with proper interval graphs. Our FPT algorithms also add to this study of structural parameterizations for these classical problems.

Theorem 10. PATH COVER *and* CYCLE COVER *parameterized by the size k of a proper interval deletion set can be solved in* $\mathcal{O}^*(2^{\mathcal{O}(k \log k)})$ *time.*

We assume that the proper interval deletion set T is part of the input. This assumption is reasonable as given a graph H and an integer k, there is an algorithm that, in $\mathcal{O}^*(6^k)$ time, outputs a proper interval deletion set of size at most k (if one exists) [18, 87]. Our algorithms use several interesting properties of proper interval graphs and a dynamic programming procedure over clique partitions to solve these problems in the mentioned time. As a consequence, we get the same fixed-parameter tractability results for HAMILTONIAN CYCLE and HAMILTONIAN PATH problems with the same parameterization.

Overview of our Algorithm and Techniques. Consider an instance $\mathscr{I} = (H, T, r)$ of PATH COVER and CYCLE COVER. Let $\mathscr{P}$ be a minimum path cover of H that we are looking for. We first guess the following properties of $\mathscr{P}$. Intialize the set of variables $\mathscr{S}$

to be the empty set. Let G denote $H - T$.

1. We guess the number ℓ of paths in $\mathscr{P}$ that have a vertex from T. Clearly, $\ell \leq k$ and the number of choices for ℓ is k. Let $\mathscr{P}_m = \{P_1, \dots, P_\ell\}$ denote the set of these paths.

2. For each $P_i \in \mathscr{P}_m$, we guess if P_i has zero, one or two endpoints in T. The number of possible choices in this step is $2^{\mathscr{O}(k)}$.

3. For each $P_i \in \mathscr{P}_m$, we guess the order $\lambda(P_i)$ of vertices of $V(P_i) \cap T$. The number of possible choices in this step is $2^{\mathscr{O}(k \log k)}$.

4. For each $P_i \in \mathscr{P}_m$ that starts at a vertex in G, we add the variable $S^i(\$, t)$ to $\mathscr{S}$ where t is the first vertex according to $\lambda(P_i)$. The variable $S^i(\$, t)$ indicates that we need to assign it a path in G that ends in a neighbour of t.

5. For each $P_i \in \mathscr{P}_m$ that ends at a vertex in G, we add the variable $S^i(t', \$)$ to $\mathscr{S}$ where t' is the last vertex according to $\lambda(P_i)$. The variable $S^i(t', \$)$ indicates that we need to assign it a path in G that starts in a neighbour of t'.

6. For each $P_i \in \mathscr{P}_m$, for each pair of vertices $t \in T$ and $t' \in T$ that are consecutive according to $\lambda(P_i)$, we add a variable $S^i(t, t')$ to $\mathscr{S}$ indicating that $S^i(t, t')$ should be assigned a path in G that is between a neighbour of t and a neighbour of t'.

Clearly, $|\mathscr{S}| = \mathscr{O}(k)$ and the task of finding a minimum path cover of H reduces to the problem of finding an assignment of vertex-disjoint paths in G to the variables in $\mathscr{S}$ satisfying the appropriate endpoint constraints while minimizing the size of a minimum path cover of $G[X]$ where $V(G) \setminus X$ is the set of vertices that are in the path assigned to some variable in $\mathscr{S}$.

Similarly, let $\mathscr{C}$ be a minimum cycle cover of H that we are looking for. We first guess the following properties of $\mathscr{C}$. Intialize the set of variables $\mathscr{S}$ to be the empty set.

1. We guess the number ℓ of cycles in $\mathscr{C}$ that have a vertex from T. Clearly, $\ell \leq k$ and the number of choices for ℓ is k. Let $\mathscr{C}_m = \{C_1, \ldots, C_\ell\}$ denote the set of these cycles.

2. For each $C_i \in \mathscr{C}_m$, we guess the order $\lambda(C_i)$ of vertices of $V(C_i) \cap T$. The number of possible choices in this step is $2^{\mathscr{O}(k \log k)}$.

3. For each $C_i \in \mathscr{C}_m$ with $|V(C_i) \cap T| \geq 2$, for each pair $t \in T$ and $t' \in T$ that are consecutive according to $\lambda(C_i)$, we add a variable $S^i(t,t')$ to $\mathscr{S}$ indicating that $S^i(t,t')$ should be assigned a path in G that is between a neighbour of t and a neighbour of t'. Note that the first and last vertices in $\lambda(C_i)$ are also considered to be consecutive.

4. For each $C_i \in \mathscr{C}_m$ with $V(C_i) \cap T = \{t\}$, we add a variable $S^i(t,t)$ to $\mathscr{S}$ indicating that $S^i(t,t)$ should be assigned a path in G that is between two neighbours of t.

Once again, $|\mathscr{S}| = \mathscr{O}(k)$ and the task of finding a minimum cycle cover of H reduces to the problem of finding an assignment of vertex-disjoint paths in G to the variables in $\mathscr{S}$ satisfying the appropriate endpoint constraints while minimizing the size of a minimum cycle cover of $G[X]$ where $V(G) \setminus X$ is the set of vertices that are in the path assigned to some variable in $\mathscr{S}$.

Thus, both the problems of finding a minimum path cover and a minimum cycle cover of H boil down to the problem of finding certain constrained paths in G which is a proper interval graph. We first show that these paths are very structured due to the properties given by a proper interval ordering. Then, we describe a dynamic programming procedure to find such structured paths.

Road Map. In Section 3.5, we list some fundamental terminology and properties related to proper interval graphs. In Section 3.5.1, we list important properties of paths and cycles in proper interval graphs. In Section 3.6, we describe an FPT algorithm for PATH COVER and in Section 3.7, we describe an FPT algorithm for CYCLE COVER. Finally, we conclude with some remarks.

Notations: We state the following notations that are specifically used throughout this section. A permutation/ordering π of a set $A = \{a_1, \ldots, a_{|A|}\}$ is denoted by a sequence $(a_{i_1}, a_{i_2}, \ldots, a_{i_{|A|}})$ where $\{i_1, i_2, \ldots, i_{|A|}\} = [|A|]$. Given a permutation π of A and an element $a \in A$, $\pi(a)$ denotes the position (between 1 to $|A|$) of a in π.

3.5.1 Structure of Path Covers and Cycle Covers

Recall that a path cover $\mathscr{P}$ (cycle cover $\mathscr{C}$) of a graph H is a set of vertex-disjoint paths (cycles) in H such that $V(\mathscr{P}) = V(H)$ ($V(\mathscr{C}) = V(H)$). Consider an instance $\mathscr{I} = (H, T, r)$ of PATH COVER and CYCLE COVER. Let π and $\mathscr{Q} = \{Q_1, \cdots, Q_q\}$ denote the proper interval ordering and clique partition of $G = H - T$, respectively, obtained in polynomial time [44, 58]. Let $T = \{t_1, \ldots, t_k\}$.

Paths and Cycles in Proper Interval Graphs

In this section, we list some fundamental properties of paths and cycles in proper interval graphs. The following result is well-known in the proper interval graphs literature.

Proposition 44 ([7, 20, 53]). *Every connected proper interval graph has a Hamiltonian path, and a proper interval graph has a Hamiltonian cycle if and only if it is 2-connected with at least three vertices.*

This results leads to the following proposition.

Proposition 45. *Let G be a connected proper interval graph with proper interval ordering π where u is the leftmost vertex in G and v is the rightmost vertex in G. Then, G has a Hamiltonian path from u to v.*

Definition 46. ***(Monotone path)*** *Let G be a proper interval graph with proper interval ordering π. A path $P = (v_1, \dots, v_r)$ in G is called* monotone *if $\pi(v_1) < \pi(v_2) < \cdots < \pi(v_r)$ or $\pi(v_1) > \pi(v_2) > \cdots > \pi(v_r)$.*

The following holds from Proposition 45.

Observation 47. *If P is a path in a proper interval graph G with proper interval ordering π, then there is a monotone path P' in G with $V(P) = V(P')$.*

Next, we study the structures of cycles in proper interval graphs.

Definition 48. ***(Starting and ending vertices of a cycle)*** *Let $C = (v_1, \dots, v_p)$ be a cycle in a proper interval graph G with proper interval ordering π. Then, we say that C starts at v_i and ends at v_j where v_i is the leftmost vertex in $V(C)$ and v_j is the rightmost vertex in $V(C)$.*

Definition 49. ***(2-monotone cycle)*** *Let G be a proper interval graph with proper interval ordering π. A cycle $C = (v_1, v_2, \dots, v_i, v_{i+1}, \dots, v_r)$ in G is called* 2-monotone *if there is an integer $i \in [r]$ such that $(v_1, v_2, \dots, v_i)$ and $(v_1, v_r, \dots, v_i)$ are monotone paths that are internally vertex-disjoint and start and end at the same vertices.*

Observe that a 2-monotone cycle C can be obtained by the concatenation of two maximal monotone paths and this pair of subpaths is unique for C.

Proposition 50. *If G is a 2-connected proper interval graph with proper interval ordering π, then G has a 2-monotone Hamiltonian cycle.*

Proof. We claim that $C = (\pi(1), \dots, \pi(n))$ is the required Hamiltonian cycle. Consider the sequences $P_1 = (\pi(1), \pi(3), \pi(5), \dots, \pi(n))$ and $P_2 = (\pi(1), \pi(2), \pi(4), \dots, \pi(n))$. Suppose $\pi(i)$ is not adjacent to $\pi(i+1)$ for some i. Then, from Proposition 110, there does not exist a pair $i' \leq i, j' \geq i+1$ such that $\pi(i')$ and $\pi(j')$ are adjacent. Thus, there is no edge between a vertex in $\mathscr{C}_1 = \bigcup_{j=1}^{i} \{\pi(j)\}$ and a vertex in $\mathscr{C}_2 = \bigcup_{j=i+1}^{n} \{\pi(j)\}$

implying that G is disconnected. This leads to a contradiction. Similarly, suppose for some i, $\pi(i)$ is not adjacent to $\pi(i+2)$. Then, from Proposition 110, there does not exist a pair $i' \leq i, j' \geq i+2$ such that $\pi(i')$ and $\pi(j')$ are adjacent. Thus, there is no edge between a vertex in $\mathscr{C}_3 = \bigcup\limits_{j=1}^{i} \{\pi(j)\}$ and a vertex in $\mathscr{C}_4 = \bigcup\limits_{j=i+2}^{n} \{\pi(j)\}$ in $G - \pi(i+1)$ implying that G is not 2-connected leading to a contradiction. Thus, P_1 and P_2 are paths in G. Clearly, P_1 and P_2 are internally vertex-disjoint and monotone by definition. Thus, C is a 2-monotone Hamiltonian cycle. □ □

Observation 51. *If C is a cycle in a proper interval graph G with proper interval ordering π, then there is a 2-monotone cycle C' in G such that $V(C) = V(C')$.*

Next, we define the notion of i-monotone paths in proper interval graphs.

Definition 52. ***(i-Monotone path)*** *Let G be a proper interval graph with proper interval ordering π. For a positive integer i, a path P is called* i-monotone *if P is the concatenation of i monotone paths.*

Observe that monotone paths are 1-monotone.

Proposition 53. *If P is a path from a vertex s in a proper interval graph G with proper interval ordering π, then there is an i-monotone path P' in G from s with $V(P) = V(P')$ for some $i \in [2]$.*

Proof. Let v denote the other endpoint of P. Let v_l and v_r be the leftmost and rightmost vertices of P. Note that v_l, v_r, s, v may not all be distinct. Let Z denote the set of these (at most 4) vertices.

Case 1: The order of appearance of Z in P is s, v_r, v_l, v.

Let S' be the subpath of P from s to v_r. Let $X = \{v \in V(S') : \pi(v) \geq \pi(s)\}$ and $Y = (V(P) \setminus X) \cup \{v_r\}$. Observe that $G[X]$ and $G[Y]$ are connected. Therefore, there are monotone paths S_1 from s to v_r and and S_2 from v_r to v_l such that $V(S_1) \cup V(S_2) = V(P)$. Then, the concatenation of paths S_1 and S_2 is an i-monotone (s, v_l)-path P' with $V(P) = V(P')$ for some $i \in [2]$. See Figure 3.10 for an illustration.

Case 2: The order of appearance of Z in P is s, v_l, v_r, v.

Let S' be the subpath of P from s to v_l. Let $X = \{v \in V(S') : \pi(v) \leq \pi(s)\}$ and $Y = (V(P) \setminus X) \cup \{v_l\}$. Observe that $G[X]$ and $G[Y]$ are connected. Therefore, there are monotone paths S_1 from s to v_l and S_2 from v_l to v_r such that $V(S_1) \cup V(S_2) = V(P)$. Then, the concatenation of paths S_1 and S_2 is an i-monotone (s, v_r)-path P' with $V(P) = V(P')$ for some $i \in [2]$. □ □

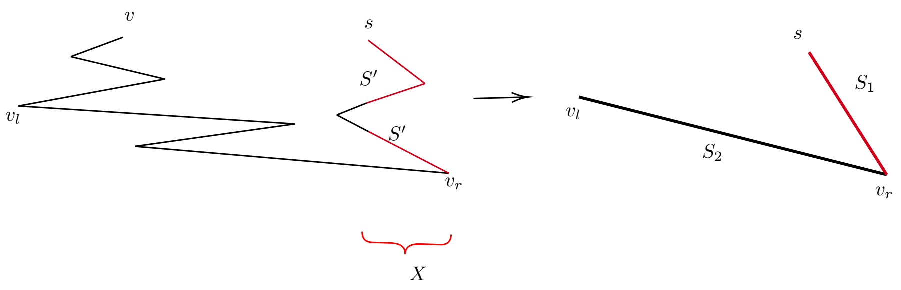


Fig 3.10: A 2-monotone path from s

Proposition 54. *If P is a (s,t)-path in a proper interval graph G with proper interval ordering π, then there is an i-monotone (s,t)-path P' in G with $V(P) = V(P')$ for some $i \in [3]$.*

Proof. Let v_l and v_r be the left most and right most vertices of P. Note that v_l, v_r, s, t may not all be distinct. Let the four vertices appear in P in the order (s, a, b, t) where $a, b \in \{v_l, v_r\}$. Define the following subpaths of P: S_1 from s to a, S_2 from a to b and S_3 from b to t. Let $X = \{v \in V(S_1) : \pi(v) \leq \pi(s)\}$ and $Y = \{v \in V(S_3) : \pi(v) \geq \pi(t)\}$. Let $Z = V(P) \setminus (X \cup Y) \cup \{a, b\}$. Observe that $G[X]$, $G[Y]$ and $G[Z]$ are connected. Therefore, there are monotone paths S'_1 from s to a, S'_2 from a to b and S'_3 from b to t. Thus, the concatenation of paths S'_1, S'_2 and S'_3 is an i-monotone path P' from s to t for some $i \in [3]$. See Figure 3.11 for an illustration. □ □

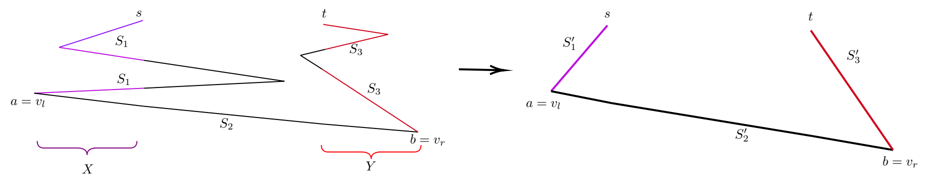


Fig 3.11: A 3-monotone path between s and t

3.5.2 Canonical Minimum Path Covers

For a path cover $\mathscr{P}$ of H, define the following sets.

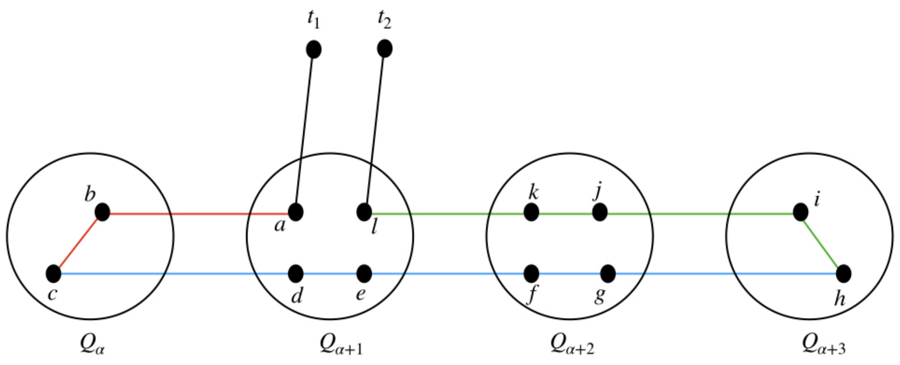
- $\mathscr{P}_o = \{P_i \in \mathscr{P} : V(P_i) \cap T = \emptyset\}$, the set of paths in $\mathscr{P}$ that are completely contained in G.

- $\mathscr{P}_m = \mathscr{P} \setminus \mathscr{P}_o$, the set of paths in $\mathscr{P}$ that have at least one vertex from T.

- $M(\mathscr{P})$ is the set of maximal subpaths of paths in $\mathscr{P}_m$ that are contained in G. That is, for each P in $\mathscr{P}_m$, a subpath S of P with $V(S) \subseteq V(G)$ is in $M(\mathscr{P})$ if and only if there is no subpath S' of P such that $V(S') \subseteq V(G)$ and $V(S) \subset V(S')$.

- $S(\mathscr{P})$ is the set of maximal subpaths of paths in $M(\mathscr{P})$ that are monotone. That is, for each P in $M(\mathscr{P})$, a subpath S of P is in $S(\mathscr{P})$ if and only if S is monotone and there is no monotone subpath S' of P such that $V(S) \subset V(S')$.

We refer to elements of $S(\mathscr{P})$ as segments of $\mathscr{P}$. In the example shown below, $\mathscr{P}_m$ contains a path P from t_1 to t_2 and $M(\mathscr{P}) = \{S\}$ where $S = (a,b,c,d,e,f,g,h,i,j,k,l)$. The set of segments is $S(\mathscr{P}) = \{S_1, S_2, S_3\}$ where $S_1 = (c,b,a)$, $S_2 = (c,d,e,f,g,h)$ and $S_3 = (l,k,j,i,h)$.

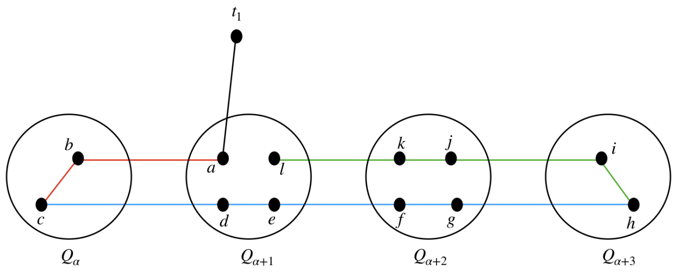


Definition 55. ***(Pseudo-consecutive vertices)*** *Two vertices $u, v \in T$ are said to be pseudo-consecutive if u and v are in the same path P in $\mathscr{P}$ and there is no other vertex of T that is in the subpath of P between u and v.*

In the example shown above, t_1 are t_2 are pseudo-consecutive.

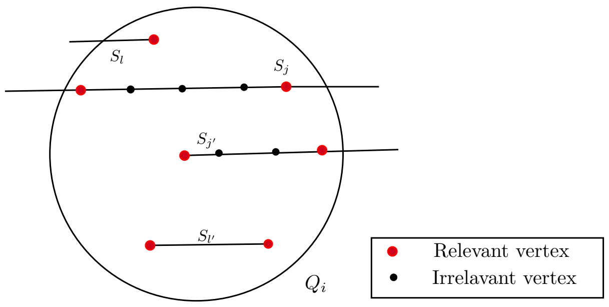
Definition 56. ***(Pseudo-adjacent vertices)*** *Let y be a vertex in G that is an endpoint of some path in $\mathscr{P}$. A vertex $x \in T$ is said to be pseudo-adjacent to y if x and y are in the same path P in $\mathscr{P}$ and there is no other vertex of T that is in the subpath of P between y and x.*

In the example shown below, t_1 are l are pseudo-adjacent.



Definition 57. (**Relevant and irrelevant vertices in** Q_i) *For a path S in $S(\mathscr{P}) \cup \mathscr{P}_o$ that contains at least one vertex from Q_i, the set $R_i(\mathscr{P}, S)$ of relevant vertices is $V(S) \cap Q_i$ if $|V(S) \cap Q_i| \leq 2$, otherwise $R_i(\mathscr{P}, S)$ consists of the leftmost and the rightmost vertices of $V(S) \cap Q_i$. The collection $\mathscr{R}_i(\mathscr{P})$ of relevant vertices contains the set $R_i(\mathscr{P}, S)$ of every path S in $S(\mathscr{P}) \cup \mathscr{P}_o$ that contains at least one vertex from Q_i. A vertex in Q_i that is not in $\mathscr{R}_i(\mathscr{P})$ called irrelevant.*

An example is shown in the following figure.



Definition 58. ***(Nice path cover)*** *A path cover $\mathscr{P}$ is said to be nice if the following properties hold.*

- **(NP.1)** *Every path in $\mathscr{P}_o$ is monotone.*
- **(NP.2)** *For any $i \in [q]$, there is at most one path P in $\mathscr{P}_o$ such that $Q_i \cap V(P) \neq \emptyset$.*
- **(NP.3)** *For every path P in $\mathscr{P}_m$, for every pair of pseudo-consecutive modulator vertices t, t' in P that are not consecutive in P, the maximal subpath of P between t and t' that is contained in G is i-monotone for some $i \in [3]$.*
- **(NP.4)** *For every path P in $\mathscr{P}_m$ starting (or ending) at a vertex s in G whose pseudo-adjacent modulator vertex is t, the maximal subpath of P contained in G that is between s and the neighbour of t in P is i-monotone for some $i \in [2]$.*
- **(NP.5)** *For any $i \in [q]$, if $|Q_i| > 10k$, then each segment $S \in S(\mathscr{P})$ with $V(S) \cap Q_i \neq \emptyset$ that neither starts nor ends at a vertex in Q_i satisfies $|V(S) \cap Q_i| \geq 2$.*

Lemma 59. *Given a path cover $\mathscr{P}$ of H, a nice path cover $\mathscr{P}^*$ of H with $|\mathscr{P}^*| \leq |\mathscr{P}|$ can be obtained in polynomial time.*

Proof. Every path P in $\mathscr{P}_o$ not satisfying **(NP.1)** can be replaced by a path P' satisfying **(NP.1)** using Observation 47. If there are two paths P and P' in $\mathscr{P}_o$ containing vertices from Q_i, then they can be replaced by the Hamiltonian path of $G[V(P) \cup V(P')]$ given by Proposition 45. Applying these replacement rules as long as possible results in a path cover $\mathscr{P}^*$ of H with $|\mathscr{P}^*| \leq |\mathscr{P}|$ satisfying **(NP.1)** and **(NP.2)**. Suppose S is a subpath of a path in $\mathscr{P}^*_m$ between two pseudo-consecutive modulator vertices u and v. Let s and t be the neighbors of u and v respectively in S. If the subpath S' of S between s and t is not i-monotone for some $i \in [3]$, we replace S' by an i-monotone (s,t)-path S'' with $V(S) = V(S'')$ for some $i \in [3]$ given by Proposition 54. Applying this replacement rule as long as possible results in a path cover $\mathscr{P}^\star$ of H with $|\mathscr{P}^\star| \leq |\mathscr{P}^*|$ satisfying **(NP.1)**, **(NP.2)** and **(NP.3)**. Similarly, using Proposition 53, we may assume that this path cover also satisfies **(NP.4)**. Suppose s is a vertex in G that is an endpoint of some path $P \in \mathscr{P}^\star$. Let t be its pseudo-adjacent modulator vertex and let S be the subpath of P between s and t. Let v be the neighbour of t in S and S' denote the subpath of S from v to s. Then, from Proposition 53, there is an i-monotone path S'' with v as an endpoint and $V(S'') = V(S')$ for some $i \in [2]$. We replace the subpath S' by S'' in P. Applying this replacement rule as long as possible results in a path cover $\widehat{\mathscr{P}}$ of H with $|\widehat{\mathscr{P}}| \leq |\mathscr{P}^\star|$ satisfying **(NP.1)**, **(NP.2)**, **(NP.3)** and **(NP.4)**.

Suppose there is a segment $S \in S(\widehat{\mathscr{P}})$ that neither starts nor ends in Q_i but has exactly one vertex x from Q_i. Let P be the path in $\mathscr{P}$ that contains S. As $|Q_i| > 10k$, from Lemma 61, there is an irrelevant vertex $y \in Q_i$ that is in a monotone path $S' \in S(\mathscr{P}) \cup \mathscr{P}_o$. Let P' be the path in $\mathscr{P}$ that contains S'. We delete y from S' to get another monotone path S''. We replace S' with S'' in P' and P' remains a path with the same monotonicity. Then, we replace $S = (x)$ with $S^* = (x,y)$ if $\pi(x) \leq \pi(y)$, otherwise we replace $S = (x)$ with $S^* = (y,x)$. We claim that P remains a path with the same monotonicity. Without loss of generality, let $S^* = (y,x)$ and let the last vertex of $V(P) \cap Q_{i-1}$ be z. Since z is adjacent to x, z is also adjacent to y as $\pi(y) \leq \pi(x)$. The other case when $S^* = (x,y)$ can be argued similarly. Applying this replacement rule as long as possible results in a path cover $\widehat{\mathscr{P}^\star}$ of H with $|\widehat{\mathscr{P}^\star}| \leq |\widehat{\mathscr{P}}|$ satisfying **(NP.1)**, **(NP.2)**, **(NP.3)**, **(NP.4)** and **(NP.5)**. □ □

Corollary 60. *Given a minimum path cover $\mathscr{P}$ of H, a nice minimum path cover $\mathscr{P}^*$ of H with $|\mathscr{P}^*| \leq |\mathscr{P}|$ can be obtained in polynomial time.*

Lemma 61. *If $\mathscr{P}$ is a nice minimum path cover of H, then $|S(\mathscr{P})| \leq 4k$ and for any $i \in [q]$, $|\mathscr{R}_i(\mathscr{P})| \leq 8k+2$.*

Proof. From Propositions 53 and 54, each vertex $v \in T$ is preceeded by at most 3 segments and succeeded by at most 3 segments in $\mathscr{P}$. Further, each vertex in T that has no other vertex in T preceeding it is preceeded by at most 2 segments. Similarly, each vertex in T that has no other vertex in T succeeding it is succeeded by at most 2 segments. Thus $S(\mathscr{P}) \leq 4k$. In any minimum path cover, there is at most one path in $\mathscr{P}_o$ that intersects Q_i (from **NP.2**) and it has at most 2 relevant vertices in it. Hence $|\mathscr{R}_i(\mathscr{P})| \leq 8k+2$. $\square$ $\square$

Definition 62. ***(Leftmost and rightmost set of vertices)*** *Consider a subset S of vertices of G. If $|S| > 10k$, then let $LM(S)$ denote the $10k$ leftmost vertices of S and $RM(S)$ denote the $10k$ rightmost vertices of S. Otherwise, $LM(S) = RM(S) = S$.*

Definition 63. ***(Boundary vertices of Q_i)*** *Consider the following sets.*

- $L^i = LM(Q_i)$.
- $R^i = RM(Q_i)$.
- $L^i_x = LM(Q_i \cap N(x))$ *for each* $x \in T$.
- $R^i_x = RM(Q_i \cap N(x))$ *for each* $x \in T$.
- $L^i_{xy} = LM(Q_i \cap N(x) \cap N(y))$ *for each* $x, y \in T$.
- $R^i_{xy} = RM(Q_i \cap N(x) \cap N(y))$ *for each* $x, y \in T$.

The set $B(Q_i) = L^i \cup R^i \bigcup_{x \in T} (L^i_x \cup R^i_x) \bigcup_{y,z \in T} (L^i_{yz} \cup R^i_{yz})$ is called the boundary vertices of Q_i.

Notice that for any $i \in [q]$, the size of $B(Q_i)$ is $\mathscr{O}(k^3)$.

Definition 64. *For every vertex $v \in V(G)$, let $\rho(v)$ denote the tuple (m_1, m_2) defined as follows.*

- m_1 *is the vertex in* T *that preceeds* v *in the path in* $\mathscr{P}$ *that contains* v*. If no such vertex exists, then* $m_1 = \$$.

- m_2 *is the vertex in* T *that succeeds* v *in the path in* $\mathscr{P}$ *that contains* v*. If no such vertex exists, then* $m_2 = \$$.

Definition 65. ***(Canonical path cover)*** *A nice path cover* $\mathscr{P}$ *is said to be canonical if for each* $i \in [q]$ *and each* $S \in S(\mathscr{P})$, $\mathscr{R}_i(\mathscr{P}, S) \subseteq B(Q_i)$ *and the following properties are satisfied.*

- **(CP.1)** *For a segment* $S \in S(\mathscr{P})$ *passing through* Q_i*, its first and last vertices in* Q_i *are in* $L^i \cup R^i$.

- **(CP.2)** *For a segment* $S \in S(\mathscr{P})$ *that ends in* Q_i*, the following hold.*

  - *If* $S \cap Q_i = (a)$ *and* $\rho(a) = (\$, \$)$*, then* $a \in L^i$.
  - *If* $S \cap Q_i = (a)$ *and* $\rho(a) = (m_1, m_2)$*, then* $a \in L^i_{m_1 m_2} \cup R^i_{m_1 m_2}$.
  - *If* $S \cap Q_i = (a)$ *and* $\rho(a) = (m_1, \$)$*, then* $a \in L^i_{m_1} \cup R^i_{m_1}$.
  - *If* $S \cap Q_i = (a)$ *and* $\rho(a) = (\$, m_2)$*, then* $a \in L^i_{m_2} \cup R^i_{m_2}$.
  - *If* $\mathscr{R}_i(\mathscr{P}, S) = (a, b)$ *and* $\rho(a) = (m_1, \$)$, $\rho(b) = (\$, m_2)$*, then* $a \in L^i_{m_1} \cup R^i_{m_1}, b \in R^i_{m_2} \cup L^i_{m_2}$.
  - *If* $\mathscr{R}_i(\mathscr{P}, S) = (a, b)$ *and* $\rho(a) = (\$, \$)$, $\rho(b) = (\$, m_2)$*, then* $a \in L^i, b \in R^i_{m_2} \cup L^i_{m_2}$.
  - *If* $\mathscr{R}_i(\mathscr{P}, S) = (a, b)$ *and* $\rho(a) = (m_1, \$)$, $\rho(b) = (\$, \$)$*, then* $a \in L^i_{m_1} \cup R^i_{m_1}, b \in R^i \cup L^i$.
  - *If* $\mathscr{R}_i(\mathscr{P}, S) = (a, b)$ *and* $\rho(a) = (\$, \$)$, $\rho(b) = (\$, \$)$*, then* $a \in L^i, b \in R^i$.

- **(CP.3)** *For a segment* $S \in S(\mathscr{P})$ *that starts in* Q_i*, the following hold.*

  - *If* $S \cap Q_i = (a)$ *and* $\rho(a) = (\$, \$)$*, then* $a \in R^i$.
  - *If* $S \cap Q_i = (a)$ *and* $\rho(a) = (m_1, \$)$*, then* $a \in L^i_{m_1} \cup R^i_{m_1}$.

– *If $S \cap Q_i = (a)$ and $\rho(a) = (\$, m_2)$, then $a \in R^i_{m_2} \cup L^i_{m_2}$.*

– *If $\mathscr{R}_i(\mathscr{P}, S) = (a,b)$ and $\rho(a) = (\$,\$)$, $\rho(b) = (\$, m_2)$, then $a \in L^i \cup R_i, b \in R^i_{m_2} \cup L^i_{m_2}$.*

– *If $\mathscr{R}_i(\mathscr{P}, S) = (a,b)$ and $\rho(a) = (m_1, \$)$, $\rho(b) = (\$,\$)$, then $a \in L^i_{m_1} \cup R^i_{m_1}, b \in R^i$.*

– *If $\mathscr{R}_i(\mathscr{P}, S) = (a,b)$ and $\rho(a) = (\$,\$)$, $\rho(b) = (\$,\$)$, then $a \in L^i, b \in R^i$.*

Lemma 66. *Given a nice path cover $\mathscr{P}$ of H, a canonical path cover $\mathscr{P}^*$ of H with $|\mathscr{P}^*| \leq |\mathscr{P}|$ can be obtained in polynomial time.*

Proof. From Lemma 61, as $\mathscr{P}$ is a nice minimum path cover of H, we have $|S(\mathscr{P})| \leq 4k$ and for any $i \in [q]$, $|\mathscr{R}_i(\mathscr{P})| \leq 8k+2$. For any $i \in [q]$ such that $|Q_i| \leq 10k$, Properties **(CP.1)**, **(CP.2)** and **(CP.3)** vacously hold as $L^i = R^i = Q_i$, $L^i_x = R^i_x = Q_i \cap N(x)$, $L^i_{xy} = R^i_{xy} = Q_i \cap N(x) \cap N(y)$. Now, consider a clique Q_i such that $|Q_i| > 10k$.

Suppose $S(\mathscr{P})$ has a segment S that intersects Q_i but neither starts nor ends in it. Suppose the first vertex a in $V(S) \cap Q_i$ is not in L^i. Since Q_i has more than $10k$ vertices, $|L^i| = 10k$ and L^i has a vertex a' that is irrelevant in some $S_l \in S(\mathscr{P}) \cup \mathscr{P}_o$. Delete a' from S_l to get S'_l and add a' to S between a and the last vertex of $V(S) \cap Q_{i-1}$ to get S'. As $\pi(a') < \pi(a)$ and S_l has two relevant vertices in Q_i, it follows that S' and S'_l are monotone paths, starting and ending at the same vertices as before while together covering the same set of vertices. After this preprocessing, the leftmost relevant vertex of S is in L^i. We can apply a similar preprocessing to show that the rightmost vertex of $V(S) \cap Q_i$ is in R^i.

Suppose $S(\mathscr{P})$ has a segment S such that $V(S) \cap Q_i = (a)$ where a is adjacent to two modulator vertices m_1, m_2 but $a \notin L^i_{m_1m_2} \cup R^i_{m_1m_2}$. As $a \notin L^i_{m_1m_2}$, $|L^i_{m_1m_2}| = 10k$ and $L^i_{m_1m_2}$ has a vertex a' that is irrelevant in some $S_l \in S(\mathscr{P}) \cup \mathscr{P}_o$. Let $V(S_l) \cap Q_i = (x, \ldots, x_i, x_{i+1}, \ldots y)$ where x and y are the two relevant vertices of S_l in Q_i. First we obtain S' from S by replacing a with a'. Then, we obtain S'_l from S_l depending on the following cases.

- Case $S_l \in S(\mathscr{P})$ and $\pi(x) \leq \pi(a) \leq \pi(y)$: Then, S'_l is obtained from S_l by adding

a between two consecutive vertices x_i and x_{i+1} in S_l where $\pi(x) \leq \ldots \leq \pi(x_i) \leq \pi(a) \leq \pi(x_{i+1}) \leq \ldots \leq \pi(y)$. Then, S_l' is a segment starting and ending with the same vertices as S_l and $V(S_l') \cup V(S') = V(S_l) \cup V(S)$.

- Case $S_l \in S(\mathscr{P})$ and $\pi(a) \geq \pi(y)$: In this case we consider the following possibilities. Refer to Figure 3.12 for an illustration.

  - Case 1: If S_l does not end in Q_i, then we add the vertex a after y to get the monotone path S_l'.
  - Case 2: If S_l ends in Q_i and there is no modulator vertex after y and there is no other segment ending at y, then we add a after y to get the monotone path S_l'.
  - Case 3: If S_l ends in Q_i and there is no modulator vertex in after y but there is another segment S_r ending at y, then we add a to the end of S_r after y and replace y with a in S_l to get S_l'.
  - Case 4: If S_l ends in Q_i and there is modulator vertex m_3 after y, then we add the new monotone segment (y,a) and replace y in S_l with a. We concatenate these two segments to get a path $(x \ldots a, y)$ and replace the segment S_l in $\mathscr{P}$ with this new path.

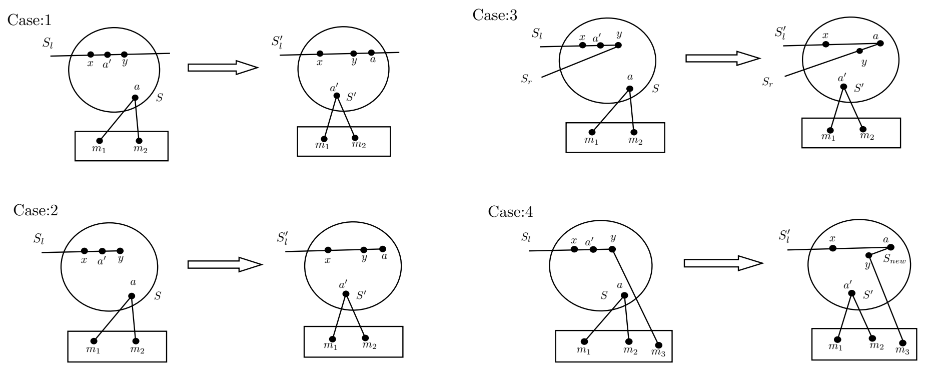


Fig 3.12: Reconstructing Segments

- Case $S_l \in S(\mathscr{P})$ and $\pi(a) \leq \pi(x)$: This case is similar to the previous case.

- Case $S_l \notin S(\mathscr{P})$: In this case S_l is a monotone path from $\mathscr{P}_o$. Since S_l has 2 vertices in Q_i, $V(S_l) \cup \{a\} \setminus \{a'\}$ induces a connected subgraph and hence has a Hamiltonian path S_l'.

The other properties can be proved in a similar manner. Note that if there are two segments that start at the same vertex in Q_i and some property does not hold true for the first relevant vertex in Q_i, then we apply the replacement procedure for the first vertex of both these segments together. We do the same for two segments that end at the same vertex in Q_i. □ □

Corollary 67. *Given a nice minimum path cover $\mathscr{P}$ of H, a canonical minimum path cover $\mathscr{P}^*$ of H with $|\mathscr{P}^*| \leq |\mathscr{P}|$ can be obtained in polynomial time.*

3.5.3 Canonical Minimum Cycle Covers

For a cycle cover $\mathscr{C}$ of H, define the following sets.

- $\mathscr{C}_o = \{C_i \in \mathscr{C} : V(C_i) \cap T = \emptyset\}$, the set of cycles in $\mathscr{C}$ that are completely contained in G.
- $\mathscr{C}_m = \mathscr{C} \setminus \mathscr{C}_o$, the set of cycles that have at least one vertex from T.
- $M(\mathscr{C})$ is the set of maximal subpaths of cycles in $\mathscr{C}_m$ that are contained inside G. That is, for each $C \in \mathscr{C}_m$, a subpath S of C is in $M(\mathscr{C})$ if and only if $V(S) \subseteq V(G)$ and there is no subpath S' of C such that $V(S') \subseteq V(G)$ and $V(S) \subset V(S')$.
- $S(\mathscr{C})$ is the set of maximal subpaths of paths in $M(\mathscr{C})$ that are monotone. That is, for each P in $M(\mathscr{C})$, a subpath S of P is in $S(\mathscr{C})$ if and only if S is monotone and there is no monotone subpath S' of P such that $V(S) \subset V(S')$.

We refer to elements of $S(\mathscr{C})$ as segments of $\mathscr{C}$.

Definition 68. ***(Pseudo-consecutive vertices)*** *Two vertices $u, v \in T$ are said to be pseudo-consecutive if u and v are in the same cycle C in $\mathscr{C}$ and there is a subpath in C between u and v with no other vertex of T.*

Definition 69. ***(Relevant and irrelevant vertices in Q_i)*** *For a path $P \in S(\mathscr{C})$ that contains at least one vertex from Q_i, the set $R_i(\mathscr{C}, P)$ of relevant vertices is $V(P) \cap Q_i$ if $|V(P) \cap Q_i| \leq 2$, otherwise $R_i(\mathscr{C}, P)$ consists of the leftmost and the rightmost vertices of $V(P) \cap Q_i$. The relevant vertices $R_i(\mathscr{C}, C)$ of a 2-monotone cycle $C \in \mathscr{C}_o$ is the collection of relevant vertices of its two maximal internally vertex-disjoint monotone paths in Q_i. We denote the set of relevant vertices in Q_i by $\mathscr{R}_i(\mathscr{C}) = \mathscr{R}_{i_m} \cup \mathscr{R}_{i_o}$, where $\mathscr{R}_{i_m}$ is the collection of the relevant vertices of all the segments in $S(\mathscr{C})$ in Q_i and $\mathscr{R}_{i_o}$ is the collection of all the relevant vertices of the 2-monotone cycles of $\mathscr{C}_o$ in Q_i. A vertex in Q_i that is not in $\mathscr{R}_i(\mathscr{C})$ called irrelevant.*

See **??** for an illustration.

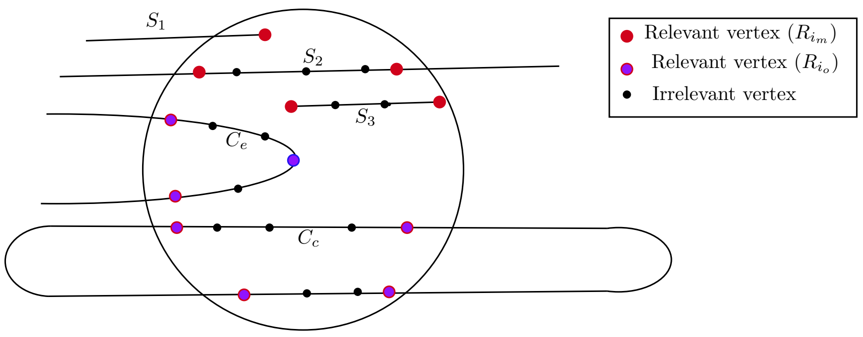


Fig 3.13: Relevant vertices of Q_i

Proposition 70. *For any minimum cycle cover $\mathscr{C} = \mathscr{C}_o \cup \mathscr{C}_m$ and for any $i \in [q]$, there is at most 1 cycle in $\mathscr{C}_o$ from each of the following sets.*

- *(Type 1) Cycles that are contained in $G[Q_i]$.*
- *(Type 2) Cycles that do not start but end in Q_i.*
- *(Type 3) Cycles that start but do not end in Q_i.*

- *(Type 4) Cycles that intersect Q_i, but neither start nor end in it.*

Moreover, for each $i \in [q]$, if there is a cycle of Type 2 (Type 4) in $\mathscr{C}_o$, then there is no cycle of Type 4 (Type 2). Similarly, for each $i \in [q]$, if there is a cycle of Type 3 (Type 4) in $\mathscr{C}_o$, then there is no cycle of Type 4 (Type 3).

Proof. Suppose there are two cycles C and C' in $\mathscr{C}_o$ that are in $G[Q_i]$. Then, $G[V(C) \cup V(C')]$ is 2-connected and hence has a 2-monotone Hamiltonian cycle C^* by Proposition 50. Then $\mathscr{C} \setminus \{C, C'\} \cup \{C^*\}$ is a cycle cover of H smaller than $\mathscr{C}$ leading to a contradiction.

Suppose there are two cycles $C, C' \in \mathscr{C}_o$ that do not start but end in Q_i. We claim that $G[V(C) \cup V(C')]$ is 2-connected. Let $x \in V(C) \cup V(C')$. As $G[V(C)]$ and $G[V(C')]$ are 2-connected, it follows that $G[V(C)] - x$ and $G[V(C')] - x$ are connected. If $x \in Q_i$, then $V(C) \setminus \{x\}$ and $V(C') \setminus \{x\}$ have vertices in Q_{i-1} and hence $G[V(C) \cup V(C')] - x$ is connected. If $x \notin Q_i$, then $V(C) \setminus \{x\}$ and $V(C') \setminus \{x\}$ have vertices in Q_i and hence $G[V(C) \cup V(C')] - x$ is connected. Thus, $G[V(C) \cup V(C')]$ is 2-connected. Then, from Proposition 50, we can replace C and C' by a Hamiltonian cycle in G' leading to a cycle cover smaller than $\mathscr{C}$ which leads to a contradiction. A similar argument proves that there is at most one cycle of Type 3 and there is at most one cycle of Type 4 in $\mathscr{C}_o$.

Suppose there is a Type 2 cycle C and a Type 4 cycle C' in $\mathscr{C}_o$. Note that each of C and C' has vertices in Q_{i-1} and Q_i. Therefore, a similar argument to the above shows that $G[V(C) \cup V(C')]$ is 2-connected and hence by Proposition 50 has a Hamiltonian cycle. Replacing C and C' with this cycle results in a cycle cover smaller than $\mathscr{C}$ which leads to a contradiction. A similar argument proves that $\mathscr{C}_o$ cannot have a cycle of Type 3 and a cycle of Type 4. □ □

Definition 71. ***(Nice cycle cover)*** *A cycle cover $\mathscr{C} = \mathscr{C}_m \cup \mathscr{C}_o$ is said to be nice if the following properties hold.*

- **(NC.1)** *Every cycle in $\mathscr{C}_o$ is 2-monotone.*

- **(NC.2)** *The cycle cover $\mathscr{C}$ satisfies the properties given by Proposition 70.*
- **(NC.3)** *For every cycle C in $\mathscr{C}_m$, for every pair of pseudo-consecutive modulator vertices t, t' in C that are not consecutive in C, the maximal subpath of P between t and t' that is contained in G is i-monotone for some $i \in [3]$.*
- **(NC.4)** *For any $i \in [q]$, if $|Q_i| > 10k$, then each $S \in S(\mathscr{C}^*)$ with $V(S) \cap Q_i \neq \emptyset$ that neither starts nor ends at a vertex in Q_i satisfies $|V(S) \cap Q_i| \geq 2$.*

Lemma 72. *If $\mathscr{C}$ is a nice minimum cycle cover of H, then $|S(\mathscr{C})| \leq 3k$ and for any $i \in [q]$, $|\mathscr{R}_i(\mathscr{C})| \leq 6k + 16$.*

Proof. From Proposition 53, we may assume that the path between any two pseudo-consecutive vertices can have at most 3 monotone segments in $S(\mathscr{C})$. Hence any cycle in $\mathscr{C}$ having t vertices from T has at most $3t$ segments in $S(\mathscr{C})$. This upper bounds the number of maximal monotone paths in $S(\mathscr{C})$ by $3k$. By definition, each segment has at most 2 relevant vertices in Q_i. From Proposition 70, there are at most four 2-monotone cycles from $\mathscr{C}_o$ that intersect Q_i and each such cycle has at most 4 relevant vertices in Q_i. This upper bounds the size of $\mathscr{R}_i(\mathscr{C})$ by $6k + 16$. □ □

Lemma 73. *Given a cycle cover $\mathscr{C}$ of H, a nice cycle cover $\mathscr{C}^*$ of H with $|\mathscr{C}^*| \leq |\mathscr{C}|$ can be obtained in polynomial time.*

Proof. Every cycle C in $\mathscr{C}_o$ not satisfying **(NC.1)** can be replaced by a cycle C' satisfying **(NC.1)** using Observation 51. Property **(NC.2)** is a direct consequence of Proposition 70. Let us next consider **(NC.3)**. Suppose S is a subpath of a path in $\mathscr{C}_m^*$ between two pseudo-consecutive modulator vertices u and v. Let s and t be the neighbors of u and v respectively in S. If the subpath S' of S between s and t is not i-monotone for some $i \in [3]$, we replace S' by an i-monotone (s,t)-path S'' with $V(S) = V(S'')$ for some $i \in [3]$ given by Proposition 54. Applying this replacement rule as long as possible results in a cycle cover $\mathscr{C}^\star$ of H with $|\mathscr{C}^*| \leq |\mathscr{C}|$ satisfying **(NC.1)**, **(NC.2)** and **(NC.3)**.

Suppose there is a segment $S \in S(\mathscr{C}^*)$ that neither starts nor ends in Q_i but has just one vertex x in it. As $|Q_i| > 10k$, from Lemma 72, there an irrelevant vertex y in Q_i that is in some maximal monotone path S_l which is a subpath of some cycle in $\mathscr{C}^*$. We consider the two following cases.

Case 1: **($S_l \in S(\mathscr{C}^*)$)** First, we delete y from S_l to get S_l'. Then, if $\pi(x) \leq \pi(y)$, we add y before x in S to get S'. Otherwise, we add y after x in S to get S'. Without loss of generality, let $S \cap Q_i = (y, x)$ and the last vertex of $S \cap Q_{i-1}$ be z. Since z is adjacent to x, z is also adjacent to y as $\pi(z) \leq \pi(y) \leq \pi(x)$. The other case when y is added after x can be argued similarly. Thus, S' and S_l' are segments that cover the same set of vertices as S and S_l. Hence, after these updates $\mathscr{C}^*$ is still a minimum cycle cover.

Case 2: **($S_l \notin S(\mathscr{C}^*)$)** In this case, S_l is one of the two maximal monotone subpaths of a 2-monotone cycle C in $\mathscr{C}_o$. Deleting the irrelevant vertex y from S_l and adding it to S in the above mentioned way results in a new minimum cycle cover where S has at least 2 vertices from Q_i. We apply this procedure exhaustively to make sure that property **(NC.4)** holds. □ □

Corollary 74. *Given a minimum cycle cover $\mathscr{C}$ of H, a nice minimum cycle cover $\mathscr{C}^*$ of H with $|\mathscr{C}^*| \leq |\mathscr{C}|$ can be obtained in polynomial time.*

Definition 75. ***(Leftmost and rightmost set of vertices)*** *Consider a subset S of vertices of G. If $|S| > 10k$, then let $LM(S)$ denote the $10k$ leftmost vertices of S and $RM(S)$ denote the $10k$ rightmost vertices of S. Otherwise, $LM(S) = RM(S) = S$.*

Definition 76. ***(Boundary vertices of Q_i)*** *Consider the following sets.*

- $L^i = LM(Q_i)$.
- $R^i = RM(Q_i)$.
- $L^i_x = LM(Q_i \cap N(x))$ *for each* $x \in T$.
- $R^i_x = RM(Q_i \cap N(x))$ *for each* $x \in T$.

- $L^i_{xy} = LM(Q_i \cap N(x) \cap N(y))$ *for each* $x, y \in T$.
- $R^i_{xy} = RM(Q_i \cap N(x) \cap N(y))$ *for each* $x, y \in T$.

The set $B(Q_i) = L^i \cup R^i \bigcup_{x \in T} (L^i_x \cup R^i_x) \bigcup_{y,z \in T} (L^i_{yz} \cup R^i_{yz})$ *is called the boundary vertices of* Q_i. *Notice that for any* $i \in [q]$, *the size of* $B(Q_i)$ *is* $\mathscr{O}(k^3)$.

Definition 77. *For every vertex* $v \in V(G)$, *let* $\rho(v)$ *denote the tuple* (m_1, m_2) *defined as follows.*

- m_1 *is the vertex in* T *that preceeds* v *in a subpath of the cycle in* $\mathscr{C}$ *that contains* v. *If no such vertex exists, then* $m_1 = \$$.
- m_2 *is the vertex in* T *that succeeds* v *in a subpath of the cycle in* $\mathscr{C}$ *that contains* v. *If no such vertex exists, then* $m_2 = \$$.

Definition 78. ***(Canonical cycle cover)*** *A nice cycle cover* $\mathscr{P}$ *is said to be canonical if for each* $i \in [q]$ *and each* $S \in S(\mathscr{C})$, $\mathscr{R}_i(\mathscr{C}, S) \subseteq B(Q_i)$, *the following properties are satisfied.*

- **(CC.1)** *For a segment* $S \in S(\mathscr{C})$ *passing through* Q_i, *its first and last vertices in* Q_i *are in* $L^i \cup R^i$.
- **(CC.2)** *For a segment* $S \in S(\mathscr{C})$ *that ends in* Q_i, *the following hold.*
  - *If* $S \cap Q_i = (a)$ *and* $\rho(a) = (\$, \$)$, *then* $a \in L^i$.
  - *If* $S \cap Q_i = (a)$ *and* $\rho(a) = (m_1, m_2)$, *then* $a \in L^i_{m_1 m_2} \cup R^i_{m_1 m_2}$.
  - *If* $S \cap Q_i = (a)$ *and* $\rho(a) = (m_1, \$)$, *then* $a \in L^i_{m_1} \cup R^i_{m_1}$.
  - *If* $S \cap Q_i = (a)$ *and* $\rho(a) = (\$, m_2)$, *then* $a \in L^i_{m_2} \cup R^i_{m_2}$.
  - *If* $\mathscr{R}_i(\mathscr{C}, S) = (a, b)$ *and* $\rho(a) = (m_1, \$)$, $\rho(b) = (\$, m_2)$, *then* $a \in L^i_{m_1} \cup R^i_{m_1}, b \in L^i_{m_2} \cup R^i_{m_2}$.
  - *If* $\mathscr{R}_i(\mathscr{C}, S) = (a, b)$ *and* $\rho(a) = (\$, \$)$, $\rho(b) = (\$, m_2)$, *then* $a \in L^i, b \in L^i_{m_2} \cup R^i_{m_2}$.

- *If $\mathscr{R}_i(\mathscr{C},S)(\mathscr{C}) = (a,b)$ and $\rho(a) = (m_1,\$)$, $\rho(b) = (\$,\$)$, then $a \in L^i_{m_1} \cup R^i_{m_1}, b \in R^i$.*
- *If $\mathscr{R}_i(\mathscr{C},S) = (a,b)$ and $\rho(a) = (\$,\$)$, $\rho(b) = (\$,\$)$, then $a \in L^i, b \in R^i$.*

- **(CC.3)** *For a segment $S \in S(\mathscr{C})$ that starts in Q_i, the following hold.*
  - *If $S \cap Q_i = (a)$ and $\rho(a) = (\$,\$)$, then $a \in R^i$.*
  - *If $S \cap Q_i = (a)$ and $\rho(a) = (m_1,\$)$, then $a \in L^i_{m_1} \cup R^i_{m_1}$.*
  - *If $S \cap Q_i = (a)$ and $\rho(a) = (\$,m_2)$, then $a \in L^i_{m_2} \cup R^i_{m_2}$.*
  - *If $\mathscr{R}_i(\mathscr{C},S) = (a,b)$ and $\rho(a) = (\$,\$)$, $\rho(b) = (\$,m_2)$, then $a \in L^i, b \in L^i_{m_2} \cup R^i_{m_2}$.*
  - *If $\mathscr{R}_i(\mathscr{C},S) = (a,b)$ and $\rho(a) = (m_1,\$)$, $\rho(b) = (\$,\$)$, then $a \in L^i_{m_1} \cup R^i_{m_1}, b \in R^i$.*
  - *If $\mathscr{R}_i(\mathscr{C},S) = (a,b)$ and $\rho(a) = (\$,\$)$, $\rho(b) = (\$,\$)$, then $a \in L^i, b \in R^i$.*

Lemma 79. *Given a nice cycle cover $\mathscr{C}$ of H, a canonical cycle cover $\mathscr{C}^*$ of H with $|\mathscr{C}^*| \leq |\mathscr{C}|$ can be obtained in polynomial time.*

Proof. From Lemma 72, as $\mathscr{C}$ is a nice minimum cycle cover of H, we have $|S(\mathscr{C})| \leq 3k$ and for any $i \in [q]$, $|\mathscr{R}_i(\mathscr{C})| \leq 6k+16$. For any $i \in [q]$ such that $|Q_i| \leq 10k$, Properties **(CC.1)**, **(CC.2)** and **(CC.3)** vacously hold as $L^i = R^i = V(Q_i)$, $L^i_x = R^i_x = V(Q_i) \cap N(x)$, $L^i_{xy} = R^i_{xy} = V(Q_i) \cap N(x) \cap N(y)$. Now, consider a clique Q_i such that $|Q_i| > 10k$. Suppose $S(\mathscr{C})$ has a segment S that intersects Q_i but neither starts nor ends in it. Suppose the first vertex a in $V(S) \cap Q_i$ is not in L^i. Since Q_i has more than $10k$ vertices, $|L^i| = 10k$ and L^i has an irrelevant vertex a' that is in a maximal monotone path S_l which is a subpath of some cycle in $\mathscr{C}^*$. If $S_l \in S(\mathscr{C})$, then we delete a' from S_l and add it to S between a and the last vertex of $S \cap Q_{i-1}$. As $\pi(a') < \pi(a)$ and S_l has two relevant vertices in Q_i, it follows that the updated S and S_l are still monotone paths starting and ending at the same vertices as before while together covering the same set of vertices. After this preprocessing the leftmost relevant vertex of S is in L^i. Now, we are in the case when $S_l \notin S(\mathscr{C})$. Then, S_l is one of the two maximal monotone paths of a 2-monotone cycle $C \in \mathscr{C}_o$ and we delete a' from S_l and add it to S as mentioned earlier. After this preprocessing the leftmost vertex of $V(S) \cap Q_i$ is in L^i. We can apply a similar preprocessing to show that the rightmost vertex of $V(S) \cap Q_i$ is in R^i.

Suppose $S(\mathscr{C})$ has a segment S such that $S \cap Q_i = (a)$, where a is adjacent to two modulator vertices m_1 and m_2 in $\mathscr{C}$, but $a \notin L^i_{m_1m_2} \cup R^i_{m_1m_2}$. As $a \notin L^i_{m_1m_2}$, $|L^i_{m_1m_2}| = 10k$ and $L^i_{m_1m_2}$ has an irrelevant vertex a' in S_l between its two relevant vertices x and y, where S_l is either a segment or a maximal monotone path of some 2-monotone cycle in $\mathscr{C}_o$. We interchange the vertices a and a' between S and S_l to construct a new cycle cover. Let S' denote the segment obtained from S by replacing a with a'. Then we obtain S'_l from S_l depending on the following cases.

- Case $S_l \in S(\mathscr{C})$ and $\pi(x) \leq \pi(a) \leq \pi(y)$: Then, S'_l is obtained from S_l by adding a between two consecutive vertices x_i and x_{i+1} in S_l where $\pi(x) \leq \ldots \leq \pi(x_i) \leq \ldots \leq \pi(a) \leq \pi(x_{i+1}) \leq \ldots \leq \pi(y)$. Then, S'_l is a segment starting and ending with the same vertices as S_l and $V(S'_l) \cup V(S') = V(S_l) \cup V(S)$.
- Case $S_l \in S(\mathscr{C})$ and $\pi(a) \geq \pi(y)$: In this case we consider the following possibilities. Refer to Figure 3.12 for an illustration.
  - If S_l does not end in Q_i, then we add the vertex a after y to get the monotone path S'_l.
  - If S_l ends in Q_i and there is no modulator vertex after y and there is no other segment ending at y, then we add a after y to get the monotone path S'_l.
  - If S_l ends in Q_i and there is no modulator vertex in after y but there is another segment S_r ending at y, then we add a to the end of S_r after y and replace y with a in S_l to get S'_l.
  - If S_l ends in Q_i and there is modulator vertex m_3 after y, then we add the new

monotone segment (y,a) and replace y in S_l with a. We concatenate these two segments to get a path $(x\ldots a,y)$ and replace the segment S_l in $\mathscr{C}$ with this new path.

- Case $S_l \in S(\mathscr{C})$ and $\pi(a) \leq \pi(x)$: This case is similar to the previous case.
- Case: $S_l \notin S(\mathscr{C})$: In this case S_l is a maximal monotone path of some 2-monotone cycle $C \in \mathscr{C}_o$. Since C has at least 2 vertices in Q_i other than a', $V(C) \cup \{a\} \setminus \{a'\}$ induces a 2-connected component. Hence we can get a new 2-monotone cycle that covers all the vertices in $V(C) \cup \{a\} \setminus \{a'\}$ from Observation 50.

The other properties can be proved in a similar manner. Note that if there are two segments that start at the same vertex in a clique Q_i and some property does not hold true for their first relevant vertex in Q_i, then we apply the above replacement procedure for the first vertex of both these segments together. We do the same for two segments that end at the same vertex in Q_i. □ □

Corollary 80. *Given a minimum cycle cover $\mathscr{C}$ of H, a canonical minimum cycle cover $\mathscr{C}^*$ of H with $|\mathscr{C}^*| \leq |\mathscr{C}|$ can be obtained in polynomial time.*

3.6 Path Cover Parameterized by Proper Interval Deletion Set

Let $\mathscr{P}$ denote a minimum canonical path cover of H. We define the following functions that help us to understand the relationship between the segments of a canonical path cover $\mathscr{P}$.

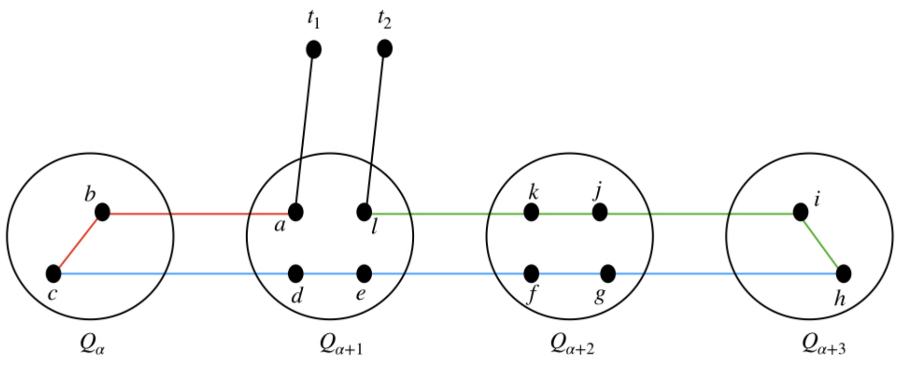
- $\mathscr{F} : S(\mathscr{P}) \times S(\mathscr{P}) \rightarrow \{0,1\}$ where $\mathscr{F}(S,S') = 1$ if and only if S and S' start at the same vertex.

- $\mathscr{L} : S(\mathscr{P}) \times S(\mathscr{P}) \rightarrow \{0,1\}$ where $\mathscr{L}(S,S') = 1$ if and only if S and S' end at the same vertex.

- $\mathscr{F}_1 : S(\mathscr{P}) \rightarrow T \cup \{0\}$ where $\mathscr{F}_1(S) = t$ if S starts immediately after t, otherwise $\mathscr{F}_1(S) = 0$.

- $\mathscr{L}_1 : S(\mathscr{P}) \rightarrow T \cup \{0\}$ where $\mathscr{L}_1(S) = t$ if S ends just before t, otherwise $\mathscr{L}_1(S) = 0$.

We remark that $\mathscr{F}$ and $\mathscr{L}$ are symmetric functions. In the example given below, the segment S_1 ends just before t_1 and the segment S_3 starts just after t_2. Further, segments S_1 and S_2 have the same starting vertex while segments S_2 and S_3 have the same ending vertex.



Given $\mathscr{P}$, determining $S(\mathscr{P})$ is easy and in turn given $S(\mathscr{P})$, determining $\mathscr{F}$, $\mathscr{F}_1$, $\mathscr{L}$ and $\mathscr{L}_1$ is easy. It is now natural to ask what choices of $(\mathscr{F}, \mathscr{L}, \mathscr{F}_1, \mathscr{L}_1)$ lead to a set $S(\mathscr{P})$ that in turn leads to a minimum canonical path cover $\mathscr{P}$. Let us first guess the size of $S(\mathscr{P})$. From Lemma 61, it is at most $4k$. For a correct choice of this number, the choice $(\mathscr{F}, \mathscr{L}, \mathscr{F}_1, \mathscr{L}_1)$ that minimizes the size of a minimum path cover of $G[X]$ where $V(G) \setminus X$ is the set of vertices that are in some segment assigned to a variable in $\mathscr{S}$ is the one the results in a minimum canonical path cover $\mathscr{P}$.

3.6.1 The Guessing Phase

With this information, we proceed as follows. Let $\mathscr{P}$ be a minimum canonical path cover that we are looking for. We first guess the following properties of $\mathscr{P}$. Intialize $\mathscr{S}$ to be the empty set.

1. We guess the number ℓ of paths in $\mathscr{P}_m$. Clearly, $\ell \leq k$ and the number of choices for ℓ is k. Let $P_1, \dots, P_\ell$ denote the paths in $\mathscr{P}_m$.

2. For each $P_i \in \mathscr{P}_m$, we guess if P_i has zero, one or two endpoints in T. The number of possible choices in this step is $\mathscr{O}(3^k)$.

3. For each $P_i \in \mathscr{P}_m$, we guess the order of vertices of $V(P_i) \cap T$. The number of possible choices in this step is $2^{\mathscr{O}(k \log k)}$.

4. For each $P_i \in \mathscr{P}_m$, for each pair of pseudo-consecutive vertices t and t' in P_i, we guess if t and t' are consecutive in P_i (in which case t and t' must be adjacent) or not. It t and t' are not consecutive in P_i, then we guess if the maximal subpath P of the (t,t')-path that is contained in G is 1-monotone or 2-monotone or 3-monotone. The number of possible choices in this step is $\mathscr{O}(3^k)$.

   - If P is 1-monotone, then we add the variable S^i to $\mathscr{S}$ and set $\mathscr{F}_1(S^i) = t$, $\mathscr{L}_1(S^i) = t'$.
   - If P is 2-monotone, then we add the variables S_1^i and S_2^i to $\mathscr{S}$. We guess one of the following two choices.
     - Set $\mathscr{L}_1(S_1^i) = t$, $\mathscr{F}_1(S_1^i) = 0$, $\mathscr{F}_1(S_2^i) = 0$, $\mathscr{L}_1(S_2^i) = t'$, $\mathscr{F}(S_1^i, S_2^i) = 1$, $\mathscr{L}(S_1^i, S_2^i) = 0$.
     - Set $\mathscr{F}_1(S_1^i) = t$, $\mathscr{L}_1(S_1^i) = 0$, $\mathscr{F}_1(S_2^i) = t'$, $\mathscr{L}_1(S_2^i) = 0$, $\mathscr{L}(S_1^i, S_2^i) = 1$, $\mathscr{F}(S_1^i, S_2^i) = 0$.
   - If P is 3-monotone, then we add the variables S_1^i, S_2^i and S_3^i to $\mathscr{S}$. We set $\mathscr{L}_1(S_1^i) = t$, $\mathscr{F}_1(S_1^i) = 0$, $\mathscr{F}_1(S_2^i) = 0$, $\mathscr{L}_1(S_2^i) = 0$, $\mathscr{F}_1(S_3^i) = t'$, $\mathscr{L}_1(S_3^i) = 0$,

$\mathscr{F}(S^i_1,S^i_2)=1$, $\mathscr{L}(S^i_2,S^i_3)=1$, $\mathscr{F}(S^i_1,S^i_3)=0$, $\mathscr{F}(S^i_2,S^i_3)=0$, $\mathscr{L}(S^i_1,S^i_2)=0$, $\mathscr{L}(S^i_1,S^i_3)=0$.

5. For each $P_i \in \mathscr{P}_m$, for each ordered pair of pseudo-adjacent vertices $x \in T$ and $y \in V(G)$, we guess if the maximal subpath P of the (x,y)-path that is contained in G is 1-monotone or 2-monotone. The number of possible choices in this step is $\mathscr{O}(2^k)$.

   - If P is 1-monotone, then we add the variable S^i to $\mathscr{S}$.
     - If $x \notin T$ and $y \in T$, then set $\mathscr{F}_1(S^i)=0$ and $\mathscr{L}_1(S^i)=y$.
     - If $x \in T$ and $y \notin T$, then set $\mathscr{F}_1(S^i)=x$ and $\mathscr{L}_1(S^i)=0$.
   - If P is 2-monotone, then we add the variables S^i_1 and S^i_2 to $\mathscr{S}$.
     - If $x \notin T$ and $y \in T$, then we guess one of the following two choices.
       * Set $\mathscr{F}_1(S^i_1)=0$, $\mathscr{L}_1(S^i_1)=0$, $\mathscr{F}_1(S^i_2)=y$, $\mathscr{L}_1(S^i_2)=0$, $\mathscr{F}(S^i_1,S^i_2)=0$, $\mathscr{L}(S^i_1,S^i_2)=1$.
       * Set $\mathscr{F}_1(S^i_1)=0$, $\mathscr{L}_1(S^i_1)=0$, $\mathscr{F}_1(S^i_2)=0$, $\mathscr{L}_1(S^i_2)=y$, $\mathscr{F}(S^i_1,S^i_2)=1$, $\mathscr{L}(S^i_1,S^i_2)=0$.
     - If $x \in T$ and $y \notin T$, then we guess one of the following two choices.
       * Set $\mathscr{F}_1(S^i_1)=x$, $\mathscr{L}_1(S^i_1)=0$, $\mathscr{F}_1(S^i_2)=0$, $\mathscr{L}_1(S^i_2)=0$, $\mathscr{F}(S^i_1,S^i_2)=0$, $\mathscr{L}(S^i_1,S^i_2)=1$.
       * Set $\mathscr{F}_1(S^i_1)=0$, $\mathscr{L}_1(S^i_1)=x$, $\mathscr{F}_1(S^i_2)=0$, $\mathscr{L}_1(S^i_2)=0$, $\mathscr{F}(S^i_1,S^i_2)=1$, $\mathscr{L}(S^i_1,S^i_2)=0$.

6. For each pair S and S' of variables in $\mathscr{S}$ such that $\mathscr{F}(S,S')$ (or $\mathscr{L}(S,S')$) is not yet set is set to 0. Similarly, for each variable S in $\mathscr{S}$ such that $\mathscr{F}_1(S)$ (or $\mathscr{L}_1(S)$) is not yet set is set to 0.

As $|\mathscr{P}_m| \leq k$, the maximum number of guesses in Steps 1-3 is $2^{\mathscr{O}(k\log k)}$. The maximum number of guesses in Steps 4-6 is asymptotically upper bounded by the maximum number of choices for $(\mathscr{S},\mathscr{F},\mathscr{L},\mathscr{F}_1,\mathscr{L}_1)$ which is $2^{\mathscr{O}(k)}$. Once these choices are fixed, the problem of finding a mnimum path cover $\mathscr{P}$ now reduces to the problem of finding an assignment of segments to variables in $\mathscr{S}$ that satisfy the relationships given by $(\mathscr{S},\mathscr{F},\mathscr{L},\mathscr{F}_1,\mathscr{L}_1)$ while minimizing the size of a minimum path cover of $G-X$ where X is the set of vertices of H that are in a segment assigned to some variable in $\mathscr{S}$. In other words, we find an assignment of segments to variables in $\mathscr{S}$ that satisfy the relationships given by $(\mathscr{S},\mathscr{F},\mathscr{L},\mathscr{F}_1,\mathscr{L}_1)$ resulting in a set of paths $\mathscr{P}_m$ while minimizing the number of paths in $\mathscr{P}_o$. Note that not all choices of $(\mathscr{S},\mathscr{F},\mathscr{L},\mathscr{F}_1,\mathscr{L}_1)$ may necessarily lead to a minimum path cover $\mathscr{P}_o \cup \mathscr{P}_m$ of H. However, at least one of the choices that we generate leads to one.

Consider a particular choice for Steps 1-3 and a choice of $(\mathscr{S},\mathscr{F},\mathscr{L},\mathscr{F}_1,\mathscr{L}_1)$. This fixes how the paths in $\mathscr{P}_m$ interact with T. That is, for any path P in $\mathscr{P}_m$, the vertices of T that are in P and their order in P are fixed. Furthernore, the paths between any two pseudo-consecutive vertices and the paths between any two pseudo-adjacent vertices are also fixed. This also fixes the number of segments and the relationship among the segments. We will now describe a dynamic programming algorithm that finds a minimum canonical path cover respecting this choice $\vartheta = (\mathscr{S},\mathscr{F},\mathscr{L},\mathscr{F}_1,\mathscr{L}_1)$.

3.6.2 Finding an Assignment of Segments for $\vartheta = (\mathscr{S},\mathscr{F},\mathscr{L},\mathscr{F}_1,\mathscr{L}_1)$

Let $Q_0 = \emptyset$. For each $i \in [q]$, let G_i denote the graph $G[Q_1 \cup \cdots \cup Q_i]$. Let us first understand the interaction of the solution (minimum canonical path cover with the properties given by $(\mathscr{S},\mathscr{F},\mathscr{L},\mathscr{F}_1,\mathscr{L}_1)$) with G_i. Subsequently, we refer to $\mathscr{S}$ as segments instead of variables that have to be assigned segments.

Index of an Entry: An entry in the table T_i is indexed by a tuple $(\mathscr{S}_f,\mathscr{X},X_o,\mathscr{A},B)$ with the following interpretation.

- $\mathscr{S}_f \subseteq \mathscr{S}$ denotes the segments that have no vertex from $Q_{i+1} \cup \ldots \cup Q_q$. That is, these segments are completely contained in G_i.

- $\mathscr{X}$ denotes the set of relevant vertices of all segments from $\mathscr{S}$ in Q_i. That is, for every $S \in \mathscr{S}$, X_S in $\mathscr{X}$ is the set of relevant vertices of the segment S in Q_i.
  - If X_S is the empty set, then the segment corresponding to S has no vertex from Q_i.
  - Otherwise, X_S has a single vertex or an ordered pair of vertices.
    - If X_S is an ordered pair of vertices (v_1, v_2), we call v_1 the first relevant vertex (denoted by $X_S(1)$) and v_2 the last relevant vertex (denoted by $X_S(2)$) of S in Q_i.
    - If X_S has a single vertex v, we call v both first and last relevant vertex of S in Q_i.
- X_o denotes the set of relevant vertices of the unique monotone path in $\mathscr{P}_o$ that has a vertex from Q_i.
- $\mathscr{A} \in [0,1,2,3,4]^{|\mathscr{S}|}$ represents the interactions of the segments from $\mathscr{S}$ with Q_i.
  - $a_S = 0$ iff the segment S does not intersect Q_i.
  - $a_S = 1$ iff the segment S has at least one vertex from Q_i but neither starts nor ends in Q_i.
  - $a_S = 2$ iff the segment S starts but does not end in Q_i.
  - $a_S = 3$ iff the segment S ends but does not start in Q_i.
  - $a_S = 4$ iff the segment S starts and ends in Q_i.
- $B \in \{0,1,2,3,4\}$ represents the interaction of the monotone path P in $\mathscr{P}_o$ with Q_i.
  - $B = 0$ iff the segment P does not intersect Q_i.
  - $B = 1$ iff the segment P has at least one vertex from Q_i but neither starts nor ends in Q_i.

- $B = 2$ iff the segment P starts but does not end in Q_i.
- $B = 3$ iff the segment P ends but does not start in Q_i.
- $B = 4$ iff the segment P starts and ends in Q_i.

Valid Indices: An index $(\mathscr{S}_f, \mathscr{X}, X_o, \mathscr{A}, B)$ corresponding to an entry in T_i is valid if the following conditions are satisfied.

- **(VI.1)** Two sets X_S and $X_{S'}$ are vertex-disjoint iff $\mathscr{F}(S,S') = \mathscr{L}(S,S') = 0$.
- **(VI.2)** If $\mathscr{F}(S,S') = 1$ and $a_S \in \{2,4\}$, then $a_{S'} \in \{2,4\}$ and X_S and $X_{S'}$ have the same first vertex.
- **(VI.3)** If $\mathscr{L}(S,S') = 1$ and $a_S \in \{3,4\}$, then $a_{S'} \in \{3,4\}$ and X_S and $X_{S'}$ have the same last vertex.
- **(VI.4)** $a_S = 0$ iff $X_S = \emptyset$ and $B = 0$ iff $X_o = \emptyset$.
- **(VI.5)** $a_S \in \{3,4\}$ iff $S \in \mathscr{S}_f$ and $a_S \in \{1,2\}$ iff $S \notin \mathscr{S}_f$.
- **(VI.6)** The set X_S along with a_S for any segment S satisfy the canonical solution properties given by Definition 65. For example, if $X_S = (a,b)$, $a_S = 2$, then $a \in L^i$, $b \in R^i$.
- **(VI.7)** For every vertex $v \in Q_i \setminus (V(\mathscr{X}) \cup V(X_o))$, either there is an element $S \in \mathscr{S}$ such that $\pi(X_S(1)) \leq \pi(v) \leq \pi(X_S(2))$ or $\pi(X_o(1)) \leq \pi(v) \leq \pi(X_o(2))$.

Observe that all segments are disjoint except possibly at the start and end vertices. **(VI.1)** ensures that if two segments do not have common start and end vertices then they are vertex-disjoint. In particular, their relevant vertices are disjoint. **(VI.2)** and **(VI.3)** ensure that the choice of $(\mathscr{F}, \mathscr{L})$ is respected. **(VI.4)** and **(VI.5)** ensure that the intended interpretation of $\mathscr{A}$ is respected. In particluar, **(VI.5)** makes sure that only the segments from $\mathscr{S}$ which are ending in Q_i are taken into $\mathscr{S}_f$. **(VI.6)** follows from Lemma 66 and it ensures that the relevant vertices of segments are chosen from $B(Q_i)$ and these segments can be concatenated to the modulator vertices appropriately. **(VI.7)** makes sure that any irrelevant vertex can always be added in between the relevant vertices of some segment in $\mathscr{S}$ or some monotone path in $\mathscr{P}_o$.

Observation 81. *For each $i \in [q]$, the maximum number of valid indices is $\mathscr{O}^*(2^{\mathscr{O}(k \log k)})$.*

Proof. Let us bound the number of valid entries $\alpha = (\mathscr{S}_f, \mathscr{X}, X_o, \mathscr{A}, B)$ in T_i. There are $2^{\mathscr{O}(k)}$ choices for $\mathscr{S}_f$. Every relevant set in $\mathscr{X}$ is chosen from a set of at most $10k$ vertices. Hence there are $2^{\mathscr{O}(k \log k)}$ choices for $\mathscr{X}$. We have $\mathscr{O}(n^2)$ ways to select an X_o and there are $2^{\mathscr{O}(k)}$ choices for $\mathscr{A}$ and B. Thus, the maximum number of entries in T_i is $2^{\mathscr{O}(k \log k)} n^{\mathscr{O}(1)}$. □ □

(Optimum) Partial Solutions: For $\sigma = (\mathscr{S}_f, \mathscr{X}, X_o, \mathscr{A}, B)$, a collection of paths $\mathscr{P}_d \uplus \mathscr{P}_u$ is a partial solution of $T_i(\sigma)$ if the following conditions hold. Let $h : \mathscr{S}_f \uplus \{S \in \mathscr{S} : a_S \in \{1,2\}\} \to \mathscr{P}_u$ denote the assignment of paths in $\mathscr{P}_u$ to variables in $\mathscr{S}_f \uplus \{S \in \mathscr{S} : a_S \in \{1,2\}\}$.

- **(PS.1)** $|\mathscr{P}_u| = |\mathscr{S}_f| + |\{S \in \mathscr{S} : a_S \in \{1,2\}\}|$ and h is injective.
- **(PS.2)** Every path P in $\mathscr{P}_d \cup \mathscr{P}_u$ is monotone and satisfies $V(P) \subseteq Q_1 \cup \ldots Q_i$.
- **(PS.3)** Every vertex in $Q_1 \cup \ldots \cup Q_i$ is in a path in $\mathscr{P}_d \cup \mathscr{P}_u$.
- **(PS.4)** Every pair P_i, P_j of distinct paths in $\mathscr{P}_d \cup \mathscr{P}_u$ are internally vertex-disjoint. Further, they are vertex-disjoint except when $\mathscr{F}(P_i, P_j) = 1$ or $\mathscr{L}(P_i, P_j) = 1$.
- **(PS.5)** The paths in $\mathscr{P}_u$ take their respective relevant vertices in Q_i according to the assignment $\mathscr{X}$ i.e. the first and last vertices of X_S are the first and last vertices $V(h(S)) \cap Q_i$.
- **(PS.6)** For each $S \in \text{dom}(h)$, $h(S)$ starts at a vertex in $N(m_1)$ if $\mathscr{F}_1(S) = m_1$.

- **(PS.7)** For each $S \in \mathscr{S}_f$, $h(S)$ ends at a vertex in $N(m_2)$ if $\mathscr{L}_1(S) = m_2$.
- **(PS.8)** If $\mathscr{F}(S_i, S_j) = 1$ and $S_i \in \mathrm{dom}(h)$, then $S_j \in \mathrm{dom}(h)$ and $h(S_i)$ and $h(S_j)$ start at the same vertex.
- **(PS.9)** If $\mathscr{L}(S_i, S_j) = 1$ and $S_i \in \mathscr{S}_f$, then $S_j \in \mathscr{S}_f$ and $h(S_i)$ and $h(S_j)$ end at the same vertex.
- **(PS.10)** At most one path from $\mathscr{P}_d$ has relevant vertices in Q_i and these vertices are given by X_o.
- **(PS.11)** Any path $P = h(S)$ with $S \in \mathrm{dom}(h)$ and at most one path from $\mathscr{P}_d$ start and end in Q_i iff a_S (and/or B) is in $\{2, 4\}$.
- **(PS.12)** Any path $P = h(S)$ with $S \in \mathrm{dom}(h)$ and at most one path from $\mathscr{P}_d$ do not start but end in Q_i iff (and/or B) is in $\{1, 3\}$.

Let $B^* = 1$ if $B \in \{3, 4\}$, 0 otherwise. Over all possible partial solutions $\mathscr{P}_d \cup \mathscr{P}_u$, $T_i(\sigma)$ stores the one that minimizes the value of $|\mathscr{P}_d| - (1 - B^*)$. Such a partial solution is called an **optimum partial solution**. In other words, $T_i(\sigma)$ stores a partial solution that minimizes the number of paths contained inside $\mathscr{P}_d$ that end in G_i. If there is a path in $\mathscr{P}_d$ with B value either 1 or 2 in Q_i, it has a vertex in Q_{i+1} and hence not counted. We also store the size of an optimum solution denoted by $|T_i(\sigma)|$.

Clearly, an optimum solution for an entry in T_q where every path in the solution has ended in $G_q = G$ gives the required answer. In the dynamic programming algorithm that we will subsequently describe, we compute partial solutions for G_i and use them to compute partial solutions for G_{i+1}. So let us understand how the solution that we are looking for interacts with G_i. A partial solution for G_i is the one that minimizes the number of paths contained in $\mathscr{P}_o$ when restricted to G_i. To determine such a partial solution we need to know the segments that have already terminated before Q_i and the interaction of segments in $S(\mathscr{P}) \cup \mathscr{P}_o$ with Q_i. Refer to Figure 3.14 for an illustration.

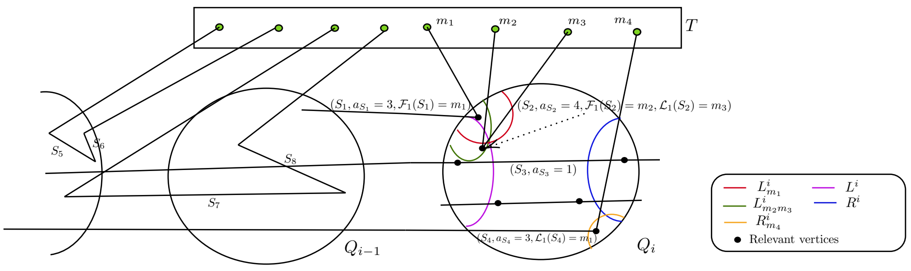

m_1
m_2
m_3
m_4
T
$(S_1, a_{S_1} = 3, \mathcal{F}_1(S_1) = m_1)$
$(S_2, a_{S_2} = 4, \mathcal{F}_1(S_2) = m_2, \mathcal{L}_1(S_2) = m_3)$
$(S_3, a_{S_3} = 1)$
$(S_4, a_{S_4} = 3, \mathcal{L}_1(S_4) = m_1)$
S_5
S_6
S_7
S_8
Q_{i-1}
Q_i
$L^i_{m_1}$
$L^i_{m_2 m_3}$
$R^i_{m_4}$
L^i
R^i
● Relevant vertices

Fig 3.14: Interaction of a canonical path cover with $Q_1 \cup \cdots \cup Q_i$

Compatible Entries: A valid entry in T_i with index $(\mathscr{S}_f, \mathscr{X}, X_o, \mathscr{A}, B)$ is compatible with a valid entry in T_{i-1} with index $(\mathscr{S}'_f, \mathscr{X}', X'_o, \mathscr{A}', B')$ if the following conditions hold.

- **(CI.1)** $\mathscr{S}_f = \mathscr{S}'_f \cup \{S : a_S \in \{3,4\}\}$.
- **(CI.2)** If $a_S = 0$ ($B = 0$), then $a'_S \in \{0,3,4\}$ ($B' \in \{0,3,4\}$).
- **(CI.3)** If $a_S = 1$ ($B = 1$), then $a'_S \in \{1,2\}$ ($B' \in \{1,2\}$) and the first relevant vertex of X_S (X_o) is adjacent to the last relevant vertex of X'_S (X'_o).
- **(CI.4)** If $a_S = 2$ ($B = 2$), then $a'_S \in \{0\}$ ($B' \in \{0\}$).
- **(CI.5)** If $a_S = 3$ ($B = 3$), then $a'_S \in \{1,2\}$ ($B' \in \{1,2\}$) and the first relevant vertex of X_S (X_o) is adjacent to the last relevant vertex of X'_S (X'_o).
- **(CI.6)** If $a_S = 4$ ($B = 4$), then $a'_S = 0$ ($B' = 0$).
- **(CI.7)** If $a_S \neq 0$, then $S \notin S'_f$.
- **(CI.8)** $X_o \neq \emptyset$ and $B \in \{1,3\}$ iff $X'_o \neq \emptyset$, $B' \in \{1,2\}$ and the first relevant vertex of X_o is adjacent to the last relevant vertex X'_o.

(CI.1) makes sure that the segments completely contained in G_i are the segments completed before Q_i and the segments completed in Q_i. **(CI.2)** ensures that if a segment/monotone path does not intersect Q_i, it is not continued from Q_{i-1}. **(CI.3)** and **(CI.5)** ensure we can extend a segment/monotone path from Q_{i-1} to Q_i. **(CI.4)**, **(CI.6)** and **(CI.7)**

make sure that the continuing segments/monotone paths in Q_i are disjoint from the segments/monotone paths that are finished till Q_{i-1}. **(CI.8)** makes sure we can extend the monotone path in $\mathscr{P}_o$ from Q_{i-1} to Q_i.

Lemma 82. *$|T_0(\alpha)| = 0$ and for each $i \in [q]$, $|T_i(\alpha)| = \min\{|T_{i-1}(\beta)| + B^*\}$ over all valid β compatible with α. Further, all the entries in T can be computed in $\mathscr{O}^*(2^{\mathscr{O}(k \log k)})$ time.*

Proof. Since Q_0 is the empty clique, 0 paths are required to cover vertices of G_0 and hence $|T_0(\alpha)| = 0$. Let $i \in [q]$. Suppose $T_{i-1}(\beta)$ has been correctly computed for G_{i-1} for all valid β. Consider an entry in $T_{i-1}(\beta')$ that is compatible with $T_i(\alpha)$. Let $\alpha = (\mathscr{S}_f, \mathscr{X}, X_o, \mathscr{A}, B)$ and $\beta' = (\mathscr{S}'_f, \mathscr{X}', X'_o, \mathscr{A}', B')$. Let $\mathscr{P}'_d \cup \mathscr{P}'_u$ be an optimal partial solution for $T_{i-1}(\beta')$. We will construct a partial solution for $T_i(\alpha)$ from $\mathscr{P}'_d \cup \mathscr{P}'_u$ in the following manner.

- If $X_S \neq \emptyset$ and $X'_S \neq \emptyset$ for some S in $\mathscr{S}$, we delete P from $\mathscr{P}'_u$ corresponding to S and add the path obtained by concatenating P and $(X'_S(2)X_S(1)X_S(2))$ to $\mathscr{P}_u$.
- If $X'_S = \emptyset$ and $X_S \neq \emptyset$ for some S in $\mathscr{S}$, we add the new path $(X_S(1)X_S(2))$ to the set $\mathscr{P}_u$.
- If $X'_o \neq \emptyset$ and $X_o \neq \emptyset$, then there is a monotone path P in $\mathscr{P}'_d$ that intersects Q_{i-1} with $B' \in \{1,2\}$. We delete P from $\mathscr{P}'_d$ and add the path obtained by concatenating P and $(X'_o(2)X_o(1)X_o(2))$ to $\mathscr{P}_d$.
- If $X'_o = \emptyset, X_o \neq \emptyset$, we add the path $(X_o(1)X_o(2))$ to $\mathscr{P}_d$.
- All the paths that are not modified in $\mathscr{P}'_d$ (and $\mathscr{P}'_u$) are included into $\mathscr{P}_d$ (and $\mathscr{P}_u$). If there are vertices in Q_i which are not in any path in $\mathscr{P}_u \cup \mathscr{P}_d$, then we add them to the paths in between their relevant vertices while preserving monotonicity. This is always possible due to **(VI.9)**.

Thus, we get a collection of monotone paths $\mathscr{P}_d \cup \mathscr{P}_u$ that cover all the vertices of G_i while satisfying the required properties given by α. Hence, $|T_i(\alpha)| \leq |T_{i-1}(\beta')| + B^* \leq \min\{|T_{i-1}(\beta)| + B^*\}$ (over all valid and compatible β).

Conversely, let $\mathscr{P}_d \cup \mathscr{P}_u$ be an optimal partial solution for $T_i(\alpha)$. Let us restrict the paths in $\mathscr{P}_d \cup \mathscr{P}_u$ to G_{i-1} and get $\mathscr{P}'_d \cup \mathscr{P}'_u$. Then, $\mathscr{P}'_d \cup \mathscr{P}'_u$ is a partial solution for $T_{i-1}(\beta')$ where $\beta' = (\mathscr{S}'_f, \mathscr{X}', X'_o, \mathscr{A}', B')$ such that

- $\mathscr{S}'_f \subseteq \mathscr{S}_f$ is the set of segments that are completely contained in G_{i-1}.
- For each $S \in \mathscr{S}$, X'_S is the ordered set of the first and last relevant vertices (not necessarily distinct) of the path corresponding to S in Q_{i-1}. The set $\mathscr{X}'$ is the set $\{X'_S : S \in \mathscr{S}\}$.
- X'_o is the ordered set of the first and last relevant vertices (not necessarily distinct) in Q_{i-1} of the path P in $\mathscr{P}_d$ with $V(P) \cap Q_{i-1} \neq \emptyset$.
- The string $\mathscr{A}'$ is defined as follows.
  - $a'_S = 0$ iff the path corresponding to S in $\mathscr{P}'_u$ has no vertex from Q_{i-1}.
  - $a'_S = 1$ iff the path P corresponding to S in $\mathscr{P}'_u$ neither starts nor ends in Q_{i-1} but has a vertex from Q_{i-1}.
  - $a'_S = 2$ iff the path P corresponding to S in $\mathscr{P}'_u$ starts but does not end in Q_{i-1}.
  - $a'_S = 3$ iff the path P corresponding to S in $\mathscr{P}'_u$ ends but does not start in Q_{i-1}.
  - $a'_S = 4$ iff the path P corresponding to S in $\mathscr{P}'_u$ starts and ends in Q_{i-1}.
- B' takes a value in $\{0,1,2,3,4\}$ in the same way as any a'_S does with P corresponding to the path in $\mathscr{P}_d$ that intersects Q_{i-1}.

Then β' is compatible with α and $|T_{i-1}(\beta')| + (1 - B^*) = |T_i(\alpha)|$ and hence $\min\{|T_{i-1}(\beta)| + (1 - B^*)\} \leq |T_i(\alpha)|$ (over all valid and compatible β).

Finally, from Observation 81, the maximum number of entries in T is $\mathscr{O}^*(2^{\mathscr{O}(k \log k)})$ and each entry (that only involves a minimum computation over $\mathscr{O}^*(2^{\mathscr{O}(k \log k)})$ entries) can be computed in $\mathscr{O}^*(2^{\mathscr{O}(k \log k)})$ time. □ □

3.6.3 Overall Algorithm

Now, we are ready to prove the main result of this section.

Theorem 11. PATH COVER *parameterized by the size k of a proper interval deletion set can be solved in $\mathcal{O}^*(2^{\mathcal{O}(k\log k)})$ time.*

Proof. Consider an instance $\mathscr{I} = (H,T,r)$ of PATH COVER. Let π and $\mathscr{Q} = \{Q_1,\cdots,Q_q\}$ denote the proper interval ordering and clique partition of $G = H - T$. Let $T = \{t_1,\ldots,t_k\}$. From Corollaries 60 and 67, there is a minimum path cover $\mathscr{P}$ of H that is canonical satisfying the properties listed in Definition 65. We first guess the properties of $\mathscr{P}$ given in Section 3.6.1. There are $2^{\mathcal{O}(k\log k)}$ choices for such a guess and for each such choice we get a tuple $\vartheta = (\mathscr{S},\mathscr{F},\mathscr{L},\mathscr{F}_1,\mathscr{L}_1)$. The task is now to find an assignment of segments to variables in $\mathscr{S}$ that satisfy the relationships given by ϑ resulting in a set of paths $\mathscr{P}_m$ while minimizing the number opt_ϑ of paths in $\mathscr{P}_o$. This task is accomplished by the dynamic programming procedure described in Section 3.6.2. Following is the description of the algorithm and an illustration of the same is given in Figure 3.15.

Algorithm for PATH COVER

1. Generate all valid tuples $(\mathscr{S},\mathscr{F},\mathscr{L},\mathscr{F}_1,\mathscr{L}_1)$ using the guessing phase described in Section 3.6.1.

2. For every valid $\vartheta = (\mathscr{S},\mathscr{F},\mathscr{L},\mathscr{F}_1,\mathscr{L}_1)$, run the following subroutine.

   Subroutine(ϑ)

   - Initialize $opt = n$ and $T_0(\alpha) = 0$ for each valid α.
   - For all $i \in [q]$

```
    for each valid $\alpha$
        $T_i(\alpha) = \infty$.
```

```
• For each $i \in [q]$ do
    for each valid entry $T_i(\alpha)$ where $\alpha = (\mathscr{S}_f, \mathscr{X}, X_o, \mathscr{A}, B)$
        for each valid entry $T_{i-1}(\beta)$ compatible with $\alpha$
            Let $\beta = (\mathscr{S}'_f, \mathscr{X}', X'_o, \mathscr{A}', B')$.
            if $(T_i(\alpha) \geq T_{i-1}(\beta) + B^*)$
                $T_i(\alpha) = T_{i-1}(\beta) + B^*$ where $B^* = 1$ if $B \in \{3,4\}$
                                                    and 0 otherwise.
• For each valid entry $T_q(\alpha)$ where $\alpha = (\mathscr{S}_f, \mathscr{X}, X_o, \mathscr{A}, B)$
    if $\mathscr{S}_f = \mathscr{S}$ and $B \in \{0,3,4\}$
        if $opt \geq T_q(\alpha)$
            $opt = T_q(\alpha)$
• $opt_\vartheta = opt + \ell$ where $\ell = |\mathscr{P}_m|$ as guessed in Step 1.

3. Return the smallest $opt_\vartheta$ among all valid $\vartheta$ generated in Step 1.
```

Let us now analyze the running time for the dynamic programming procedure. From Observation 81, the maximum number of entries in the dynamic programming table is $\mathscr{O}^*(2^{\mathscr{O}(k \log k)})$. Computing an optimum solution for a single entry takes $2^{\mathscr{O}(k \log k)} n^{\mathscr{O}(1)}$ time from Lemma 82. Then, the size of a minimum path cover of H is given by the minimum value of $|\mathscr{P}_m| + opt_\vartheta$ over all choices for ϑ.

Finally, we show how to construct a minimum path cover of H given an assignment to segments corresponding to an optimum $\vartheta = (\mathscr{S}, \mathscr{F}, \mathscr{L}, \mathscr{F}_1, \mathscr{L}_1)$. Let $\mathscr{P}_d \cup \mathscr{P}_u$ be an optimum solution of an entry in T_q. Let $r = |\mathscr{P}_m|$ as guessed in Step 1. We construct a path cover of H in the following way.

- Initialize $\mathscr{P}_m$ as $\mathscr{P}_u$. For a path $P \in \mathscr{P}_m$, if $\mathscr{F}_1(P) = m_1$, add the vertex m_1 to the beginning of the path P. Similarly if $\mathscr{L}_1(P) = m_2$, add the vertex m_2 to the end of the path P.

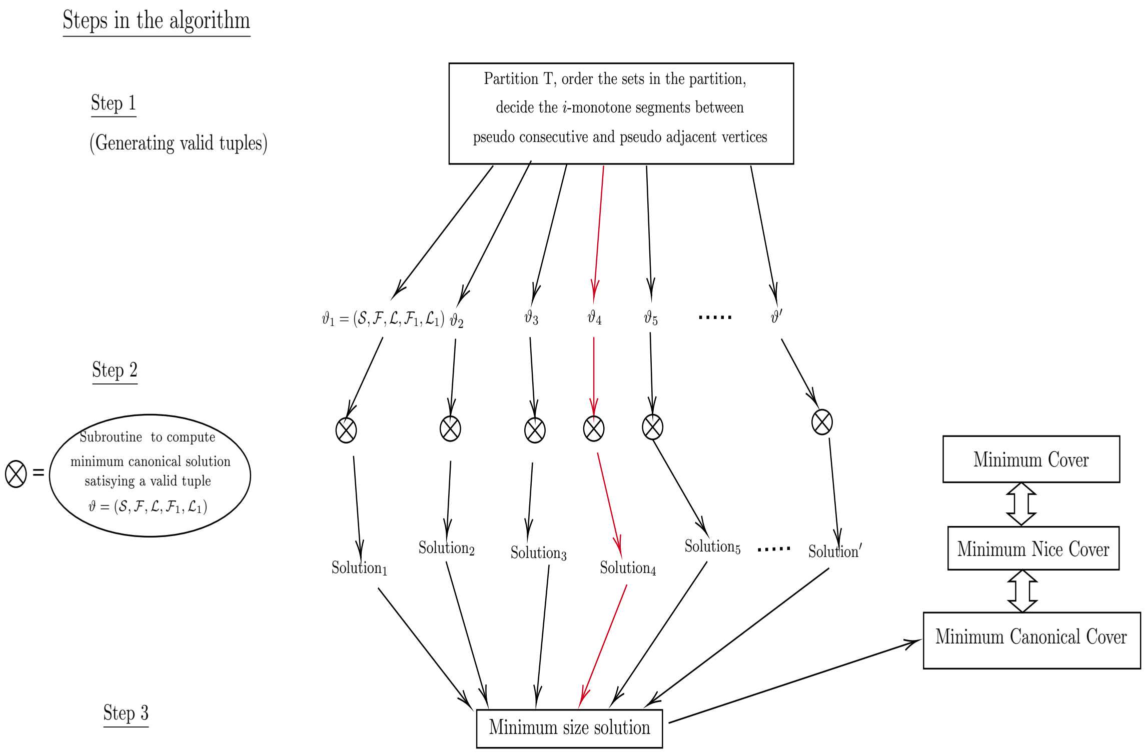

Steps in the algorithm
Step 1
(Generating valid tuples)
Partition T, order the sets in the partition, decide the i-monotone segments between pseudo consecutive and pseudo adjacent vertices
$\vartheta_1 = (\mathcal{S}, \mathcal{F}, \mathcal{L}, \mathcal{F}_1, \mathcal{L}_1)$ ϑ_2 ϑ_3 ϑ_4 ϑ_5 ..... ϑ'
Step 2
$\otimes$ = Subroutine to compute minimum canonical solution satisying a valid tuple $\vartheta = (\mathcal{S}, \mathcal{F}, \mathcal{L}, \mathcal{F}_1, \mathcal{L}_1)$
Solution$_1$
Solution$_2$
Solution$_3$
Solution$_4$
Solution$_5$ .....
Solution$'$
Minimum Cover
Minimum Nice Cover
Minimum Canonical Cover
Step 3
Minimum size solution

Fig 3.15: Outline of the Algorithm

- If there are two paths $P_1, P_2 \in \mathscr{P}_m$ that start/end at the same vertex, then concatenate P_1 and P_2 to get a new path P and add it to $\mathscr{P}_m$. Delete P_1 and P_2 from $\mathscr{P}_m$.

Now, $\mathscr{P}_d \cup \mathscr{P}_m$ is a path cover for H of size $|\mathscr{P}_d| + \ell$ where $\ell = |\mathscr{P}_m|$. Hence the overall running time of the algorithm is $2^{\mathscr{O}(k \log k)} n^{\mathscr{O}(1)}$. □ □

3.7 Cycle Cover Parameterized by Proper Interval deletion Set

Let $\mathscr{C}$ denote a minimum canonical cycle cover of H. We define the following functions that help us to understand the relationship between the segments of a canonical cycle cover $\mathscr{C}$.

- $\mathscr{F} : S(\mathscr{C}) \times S(\mathscr{C}) \to \{0,1\}$ where $\mathscr{F}(S, S') = 1$ if and only if S and S' start at the same vertex.

- $\mathscr{L} : S(\mathscr{C}) \times S(\mathscr{C}) \to \{0,1\}$ where $\mathscr{L}(S,S') = 1$ if and only if S and S' end at the same vertex.

- $\mathscr{F}_1 : S(\mathscr{C}) \to T \cup \{0\}$ where $\mathscr{F}_1(S) = t$ if S starts immediately after t, otherwise $\mathscr{F}_1(S) = 0$.

- $\mathscr{L}_1 : S(\mathscr{C}) \to T \cup \{0\}$ where $\mathscr{L}_1(S) = t$ if S ends just before t, otherwise $\mathscr{L}_1(S) = 0$.

Given $\mathscr{C}$, determining $\mathscr{F}$, $\mathscr{F}_1$, $\mathscr{L}$ and $\mathscr{L}_1$ is easy. It is now natural to ask what choices of $(\mathscr{F}, \mathscr{L}, \mathscr{F}_1, \mathscr{L}_1)$ lead to a minimum canonical cycle cover. Let us first guess the size of $S(\mathscr{C})$. From Lemma 72, it is at most $3k$. For a correct choice of this number, the choice $(\mathscr{F}, \mathscr{L}, \mathscr{F}_1, \mathscr{L}_1)$ that minimizes the size of a minimum cycle cover of $G[X]$ where $V(G) \setminus X$ is the set of vertices that are in some segment assigned to a variable in $\mathscr{S}$ is the one that results in a minimum canonical cycle cover $\mathscr{C}$.

3.7.1 The Guessing Phase

With this information, we proceed as follows. Let $\mathscr{C}$ be a minimum canonical cyle cover that we are looking for. We first guess the following properties of $\mathscr{C}$. Intialize $\mathscr{S}$ to be the empty set.

1. We guess the number ℓ of cycles in $\mathscr{C}_m$. Clearly, $\ell \leq k$ and the number of choices for ℓ is k. Let $C_1, \ldots, C_\ell$ denote the cycles in $\mathscr{C}_m$.

2. For each $C_i \in \mathscr{C}_m$, we guess the order of vertices of $V(C_i) \cap T$. The number of possible choices in this step is $2^{\mathscr{O}(k \log k)}$.

3. For each $C_i \in \mathscr{C}_m$ with $V(C_i) \cap T = \{t\}$, we guess if the path P in C_i between the two neigbours of t is 1-monotone or 2-monotone or 3-monotone. The number of possible choices in this step is $\mathscr{O}(3^k)$.

- If P is 1-monotone, then we add the variable S^i to $\mathscr{S}$ and set $\mathscr{F}_1(S^i) = t$, $\mathscr{L}_1(S^i) = t$.
- If the P is 2-monotone, then we add the variables S^i_1 and S^i_2 to $\mathscr{S}$. We guess one of the following two choices.
  - Set $\mathscr{L}_1(S^i_1) = t$, $\mathscr{F}_1(S^i_1) = 0$, $\mathscr{F}_1(S^i_2) = 0$, $\mathscr{L}_1(S^i_2) = t$, $\mathscr{F}(S^i_1, S^i_2) = 1$, $\mathscr{L}(S^i_1, S^i_2) = 0$.
  - Set $\mathscr{F}_1(S^i_1) = t$, $\mathscr{L}_1(S^i_1) = 0$, $\mathscr{F}_1(S^i_2) = t$, $\mathscr{L}_1(S^i_2) = 0$, $\mathscr{L}(S^i_1, S^i_2) = 1$, $\mathscr{F}(S^i_1, S^i_2) = 0$.
- If P is 3-monotone, then we add the variables S^i_1, S^i_2 and S^i_3 to $\mathscr{S}$. We set $\mathscr{L}_1(S^i_1) = t$, $\mathscr{F}_1(S^i_1) = 0$, $\mathscr{F}_1(S^i_2) = 0$, $\mathscr{L}_1(S^i_2) = 0$, $\mathscr{F}_1(S^i_3) = t$, $\mathscr{L}_1(S^i_3) = 0$, $\mathscr{F}(S^i_1, S^i_2) = 1$, $\mathscr{L}(S^i_2, S^i_3) = 1$, $\mathscr{F}(S^i_1, S^i_3) = 0$, $\mathscr{F}(S^i_2, S^i_3) = 0$, $\mathscr{L}(S^i_1, S^i_2) = 0$, $\mathscr{L}(S^i_1, S^i_3) = 0$.

4. For each $C_i \in \mathscr{C}_m$ with $|V(C_i) \cap T| \geq 2$, for each pair of pseudo-consecutive vertices t and t' in C_i, we guess if t and t' are consecutive in C_i (in which case t and t' must be adjacent) or not. It t and t' are not consecutive in C_i, then we guess if the maximal subpath P of the (t,t')-path that is contained in G is 1-monotone or 2-monotone or 3-monotone. The number of possible choices in this step is $\mathscr{O}(3^k)$.
   - If P is 1-monotone, then we add the variable S^i to $\mathscr{S}$ and set $\mathscr{F}_1(S^i) = t$, $\mathscr{L}_1(S^i) = t'$.
   - If P is 2-monotone, then we add the variables S^i_1 and S^i_2 to $\mathscr{S}$. We guess one of the following two choices.
     - Set $\mathscr{L}_1(S^i_1) = t$, $\mathscr{F}_1(S^i_1) = 0$, $\mathscr{F}_1(S^i_2) = 0$, $\mathscr{L}_1(S^i_2) = t'$, $\mathscr{F}(S^i_1, S^i_2) = 1$, $\mathscr{L}(S^i_1, S^i_2) = 0$.
     - Set $\mathscr{F}_1(S^i_1) = t$, $\mathscr{L}_1(S^i_1) = 0$, $\mathscr{F}_1(S^i_2) = t'$, $\mathscr{L}_1(S^i_2) = 0$, $\mathscr{L}(S^i_1, S^i_2) = 1$, $\mathscr{F}(S^i_1, S^i_2) = 0$.
   - If P is 3-monotone, then we add the variables S^i_1, S^i_2 and S^i_3 to $\mathscr{S}$. We set $\mathscr{L}_1(S^i_1) = t$, $\mathscr{F}_1(S^i_1) = 0$, $\mathscr{F}_1(S^i_2) = 0$, $\mathscr{L}_1(S^i_2) = 0$, $\mathscr{F}_1(S^i_3) = t'$, $\mathscr{L}_1(S^i_3) = 0$,

$\mathscr{F}(S_1^i, S_2^i) = 1$, $\mathscr{L}(S_2^i, S_3^i) = 1$, $\mathscr{F}(S_1^i, S_3^i) = 0$, $\mathscr{F}(S_2^i, S_3^i) = 0$, $\mathscr{L}(S_1^i, S_2^i) = 0$, $\mathscr{L}(S_1^i, S_3^i) = 0$.

5. For each pair S and S' of variables in $\mathscr{S}$ such that $\mathscr{F}(S,S')$ (or $\mathscr{L}(S,S')$) is not yet set is set to 0. Similarly, for each variable S in $\mathscr{S}$ such that $\mathscr{F}_1(S)$ (or $\mathscr{L}_1(S)$) is not yet set is set to 0.

As $|\mathscr{C}_m| \leq k$, the maximum number of guesses in Steps 1-2 is $2^{\mathscr{O}(k \log k)}$. The maximum number of guesses in Steps 3-5 is asymptotically upper bounded by the maximum number of choices for $(\mathscr{S}, \mathscr{F}, \mathscr{L}, \mathscr{F}_1, \mathscr{L}_1)$ which is $2^{\mathscr{O}(k)}$. Once these choices are fixed, the problem of finding a mnimum cycle cover $\mathscr{C}$ now reduces to the problem of finding an assignment of segments to variables in $\mathscr{S}$ that satisfy the relationships given by $(\mathscr{S}, \mathscr{F}, \mathscr{L}, \mathscr{F}_1, \mathscr{L}_1)$ while minimizing the size of a minimum cycle cover of $G - X$ where X is the set of vertices of H that are in a segment assigned to some variable in $\mathscr{S}$. In other words, we find an assignment of segments to variables in $\mathscr{S}$ that satisfy the relationships given by $(\mathscr{S}, \mathscr{F}, \mathscr{L}, \mathscr{F}_1, \mathscr{L}_1)$ resulting in a set of paths $\mathscr{C}_m$ while minimizing the number of paths in $\mathscr{C}_o$. Note that not all choices of $(\mathscr{S}, \mathscr{F}, \mathscr{L}, \mathscr{F}_1, \mathscr{L}_1)$ may necessarily lead to a minimum cycle cover $\mathscr{C}_o \cup \mathscr{C}_m$ of H. However, at least one of the choices that we generate leads to one.

3.7.2 Finding an Assignment of Segments for $\vartheta = (\mathscr{S}, \mathscr{F}, \mathscr{L}, \mathscr{F}_1, \mathscr{L}_1)$

Let $Q_0 = \emptyset$. For each $i \in [q]$, let G_i denote the graph $G[Q_1 \cup \cdots \cup Q_i]$. Let us first understand the interaction of the solution (minimum canonical cycle cover with the properties given by $(\mathscr{S}, \mathscr{F}, \mathscr{L}, \mathscr{F}_1, \mathscr{L}_1)$) with G_i. Subsequently, we refer to $\mathscr{S}$ as segments instead of variables that have to be assigned segments.

Index of an Entry: An entry in the table T_i is indexed by a tuple $(S_f, \mathscr{X}, \mathscr{A}, \mathscr{X}_o = (C_s, C_c, C_e, C_{in}))$ with the following interpretation.

- $\mathscr{S}_f \subseteq \mathscr{S}$ denotes the segments that have no vertex from $Q_{i+1} \cup \ldots \cup Q_q$. That is, these segments are completely contained in G_i.

- $\mathscr{X}$ denotes the set of relevant vertices of all segments from $\mathscr{S}$ in Q_i. That is, for every $S \in \mathscr{S}$, X_S in $\mathscr{X}$ is the set of relevant vertices of the segment S in Q_i.

  - If X_S is the empty set, then the segment corresponding to S has no vertex from Q_i.
  - Otherwise, X_S has a single vertex or an ordered pair of vertices.
    * If X_S is an ordered pair of vertices (v_1, v_2), we call v_1 the first relevant vertex (denoted by $X_S(1)$) and v_2 the last relevant vertex of S in Q_i (denoted by $X_S(2)$).
    * If X_S has a single vertex v, we call v both first and last relevant vertex of S in Q_i.

- $\mathscr{A} \in [0,1,2,3,4]^{|S(\mathscr{C})|}$ represents the interactions of the segments from $\mathscr{S}$ with Q_i.

  - $a_S = 0$ iff the segment S does not intersect Q_i.
  - $a_S = 1$ iff the segment S has at least one vertex from Q_i but neither starts nor ends in Q_i.
  - $a_S = 2$ iff the segment S starts but does not end in Q_i.
  - $a_S = 3$ iff the segment S ends but does not start in Q_i.
  - $a_S = 4$ iff the segment S starts and ends in Q_i.

- $\mathscr{X}_o$ contains the sets of relevant vertices of the 2-monotone cycles in $\mathscr{C}_o$ that intersect Q_i. C_{in} consists of an ordered set of three distinct vertices corresponding to two relevant vertices of the cycle in $\mathscr{C}_o$ of Type 1 (defined in Proposition 70) and an irrelevant vertex of this cycle. The sets C_s, C_e and C_c contain the relevant vertices of the cycles in $\mathscr{C}_o$ of Types 3, 2 and 4, respectively.

We denote the two pairs of relevant set of vertices of the two maximal monotone segments of Type 3 cycle by $C_s[1]$ and $C_s[2]$. Similarly, $C_e[p], C_c[p], C_{in}[p]$ are defined for $p \in [2]$. Also, $C_\gamma[i](j)$ denotes the jth relevant vertex of the ith relevant set of vertices of the cycle of type indicated by γ. If there is no cycle of a particluar type (say Type 3), then C_s has empty entries. The set $\mathscr{X}$ together with $\mathscr{X}_o$ denote all the relevant vertices of the cycle cover (that we are looking for) in Q_i.

Valid Indices: An index $(\mathscr{S}_f, \mathscr{X}, \mathscr{A}, \mathscr{X}_o = (C_s, C_c, C_e, C_{in}))$ corresponding to an entry in T_i is valid if the following conditions are satisfied.

- **(VI.1)** Two sets X_S and $X_{S'}$ are vertex-disjoint if $\mathscr{F}(S,S') = \mathscr{L}(S,S') = 0$.
- **(VI.2)** The sets C_s, C_e, C_c, C_{in} are disjoint from each other as well as from the sets in $\mathscr{X}$.
- **(VI.3)** If $\mathscr{F}(S,S') = 1$ and $a_S \in \{2,4\}$, then $a_{S'} \in \{2,4\}$ and X_S and $X_{S'}$ have the same first vertex.
- **(VI.4)** If $\mathscr{L}(S,S') = 1$ and $a_S \in \{3,4\}$, then $a_{S'} \in \{3,4\}$ then X_S and $X_{S'}$ have the same last vertex.
- **(VI.5)** $a_S = 0$ iff $X_S = \emptyset$.
- **(VI.6)** $a_S \in \{3,4\}$ iff $S \in \mathscr{S}_f$ and $a_S \in \{1,2\}$ iff $S \notin \mathscr{S}_f$.
- **(VI.7)** The set X_S along with a_S for any segment S satisfy the canonical solution properties given by Definition 78. For example, if $X_S = (a,b)$, $a_S = 2$, then $a \in L^i$, $b \in R^i$.
- **(VI.8)** For every irrevant vertex $v \in Q_i$, either there exists $S \in \mathscr{S}$ such that $\pi(X_S(1)) \leq \pi(x) \leq \pi(X_S(2))$ or there is a pair (a,b) of relevant vertices in $\mathscr{X}_o$ such that the $\pi(a) \leq \pi(v) \leq \pi(b)$.

- **(VI.9)** If $C_s \neq \emptyset$, then either $C_s = ((a,a),(a,a))$ or $C_s = ((a,b),(a,c))$ where a,b,c are distinct vertices.
- **(VI.10)** If $C_e \neq \emptyset$, then either $C_e = ((a,a),(a,a))$ or $C_e = ((b,a),(c,a))$ where a,b,c are distinct vertices.
- **(VI.11)** If $C_c \neq \emptyset$ then $C_c \in \{((a,a),(b,b)),((a,a),(b,c)),((a,b),(c,c)),$ $((a,b),(c,d))\}$ where a,b,c,d are distinct vertices.
- **(VI.12)** If $C_{in} \neq \emptyset$, then $|C_{in}| = 3$.
- **(VI.13)** Both C_c and C_e cannot be nonempty. Similary both C_c and C_s cannot be nonempty.

Observe that all segments are disjoint except possibly at the start and end vertices. This is encoded by **(VI.1)** and **(VI.2)**. In particluar, **(VI.1)** ensures that if two segments do not have common start and end vertices then they are vertex-disjoint. That is, their relevant vertices are disjoint. **(VI.3)** and **(VI.4)** ensure that the choice of $(\mathscr{F},\mathscr{L})$ is respected. **(VI.5)** and **(VI.6)** ensure that the intended interpretation of $\mathscr{A}$ is respected. In particluar, **(VI.6)** makes sure that only the segments from $\mathscr{S}$ which are ending in Q_i are taken into $\mathscr{S}_f$. **(VI.7)** follows from Lemma 78 and it ensures that the relevant vertices of segments are chosen from $B(Q_i)$ and these segments can be concatenated to the modulator vertices appropriately. **(VI.8)** makes sure that any irrelevant vertex can always be added in between the relevant vertices of some segment in $\mathscr{S}$ or some monotone path of a 2-monotone cycle in $\mathscr{C}_o$. **(VI.9)** to **(VI.12)** make sure that the two maximal monotone paths forming a 2-monotone cycle together are internally vertex-disjoint from each other and start/end at the same vertices. **(VI.13)** follows from Proposition 70.

Observation 83. *For each $i \in [q]$, the maximum number of valid indices is $\mathscr{O}^*(2^{\mathscr{O}(k\log k)})$.*

Proof. Let us bound the number of valid entries $\alpha = (\mathscr{S}_f, \mathscr{X}, \mathscr{A}, \mathscr{X}_o)$ in T_i. There are $2^{\mathscr{O}(k)}$ choices for $\mathscr{S}_f$. Every relevant set in $\mathscr{X}$ is chosen from a set of at most $20k$ vertices.

Hence there are $2^{\mathscr{O}(k \log k)}$ choices for $\mathscr{X}$. We have $\mathscr{O}(n^16)$ ways to select $\mathscr{X}_o$ and there are $2^{\mathscr{O}(k)}$ choices for $\mathscr{A}$. Thus, the maximum number of entries in T_i is $2^{\mathscr{O}(k \log k)} n^{\mathscr{O}(1)}$. $\square$ $\square$

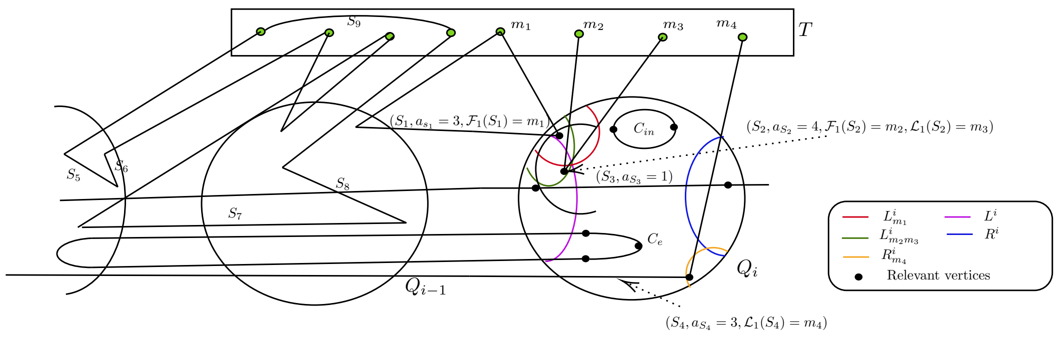
(Optimum) Partial Solutions: A collection of paths and cycles $\mathscr{C}_d \uplus \mathscr{P}_u \uplus P_2$ is a partial solution of $T_i(\sigma)$ where $\sigma = (\mathscr{S}_f, \mathscr{X}, \mathscr{A}, \mathscr{X}_o = (C_s, C_c, C_e, C_{in}))$ if the following conditions hold. Let $h : \mathscr{S}_f \uplus \{S \in \mathscr{S} : a_S \in \{1,2\}\} \to \mathscr{P}_u$ denote the assignment of paths in $\mathscr{P}_u$ to variables in $\mathscr{S}_f \uplus \{S \in \mathscr{S} : a_S \in \{1,2\}\}$.

- **(PS.1)** $|\mathscr{P}_u| = |\mathscr{S}_f| + |\{S \in \mathscr{S} : a_S \in \{1,2\}\}|$ and h is injective.
- **(PS.2)** $\mathscr{P}_u$ is a set of monotone paths and $\mathscr{C}_d$ is a collection of 2-monotone cycles.
- **(PS.3)** Every path/cycle in $\mathscr{C}_d \uplus \mathscr{P}_u \uplus P_2$ is contained in G_i.
- **(PS.4)** P_2 is either empty or contains one 2-monotone path with both endpoints in Q_i.
- **(PS.5)** Every vertex from $Q_1 \cup \ldots \cup Q_i$ is in some element in $\mathscr{C}_d \cup \mathscr{P}_u \cup P_2$.
- **(PS.6)** Every pair of distinct elements in $\mathscr{C}_d \cup P_2$ are vertex-disjoint and an element from $\mathscr{C}_d \cup P_2$ is vertex-disjoint with an element in $\mathscr{P}_u$. Further, two paths $P, P' \in \mathscr{P}_u$ are internally vertex-disjoint, moreover, they are vertex-disjoint except when $\mathscr{F}(P,P') = 1$ or $\mathscr{L}(P,P') = 1$.
- **(PS.7)** For each $S \in \text{dom}(h)$, $h(S)$ starts at a vertex in $N(m_1)$ if $\mathscr{F}_1(S) = m_1$.
- **(PS.8)** For each $S \in \mathscr{S}_f$, $h(S)$ ends at a vertex in $N(m_2)$ if $\mathscr{L}_1(S) = m_2$.
- **(PS.9)** If $\mathscr{F}(S,S') = 1$ and $S \in \text{dom}(h)$, then $S' \in \text{dom}(h)$ and $h(S)$ and $h(S')$ start at the same vertex.
- **(PS.10)** If $\mathscr{L}(S,S') = 1$ and $S \in \mathscr{S}_f$, then $S' \in \mathscr{S}_f$ and $h(S)$ and $h(S')$ end at the same vertex.

- **(PS.11)** The paths in $\mathscr{P}_u$ take their respective relevant vertices in Q_i according to the assignment $\mathscr{X}$, i.e. the first and last vertices of X_S are the first and last vertices of $h(S)$ in Q_i.
- **(PS.12)** Any path $S \in \mathscr{P}_u$ starts and ends in Q_i iff $a_s \in \{2,4\}$.
- **(PS.13)** Any path $S \in \mathscr{P}_u$ does not start but ends in Q_i iff $a_s \in \{1,3\}$.
- **(PS.14)** If $C_{in} \neq \emptyset$, then there is a cycle in $\mathscr{C}_d$, contained in $G[Q_i]$ containing the first and last vertices of C_{in} as relevant vertices.
- **(PS.15)** If $C_e \neq \emptyset$, then there is a cycle in $\mathscr{C}_d$ that ends but does not start in Q_i and its relevant vertices in Q_i are given by C_e.
- **(PS.17)** If $C_s \neq \emptyset$, then the path in P_2 starts in Q_i and its relevant vertices in Q_i are given by C_s.
- **(PS.18)** If $C_c \neq \emptyset$, then the path in P_2 starts before Q_i and its relevant vertices in Q_i are given by C_c.

Let $B_1^* = 1$ iff $C_{in} \neq \emptyset$, 0 otherwise. Similarly $B_2^* = 1$ iff $C_e \neq \emptyset$, 0 otherwise. Over all possible partial solutions $\mathscr{C}_d \cup \mathscr{P}_u \cup P_2$, $T_i(\sigma)$ stores the one that minimizes the value of $|\mathscr{C}_d| + B_1^* + B_2^*$. Such a partial solution is called an **optimum partial solution**. In other words, $T_i(\sigma)$ stores a partial solution that minimizes the number of cycles contained inside $\mathscr{C}_d$ that end in G_i. Also, as $C_e \neq \emptyset$ implies a cycle finishing in Q_i, we add this cycle to our solution and increase its value by 1. A similar contribution is for C_{in} too.

Clearly, an optimum solution for an entry in T_q where every cycle in the solution has ended in $G_q = G$ gives the required answer. In the dynamic programming algorithm that we will subsequently describe, we compute partial solutions for G_i and use them to compute partial solutions for G_{i+1}. So let us understand how the solution that we are looking for interacts with G_i. A partial solution for G_i is the one that minimizes the number of cycles contained in $\mathscr{C}_o$ when restricted to G_i. To determine such a partial solution we need to know the segments that have already terminated before Q_i and the interaction of segments in $S(\mathscr{P}) \cup \mathscr{C}_o$ with Q_i. Refer to the following for an illustration.



S_9
m_1
m_2
m_3
m_4
T
$(S_1, a_{s_1} = 3, \mathcal{F}_1(S_1) = m_1)$
C_{in}
$(S_2, a_{S_2} = 4, \mathcal{F}_1(S_2) = m_2, \mathcal{L}_1(S_2) = m_3)$
S_5
S_6
$(S_3, a_{S_3} = 1)$
S_8
S_7
C_e
Q_i
Q_{i-1}
$L^i_{m_1}$
$L^i_{m_2m_3}$
$R^i_{m_4}$
L^i
R^i
Relevant vertices
$(S_4, a_{S_4} = 3, \mathcal{L}_1(S_4) = m_4)$

Fig 3.16: Interaction of a canonical cycle cover with $Q_1 \cup \ldots \cup Q_i$

Compatible Entries: A valid entry in T_i with index $(\mathscr{S}_f, \mathscr{X}, \mathscr{A}, \mathscr{X}_o = (C_s, C_c, C_e, C_{in}))$ is compatible with a valid entry in T_{i-1} with index $(\mathscr{S}'_f, \mathscr{X}', \mathscr{A}', \mathscr{X}'_o = (C'_s, C'_c, C'_e, C'_{in}))$ if the following conditions hold.

- **(CI.1)** $\mathscr{S}_f = \mathscr{S}'_f \cup \{S : a_S \in \{3,4\}\}$.
- **(CI.2)** If $a_S = 0$, then $a'_S \in \{0,3,4\}$.
- **(CI.3)** If $a_S = 1$, then $a'_S \in \{1,2\}$ and the first relevant vertex of X_S is adjacent to the last relevant vertex X'_S.
- **(CI.4)** If $a_S = 2$, then $a'_S = 0$.
- **(CI.5)** If $a_S = 3$, then $a'_S \in \{1,2\}$ and the first relevant vertex of X_S is adjacent to the last relevant vertex X'_S.
- **(CI.6)** If $a_S = 4$, then $a'_S = 0$.
- **(CI.7)** If $a_S \neq 0$, then $S \notin S'_f$.
- **(CI.8)** Exactly one of C_c and C_e (say C) is non-empty iff exactly one of C'_c and C'_s (say C') is non-empty. Moreover, $C'[1](2)$ and $C'[2](2)$ are adjacent to $C[1](1)$ and $C[2](1)$, respectively.

(CI.1) makes sure that the segments completely contained in G_i are the segments completed before Q_i and the segments that end in Q_i. **(CI.2)** ensures that if a segment does not intersect Q_i, it is not continued (unfinished) from Q_{i-1}. **(CI.3)** and **(CI.5)** ensure we can extend a segment from Q_{i-1} to Q_i. **(CI.4)**, **(CI.6)** and **(CI.7)** make sure that the segments/monotone paths that have vertices in Q_i are disjoint from the ones that end before Q_i. **(CI.8)** makes sure that if a cycle continues or ends in Q_i, then it must have continued from Q_{i-1} i.e. it ensures we can extend the cycles in $\mathscr{C}_o$ from Q_{i-1} to Q_i.

Lemma 84. *$T_0(\alpha) = 0$ and for each $i \in [q]$, $T_i(\alpha) = \min\{T_{i-1}(\beta) + B_1{}^* + B_2{}^*\}$ over all valid β compatible with α. Further, all the entries in T can be computed in $\mathscr{O}^*(2^{\mathscr{O}(k \log k)})$ time.*

Proof. Since Q_0 is the empty clique, 0 cycles are required to cover vertices of G_0, hence $T_0(\alpha) = 0$. Let $i \in [q]$. Suppose $T_{i-1}(\beta)$ has been correctly computed for G_{i-1} for all valid β. Consider an entry in $T_{i-1}(\beta')$ that is compatible with $T_i(\alpha)$. Let $\alpha = (\mathscr{S}_f, \mathscr{X}, \mathscr{A}, \mathscr{X}_o = (C_s, C_c, C_e, C_{in}))$ and $\beta' = (\mathscr{S}'_f, \mathscr{X}', \mathscr{A}', \mathscr{X}'_o = (C'_s, C'_c, C'_e, C'_{in}))$. Let $\mathscr{C}'_d \cup \mathscr{P}'_u \cup P'_2$ be an optimal partial solution for $T_{i-1}(\beta')$. We will construct a partial solution for $T_i(\alpha)$ from $\mathscr{C}'_d \cup \mathscr{P}'_u \cup P'_2$ in the following manner.

- If $X_S \neq \emptyset$ and $X'_S \neq \emptyset$ for some S in $\mathscr{S}$, we delete P from $\mathscr{P}'_u$ corresponding to S and add the path obtained by concatenating P and $(X'_S(2)X_S(1)X_S(2))$ to $\mathscr{P}_u$.
- If $X'_S = \emptyset$ and $X_S \neq \emptyset$ for some S in $\mathscr{S}$, we add the new path $(X_S(1)X_S(2))$ to the set $\mathscr{P}_u$.
- If $C_c \neq \emptyset$, then we update the 2-monotone path S in $\mathscr{P}'_2$ by connecting the last relevant vertices of the maximal monotone paths of S in Q_{i-1} to their first relevant vertices of C_c. Also we connect the second relevant vertices of C_c to the updated monotone paths in S. We add the updated 2-monotone path S to $\mathscr{P}_2$.
- If $C_e \neq \emptyset$, then we update the 2-monotone path S in $\mathscr{P}'_2$ by connecting the last relevant vertices of the maximal monotone paths of S in Q_{i-1} to their first relevant

vertices of C_c. Also we connect the second relevant vertices of C_c to the updated monotone paths in S. Since the 2-monotone path S is updated to a cycle, we add the cycle to $\mathscr{C}_d$.

- If $C_s \neq \emptyset$, then we add a new 2-monotone path S to $\mathscr{P}_2$ where S consists of two monontone paths that have their first vertices (same vertex) from first relevant vertices of C_s and their next vertices from the second relevant vertices from C_s.

- If $C_{in} \neq \emptyset$, we add a new cycle C to $\mathscr{C}_d$ where C consists of the vertices from C_{in}.

- All the paths and cycles that are not modified in $\mathscr{C}'_d \cup \mathscr{P}'_u \cup P'_2$ are included into $\mathscr{C}_d \cup \mathscr{P}_u \cup P_2$. If there are vertices in Q_i, which are not in any paths or cycles from $\mathscr{C}_d \cup \mathscr{P}_u \cup P_2$, we add them to the paths or cycles in between their relevant vertices while preserving the monotonicity of paths. Because of **(VI.9)** we will always be able to add all the remaining vertices.

This is how we create a partial solution $\mathscr{C}_d \cup \mathscr{P}_u \cup P_2$. From the construction $\mathscr{C}_d \cup \mathscr{P}_u \cup P_2$ covers all the vertices in $Q_1 \cup \ldots Q_i$, as well as satisfies all other conditions to be a partial canonical solution for α. Hence $T_i(\alpha) \leq T_{i-1}(\beta') + B_1{}^* + B_2{}^* \leq min\{T_{i-1}(\beta) + B_1{}^* + B_2{}^*\}$(over all valid and compatible β).

Converesely, let $\mathscr{C}_d \cup \mathscr{P}_u \cup P_2$ be an optimal partial solution for $T_i(\alpha)$. Let us restrict the cycles and paths in $\mathscr{C}_d \cup \mathscr{P}_u \cup P_2$ to G_{i-1} and get $\mathscr{C}'_d \cup \mathscr{P}'_u \cup P'_2$. Then, $\mathscr{C}'_d \cup \mathscr{P}'_u \cup P'_2$ is a partial solution for $T_{i-1}(\beta')$ where $\beta' = (\mathscr{S}'{}_f, \mathscr{X}', \mathscr{A}', \mathscr{X}'_o = (C'_s, C'_c, C'_e, C'_{in}))$ such that

- $\mathscr{S}'_f \subseteq \mathscr{S}_f$ is the set of segments that are completely contained in G_{i-1}.

- For each $S \in \mathscr{S}$, X'_S is the ordered set of the first and last relevant vertices (not necessarily distinct) of the path corresponding to S in Q_{i-1}. The set $\mathscr{X}'$ is the set $\{X'_S : S \in \mathscr{S}\}$.

- The string $\mathscr{A}'$ is defined as follows.

  - $a'_S = 0$ iff the path corresponding to S in $\mathscr{P}'_u$ has no vertex from Q_{i-1}.
  - $a'_S = 1$ iff the path P corresponding to S in $\mathscr{P}'_u$ neither starts nor ends in Q_{i-1} but has a vertex from Q_{i-1}.
  - $a'_S = 2$ iff the path P corresponding to S in $\mathscr{P}'_u$ starts but does not end in Q_{i-1}.
  - $a'_S = 3$ iff the path P corresponding to S in $\mathscr{P}'_u$ ends but does not start in Q_{i-1}.
  - $a'_S = 4$ iff the path P corresponding to S in $\mathscr{P}'_u$ starts and ends in Q_{i-1}.

- C'_s is the relevant set of vertices corresponding to the two maximal monotone paths of a cycle in $\mathscr{C}_d$ or the 2-monotone path in P_2 that starts but does not end in Q_{i-1}.

- C'_c is the relevant set of vertices corresponding to the two maximal monotone paths of a cycle in $\mathscr{C}_d$ or the 2-monotone path in P_2 that intersects Q_{i-1} but neither starts nor ends in it.

- C'_e is the relevant set of vertices corresponding to the two maximal monotone paths of a cycle in $\mathscr{C}_d$ that does not start but ends in Q_{i-1}.

- C'_{in} is the relevant set of vertices corresponding to the two maximal monotone paths of a cycle in $\mathscr{C}_d$ that is completely inside Q_{i-1}.

Then β' is compatible with α and $T_{i-1}(\beta') + {B_1}^* + {B_2}^* = T_i(\alpha)$ and hence $min\{T_{i-1}(\beta) + {B_1}^* + {B_2}^*\} \leq T_i(\alpha)$(over all valid and compatible β).

Finally, from Observation 83, the maximum number of entries in T is $\mathscr{O}^*(2^{\mathscr{O}(k\log k)})$ and each entry (that only involves a minimum computation over $\mathscr{O}^*(2^{\mathscr{O}(k\log k)})$ entries) can be computed in $\mathscr{O}^*(2^{\mathscr{O}(k\log k)})$ time. □ □

3.7.3 Overall Algorithm

Now, we are ready to prove the main result of this section.

Theorem 12. CYCLE COVER *parameterized by the size k of a proper interval deletion set can be solved in* $\mathcal{O}^*(2^{\mathcal{O}(k \log k)})$ *time.*

Proof. Consider an instance $\mathcal{I} = (H, T, r)$ of CYCLE COVER. Let π and $\mathcal{Q} = \{Q_1, \cdots, Q_q\}$ denote the proper interval ordering and clique partition of $G = H - T$. Let $T = \{t_1, \ldots, t_k\}$. From Corollaries 74 and 80, there is a minimum cycle cover $\mathcal{C}$ of H that is canonical satisfying the properties listed in Definition 78. We first guess the properties of $\mathcal{C}$ given in Section 3.7.1. There are $2^{\mathcal{O}(k \log k)}$ choices for such a guess and for each such choice we get a tuple $\vartheta = (\mathcal{S}, \mathcal{F}, \mathcal{L}, \mathcal{F}_1, \mathcal{L}_1)$. The task is now to find an assignment of segments to variables in $\mathcal{S}$ that satisfy the relationships given by ϑ resulting in a set of cycles $\mathcal{C}_m$ while minimizing the number opt_ϑ of cycles in $\mathcal{C}_o$. This task is accomplished by the dynamic programming procedure described in Section 3.7.2. Following is the description of the overall algorithm.

Algorithm for CYCLE COVER

1. Generate all valid tuples $(\mathcal{S}, \mathcal{F}, \mathcal{L}, \mathcal{F}_1, \mathcal{L}_1)$ using the guessing phase described in Section 3.7.1.

2. For every valid $\vartheta = (\mathcal{S}, \mathcal{F}, \mathcal{L}, \mathcal{F}_1, \mathcal{L}_1)$, run the following subroutine.

   Subroutine(ϑ)

```
• Intialize $opt = \infty$ and $T_0(\alpha) = 0$ for all valid $\alpha$.
• For all $i \in [q]$
      for all valid $\alpha$
          $T_i(\alpha) = \infty$.
• For each $i \in [q]$ do
      for each valid entry $T_i(\alpha = (\mathcal{S}_f, \mathcal{X}, \mathcal{A}, \mathcal{X}_o))$
          for each valid entry $T_{i-1}(\beta)$ compatible with $\alpha$
              Let $\beta = (\mathcal{S}'_f, \mathcal{X}', \mathcal{A}', \mathcal{X}'_o)$.
```

```
            if ($T_i(\alpha) \geq T_{i-1}(\beta) + B_1{}^* + B_2{}^*$)
                $T_i(\alpha) = T_{i-1}(\beta) + B_1{}^* + B_2{}^*$ where $B_1{}^*/B_2{}^* = 1$
                                                  iff $C_{in}/C_e \neq \emptyset$ and
                                                      0 otherwise.

    • For each valid entry $T_q(\alpha)$ where $\alpha = (\mathscr{S}_f, \mathscr{X}, \mathscr{A}, \mathscr{X}_o'))$
        if ($S_f = \mathscr{S}$ and $C_s = C_c = \emptyset$)
            if $opt \geq T_q(\alpha)$
                $opt = T_q(\alpha)$.
    • $opt_\vartheta = opt + \ell$ where $\ell = |\mathscr{C}_m|$ as guessed in Step 1.

3. Return the smallest $opt_\vartheta$ among all valid $\vartheta$ generated in Step 1.
```

Let us now analyze the running time for the dynamic programming procedure. From Observation 83, the maximum number of entries in the dynamic programming table is $\mathscr{O}^*(2^{\mathscr{O}(k \log k)})$. Computing an optimum solution for a single entry takes $\mathscr{O}^*(2^{\mathscr{O}(k \log k)})$ time from Lemma 84. Then, the size of a minimum cycle cover of H is given by the minimum value of $|\mathscr{C}_m|+opt_\vartheta$ over all choices for ϑ.

Finally, we show how to construct a minimum cycle cover of H given an assignment to segments corresponding to an optimum $\vartheta = (\mathscr{S}, \mathscr{F}, \mathscr{L}, \mathscr{F}_1, \mathscr{L}_1)$. Let $\mathscr{C}_d \cup \mathscr{P}_u \cup P_2$ be an optimum solution. We construct a cycle cover for H in the following way. Notice that $P_2 = \emptyset$ when we compute the optimum solution on G.

- Initialize $\mathscr{C}_m$ as $\mathscr{P}_u$. For a path $P \in \mathscr{C}_m$, if $\mathscr{F}_1(P) = m_1$, add the vertex m_1 to the beginning of the path P. Similarly if $\mathscr{L}_1(P) = m_2$, add the vertex m_2 to the end of the path P.

- If there are two paths $P_1, P_2 \in \mathscr{C}_m$ that start/end at the same vertex, then concatenate P_1 and P_2 to get a new path P and add it to $\mathscr{C}_m$. Delete P_1 and P_2 from $\mathscr{C}_m$.

- For any two paths $P_1, P_2 \in \mathscr{C}_m$ that start and end at the same vertex, concatenate P_1 and P_2 to get a cycle and add it to $\mathscr{C}_m$. Delete P_1 and P_2 from $\mathscr{C}_m$.

From the construction $\mathscr{C}_d \cup \mathscr{C}_m$ is a cycle cover for H of size $|\mathscr{C}_d| + \ell$ where $\ell = |\mathscr{C}_m|$. Hence the overall running time of the algorithm is $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$. □ □

3.8 Conclusion

We designed linear kernel for CYCLE PACKING in tournaments and polynomial kernel in α-bounded digraphs. The questions about lower bounds and improving the kernel size in α-bounded digraphs still remain open. We also described an FPTalgorithm for CYCLE PACKING parameterized by the size of a proper interval deletion set. As mentioned earlier, CYCLE PACKING parameterized by the size of a proper interval deletion set does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly. Recently, a kernelization framework (called lossy kernelization) that is less stringent than the notion of polynomial kernels was introduced in [73]. It was shown that there are many problems (including CYCLE PACKING parameterized by the solution size) without classical polynomial kernels that admit lossy polynomial kernels. It is interesting to explore the possibility of such a kernel for our problem.

Finally, the status of CYCLE PACKING on interval graphs is an inevitable line of study. Similarly for COVERING problem parameterization by distance to interval graphs would be an interesting idea to pursue.

Chapter 4

Mixed Domination (Total Covering)

In this chapter we study a specific version of COVERING problem known as TOTAL COVERING/ MIXED DOMINATION (MDS) parameterized by various parameters where the objective is to cover a graph with both vertices and edges. We provide a formal definition of the problem in the introduction. Firstly we give an FPTalgorithm running in time $7.465^k n^{\mathcal{O}(1)}$ on general graphs parameterized by solution size k. We complement this result by showing that MDS does not admit an algorithm with running time $2^{o(k)}n^{\mathcal{O}(1)}$ unless the Exponential Time Hypothesis (ETH) fails, and that it does not admit a polynomial kernel unless coNP $\subseteq$ NP/poly. In addition, we provide an algorithm which, given a graph G together with a tree decomposition of width tw, solves MDS in time $6^{tw(G)}n^{\mathcal{O}(1)}$. We finally show that unless the Set Cover Conjecture (SeCoCo) fails, MDS does not admit an algorithm with running time $\mathcal{O}((2-\varepsilon)^{tw(G)}n^{\mathcal{O}(1)})$ for any $\varepsilon > 0$, where tw(G) is the tree-width of G. Next we look at the restriction of MDS to several graph classes and establish the following results.

- On proper interval graphs, MDS is polynomial time solvable.
- On graphs that exclude $K_{d,d}$ as a subgraph, MDS admits a kernel of size $\mathcal{O}(k^d)$.
- On split graphs, MDS does not admit a polynomial kernel unless coNP $\subseteq$ NP/poly.

Dominating	By	Problem	PC	Poly Kernel
Vertices	Vertices	DOMINATING SET	W[2]-hard	No
Vertices	Edges	EDGE COVER	P	$\mathcal{O}(1)$
Edges	Edges	EDGE DOMINATING SET	FPT	Yes
Edges	Vertices	VERTEX COVER	FPT	Yes
Edges+Vertices	Vertices	VERTEX COVER	FPT	Yes
Edges+Vertices	Edges	EDGE COVER	P	$\mathcal{O}(1)$
Edges+Vertices	Edges+Vertices	MIXED DOMINATING SET	FPT	No

Table 4.1: Different domination problems and their FPT and kernelization status.

In addition, we show that on general graphs, MDS admits an exact algorithm with running time $2^n n^{\mathcal{O}(1)}$.

4.1 Introduction

Dominating (or covering) objects such as vertices and edges in a graph by vertices or edges give rise to several classic problems, such as VERTEX COVER, EDGE COVER, DOMINATING SET and EDGE DOMINATING SET (see Table 4.1). All these problems and their numerous variants have been studied extensively from structural as well as algorithmic points of view. However, all these problems except EDGE COVER are known to be NP-complete [43, 89], and thus, they have been subjected to intense scrutiny in all the algorithmic paradigms meant for coping with NP-hardness, including approximation algorithms and parameterized complexity. In this paper we consider a well-studied variant of these problems, where the objective is to dominate *vertices and edges* by *vertices and edges*.

In order to define the problems formally, we first define the notion of domination, that is, what a vertex or an edge can dominate. A *vertex dominates* itself, all its neighbors and all the edges incident with it. On the other hand, an *edge dominates* its two endpoints, and all the edges incident with either of its endpoints. We first define the problem of dominating vertices by vertices. A *dominating set* of a graph G is a set $S \subseteq V(G)$ such that every vertex $v \in V(G) \setminus S$ is adjacent to at least one vertex in S. In the DOMINATING

SET problem, we are given an input graph G, a positive integer k, and the objective is to check whether there exists a dominating set of size at most k. The edge counterpart of DOMINATING SET is called EDGE DOMINATING SET. The problem we study in this paper is a variant of these domination problems. Towards that we first define the notion of *mixed dominating set (mds)*. Given a graph G, and a set $X \subseteq V(G) \cup E(G)$, X is called a *mds* if every element $x \in (V(G) \cup E(G)) \setminus X$ is either adjacent to or incident with an element of X. More formally, we study the following problem in the parameterized complexity framework.

MIXED DOMINATING SET (MDS) **Parameter:** k or $\mathsf{tw}(G)$
Input: A graph G on n vertices and m edges and a positive integer k.
Question: Does there exist a MDS of size at most k in G?

Previous work: The notion of MDS (also called total cover) was introduced in the 70s by Alavi et al. [3] as a generalization of matching and covering, and after that it has been studied extensively in graph theory [4, 35, 79, 83]. See the chapter in [51] for a survey on MDS. The algorithmic complexity of MDS was first considered by Majumdar [76], where he showed that the problem is NP-complete on general graphs and admits a linear-time algorithm on trees. Hedetniemi et al. [52] and Manlove [78] showed that MDS remains NP-complete on bipartite and chordal graphs and on planar bipartite graphs of maximum degree 4, respectively. A decade and half later, Zhao et al. [90] considered MDS and showed that it remains NP-complete on split graphs. Unaware of the older result, they also designed an $\mathcal{O}(n \log n)$ time algorithm on trees. Lan and Chang [66] extended this result and gave a linear time algorithm for MDS on cacti (an undirected graph where any two cycles have at most one vertex in common). Hatami [50] gave a factor 2 approximation algorithm for MDS on general graphs. Recently, Rajaati et al. [84] studied MDS parameterized by the treewidth of the input graph and designed an algorithm with running time $\mathcal{O}^\star(3^{\mathsf{tw}(G)^2})$[*].

We obtain MDS parameterized by solution size (k) and treewidth ($tw(G)$) of the graphs.

[*] $\mathcal{O}^\star$ notation suppresses the polynomial factor. That is, $\mathcal{O}(f(k)n^{\mathcal{O}(1)}) = \mathcal{O}^\star(f(k))$.

We also consider the problem on various graph classes and obtain the following results.

Our results:

1. MDS admits an algorithm with running time $\mathcal{O}^\star(7.465^k)$. We complement the FPT result by observing that (a) MDS does not admit an algorithm with running time $2^{o(k)}n^{\mathcal{O}(1)}$ unless ETH [54] fails; and (b) it does not admit a polynomial kernel unless $\mathsf{coNP} \subseteq \mathsf{NP/poly}$. See the last row of Table 4.1.

2. We design an algorithm with running time $\mathcal{O}^\star(6^{\mathsf{tw}(G)})$ for MDS parameterized by $\mathsf{tw}(G)$. This algorithm is a significant improvement over the $\mathcal{O}^\star(3^{\mathsf{tw}(G)^2})$ algorithm of Rajaati et al. [84]. We also show that it does not admit an algorithm with running time $\mathcal{O}^\star((2-\varepsilon)^{\mathsf{tw}}(G))$, for any $\varepsilon > 0$, unless SeCoCo fails [30].

3. On proper interval graphs, MDS is polynomial time solvable. We utilize the property of proper interval graphs' admitting a *clique-partition*—a partition of the vertex set into an ordered sequence such that the graph induced by each part is a clique, and each edge of the graph is contained within one clique or between consecutive cliques. Then we do dynamic programming over the clique-partition to show polynomial time solvability of MDS.

4. On graphs that do not contain $K_{d,d}$ as a subgraph (biclique-free graphs), MDS admits a kernel of size $\mathcal{O}(k^d)$. Biclique-free graphs contain well known sparse graph classes such as graphs of bounded expansion and nowhere dense graphs. Our kernel relies on a crucial relationship between vertex cover and mixed dominating set, namely, if a graph G has an mds of size at most k, then G has a vertex cover of size at most $2k$.

5. On split graphs, MDS does not admit a polynomial kernel unless $\mathsf{coNP} \subseteq \mathsf{NP/Poly}$. The proof is by a reduction from the RED-BLUE DOMINATING SET problem, parameterized by the number of red vertices.

6. We use the standard branching technique to design an exact algorithm with running time $2^n n^{\mathcal{O}(1)}$ for MDS on general graphs.

For references to algorithms and hardness mentioned in Table 4.1, we refer to [26].

4.2 Preliminaries

For a graph G and $R \subseteq V(G)$, we use $E(R)$ to denote the set of edges incident with at least one vertex in R. The following observation follows directly from the definition of a mixed dominating set, and we shall use it throughout the paper.

Observation 85. *Let G be a graph and $V' \cup E'$ be an mds of size k, where $V' \subseteq V(G)$ and $E' \subseteq E(G)$. Let $E_1 \subseteq E'$. Then,*

(i) $V' \cup V(E')$ is a vertex cover of G of size at most $2k$,

(ii) any vertex v of degree at least $2k+1$ belongs to $V' \cup V(E')$,

(iii) $V' \cup (E' \setminus E_1) \cup E_2$ is an mds of G for any $E_2 \subseteq E(G)$ with $V(E_2) = V(E_1)$,

(iv) if G is a complete graph, then $V' \cup V(E')$ must contain all but one vertices of G.

Now we define a *special mixed dominating set* and prove that there is in fact an optimum mds which is also a special mds.

Definition 86. *A mixed dominating set $V' \cup E'$ of a graph G, where $V' \subseteq V(G)$ and $E' \subseteq E(G)$ is said to be a* special mixed dominating set *if E' is a matching and $V' \cap V(E') = \emptyset$.*

Observation 87. *A graph G has an mds of size k if and only if G has a special mds of size k.*

Proof. Among all the k-sized mixed dominating sets of G, let S be an mds such that $|E(G) \cap S|$ is minimum. We claim that S is a special mds. If S contains two edges uv and uw, then $(S \setminus \{uv\}) \cup \{v\}$ is also an mds of size k with less number of edges than that in S. If S contains a vertex v and an edge uv, then $(S \setminus \{uv\}) \cup \{u\}$ is also an mds of size k with less number of edges than that in S. In either case, we have a contradiction to the assumption. $\square$

4.3 Algorithm for MDS parameterized by the solution size

In this section we design an algorithm for MDS parameterized by the solution size. We start with a simple observation that vertices and endpoints of edges in an mds form a vertex cover.

Lemma 88. *Let G be a graph and $S = V' \cup E'$ be an mds of G. Then $V' \cup V(E')$ is a vertex cover of G, of cardinality at most $2|S|$.*

Proof. Since $S = V' \cup E'$ is an mds of G, where $V' \subseteq V(G)$ and $E' \subseteq E(G)$, every edge in G has at least one of its endpoints in $V' \cup V(E')$. This implies that $V' \cup V(E')$ is a vertex cover of G, of cardinality at most $2|S|$. □

In order to get a handle on an optimal solution we define what we call a *nice mds*.

> *Among all* ***minimum sized*** *mixed dominating sets of G, pick the one with* ***the least number of vertices****. Such an mds is called a* ***nice mds****. To re-emphasize, a nice mds by definition is minimum sized.*

We now prove the following lemma, which forms the crux of our algorithm.

Lemma 89. *Let G be a connected graph and $V' \cup E'$ be a nice mds of G. Then, there is a minimal vertex cover C of G such that $V' \subseteq C \subseteq V' \cup V(E')$.*

Proof. Let $S = V' \cup E'$. Since S is mds, by Lemma 88, $V' \cup V(E')$ is a vertex cover of size at most $2|S|$. Any edge incident on $v \in V'$ dominates v as well as all the edges incident on v. Therefore, if v is such that $S \setminus \{v\}$ dominates $N_G(v)$, then by replacing v in S with some edge incident on v (this is possible since G is connected), we get another minimum sized mds with fewer vertices. This implies every vertex in V' must dominate at least one vertex (other than itself) which no other element in $V' \cup E'$ dominates. More specifically, for every

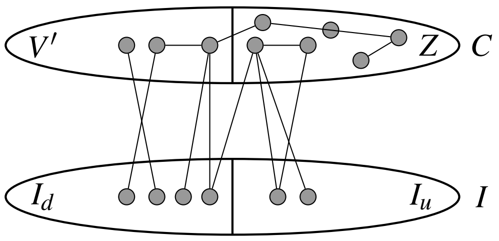


Fig 4.1: Partition of $V(G)$ into minimal vertex cover C and independent set I, where $C = V' \uplus Z$ and $I = I_d \uplus I_u$.

$v \in V'$, there is a vertex $v' \in V(G)$ such that $vv' \in E(G)$ and $v' \notin N_G[(V' \setminus \{v\})] \cup V(E')$. This means, every minimal vertex cover contained in $V' \cup V(E')$ must contain V', because if $C \subseteq V' \cup V(E')$ does not contain $v \in V'$, then edge vv' is not covered by C. Therefore, if the vertex cover $V' \cup V(E')$ is not minimal, keep removing vertices from $V(E') \setminus V'$ until we are left with a minimal vertex cover. □

Let $V' \cup E'$ be a *nice* mds and C be a minimal vertex cover such that $V' \subseteq C \subseteq V' \cup V(E')$. Let $I = V(G) \setminus C$. Note that I is an independent set and it can partitioned into two sets I_d and I_u, where I_d is the set of vertices dominated by V' and $I_u = I \setminus I_d$. That is, $I_d = N_G(V') \cap I$, and $I_u = I \setminus I_d$. Also, let $Z = C \setminus V'$. We thus have a partition of $V(G)$ into V', Z, I_d and I_u. We call the quadruple (V', Z, I_d, I_u) a *nice partition* of $V(G)$ with respect to the mds $V' \cup E'$ and the minimal vertex cover C (see Fig. 4.1). Also, we refer to the graph $G' = G[Z \cup I_u]$ as the *companion graph* of G with respect to V' and C.

Now let us define a new kind of domination called *special domination*. We say a vertex *special dominates* only itself, and an edge *special dominates* its endpoints as well as all the edges incident to at least one of its endpoints. Consequently, we can define a *special dominating set (sds)* as follows. An *sds* of a graph G' is a set $Q' \subseteq V(G') \cup E(G')$ such that every element $x \in (V(G') \cup E(G')) \setminus Q'$ is either adjacent to or incident on an *edge* in Q'. The next lemma shows the relation between mds and sds.

Lemma 90. *Let $V' \cup E'$ be a nice mds of G and C be a minimal vertex cover of G such that $V' \subseteq C \subseteq V' \cup V(E')$. Let (V', Z, I_d, I_u) be a nice partition of $V(G)$ with respect to $V' \cup E'$*

and C. Then G has an mds of size at most k if and only if $G' = G[Z \cup I_u]$ has an sds of size at most $k - |V'|$.

Proof. Assume G has an mds of size at most k. Since $V' \cup E'$ is a nice mds, $|V' \cup E'| \leq k$. We can construct an sds Q' for G' as follows. If an edge $e \in E'$ has both its endpoints in $V(G')$, add e to Q'. If an edge $e \in E'$ has exactly one endpoint in $V(G')$, then add that endpoint to Q'.

We now claim that Q' is indeed an sds for G'. Since E' dominates every vertex in $V(E') \supseteq Z \cup I_u = V(G')$, Q' special dominates all vertices of G'. If $e = uv$ is an edge of G' such that there exists an edge $uw \in E'$ (or $vw \in E'$) for some $w \in V(G')$, then $uw \in Q'$ (or $vw \in Q'$) and hence Q' special dominates e.

We claim that Q' special dominates all the edges in G'. By way of contradiction, suppose $e = uv$ is an edge of G' such that there is no edge uw' or vw' in E' for any $w' \in V(G')$. Note that this also means $uv \notin E'$. But $u, v \in V(E') \supseteq Z \cup I_u$. In that case, there must exist $xu, yv \in E'$, where $x, y \in V' \cup I_d$. Then we claim that $S = V' \cup ((E' \setminus \{xu, yv\}) \cup \{uv\})$ is an mds of G. Notice that $\{xu, yv\}$ dominate the set of vertices $R = \{x, u, y, v\}$ and all the edges $E(R)$ incident to at least one vertex in $R = \{x, u, y, v\}$. Since $V' \cup E'$ is an mds of G, to prove S is an mds of G, it is enough to show that S dominates R and $E(R)$. Since $S \supseteq V' \cup \{uv\}$ and $x, y \in I_d$, we have that $x, y \in N_G[V']$. This implies that S dominates R. Now, what is left to prove is, S dominates $E(R)$. Since $uv \in S$, all the edges incident to at least one of u or v is dominated by S. Finally, we show that S dominates all the edges incident to at least one of x or y. Let e be an edge incident on $z \in \{x, y\}$. If $z \in V'$, then S dominates e, because $z \in V' \subseteq S$. Otherwise $z \in I_d$, because $z \in \{x, y\} \subseteq V' \cup I_d$. Let $e = zw$. Since $z \in I_d$ and $V' \cup Z$ is a vertex cover of G, we have that $w \in V' \cup Z$. If $w \in V'$, then S dominates $e = zw$, because $w \in V' \subseteq S$. If $w \in \{u, v\}$, then S dominates $e = zw$, because $uv \in S$. Otherwise $w \in Z \setminus \{u, v\} \subseteq V(E') \setminus \{u, v\}$. Since $w \in V(E') \setminus \{u, v\}$, there is an edge in $E' \setminus \{xu, yv\} \subseteq S$. This implies that S dominates e. Thus we have shown that S is an mds of cardinality strictly less than that of $V' \cup E'$, a contradiction. Hence we conclude that Q' is an sds of G'.

To prove the other direction, suppose G' has an sds Q' of size atmost $k-|V'|$. We claim that $V' \cup Q'$ is an mds of G. Note that Q' dominates all vertices and edges in graph G' as well as all edges incident on $Z \cup I_u$, and V' dominates all vertices in $V' \cup I_d$ as well as all edges incident on V'. Therefore, $V' \cup Q'$ is an mds of G of cardinality $|V'|+|Q'| \leq k$. □

Lemma 90 shows that given a graph G, V' and C as defined above, the problem of deciding whether G has an mds of size at most k boils down to deciding whether G' has an sds of size at most $k-|V'|$. This results in solving the following problem.

SPECIAL DOMINATING SET (SDS)

Input: An undirected graph G and a positive integer ℓ.

Question: Does there exist an sds of size at most ℓ in G?

In what follows we first design a polynomial time algorithm for SDS. Towards this, note that an edge has more "special dominating power" than a vertex has, in the sense that an edge special dominates itself, its endpoints and its adjacent edges, whereas a vertex special dominates *only* itself. Therefore, a natural strategy is to first try to special dominate as many vertices and all edges with as *few edges* as possible, and then add to the solution all those vertices that are not special dominated by any of the edges. This intuition leads to following polynomial time algorithm for SDS.

```
ALGORITHM-SDS $(G, \ell)$

Step 1. Find a maximum matching, say $M$, in $G$. Let $U = V(G) \setminus V(M)$.

Step 2. If $|M \cup U| \leq \ell$, return Yes; else return No.
```

The only time consuming step in the above algorithm is Step 1 – finding a maximum matching – and this can be done in time $\mathcal{O}(m\sqrt{n})$ [80]. Thus, ALGORITHM-SDS runs in polynomial time, and the following lemma shows the correctness of the algorithm.

Lemma 91. *Let M be a maximum matching in a graph G and let $U = V(G) \setminus V(M)$. Then*

$M \cup U$ is a minimum sized sds of G.

Proof. Since M is a maximum matching, every edge $e \in E \setminus M$ is incident to an edge in M, and thus M special dominates all edges in G. The set M also special dominates all vertices in $V(M)$, and the rest of the vertices in G are special dominated by U. Therefore, $M \cup U$ is indeed an sds of G.

Now we claim that $M \cup U$ is a minimum size sds of G. Since $V(M) \cap U = \emptyset$ and $V(G) = V(M) \cup U$, we have that $|V(G)| = 2|M| + |U|$. Towards proving the minimality of $M \cup U$, we show that any sds $E_1 \cup V_1$ of G, where $E_1 \subseteq E(G)$ and $V_1 \subseteq V(G)$, has cardinality at least $|M \cup U| = |M| + |U|$. Let M_1 be a maximum (w.r.t. E_1) matching contained in E_1. The total number of vertices special dominated by E_1 is at most $2|M_1| + |E_1 \setminus M_1| \leq |M_1| + |E_1|$. Since $E_1 \cup V_1$ is an sds of G, we have

$$\begin{aligned}
|M_1| + |E_1| + |V_1| &\geq |V(G)| \\
|E_1| + |V_1| &\geq |V(G)| - |M_1| \\
&\geq 2|M| + |U| - |M_1| \qquad \text{(because } |V(G)| = 2|M| + |U|\text{)} \\
&\geq |M| + |U|. \qquad \text{(because } |M| \geq |M_1|\text{)}
\end{aligned}$$

This completes the proof of the lemma. □

ALGORITHM-SDS together with Lemma 91 results in the following result.

Lemma 92. *SDS can be solved in time $\mathcal{O}(m\sqrt{n})$.*

We are now fully equipped to give our algorithm for MDS.

```
ALGORITHM-MDS $(G,k)$

Step 1. Enumerate all minimal vertex covers of $G$ of size at most $2k$. Let $\mathsf{C}C$ be the
        collection of such vertex covers.

Step 2. For each $C \in \mathsf{C}C$ and each $V' \subseteq C$ such that $|V'| \le k$ and $|C| \le 2k - |V'|$, use
        ALGORITHM-SDS to decide if the companion graph $G'$ (w.r.t. $C$ and $V'$) has an
        sds of size at most $k - |V'|$; if it has, return Yes.

Step 3. Otherwise return No.
```

The correctness of the algorithm follows from Lemma 90. Now, let us analyze the running time of ALGORITHM-MDS. Any graph has at most 2^{2k} minimal vertex covers of size at most $2k$. Furthermore, given G and k, all minimal vertex covers of size at most $2k$ can be enumerated in time $2^{2k}n^{\mathcal{O}(1)}$ [38]. This means, Step 1 can be executed in time $2^{2k}n^{\mathcal{O}(1)}$.

For each $C \in \mathsf{C}C$, there are at most $2^{|C|} \le 2^{2k}$ choices for V'. For each such choice of C and V', by Lemma 92, a minimum sds in G' can be found in polynomial time. Therefore, the running time of ALGORITHM-MDS (G,k) can be bounded by $2^{2k} \cdot 2^{2k} \cdot n^{\mathcal{O}(1)} = \mathcal{O}^\star(16^k)$. This, however, is a liberal estimate. A finer analysis shows that the running time can be brought down to $\mathcal{O}^\star(7.465^k)$.

Lemma 93. ALGORITHM-MDS *runs in time* $\mathcal{O}^\star(7.465^k)$.

Proof. If (G,k) is a yes-instance of MDS with a nice mds $V' \cup E'$, where $|V'| = j$, then any minimal vertex cover C such that $V' \subseteq C \subseteq V(E')$ can have size at most $|V'| + 2|E'| \le j + 2(k-j) = 2k - j$. Therefore, in Step 2, we only process those pairs (C, V') such that $|C| \le 2k - j$, where $|V'| = j$, and there are only 2^{2k-j} such C. Thus Step 2 takes time

$$\sum_{j=0}^{k} 2^{2k-j}\binom{2k-j}{j} = 2^{2k}\sum_{j=0}^{k} 2^{-j}\binom{2k-j}{j}.$$

Since for any $x > 0$, $\binom{n}{i}x^i \le \sum_{i'=0}^{n}\binom{n}{i'}x^{i'} = (1+x)^n$, we get $\binom{n}{i} \le (1+x)^n/x^i$. Using this inequality, for any $x > 0$,

$$2^{-j}\binom{2k-j}{j} \leq \frac{(1+x)^{2k-j}}{(2x)^j} = \frac{(1+x)^{2k}}{((1+x)\cdot 2x)^j}.$$

We choose $x = \frac{(\sqrt{3}-1)}{2}$ so that $(1+x)\cdot 2x = 1$. This gives $\frac{(1+x)^{2k}}{((1+x)2x)^j} \leq (1.3661)^{2k}$. Hence, Step 2 can be executed in time $2^{2k}\cdot 1.3661^{2k}\cdot n^{\mathcal{O}(1)} \leq (7.465)^k \cdot n^{\mathcal{O}(1)}$. □

Thus, we get the following theorem.

Theorem 13. MDS *parameterized by k can be solved in time* $\mathcal{O}^\star(7.465^k)$.

4.4 Algorithm for MDS parameterized by Treewidth

In this section we devise an algorithm which, given a graph G and a nice tree decomposition of G of width $\text{tw}(G)$, computes the size of a minimum mixed dominating set of G. First we give the notion of nice tree decomposition.

A tree decomposition $(\mathbb{T},\mathscr{X})$ is called a *nice tree decomposition* if $\mathbb{T}$ is a tree rooted at some node r where $X_r = \emptyset$, each node of $\mathbb{T}$ has at most two children, and each node is of one of the following kinds:

1. **Introduce vertex node**: a node t that has only one child t' where $X_t \supset X_{t'}$ and $|X_t| = |X_{t'}|+1$.
2. **Forget node**: a node t that has only one child t' where $X_t \subset X_{t'}$ and $|X_t| = |X_{t'}|-1$.
3. **Introduce edge node**: a node t, labeled with an edge $uv \in E(G)$ such that $u, v \in X_t$, and with exactly one child t' such that $X_t = X_{t'}$. We say that edge uv is introduced at t. We additionally require that every edge of $E(G)$ is introduced exactly once in the whole decomposition
4. **Join node**: a node t with two children t_1 and t_2 such that $X_t = X_{t_1} = X_{t_2}$.

5. **Leaf node**: a node $t \neq r$ such that t is a leaf of $\mathbb{T}$, and $X_t = \emptyset$.

It is well known that any tree decomposition of G can be transformed into a nice tree decomposition maintaining the same width in linear time [61, 27]. Corresponding to each node t in $\mathbb{T}$, let V_t be the union of all the bags present in the subtree of $\mathbb{T}$ rooted at t, including X_t and let G_t be the subgraph of G defined as $G_t = (V_t, E_t = \{e \in E(G) \,:\, e \text{ is introduced in the sub tree rooted at t}\})$.

Hence, we assume that input to our algorithm is a graph G and a nice tree decomposition of G of width $\mathsf{tw}(G)$. We start with following two lemmata about special properties of some solution.

Lemma 94. *Every graph has a minimum mixed dominating set such that the edges in that set form a matching.*

Proof. Let S be a minimum mixed dominating set of G, which contains edges $e = uv$ and $e' = vw$ that share a common endpoint v, otherwise the edges in S form a matching. Let $S' = (S \setminus \{e'\}) \cup \{w\}$. Notice that $|S'| = |S|$. We claim that S' is also a mixed dominating set. Notice that the only vertices dominated by edge $e' \in S$ are v and w, and the only edges dominated by e' are those incident on v or w. But, both v and w, and edges incident on them are dominated by S' as well. We can apply this procedure exhaustively to get a minimum mixed dominating set in which the edges form a matching. $\square$

Lemma 95. *Let G be a graph. There is a minimum mixed dominating set S for G such that (i) the edges in S form a matching, and (ii) the set of endpoints of the edges in S is disjoint from the set of vertices in S.*

Proof. Suppose the statement in the lemma is not true. Among the minimum mixed dominating sets with edges in it forms a matching, choose a set $D = V' \cup E'$ such that $|V' \cap V(E')|$ is minimized, where $V' \subseteq V(G)$ and $E' \subseteq E(G)$. Since, by our assumption $|V' \cap V(E')| \geq 1$, there is an edge $uv \in E'$ and $u \in V'$. Let $D' = (D \setminus \{uv\}) \cup \{v\}$. Notice that

$|D'| \leq |D|$ and $D' = U \cup F$, where $U = V' \cup \{v\}$ and $F = E' \setminus \{uv\}$. We claim that D' is a mixed dominating set with edges in it form a matching. Since E' is a matching and $F \subseteq E'$, we have that F is a matching. The edge uv dominates vertices u and v, and edges incident on them. These elements are also dominated by the set $D' = (D \setminus \{uv\}) \cup \{v\}$, because $u, v \in D'$. This implies that D' is a mixed dominating set. Also, since E' is a matching, we have that F is a matching and $U \cap V(F)$ is a strict subset of $V' \cap V(E')$, which is a contradiction to the choice of the set D. $\square$

We start with an intuition about our algorithm. Let G be the input graph and $(\mathbb{T}, \mathscr{X} = \{X_t\}_{t \in V(\mathbb{T})})$ be the given tree decomposition of G. Any standard dynamic programming over tree decomposition has three ingredients: for any node $t \in \mathbb{T}$ (i) defining partial solution, (ii) defining equivalence classes among partial solutions (or in other words defining states of DP table according to partial solutions) and (iii) computing a 'best partial solution' for each state from previously computed values. Normally, for any node $t \in \mathbb{T}$, partial solutions are defined according to the properties of the intersection of solutions with the graph G_t. In our case, a partial solution will be a subset of $V(G_t) \cup E(G_t)$. When we define equivalence classes of these partial solutions, one of the factors in consideration is how do these partial solutions intersect with the bag X_t. Since partial solutions contain edges, these choices naively turns to at least $2^{O(\mathrm{tw}^2)}$, and we can bring it down to $\mathrm{tw}^{O(\mathrm{tw})}$ using Lemma 94.

Instead, we prove that it has an equivalent characterization in terms of pairs of vertices which allows us to bound the number equivalence classes. By Lemma 95, we know that there is a minimum mixed dominating set $S = V' \cup E'$, where $V' \subseteq V(G)$ and $E' \subseteq E(G)$, with the following properties:

(a) E' is a matching, and

(b) $V' \cap V(E') = \emptyset$.

Thus, in this subsection, we use the term *solution* to represent mixed dominating set satisfying conditions (a) and (b).

Let $V' \cup E'$ be a solution. Consider the pair $(V', V(E'))$. Since $V' \cup E'$ is a solution we have that (i) $(V', V(E'))$ is a vertex cover of G, (ii) $N[V'] \cup V(E') = V(G)$, and (iii) $G[V(E')]$ has a perfect matching. In fact, any pair of vertex subsets which satisfies these three properties, can be turned to a mixed dominating set. That is these two notions are *equivalent* as formalized in the following lemma.

Lemma 96. *Let G be a graph. G has a mixed dominating set of size at most k if and only if there is a pair of vertex subsets (X, Y) such that $G[Y]$ has a perfect matching, $V(G) = N_G[X] \cup Y$, $X \cup Y$ is a vertex cover of G and $|X| + \frac{1}{2}|Y| \leq k$.*

Proof. If G has a mixed dominating set of size at most k, then by Lemma 95 we know that G has a mixed dominating set S=$V' \cup E'$ such that E' is a matching and $V' \cap V(E') = \emptyset$. Let $X = V'$ and $Y = V(E')$. By construction $G[Y]$ has a perfect matching E'. Since $S = V' \cup E'$ is a mixed dominating set of G, every vertex and edge in G is adjacent or incident to an element in S. That is, $V(G) = N_G[X] \cup Y$ and $X \cup Y$ is a vertex cover of G. Notice that $|X| + \frac{1}{2}|Y| = |V'| + |E'| \leq k$.

To prove the converse, suppose there is a pair of vertex subsets (X, Y) such that $G[Y]$ has a perfect matching, $V(G) = N_G[X] \cup Y$, $X \cup Y$ is a vertex cover of G and $|X| + \frac{1}{2}|Y| \leq k$. Let $V' = X$ and E' be an arbitrary perfect matching in $G[Y]$. Since $X \cup Y = V' \cup V(E')$ is a vertex cover of G and $V(G) = N_G[X] \cup Y = N_G[V'] \cup V(E')$, we have that $V' \cup E'$ is a mixed dominating set of G. Also notice that $|V'| + |E'| = |X| + \frac{1}{2}|Y| \leq k$. □

As a result of Lemma 96, for any node t in the tree decomposition we define partial solutions and equivalence classes among partial solutions as follows. A *partial solution* is a tuple (X, F, Y) satisfying the following conditions, where $X \subseteq V(G_t)$, $F \subseteq E(G_t)$, $Y \subseteq X_t$:

- $X \uplus Y \uplus V(F)$ is a vertex cover of G_t,
- $V(G_t) \setminus X_t \subseteq N_{G_t}[X] \cup V(F)$.

The intuitive meaning of (X,F,Y) is that there will potentially be a solution S such that $X \cup F \subseteq S$ and for each $u \in Y$, there will be an edge $uv \in S \setminus E(G_t)$.

Now we define equivalence classes of partial solutions corresponding to a node t in the tree decomposition. We define $\mathscr{P}_t[f]$, where $f : X_t \to \{1,2,2',3,3'\}$ as the set of partial solutions (X,F,Y), which satisfy the following.

1. $X_t \cap X = f^{-1}(1)$,
2. $X_t \cap V(F) = f^{-1}(2)$,
3. $Y = f^{-1}(2')$, and
4. $(N_{G_t}(X) \cap X_t) \setminus (Y \cup V(F)) \supseteq f^{-1}(3)$.

Informally, each partial solution imposes a partition of X_t, which is defined by f. The set $f^{-1}(1)$ is the set of vertices from X_t which are part of solution. The set $f^{-1}(2)$ is the set of vertices from X_t such that there are edges in the solution which are incident on vertices in $f^{-1}(2)$ and are present in the graph G_t. The set $f^{-1}(2')$ is the set of vertices from X_t such that there are edges in the solution which are incident on vertices in $f^{-1}(2')$ and these edges are not present in the graph G_t. Here, the condition 4 implies that the set $f^{-1}(3)$ is the set of vertices in X_t, which are not part of solution vertices or end points of solution edges in the partial solution, but they are already dominated. The set $f^{-1}(3')$ is the set of vertices in X_t which are not yet dominated and not in $f^{-1}(2')$. The following lemma proves that it is enough to keep one partial solution in each equivalence class.

Lemma 97. *Let D be a solution and t be a node in the tree decomposition. Let $X = D \cap V(G_t)$, $F = D \cap E(G_t)$ and $Y = \{u \in D \cap V(G_t) \mid \exists v \in V(G)$ such that $uv \in D \setminus E(G_t)\}$. Then $(X,F,Y) \in \mathscr{P}_t[f]$ for some $f : X_t \to \{1,2,2',3,3'\}$. Moreover, for any $(X',F',Y') \in \mathscr{P}_t[f]$, $X' \cup F' \cup (D \setminus (X \cup F))$ is a solution.*

Proof. Let $D = V' \cup E'$, where $V' \subseteq V(G)$ and $E' \subseteq E(G)$. First we show that (X,F,Y) is a partial solution. Since D is a mixed dominating set, for any edge $e \in E(G_t)$ at least one of its endpoints belongs to $(V' \cup V(E')) \cap V(G_t)$. This implies that $X \cup V(F) \cup Y$ is a vertex cover of G_t. Now we show that $V(G_t) \setminus X_t \subseteq N_{G_t}[X] \cup V(F)$. Since D is a mixed dominating set, for any $v \in V(G_t) \setminus X_t$, either $vu \in E'$ for some $u \in V(G)$ or $v \in N_G[V']$. Since X_t is a separator in G, $N_G(v) \subseteq V(G_t)$. This implies that $V(G_t) \setminus X_t \subseteq N_{G_t}[X] \cup V(F)$. Thus, we have proved that (X,F,Y) is a partial solution corresponding to the node t.

Let $(X,F,Y) \in \mathscr{P}_t[f]$, where $f : X_t \to \{1,2,2',3,3'\}$. Let (X',F',Y') be another partial solution in $\mathscr{P}_t[f]$. Notice that, by property 3 of $\mathscr{P}_t[f]$, we have that $Y = Y'$. We need to show that $D' = X' \cup F' \cup (D \setminus (X \cup F))$ is a solution. Let $Z = D \setminus (X \cup F)$. Let $v \in V(G)$ be a vertex.

1. If $v \in V(G_t) \setminus X_t$, then by the definition of partial solution $v \in N_{G_t}[X'] \cup V(F)$ and hence v is dominated by D'.
2. If $v \in f^{-1}(1) \cup f^{-1}(2) \cup f^{-1}(3)$, then it is dominated by $X' \cup F'$, by properties 1,2 and 4 of $\mathscr{P}_t[f]$.
3. If $v \in f^{-1}(2')$, then there an edge $e \in Z$ which is incident to v, because D is a mixed dominating set. Hence v is dominated by $D' \supseteq Z$.
4. If $v \in f^{-1}(3') \cup (V(G) \setminus V(G_t))$, then it is dominated either by Z or by $X \cap X_t = f^{-1}(1) = X' \cap X_t$; so v is dominated by $X' \cup F' \cup Z$.

Hence all vertices of G are dominated by D'. Let $e \in E(G)$ be an edge.

1. Since $X' \cup Y' \cup V(F')$ is a vertex cover of G_t, all edges in $E(G_t)$ are dominated by D'.
2. Suppose $e \in E(G) \setminus E(G_t)$ has an endpoint in $f^{-1}(1) \cup f^{-1}(2)$. Then it is dominated by $X' \cup F'$ by properties 1 and 2. If e has an endpoint in $f^{-1}(2')$, then it is incident on an edge $e' \in Z$ and hence dominated by $D' \supseteq Z$. Otherwise edge e has both its endpoints in in $f^{-1}(3) \cup f^{-1}(3') \cup (V(G) \setminus V(G_t))$. Notice that e could not be

dominated by $X \cup F$. Also, since e is dominated by the mixed dominating set D, e is dominated by Z. This implies that e is dominated by D'.

Thus $D' = X' \cup F' \cup Z$ dominates all vertices and edges of G and hence is a mixed dominating set. This completes the proof of the lemma. $\square$

Now we are ready to explain our dynamic programming (DP) algorithm A. Due to Lemma 97 it is enough to store one partial solution (X,F,Y) for each equivalence class $\mathscr{P}_t[f]$ which minimizes the quantity $|X|+|F|$. In our DP table, for every node t in the tree decomposition and for every $f : X_t \to \{1,2,2',3,3'\}$, we have an entry $c[t,f]$ which stores the the value $\min\{|X|+|F| \mid (X,F,Y) \in \mathscr{P}_t[f]\}$.

Before moving to the computation of the DP table entries, we define some notations. For a function $f : A \to B$ from A to B and elements $\{a_1,a_2,b_1,b_2\}$, we use $f_{(a_1,a_2)\to(b_1,b_2)} : A \cup \{a_1,a_2\} \to B \cup \{b_1,b_2\}$ to denote the following function.

$$f_{(a_1,a_2)\to(b_1,b_2)}(x) = \begin{cases} f(x) & \text{if } x \notin \{a_1,a_2\} \\ b_1 & \text{if } x = a_1 \\ b_2 & \text{if } x = a_2 \end{cases}$$

Note that a_1 and a_2 may or may not be in A, similarly, b_1 and b_2 may or may not be in B. For two elements a and b, and a function f, the function $f_{a\to b} : A \cup \{a\} \to B \cup \{b\}$ is defined analogously. For $A' \subseteq A$, we denote by $f|_{A'}$, the restriction of f to A'.

Computation.

Now move to the computation of the DP table entries $c[,]$. The DP table entries are computed in a bottom-up fashion of the given tree decomposition $(\mathbb{T}, \mathscr{X} = \{X_t\}_{t\in V(\mathbb{T})})$ of the input graph G. Let $t \in V(\mathbb{T})$ and $f : X_t \to \{1,2,2',3,3'\}$ be a function. We have different cases based on the kind of node t is.

- **Leaf node:** When t is a leaf node, $X_t = \emptyset$, and hence there is only one function – the empty function – on X_t. Here, we have $c[t,\emptyset] = 0$.

- **Introduce vertex node:** Let t be an introduce vertex node with child t' such that $X_t = X_{t'} \cup \{v\}$ for some $v \notin X_{t'}$. Since v is an isolated vertex in G_t, v alone can dominate itself. But it might also be dominated in future either as some edge's endpoint in the solution or by some vertex. We thus have the following formula:

$$c[t,f] = \begin{cases} 1 + c[t', f|_{X_{t'}}] & \text{if } f(v) = 1 \\ c[t', f|_{X_{t'}}] & \text{if } f(v) \in \{2', 3'\} \\ \infty & \text{if } f(v) \in \{2, 3\} \end{cases}$$

- **Introduce edge node:** Let t be an introduce edge node labeled with edge $e = uv$ and let t' be the child of t. Now graph G_t is exactly the same as graph $G_{t'}$ except for the additional edge e. If exactly one of the endpoints of e is in $f^{-1}(1)$, then e can be used to dominate the other endpoint. In the case where e is in the partial solution, we have $f(u) = f(v) = 2$, so that while taking the precomputed solution for t', we must have u and v in $f^{-1}(2')$. In addition, we also have to take care of the fact that we have added a new edge and it needs to be covered by $f^{-1}(1) \cup f^{-1}(2) \cup f^{-1}(2')$. These observations lead to the following formula.

$$c[t,f] = \begin{cases} \min\left\{1 + c[t', f_{(u,v)\to(2',2')}], c[t'f]\right\} & \text{if } f(u) = f(v) = 2 \\ \min\{c[t',f], c[t', f_{v\to 3'}]\} & \text{if } (f(u), f(v)) = (1,3) \\ \min\{c[t',f], c[t', f_{u\to 3'}]\} & \text{if } (f(u), f(v)) = (3,1) \\ \infty & \text{if } \{f(u), f(v)\} \cap \{1, 2, 2'\} = \emptyset \\ \infty & \text{if } \{f(u), f(v)\} = \{1, 3'\} \\ c[t',f] & \text{otherwise.} \end{cases}$$

- **Forget node:** Let t be a forget node with child t' such that $X_t = X_t' \setminus \{v\}$ for some $v \in X_t'$. When we are forgetting the vertex v we need to make sure that v is dominated in graph G_t, since it cannot be dominated in future. Hence, in t', v must be in $f^{-1}(1) \cup f^{-1}(2) f^{-1}(3)$ and thus we have the following formula.

$$c[t,f] = \min\{c[t', f_{v\to 1}], c[t', f_{v\to 2}], c[t', f_{v\to 3}]\}.$$

- **Join node:** Let t be a join node with children t_1 and t_2. Then $X_t = X_{t_1} = X_{t_2}$. We say that functions $f_1 : X_{t_1} \to \{1,2,2',3,3'\}$ and $f_2 : X_{t_2} \to \{1,2,2',3,3'\}$ are *consistent* with f if the following conditions hold.

  1. $f^{-1}(1) = f_1^{-1}(1) = f_2^{-1}(1)$
  2. $f^{-1}(2) = f_1^{-1}(2) \uplus f_2^{-1}(2)$
  3. $f^{-1}(2') = f_1^{-1}(2') \cap f_2^{-1}(2')$
  4. $f^{-1}(3) = f_1^{-1}(3) \cup f_2^{-1}(3)$
  5. $f^{-1}(3') = f_1^{-1}(3') \cap f_2^{-1}(3')$.

  Now we define $c[t,f]$ as follows.

$$c[t,f] = \min_{f_1,f_2}\{c[t_1,f_1] + c[t_2,f_2] - |f^{-1}(1)|\},$$

  where the minimum is over all possible functions f_1 on X_{t_1} and f_2 on X_{t_2} that are consistent with f.

This completes the description of the computation of DP table entries. Algorithm A will output $c[r,\emptyset]$. The correctness of the algorithm A follows from the following lemma.

Lemma 98. *For any $t \in V(\mathbb{T})$ and $f : X_t \to \{1,2,2',3,3'\}$, $c[t,f] = \min\{|X|+|F| \mid (X,F,Y) \in \mathscr{P}_t[f]\}$.*

Proof sketch. We prove the lemma using induction on the height of the tree rooted at t. The base case is when height of the tree rooted at t is 0. In this case t is a leaf node and we have that $\mathscr{P}_t[\emptyset]$ contains only $\emptyset$. Thus $c[t,\emptyset] = 0$. Now let the height of the tree rooted at t be $i > 0$. Here, we consider only the case when t is a join node with child nodes t_1 and t_2. Other cases can be proved using similar arguments.

First, we prove that $c[t,f] \leq \min\{|X|+|F| \mid (X,F,Y) \in \mathscr{P}_t[f]\}$. Let $(X,F,Y) \in \mathsf{C}P_t[f]$ be such that $|X|+|F| = \min\{|X'|+|F'| \mid (X',F',Y') \in \mathscr{P}_t[f]\}$. For $i \in \{1,2\}$, let $X_i = X \cap V(G_{t_i})$, $F_i = F \cap E(G_{t_i})$, and $Y_i = (X_t \cap (X \cup V(F) \cup Y)) \setminus (X_i \cup V(F_i))$. Since each edge in F is either in G_{t_1} or in G_{t_2}, but not in both the graphs G_{t_1} and G_{t_2}, we have that $F = F_1 \uplus F_2$. Let f_i, $i \in \{1,2\}$, be a function such that $(X_i,F_i,Y_i) \in \mathsf{C}P_t[f_i]$. Now we prove that f_1 and f_2 are consistent with f. Notice that for $i \in \{1,2\}$, $X_i = X \cap V(G_{t_i}) = X \cap V(G_t)$. This implies that $f^{-1}(1) = f_1^{-1}(1) = f_2^{-1}(1)$. Since $F = F_1 \uplus F_2$, we have that $f^{-1}(2) = f_1^{-1}(2) \uplus f_2^{-1}(2)$. Now we prove that $Y = Y_1 \cap Y_2$.

$$\begin{aligned}
Y = f^{-1}(2') &= (X_t \cap (X \cup V(F) \cup Y)) \setminus (X \cup V(F)) \\
&= (X_t \cap (X \cup V(F) \cup Y)) \setminus (X_1 \cup V(F_1) \cup X_2 \cup V(F_2)) \\
&= \bigcap_{i \in [2]} (X_t \cap (X \cup V(F) \cup Y)) \setminus (X_i \cup V(F_i)) \\
&= Y_1 \cap Y_2.
\end{aligned}$$

Since $Y = Y_1 \cap Y_2$, we have that $f^{-1}(2') = f_1^{-1}(2') \cap f_2^{-1}(2')$. We can argue that $f^{-1}(3) = f_1^{-1}(3) \cup f_2^{-1}(3)$. The reason is $f^{-1}(3)$ is the set of vertices in X_t which are dominated by X in G_t, but not in the set $X \cup Y \cup V(F)$. By our construction each element in $f^{-1}(3)$ is dominated by X either in G_{t_1} or in G_{t_2}. This implies that $f^{-1}(3) = f_1^{-1}(3) \cup f_2^{-1}(3)$. Now we argue that $f^{-1}(3') = f_1^{-1}(3') \cap f_2^{-1}(3')$. The reason is $f^{-1}(3')$ is the set of vertices in X_t which are not dominated by X in G_t and not in the set $Y \cup V(F)$. By our construction each element in $f^{-1}(3')$ will not be dominated by X either in G_{t_1} or in G_{t_2}. This implies that $f^{-1}(3') = f_1^{-1}(3') \cap f_2^{-1}(3')$. Hence, we have proved that f_1 and f_2 are consistent with f. Since, for $i \in \{1,2\}$, $(X_i,F_i,Y_i) \in \mathsf{C}P_t[f_i]$, by induction hypothesis, we have that

$c[t_i,f_i] \leq |X_i|+|F_i|$. Hence,

$$\begin{aligned} c[t,f] &\leq c[t_1,f_1]+c[t_2,f_2]-|f^{-1}(1)| \\ &= |X_1|+|F_1|+|X_2|+|F_2|-|f^{-1}(1)| \\ &= |X_1 \cup X_2|+|F|+|X_1 \cap X_2|-|f^{-1}(1)| \qquad \text{(Because } F_1 \cap F_2 = \emptyset\text{)} \\ &= |X|+|F| \qquad \text{(Because } f^{-1}(1) = X_1 \cap X_2\text{)} \end{aligned}$$

Now, we prove that $c[t,f] \geq \min\{|X|+|F| \mid (X,F,Y) \in \mathscr{P}_t[f]\}$. Let f_1 and f_2 are two functions, consistent with f and $c[t,f] = c[t_1,f_1]+c[t_2,f_2]-|f^{-1}(1)|$. By induction hypothesis, for $i \in \{1,2\}$ there exists $(X_i,F_i,Y_i) \in \mathsf{C}P_{t_i}[f_i]$ such that $c[t_i,f_i] = |X_i|+|F_i|$. Let $Y = f^{-1}(2')$. Since f_1 and f_2 are consistent with f, we have that $Y = f^{-1}(2') = f_1^{-1}(2') \cap f_2^{-1}(2') = Y_1 \cap Y_2$. Consider the tuple $(X = X_1 \cup X_2, F = F_1 \cup F_2, Y)$. Notice that $X_1 \cap X_2 = X_1 \cap X_t = X_2 \cap X_t = f^{-1}(1)$ and $F = F_1 \uplus F_2$. This implies that $c[t,f] = |X_1|+|F_1|+|X_2|+|F_2|-|X_1 \cap X_2| = |X|+|F|$.

Now we claim that $(X,F,Y) \in \mathscr{P}_t[f]$. Towards that first we argue that (X,F,Y) is a partial solution corresponding to node t. Since f_1 and f_2 are consistent with f, we have that $X \cup V(F) \cup Y = X_1 \cup X_2 \cup V(F_1) \cup V(F_2) \cup Y_1 \cup Y_2$. This, along with the fact that $X_i \cup V(F_i) \cup Y_i$ is a vertex cover of G_{t_i} for any $i \in \{1,2\}$, implies that $X \cup V(F) \cup Y$ is a vertex cover of G_t. Since, $(X_i,F_i,Y_i) \in \mathsf{C}P_{t_i}[f_i]$, $V(G_{t_i}) \setminus X_{t_i} \subseteq N_{G_{t_i}}[X_i] \cup V(F_i)$, we have that $V(G_t) \setminus X_t \subseteq N_{G_t}[X] \cup V(F)$. This implies that (X,F,Y) is a partial solution corresponding to node t. Now we show that $(X,F,Y) \in \mathscr{P}_t[f]$. Notice that $f^{-1}(1) = X_1 \cap X_t = X_2 \cap X_t = X \cap X_t$. Since $f^{-1}(2) = f_1^{-1}(2) \uplus f_2^{-1}(2)$ and for any $i \in \{1,2\}$, $V(F_i) \cap X_{t_i} = f_i^{-1}(2)$, we have that $f^{-1}(2) = X_t \cap V(F)$. We have already argued that $Y = f^{-1}(2')$. Since $f^{-1}(3) = f_1^{-1}(3) \cup f_2^{-1}(3)$, each vertex in $f^{-1}(3)$ is either dominated by X_1 or X_2, and thus we have that $(N_{G_t}(X) \cap X_t) \setminus (Y \cup V(F)) = f^{-1}(3)$. Hence we have that $(X,F,Y) \in \mathscr{P}_t[f]$.

Since $(X,F,Y) \in \mathscr{P}_t[f]$ and $c[t,f] = |X|+|F|$, we have that $c[t,f] \geq \min\{|X|+|F| \mid (X, F,Y) \in \mathscr{P}_t[f]\}$. This complete the proof sketch of the lemma. $\square$

Lemma 99. *Algorithm A runs in time $8^{\mathsf{tw}} n^{\mathcal{O}(1)}$.*

Proof. The number of DP table entries is upper bounded by $5^{\mathsf{tw}} n^{\mathcal{O}(1)}$. The time required to process each leaf node, introduce vertex/edge node or forget node is $5^{\mathsf{tw}} \cdot \mathsf{tw}^{O(1)}$. Because for every entry to be computed we only have to check at most 3 many entries from children nodes. However, computing the values of $c[t,\cdot]$ for a join node t is more time consuming. Note that if a pair f_1, f_2 is consistent with f, then for every $v \in X_t$, $(f(v), f_1(v), f_2(v))$ belongs to

$$\{(1,1,1),(2,2,2'),(2,2',2),(2',2',2'),(3,3,3),(3,3,3'),(3,3',3),(3',3',3')\}.$$

This implies that for any $t \in V(\mathbb{T})$, there are exactly $8^{|X_t|}$ many valid triples of functions (f, f_1, f_2) such that f_1 and f_2 are consistent with f, because for every vertex v we have 8 possibilities for $(f(v), f_1(v), f_2(v))$. The computation can be implemented as follows. Initialize $c[t,f] = \infty$ for all functions f on X_t. We iterate through all valid triples and for each such triple (f, f_1, f_2), we replace the current value of $c[t,f]$ with $\{c[t_1, f_1] + c[t_2, f_2] - |f^{-1}(1)|\}$ in case the latter value is smaller. As $|X_t| \leq \mathsf{tw} + 1$, it follows that the algorithm spends $8^{\mathsf{tw}} \cdot \mathsf{tw}^{O(1)}$ time for every join node. Hence, we get the required running time. □

Improving the runtime:

We can improve the running time of algorithm A using fast computation of cover product. The cover product of two functions $f, g : 2^V \to \mathbb{Z} \cup \{\infty\}$ is a function $(f *_c g) : 2^V \to (\mathbb{Z} \cup \{\infty\})$ such that for every $Y \subseteq V$,

$$(f *_c g)(Y) = \min_{A \cup B = Y} (f(A) + g(B)).$$

Theorem 14 ([10]). *For two functions $f, g : 2^V \to \{-M, \ldots, M\}$, where $|V| = n$, given all* *the 2^n values of f and g in the input, all the 2^n values of the cover product of f and g over the integer min-sum semi-ring* $(\mathbb{Z}, \min, +)$ *can be computed in time* $2^n \cdot n^{\mathcal{O}(1)} \cdot \mathcal{O}(M \cdot \log M \cdot \log\log M)$.

We can use Theorem 14 to accelerate run time of our dynamic programming algorithm to $6^{\mathsf{tw}} \cdot n^{\mathcal{O}(1)}$. The processing of introduce nodes and forget nodes is already fast enough. We will show here that for a join node t the values of $c[t,f]$ for all the $5^{|X_t|}$ functions $f : X_t \to \{1,2,2',3,3'\}$ can be computed in time $6^{|X_t|} \cdot |X_t|^{O(1)}$. Recall that

$$c[t,f] = \min\{c[t_1,f_1] + c[t_2,f_2] - |f^{-1}(1)|\},$$

where t_1 and t_2 are the children of t and the minimum is taken over all colorings f_1, f_2 that are consistent with f. Recall that $X_t = X_{t_1} = X_{t_2}$. From the consistency conditions we know that

1. $f^{-1}(1) = f_1^{-1}(1) = f_2^{-1}(1)$,
2. $f^{-1}(2) = f_1^{-1}(2) \uplus f_2^{-1}(2)$,
3. $f^{-1}(2') = f_1^{-1}(2') \cap f_2^{-1}(2')$,
4. $f^{-1}(3) = f_1^{-1}(3) \cup f_2^{-1}(3)$,
5. $f^{-1}(3') = f_1^{-1}(3') \cap f_2^{-1}(3')$.

Conditions 1, 2 and 3 together imply that

$$\bigcup_{x \in \{1,2,2'\}} f^{-1}(x) = \bigcup_{x \in \{1,2,2'\}} f_1^{-1}(x) = \bigcup_{x \in \{1,2,2'\}} f_2^{-1}(x).$$

Let us call this set R. If we fix the partitions for R for three functions f, f_1 and f_2, then what we want to compute resembles the cover product. Let us follow this idea. Any partition of R into 4 parts $P_1 \uplus P_2 \uplus P_3 \uplus P_4$ can be interpreted as follows: $P_1 = f^{-1}(1) = f_1^{-1}(1) =$

$f_2^{-1}(1)$, $P_2 = f_1^{-1}(2)$, $P_3 = f_2^{-1}(2)$ and $P_4 = f^{-1}(2') = f_1^{-1}(2') \cap f_2^{-1}(2')$. That is given $R = P_1 \uplus P_2 \uplus P_3 \uplus P_4$, we can uniquely determine $f_1|_R$ and $f_2|_R$ as follows:

$$\begin{aligned} f_1^{-1}(1) &= f_2^{-1}(1) = P_1 \\ f_1^{-1}(2) &= P_2 \\ f_2^{-1}(2) &= P_3 \\ f_1^{-1}(2') &= P_3 \cup P_4 \\ f_2^{-1}(2') &= P_2 \cup P_4 \end{aligned} \tag{4.1}$$

So we have $4^{|R|}$consistent partitions from a set R. Each partition in R determines the exact sets $f^{-1}(x), f_1^{-1}(x), f_2^{-1}(x)$ for all $x \in \{1,2,2'\}$, such that the first three conditions of consistency are satisfied. Thus we can partition the pair of functions (f_1, f_2) which are consistent with f according to (P_1, P_2, P_3, P_4), where $P_1 \uplus P_2 \uplus P_3 \uplus P_4 \subseteq X_t$, satisfies Equations (4.1). For every $S \subseteq X_t \setminus (P_1 \cup P_2 \cup P_3 \cup P_4)$, define

$$g_{1,S}(x) = \begin{cases} 1 & \text{if } x \in P_1 \\ 2 & \text{if } x \in P_2 \\ 2' & \text{if } x \in P_3 \cup P_4 \\ 3 & \text{if } x \in S \\ 3' & \text{otherwise} \end{cases} \qquad g_{2,S}(x) = \begin{cases} 1 & \text{if } x \in P_1 \\ 2 & \text{if } x \in P_3 \\ 2' & \text{if } x \in P_2 \cup P_4 \\ 3 & \text{if } x \in S \\ 3' & \text{otherwise} \end{cases}$$

$$g_S(x) = \begin{cases} 1 & \text{if } x \in P_1 \\ 2 & \text{if } x \in P_2 \cup P_3 \\ 2' & \text{if } x \in P_4 \\ 3 & \text{if } x \in S \\ 3' & \text{otherwise} \end{cases}$$

Let $R = P_1 \cup P_2 \cup P_3 \cup P_4)$. Now we define functions $c_{t_i}, c' : 2^{X_{t_i} \setminus R}$, where $i \in \{1,2\}$, as

$c_{t_1}(S) = c[t_1, g_{1,S}]$, $c_{t_2}(S) = c[t_2, g_{2,S}]$ and $c'(P_1, P_2, P_3, P_4, S) = (c_{t_1} *_c c_{t_2})(S)$. The values $c'(P_1, P_2, P_3, P_4, S)$ for all $S \subseteq X_t \setminus R$ can be computed in time $2^{|X_t \setminus R|} n^{\mathcal{O}(1)}$.

Having computed all $c'(P_1, P_2, P_3, P_4, S)$, the value $c[t, f]$ can be computed as,

$$c[t, f] = \min_{Q_1 \uplus Q_2 = f^{-1}(2)} \{c'(f^{-1}(1), Q_1, Q_2, P_4, S) - |f^{-1}(1)|\}.$$

The computation of $c[t, f]$ from the values $c'(P_1, P_2, P_3, P_4, S)$ takes time $\mathcal{O}(2^{|f^{-1}(2)|})$. The total running time to compute all values $c'(P_1, P_2, P_3, P_4, S)$ is upper bounded by $\sum_{R \subseteq X_t} 4^{|R|} 2^{|X_t \setminus R|} n^{\mathcal{O}(1)} = 6^{\mathsf{tw}} n^{\mathcal{O}(1)}$. For a node t, the total running time to compute all the entries $c[t, f]$ from $c'(P_1, P_2, P_3, P_4, S)$ is upper bounded by $\sum_{Q \subseteq X_t} 4^{|X_t \setminus Q|} 2^{|Q|} = 6^{\mathsf{tw}}$. Hence the total running time is $6^{\mathsf{tw}} n^{\mathcal{O}(1)}$.

Theorem 15. *Given a graph G together with a tree decomposition of width* tw*,* MDS *can be solved in time $\mathcal{O}^\star(6^{\mathsf{tw}})$.*

4.5 MDS on split graphs has no polynomial kernel

In this section, we show that MDS restricted to split graphs does not admit a polynomial kernel unless $\mathsf{coNP} \subseteq \mathsf{NP/poly}$. The proof is by a polynomial parameter transformation from RED-BLUE DOMINATING SET (RBDS), parameterized by the number of red vertices.

In the RBDS problem, the input is a bipartite graph G with bipartition $R \cup B$ and a positive integer ℓ. The objective is to test whether there exists a set $X \subseteq R$ of size at most ℓ that dominates the set B, i.e., $N(X) = B$. Such a set X is called a red-blue dominating set (rbds, for short) of G. This problem when parameterized by $|R|$ is the same as SMALL UNIVERSE HITTING SET, which does not have a polynomial kernel unless $\mathsf{coNP} \subseteq \mathsf{NP/poly}$ (see [32]). Thus we get the following result from [32].

Lemma 100 ([32])**.** RBDS *parameterized by* $|R|$ *has no polynomial kernel unless* coNP $\subseteq$ NP/poly.

We first state an auxiliary lemma, (the proof of which follows from Observation 85) and then prove the main theorem of the section.

Lemma 101. *Let G be a graph and let $X \subseteq V(G)$ be such that $G[X]$ is a complete graph. Let $S \subseteq V(G) \cup E(G)$ be such that S dominates all edges of $G[X]$. Then, $|S| \geq (|X|-1)/2$.*

Theorem 16. MDS *on split graphs does not admit a polynomial kernel, unless* coNP $\subseteq$ NP/poly.

Proof. The proof is by a polynomial parameter transformation from the RED-BLUE DOMINATING SET problem, parameterized by the number of red vertices. Consider an instance (G,ℓ) of RBDS, where $V(G) = R \cup B$. We assume that the instance (G,ℓ) has the following properties: (i) the set R contains an isolated vertex, and (ii) $|R|$ is odd and ℓ is even. It is easy to verify that these assumptions are safe.

We construct an equivalent instance (G',ℓ') of MDS as follows. Let G' be the graph obtained from G by turning $G[R]$ into a clique. That is, $V(G') = V(G) = R \cup B$ and $E(G') = E(G) \cup \{r_1r_2 \mid r_1, r_2 \in R\}$. Note that G' is a split graph. We set $\ell' = (|R|+\ell-1)/2$. We claim that G has an rbds of size at most ℓ if and only if G' has an mds of size at most $\ell' = (|R|+\ell-1)/2$.

Assume that (G,ℓ) is a yes-instance of RBDS and let $X \subseteq R$ be an rbds of G of size ℓ. Let $v \in R$ be an isolated vertex in G. If $v \in X$, then replace v with any other vertex in $R \setminus X$, which is still an rbds of G. So from now on, we assume that $v \notin X$. Now note that $|R \setminus (X \cup \{v\})| = |R| - (\ell+1)$ is even and $G'[R \setminus (X \cup \{v\})]$ is a complete graph and hence $G'[R \setminus (X \cup \{v\})]$ admits a perfect matching M. Then, $|M| = (|R|-\ell-1)/2$ and $|X \cup M| = |X| + |M| = \ell + (|R|-\ell-1)/2 = (|R|+\ell-1)/2$. We claim that $X \cup M$ is an mds of G'. Note that X dominates all vertices in B, as X is an rbds of G. Since $G'[R]$ is a clique, X dominates all vertices of R. (In fact, any one vertex of X dominates all of R.)

The set X also dominates all edges of G' that are incident with X. Thus, the only elements of $V(G') \cup E(G')$ that are not dominated by X are the edges incident with $R \setminus X$, but not incident with X. Now note that, since M is a perfect matching of $G'[R \setminus (X \cup \{v\})]$, we have $V(M) = R \setminus (X \cup \{v\})$, and hence M dominates all edges incident with $R \setminus (X \cup \{v\})$. Notice that all the edges incident with v and X are dominated by X. This implies that $M \cup X$ dominates all edges incident with $R \setminus X$. Thus, $X \cup M$ is an mds of G' of size $(|R| + \ell - 1)/2 = \ell'$.

Conversely, assume that G does not have an rbds of size at most ℓ. Let $S = V' \cup E'$ be a minimum-sized mds of G', where $V' \subseteq V(G')$ and $E' \subseteq E(G')$. We shall show that $|S| > \ell'$. Let $S' \subseteq S$ be a minimum-sized subset of S that dominates all vertices in B. Thus, S' consists of some vertices in $R \cup B$ and some edges in $E(R,B)$. Construct a set $S'' \subseteq R$ as follows. Add all vertices in $S' \cap R$ to S''. For every $xy \in S' \cap E(R,B)$, where $x \in R$ and $y \in B$, add x to S''. For every $y \in S' \cap B$, add a neighbor of y to S''. Then, $|S''| \leq |S'|$ and $S'' \supseteq V(S') \cap R$.

We claim that S'' is an rbds of G. To see this, consider $w \in B$. We shall show that $w \in N_G(S'')$. Since S is an mds of G', (i) either $w \in V'$, in which case we added a neighbor of w to S'', and hence $w \in N_G(S'')$, or (ii) S' contains an edge incident with w, say uw, in which case we added u to S'', and hence $w \in N_G(S'')$, or (iii) S' contains a neighbor (in G') of w, say u, and then $u \in S' \cap R \subseteq S''$, in which case also $w \in N_G(S'')$. Thus S'' is an rbds of G. Because of our assumption that G has no mds of size at most ℓ, we have $\ell < |S''| \leq |S'|$.

Now consider the subgraph $G'[R \setminus S'']$ of G'. Note that $G'[R \setminus S'']$ is a clique and none of its edges are dominated by S', because $S'' \supseteq V(S') \cap R$. Equivalently, (since S is an mds of G',) all edges of the clique $G'[R \setminus S'']$ are dominated by $S \setminus S'$. Therefore, by Lemma

101, $|S \setminus S'| \geq (|R \setminus S''| - 1)/2 = (|R| - |S''| - 1)/2$. Hence,

$$\begin{aligned}
|S| &= |S'| + (|R| - |S''| - 1)/2 \\
&\geq |S'| + (|R| - |S'| - 1)/2 \quad \text{(because } |S''| \leq |S'|\text{)} \\
&= (|R| + |S'| - 1)/2 \\
&> (|R| + \ell - 1)/2 \qquad\quad \text{(because } |S'| \geq |S''| > \ell\text{)} \\
&= \ell'.
\end{aligned}$$

That is $|S| > \ell'$. This completes the proof of the lemma. □

4.6 Polynomial kernel for MDS on $K_{d,d}$-free graphs

In this section, we show that MIXED DOMINATING SET admits a kernel with $\mathcal{O}(k^d)$ vertices on $K_{d,d}$-free graphs. A graph G is said to be $K_{d,d}$-free if G does not contain $K_{d,d}$ as a subgraph (not necessarily induced). Before focusing exclusively on $K_{d,d}$-free graphs, we explore some structural properties of graphs that have mixed dominating sets of size at most k, for a non-negative integer k. For a graph G and a non-negative integer k, we define the *k-induced partition* of G as follows.

Definition 102. *Let G be a graph without isolated vertices and k a non-negative integer. The k-*induced partition *of G is a triplet (H, I, R), where $H = \{v \in V(G) \mid d_G(v) \geq 2k+1\}$, $I = \{v \in V(G) \setminus H \mid N_G(v) \subseteq H\}$ and $R = V(G) \setminus (H \cup I)$.*

Notice the following immediate consequences of the above definition. First, the k-induced partition is unique, and that $H \cup I \cup R$ is indeed a partition of $V(G)$. Also, by definition, I is an independent set (i.e., $E(I) = \emptyset$) and there is no edge in G with one endpoint in I and the other in R (i.e., $E(I,R) = \emptyset$). Finally, it is easy to see that H must be contained in every vertex cover of G of size at most $2k$, if such a vertex cover exists. We now define what we call a *k-induced mixed dominating set*.

Definition 103. *Let G be a graph without isolated vertices and k be a non-negative integer. Let (H,I,R) be the k-induced partition of G. A mixed dominating set S of G is called a* k-induced mixed dominating set *if $V(S) \cap I = \emptyset$.*

The following lemma shows that if G has an mds of size at most k, then G has a k-induced mds.

Lemma 104. *Let G be a graph without isolated vertices and k a non-negative integer. Suppose (G,k) is a yes-instance of* MDS. *Then G has a k-induced mds.*

Proof. Let S' be a minimum-sized mds of G. Since(G,k) is a yes-instance, by Observation 85, we have that $H \subseteq V(S')$. We construct S from S' as follows. Add all vertices of $(H \cup R) \cap S'$ to S. Add all edges of $E(G[H \cup R]) \cap S'$ to S. For every edge $xy \in E(H,I) \cap S'$, where $x \in H$ and $y \in I$, add x to S. For every vertex $y \in S' \cap I$, add a neighbor of y to S. Then, $|S| \leq |S'|$ and $V(S) \cap I = \emptyset$. We claim that S is a mixed dominating set of G. Consider $x \in R \cup H$. Note that our construction of S ensures that if $x \in V(S')$, then $x \in V(S)$ as well. Thus S dominates $H \cup R$ and all edges incident with $H \cup R$. Therefore, in order to show that S is an mds of G, we only need to show that S dominates I. Consider $y' \in I$. Then, since S' dominates y', either $x'y' \in S'$ for some $x' \in H$, in which case $x' \in S$, and hence S dominates y'; or $y' \in S'$, in which case we added a neighbor of y' to S, and hence S dominates y; or S' contains a neighbor of y', say $z' \in H$, in which case S contains z' too, and hence S dominates y'. Thus S is a minimum-sized mds of G and $V(S) \cap I = \emptyset$. Hence, S is a k-induced mds of G. This completes the proof of the lemma. $\square$

In what follows, let (G,k) be an instance of MDS where G is a $K_{d,d}$-free graph. Note that all isolated vertices in G must belong to any mixed dominating set. Consequently, we apply the following reduction rule.

Reduction Rule 1: If $v \in V(G)$ is an isolated vertex, then delete v from G and decrease k by one.

From now on, we assume that G has no isolated vertices. Let (H, I, R) be the k-induced partition of G. We now separately bound the sizes of H, R and I in the event that (G, k) is a yes-instance. The proof of the following lemma, which bounds $|H|$, follows from Observation 85.

Lemma 105. *If $|H| > 2k$, then (G, k) is a no-instance.*

Lemma 106. *If $|R| > 8k^2$, then (G, k) is a no-instance.*

Proof. Assume that (G, k) is a yes-instance. Let S be an mds of G of size at most k. We shall show that $|R| \leq 8k^2$. Note that $V(S)$ is a vertex cover of G. Now, consider the graph $G[R]$, the subgraph of G induced by R. Then, $V(S) \cap R$ is a vertex cover of $G[R]$. Since every vertex in R has degree at most $2k$, $V(S) \cap R$ can cover at most $2k|V(S) \cap R|$ edges of $G[R]$. Since $V(S) \cap R$ is a vertex cover of $G[R]$, $V(S) \cap R$ covers all edges of $G[R]$. We thus get that $|E(G[R])| \leq 2k|V(S) \cap R| \leq 2k|V(S)| \leq 2k \times 2k = 4k^2$.

Recall that Rule 1 removed all isolated vertices from G. Observe that $G[R]$ has no isolated vertices either. If $v \in R$ were isolated in $G[R]$, then $N_G(v) \subseteq H$, which then would imply that $v \in I$, a contradiction. Since $G[R]$ has no isolated vertices and $|E(G[R])| \leq 4k^2$, we have $|V(G[R])| \leq 2|E(G[R])| \leq 8k^2$. □

As Lemmas 105 and 106 show, if either $|H| > 2k$ or $|R| > 8k^2$, then we can immediately conclude that (G, k) is a no-instance. This leads to the following reduction rule.

Reduction Rule 2: Let (G, k) be an instance of MDS and let (H, I, R) be the k-induced partition of $V(G)$. If $|H| > 2k$ or $|R| > 8k^2$, then return a trivial no-instance.

Hence from now on, we assume that $|H| \leq 2k$ and $|R| \leq 8k^2$. In order to obtain a kernel, we now need to upper bound $|I|$. Towards that end, we introduce the following reduction rule.

Reduction Rule 3: If there exist $2k+2$ distinct vertices $u_0, u_1, \ldots, u_{2k+1}$ in I such that $N_G(u_0) = N_G(u_1) = \cdots = N_G(u_{2k+1})$, then delete u_0 from G.

Lemma 107. *Reduction Rule 3 is safe.*

Proof. Consider $u_0, u_1, \ldots, u_{2k+1} \in I$ with $N_G(u_0) = N_G(u_1) = \cdots = N_G(u_{2k+1})$. We shall show that (G,k) is a yes-instance if and only if $(G - u_0, k)$ is a yes-instance. Now, if (G,k) is a yes-instance, then by Lemma 104, G has a k-induced mds, say S, of size at most k. Then $u_0 \notin V(S)$, and hence S is an mds of $G - u_0$ as well.

To see the reverse direction, assume that $(G - u_0, k)$ is a yes-instance. Then, since $d_{G-u_0}(x) \geq 2k+1$ for every $x \in H$, $(H, I \setminus \{u_0\}, R)$ is the k-induced partition of $G - u_0$. By Lemma 104, $G - u_0$ has a k-induced mds, say S', of size at most k. That is, $V(S') \cap (I \setminus \{u_0\}) = \emptyset$. We claim that S' is a mixed dominating set of G as well. Note that we only need to show that S' dominates u_0 and all edges incident with u_0. By Observation 85, we have $H \subseteq V(S')$. In particular, $N_G(u_0) \subseteq H \subseteq V(S')$, and hence S' dominates all edges incident with u_0. Now, since S' dominates u_1, and since S' is k-induced, S' must contain a neighbor of u_1, say v. But we have $N_G(u_0) = N_G(u_1)$. Thus $v \in N_G(u_0) \cap S'$, and hence S' dominates u_0. $\square$

It is easy to see that all the Reduction Rules 1–3 can be applied in polynomial time. Assume that Reduction Rules 1–3 have been applied exhaustively. Let (G', k') be the reduced instance with the k'-induced partition (H, I, R). Notice that $k' \leq k$. We partition I into two parts as follows. Let $I_{<d}$ be the set of vertices in I that have at most $(d-1)$ neighbors and $I_{\geq d}$ be the set of vertices in I that have at least d neighbors. That is, $I_{<d} = \{v \in I \mid |N_{G'}(v)| \leq d-1\}$ and $I_{\geq d} = I \setminus I_{<d} = \{v \in I \mid |N_{G'}(v)| \geq d\}$. We shall separately bound the sizes of $I_{<d}$ and $I_{\geq d}$. Let $\mathsf{C}H_{<d}$ be the family of subsets of H of size at most $(d-1)$, i.e., $\mathsf{C}H_{<d} = \{X \subset H \colon |H| \leq d-1\}$. Let $\mathsf{C}H_{=d}$ be the family of subsets of H of size exactly d, i.e., $\mathsf{C}H_{=d} = \{X \subset H \colon |H| = d\}$.

Observation 108. $|\mathsf{C}H_{<d}| \leq d(2k')^{d-1}$ *and* $|\mathsf{C}H_{=d}| \leq (2k')^d$.

Lemma 109. *$|I_{<d}| \leq d(2k'+1)(2k')^{d-1}$ and $|I_{\geq d}| \leq (d-1)(2k')^d$.*

Proof. Since Reduction Rule 3 has been applied exhaustively, for every $X \in \mathsf{C}H_{<d}$ there exist at most $2k'+1$ vertices $x \in I_{<d}$ such that $N_{G'}(x) = X$. Hence, $|I_{<d}| \leq (2k'+1)|\mathsf{C}H_{<d}|$, and then using Observation 108, we get $|I_{<d}| \leq (2k'+1)d(2k')^{d-1}$.

Note that for every $x \in I_{\geq d}$, there exists $X \in \mathsf{C}H_{=d}$ such that $N_{G'}(x) \supseteq X$; and call x and X *partners* of each other. Now, given $Y \in \mathsf{C}H_{=d}$, note that Y can have at most $(d-1)$ partners in $I_{\geq d}$. For otherwise, if Y has at least d partners, then the graph induced on Y and all its partners contains $K_{d,d}$ as a subgraph. But this is not possible as G is $K_{d,d}$-free. Thus, every $x \in I_{\geq d}$ has a partner in $\mathsf{C}H_{=d}$, and every $Y \in \mathsf{C}H_{=d}$ has at most $(d-1)$ partners. We thus have $|I_{\geq d}| \leq (d-1)|\mathsf{C}H_{=d}| \leq (d-1)(2k')^d$. □

Lemma 109 shows that $|I| = |I_{<d} \cup I_{\geq d}| \leq d(2k'+1)(2k')^{d-1} + (d-1)(2k')^d = (2k')^{d-1}$ $(4dk' - 2k' + d)$. Thus, as $k' \leq k$, we get the following theorem.

Theorem 17. MIXED DOMINATING SET *on $K_{d,d}$-free graphs admits a kernel with $(2k)^{d-1}(4dk - 2k + d) + 8k^2 + 2k = \mathcal{O}(k^d)$ vertices.*

4.7 MDS on proper interval graphs is in P

In this section, we prove the following theorem.

Theorem 18. MDS *on proper interval graphs can be solved in time $\mathcal{O}(n^{13})$, where n is the number of vertices of the input graph.*

In order to prove Theorem 18, we design a dynamic programming algorithm. In light of Observation 87, notice that we need to consider only special mixed dominating sets of the input graph.

We first state some properties of proper interval graphs that will be used throughout this section. Let G be a proper interval graph and $I(G)$ be its proper interval representation.

That is, $I(G)$ is a set of intervals such that no interval is contained in any of the other intervals in $I(G)$. For each $v \in V(G)$ with interval $I(v)$, $l(v)$ denotes the left endpoint of $I(v)$ and $r(v)$ denotes the right endpoint of $I(v)$. Observe that for two distinct vertices u and v of G, if $l(u) < l(v)$, then $r(u) < r(v)$. This imposes a natural total order on the intervals, which in turn defines an ordering $\pi : V(G) \to [|V(G)|]$ of the vertices. The ordering π is obtained by listing the vertices in the ascending order of the left (or right) endpoints of their intervals. This ordering is referred to as a *proper interval ordering* in the literature and has the following property.

Proposition 110 (folklore)**.** *Let G be a proper interval graph with proper interval ordering π. For every pair u, v of vertices with $\pi(u) < \pi(v)$, if $uv \in E(G)$, then $\{w \in V(G) \mid \pi(u) \leq \pi(w) \leq \pi(v)\}$ is a clique in G.*

Given a proper interval representation, the following result states that the vertex set of the proper interval graph can be organized into a sequence of cliques satisfying certain properties.

Proposition 111 (folklore)**.** *Given a proper interval graph G with proper interval ordering π, there is a linear-time algorithm that outputs a partition of $V(G)$ into a sequence $Q_1, \cdots, Q_q$ of (pairwise vertex-disjoint) cliques satisfying the following properties.*

(i) For each pair of vertices $u \in Q_i$, $v \in Q_j$ with $1 \leq i < j \leq q$, $\pi(u) < \pi(v)$.

(ii) For every edge $uv \in E(G)$, there exists $1 \leq i \leq q$ such that either $u, v \in Q_i$ or $u \in Q_i$ and $v \in Q_{i+1}$.

We refer to the ordered set of cliques $\mathsf{C}Q = \{Q_1, \cdots, Q_q\}$ as the *clique-partition* of G. Given a proper interval representation, a proper interval ordering π and a clique-partition $\mathsf{C}Q$ of a proper interval graph can be obtained in polynomial time [45, 59]. So we assume that $\mathsf{C}Q$ and π are given as part of the input with the proper interval graph G. We remark that neither the proper interval representation (and hence the proper interval ordering resulting from it) nor the clique-partition is unique.

Throughout this section, for a proper interval graph G, we fix an interval representation along with the proper interval ordering π and the clique-partition $\mathsf{C}Q$ obtained from it. We begin by proving the following preparatory lemmas that explore how a minimum-sized special mixed dominating set of G interacts with the clique induced by Q_i.

Lemma 112. *Let G be a proper interval graph. Then G has a minimum-sized special mds S such that for every $i \in [q-1]$, $|S \cap E(Q_i, Q_{i+1})| \leq 1$, i.e., S contains at most one edge with one endpoint in Q_i and the other in Q_{i+1}.*

Proof. Let S be a minimum-sized special mds of G. Fix $i \in [q-1]$. Suppose that $|S \cap E(Q_i, Q_{i+1})| \geq 2$. Let ab and cd be two distinct edges in S such that $a, c \in Q_i$ and $b, d \in Q_{i+1}$. Note that since $G[Q_i]$ and $G[Q_{i+1}]$ are cliques, we have $ac, bd \in E(G)$. Let S' be the set obtained from S by replacing ab and cd with ac and bd, i.e., $S' = (S \setminus \{ab, cd\}) \cup \{ac, bd\}$. Then $|S'| = |S|$ and S' is a special mds of G as edges ac and bd together dominate exactly the same vertices and edges as ab and cd together do. Apply this replacement procedure exhaustively to get a minimum-sized mds with the desired property. $\square$

Lemma 113. *Let G be a proper interval graph. Then every minimum-sized mds of G contains at most 3 vertices from Q_i for every $i \in [q]$.*

Proof. Suppose the lemma is not true. Then there exists a minimum-sized mds S of G and $i \in [q]$ such that $|S \cap Q_i| \geq 4$. Let a, b, c, d be four distinct vertices in $S \cap Q_i$. Without loss of generality, assume that $\pi(a) < \pi(b) < \pi(c) < \pi(d)$. Let e denote the edge bc. Now we claim that $S' = (S \setminus \{b, c\}) \cup \{e\}$ is an mds of G, which will contradict the fact that S is a minimum-sized mds as $|S'| < |S|$.

First, note that the set $\{b, c\}$ dominates $N[\{b, c\}]$ and all the edges incident with b or c. Now, the edge $e = bc$ dominates $\{b, c\}$ and all the edges incident with b or c. Therefore, now we only need to show that S' dominates $N(b) \cup N(c)$. We shall show that $N(b) \cup N(c) \subseteq N[\{a, d\}]$, which will imply that S' dominates $N(b) \cup N(c)$, as $a, d \in S'$.

Consider $x \in N(c)$. Since $xc \in E(G)$ and $c \in Q_i$, by Proposition 111-(ii), either $x \in Q_i$ or $x \in Q_{i+1}$ or $x \in Q_{i-1}$. If $x \in Q_i$, then x is adjacent to both a and d. If $x \in Q_{i+1}$, then, since $d \in Q_i$, by Proposition 111-(i), we get $\pi(c) < \pi(d) < \pi(x)$. But then by Proposition 110, the subgraph of G induced by $X = \{w \in V(G) \mid \pi(c) \leq \pi(w) \leq \pi(x)\}$ is a clique. Note that $d \in X$ and hence d and x are adjancet. Using symmetric arguments, we can show that if $x \in Q_{i-1}$, then a and x are adjacent. In any case, we have $x \in N[\{a,d\}]$. Similarly, we can also show that $y \in N[\{a,d\}]$ for every $y \in N(b)$. Thus, we have $N(b) \cup N(c) \subseteq N[\{a,d\}]$, and this completes the proof. □

Consider a minimum-sized mds S of G. Fix $i \in [q]$. Let $S_i \subseteq S$ be the set of all vertices and edges in S that have at least one endpoint in $Q_{\leq i} = \bigcup_{1 \leq j \leq i} Q_j$. The following lemma shows that $Q_{\leq i}$ contains at most one vertex that is not dominated by S_i.

Lemma 114. *Let G be a proper interval graph and S be a minimum-sized mds of G. Let* $S_i = V_i \cup E_i = \{x \in S \colon V(x) \cap (\bigcup_{1 \leq j \leq i} Q_j) \neq \emptyset\}$. *Then* $|\{Q_1 \cup Q_2 .... \cup Q_i\} \setminus \{N[V_i] \cup V(E_i)\}| \leq 1$.

Proof. Suppose that the lemma is not true. Then there are at least two vertices in $\bigcup_{1 \leq j \leq i} Q_j$ that are not dominated by S_i. Let a,b be two such vertices. We now claim that a and b must belong to Q_i. To see this, note that both a and b are dominated by $S \setminus S_i$. In particular, a and b are dominated by some vertices in $S \setminus S_i$. For otherwise, if an edge, say $e \in S \setminus S'$, were to dominate a, then $a \in Q_i$ must be the endpoint of $e \in S$, which means that $e \in S_i$, and hence a is dominated by S_i, a contradiction. Therefore, both a and b are dominated by $S \cap (Q_{i+1} \cup \cdots \cup Q_q)$, which means that there exist $a', b' \in S \cap (Q_{i+1} \cup \cdots \cup Q_q)$ such that $a' \in N(a)$ and $b' \in N(b)$. Then, since $a \in Q_j$ for some $j \leq i$ and $a' \in Q_{j'}$ for some $j' \geq i+1$, and $aa' \in E(G)$, by Proposition 111-(ii), we get that $a \in Q_i$ and $a' \in Q_{i+1}$. Similarly, $b \in Q_i$ and $b' \in Q_{i+1}$. Then, since $G[Q_i]$ is a clique, $ab \in E(G)$. But then, because S does not contain a,b or any edge incident with a or b, the edge ab cannot be dominated by the S, which is a contradiction. This completes the proof. □

Lemma 115. *Let* G, S, S_i *be as defined in Lemma 114. Then,* $|Q_i \setminus V(S_i)| = |Q_i \setminus V(S)| \leq 1$.

Proof. If there exist two distinct vertices $a, b \in Q_i \setminus V(S_i)$, then S does not contain a, b or any edge incident with a or b, which implies that S does not dominate the edge ab, a contradiction. $\square$

Dynamic programming algorithm for MDS.

We now design a dynamic programming algorithm that computes the size of a minimum-sized mds of a proper interval graph G. Let us first try to develop an intuitive understanding of the algorithm. Let G be the input proper interval graph and $\mathscr{Q} = \{Q_1, \dots, Q_q\}$ be the given clique-partition of G. We process the graph from left to right. That is, at stage $i \in [q]$, we consider the subgraph of G induced by $Q_{\le i} = Q_1 \cup Q_2 \cup \cdots \cup Q_i$.

Lemmas 112−115 show that G has a minimum-sized special mds S with the following properties.

(i) For any $i \in [q]$, S contains at most three vertices from Q_i ("vertices in the solution").

(ii) For any $i \in \{2, \dots, q\}$, S contains at most one edge from $E(Q_{i-1}, Q_i)$ ("past edge," has one endpoint in Q_i).

(iii) For any $i \in [q-1]$, S contains at most one edge from $E(Q_i, Q_{i+1})$ ("future edge," has one endpoint in Q_i).

(iv) For any $i \in [q]$, at most one vertex of Q_i is not dominated by S ("vertex to be dominated in the future by some vertex").

(v) For any $i \in [q]$, at most one vertex of Q_i does not belong to $V(S)$ ("vertex not part of the solution").

We try to compute the size of such an mds S. At stage i, we guess these "special vertices" in Q_i, and for each guess find an *optimal partial solution*. With all these in mind, we define a feasible tuple as follows.

Definition 116. *For $i \in [q]$, a quintuple $(F_i, P_i, Y_i, N_i, T_i)$, where $F_i, P_i, Y_i, N_i, T_i \subseteq Q_i$ and $|F_i|, |P_i|, |N_i|, |T_i| \leq 1$ and $|Y_i| \leq 3$ is said to be a* feasible tuple at the *i*th stage *if the following conditions hold.*

1. $|N_i \cup T_i| \leq 1$ *and* $N_i \cap T_i = \emptyset$.
2. $F_i \cap (N_i \cup T_i) = \emptyset$.
3. *If* $Y_i \neq \emptyset$, *then* $T_i = \emptyset$.
4. $|Q_i \setminus (F_i \cup P_i \cup Y_i \cup N_i \cup T_i)|$ *is even.*

Intuitively, think of a feasible tuple as follows. Let $S = V' \cup E'$ be an optimal solution, where $V' \subseteq V(G)$ and $E' \subseteq E(G)$. Let $S_{\leq i} = V_i \cup E_i$ be the set of vertices and edges in S that are fully contained in $G[Q_{\leq i}]$.

- $F_i = (V(E(Q_i, Q_{i+1})) \cap S) \cap Q_i$ (i.e., F_i contains the endpoint in Q_i of the future edge).
- $P_i = V(E(Q_{i-1}, Q_i) \cap S) \cap Q_i$ (i.e., P_i contains the endpoint in Q_i of the past edge).
- $Y_i = Q_i \cap S$ (i.e., Y_i is the set of vertices in Q_i that belong to the solution).
- N_i contains the vertex in Q_i that is not part of the solution, but dominated by $S \cap Q_{\leq i}$.
- T_i contains the vertex in Q_i that is not part of the solution and to be dominated in the future, i.e., the vertex in $Q_i \setminus (N[V_i] \cup V(E_i))$.

Now look at the conditions in the definition of a feasible tuple. The first condition ensures that at most one vertex from Q_i is excluded from $V_i \cup V(E_i)$ or left un-dominated (See Lemmas 114 and 115). Note that we want $F_i \subseteq V(S)$ and hence F_i is not "to be dominated in the future" or "not part of solution" and thus we have the second condition. The third condition ensures that if $S_{\leq i} \subseteq S$ contains at least one vertex from (the clique) Q_i, then all vertices of Q_i are dominated, and hence no vertex of Q_i needs to be dominated in the future. Let H denote the clique induced by $Q_i \setminus (F_i \cup P_i \cup Y_i \cup N_i \cup T_i)$. The fourth condition ensures that H admits a perfect matching. Note that we need all edges of H to be dominated by E_i, and since E_i is a matching, we need H to have a perfect matching (see Observation 85(iii)).

We now define a *valid set* for a feasible tuple and a pair of *compatible tuples* as follows.

Definition 117. *For every $i \in [q]$ and for every feasible tuple at ith stage, $\sigma = (F_i, P_i, Y_i, N_i, T_i)$, a set $S' = V' \cup E'$, where $V' \subseteq Q_{\leq i}$ and $E' \subseteq E(G[Q_{\leq i}])$ is said to be valid for σ if we the following conditions hold.*

(i) All vertices in $Q_{\leq i} \setminus (F_i \cup T_i)$ are dominated by S'.

(ii) All edges of the graph $G[Q_{\leq i} \setminus (F_i)]$ are dominated by S'.

(iii) $Q_i \setminus V(S') = F_i \cup N_i \cup T_i$.

(iv) $Q_i \cap V' = Y_i$.

(v) If $P_i = y$, then there exists $xy \in E'$ such that $x \in Q_{i-1}$, and there exists no other edge $e = x'y' \in E'$ such that $x' \in Q_{i-1}, y' \in Q_i$.

Definition 118. *A feasible tuple at the ith stage $(F_i, P_i, Y_i, N_i, T_i)$ and a feasible tuple at the $(i-1)$st stage $(F_{i-1}, P_{i-1}, Y_{i-1}, N_{i-1}, T_{i-1})$ are said to be* compatible *if the following conditions hold.*

1. *$F_{i-1} = \emptyset \Leftrightarrow P_i = \emptyset$ or if $F_{i-1} = \{x\}, P_i = \{y\}$ then x and y are adjacent.*
2. *$N_i \subseteq N[Y_{i-1} \cup Y_i]$.*
3. *$T_{i-1} \subseteq N[Y_i]$.*
4. *There are no edges between the vertices of $(N_{i-1} \cup T_{i-1})$ and $(N_i \cup T_i)$.*

Intuitively, for a feasible tuple σ at stage i, and any valid set S' for σ, S' can be derived from a tuple which is compatible with σ. Let S be a special mds for G satisfying

Lemmas 112−115. Let $S_{\leq i}$ and $S_{<i}$ be the sets of vertices and edges of S restricted to $G[Q_{\leq i}]$ and $G[Q_{\leq i-1}]$, respectively. The set $S_{\leq i}$ and $S_{<i}$ are valid sets for $(F_i, P_i, Y_i, N_i, T_i)$ and $(F_{i-1}, P_{i-1}, Y_{i-1}, N_{i-1}, T_{i-1})$, respectively, where each entry in the tuple is defined as follows.

- $F_{i-1} = Q_{i-1} \cap V(e)$, where $e \in S_{<i} \cap E(Q_{i-1}, Q_i)$,
- $P_{i-1} = Q_{i-1} \cap V(e)$, where $e \in S_{<i} \cap E(Q_{i-2}, Q_{i-1})$,
- $Y_{i-1} = S_{<i} \cap Q_{i-1}$,
- $N_{i-1} = N(S_{<i} \cap V(G)) \cap (Q_{i-1} \setminus V(S))$, and
- $T_{i-1} = Q_{i-1} \setminus (N[S_{<i} \cap V(G)] \cup V(S \cap E(G)))$.
- $F_i = Q_i \cap V(e)$, where $e \in S_{\leq i} \cap E(Q_i, Q_{i+1})$,
- $P_i = Q_i \cap V(e)$, where $e \in S_{\leq i} \cap E(Q_{i-1}, Q_i)$,
- $Y_i = S_{\leq i} \cap Q_i$,
- $N_i = N(S_{\leq i} \cap V(G)) \cap (Q_i \setminus V(S))$, and
- $T_i = Q_i \setminus (N[S_{\leq i} \cap V(G)] \cup V(S \cap E(G)))$.

One can verify that all the four conditions of Definition 118 holds for tuples $(F_i, P_i, Y_i, N_i, T_i)$ and $(F_{i-1}, P_{i-1}, Y_{i-1}, N_{i-1}, T_{i-1})$. Condition 1 follows from the fact that F_{i-1} and P_i are end points of the edge in $S \cap E(Q_{i-1}, Q_i)$, if one such edge exits, otherwise those sets will be empty. Condition 2 and 3 follows from the above definitions of N_i and T_i and the fact that S is an mds of G. If there is an edge between $N_i \cup T_i$ and $N_{i-1} \cup T_{i-1}$, then S can not dominate that edge. Thus condition (4) follows.

DP Table and its computation. For every $i \in [q]$ and for every feasible tuple at ith stage, $\sigma = (F_i, P_i, Y_i, N_i, T_i)$, we define $M[i, (F_i, P_i, Y_i, N_i, T_i)]$ to be the size of a smallest set $S = V' \cup E'$ such that S is valid for $M[i, \sigma]$. We refer to such a minimum-sized set S as a set that achieves (or realizes) $M[i, (F_i, P_i, Y_i, N_i, T_i)]$. Observe that $\min\{M[q, (F_q, P_q, Y_q, N_q, T_q)]\}$, where the minimum is over all feasible tuples at the qth stage with $F_q = T_q = \emptyset$ gives the size of a minimum mixed dominating set of G.

We now show that the DP table entries $M[i, \sigma]$ can be computed using the following recurrence.

$$M[1, (F_1, P_1, Y_1, N_1, T_1)] = \begin{cases} |Y_1| + \frac{|Q_1 \setminus (F_1 \cup Y_1 \cup N_1 \cup T_1)|}{2}, & \text{if } P_1 = \emptyset \\ \infty, & \text{otherwise.} \end{cases}$$

For $i \in \{2, 3, \ldots, q\}$,

$$\begin{aligned} M[i, (F_i, P_i, Y_i, N_i, T_i)] = \min\, \{ & M[i-1, (F_{i-1}, P_{i-1}, Y_{i-1}, N_{i-1}, T_{i-1})] + |P_i| \\ & + |Y_i| + \frac{|Q_i \setminus (F_i \cup P_i \cup Y_i \cup N_i \cup T_i)|}{2} \}, \end{aligned}$$

where the minimum is over all feasible tuples at the $(i-1)$st stage (F_{i-1}, P_{i-1}, Y_{i-1}, N_{i-1}, T_{i-1}) that are compatible with (F_i,P_i,Y_i,N_i,T_i).

Correctness of the algorithm. In what follows, we show that the above recursive formula correctly computes $M[i, (F_i, P_i, Y_i, N_i, T_i)]$. The proof is by induction on i. The base case follows easily, because the least number of elements that dominate all the edges in a clique on 2ℓ vertices is a perfect matching. Now consider the induction step. Let $\sigma = (F_i, P_i, Y_i, N_i, T_i)$ be a feasible tuple at stage i.

Let σ' be a feasible tuple at the $(i-1)$st stage such that σ' is compatible with σ and $M[i-1, \sigma'] = \min_{\sigma''} M[i-1, \sigma'']$, where the minimum is over all feasible tuples at the

$(i-1)st$ stage σ'' that are compatible with σ. Also, let $S_{\sigma'} = V_{\sigma'} \cup E_{\sigma'}$ be the set that achieves $M[i-1, \sigma']$. Define a set Z as follows. If $F_{i-1} = \emptyset$, then set $Z = \emptyset$. Otherwise let $F_{i-1} = \{z\}$. Then by compatibility conditions, we have $\emptyset \neq P_i = \{z'\}$ for some $z' \in Q_i$ and $zz' \in E(G)$. In this case we set $Z = \{zz'\}$. Let P_σ be a perfect matching in the graph induced by $Q_i \setminus (F_i \cup P_i \cup Y_i \cup N_i \cup T_i)$, which is a complete graph. We claim that $S_\sigma = S_{\sigma'} \cup Z \cup Y_i \cup P_\sigma$ is a valid set for σ.

(a) All vertices in $Q_{\leq i-1} \setminus (F_{i-1} \cup T_{i-1})$ are dominated by $S_{\sigma'}$. The set Z dominates F_{i-1} and P_i. Since σ and σ' are compatible, $T_{i-1} \subseteq N[Y_i]$. Hence S_σ dominates all the vertices in Q_{i-1} as well as P_i. By condition (2) of Definition 118, we have that $N_i \subseteq N[Y_{i-1} \cup Y_i]$ and by the definition $S_{\sigma'}$, $Y_{i-1} \subseteq S_{\sigma'}$. This implies that S_σ dominates N_i. Also notice that the perfect matching P_σ dominates $Q_i \setminus (F_i \cup P_i \cup Y_i \cup N_i \cup T_i)$. Therefore, S_σ dominates all the vertices in $Q_{\leq i} \setminus (F_i \cup T_i)$. Hence S_σ satisfies condition (i) of Definition 117.

(b) All edges of $G[Q_{\leq i-1} \setminus (F_{i-1})]$ are dominated by $S_{\sigma'}$. The set Z dominates all edges incident with F_{i-1}. Thus, $S_{\sigma'} \cup Z$ dominates all edges of $G[Q_{\leq i-1}]$. Now consider an edge $e \in E(Q_{i-1}, Q_i \setminus F_i) \cup E(Q_i \setminus F_i)$, i.e., e has at least one of its endpoints in $Q_i \setminus F_i$. If e is incident with $P_i \cup Y_i$, then $S_\sigma (\supseteq Z \cup Y_i)$ dominates e. If e is incident with $Q_i \setminus (F_i \cup P_i \cup Y_i \cup N_i \cup T_i)$, then the perfect matching P_σ dominates e. Otherwise, e is incident with $N_i \cup T_i$. Recall that $|N_i \cup T_i| \leq 1$ and $N_i \cap T_i = \emptyset$. Let u and v be the endpoints of e, where $\{u\} = N_i \cup T_i$. If $v \in Q_i$, then since $v \notin F_i$, we must have $v \in P_i \cup Y_i \cup V(P_\sigma)$. But we already covered the cases of edges incident with $P_i \cup Y_i \cup V(P_\sigma)$. Therefore, assume that $v \in Q_{i-1}$. Then by the condition (4) of Definition 118, we have that $v \notin (N_{i-1} \cup T_{i-1})$. If $v \in F_{i-1}$, then Z dominates e. Otherwise $v \in Q_{i-1} \setminus (F_{i-1} \cup N_{i-1} \cup T_{i-1})$. Note that by definition of $M[i-1, \sigma']$, we have $Q_{i-1} \setminus V(S_{\sigma'}) = (F_{i-1} \cup N_{i-1} \cup T_{i-1})$, which implies that $v \in V(S_{\sigma'})$, in which case $S_\sigma (\supseteq S_{\sigma'})$ dominates e. Hence S_σ satisfies condition (ii) of Definition 117.

(c) Notice that $Q_i \cap V(S_\sigma) = P_i \cup Y_i \cup V(P_{\sigma'})$, and therefore, $Q_i \setminus V(S_\sigma) = (F_i \cup N_i \cup T_i)$.

Hence S_σ satisfies condition (iii) of Definition 117.

(d) Notice that $S_\sigma \cap Q_i = Y_i$. Hence S_σ satisfies condition (iv) of Definition 117.

(e) Notice that $E(Q_{i-1}, Q_i) \cap S_\sigma = Z$, and $|Z| \leq 1$. Therefore, either $Z = \emptyset$, in which case S_σ contains no edge from $E(Q_{i-1}, Q_i)$, or S_σ contains exactly one edge from $E(Q_{i-1}, Q_i)$, the unique edge in Z with its endpoints being in $F_{i-1} \cup P_i$. Hence S_σ satisfies condition (v) of Definition 117.

We have thus seen that S_σ is a valid set for σ. Since $M[i,\sigma]$ is the defined to be the minimum size of such a set, we get that

$$\begin{aligned} M[i,\sigma] &\leq |S_\sigma| \\ &= M[i-1, S_{\sigma'}] + |P_i| + |Y_i| + |P_\sigma| \\ &= \min_{\substack{\sigma'' \\ \text{compatible} \\ \text{with } \sigma}} \left\{ M[i-1, S_{\sigma''}] + |P_i| + |Y_i| + \frac{|Q_i \setminus (F_i \cup P_i \cup Y_i \cup N_i \cup T_i)|}{2} \right\}. \end{aligned}$$

Now let $R_\sigma \subseteq V(G[Q_{\leq i}]) \cup E(G[Q_{\leq i}])$ be a set that achieves $M[i,\sigma]$. Let $\hat{R} = V_{\hat{R}} \cup E_{\hat{R}}$, where $V_{\hat{R}} = R_\sigma \cap (V(G[Q_{\leq i-1}])$ and $E_{\hat{R}} = R_\sigma \cap E(G[Q_{\leq i-1}]))$, i.e., $\hat{R}$ is the intersection of R_σ with the graph induced by $Q_{\leq i-1}$. There exists a feasible tuple at the $(i-1)$st stage, say $\hat{\sigma}$, such that $\hat{R}$ is a valid set for $\hat{\sigma}$, and $\hat{\sigma}$ is compatible with σ. In fact, we have $\hat{\sigma} = (F_{i-1}, P_{i-1}, Y_{i-1}, N_{i-1}, T_{i-1})$, where,

- $F_{i-1} = Q_{i-1} \cap V(e)$, where $e \in \hat{R} \cap E(Q_{i-1}, Q_i)$,
- $P_{i-1} = Q_{i-1} \cap V(e)$, where $e \in \hat{R} \cap E(Q_{i-2}, Q_{i-1})$,
- $Y_{i-1} = \hat{R} \cap Q_{i-1}$,
- $N_{i-1} = N(\hat{R} \cap V(G)) \cap (Q_{i-1} \setminus V(S))$, and
- $T_{i-1} = Q_{i-1} \setminus (N[\hat{R} \cap V(G)] \cup V(S \cap E(G)))$.

Since $\hat{R}$ is a valid set for $\hat{\sigma}$, we get that $M[i-1,\hat{\sigma}] \leq |\hat{R}|$. Since σ is compatible with $\hat{\sigma}$, we have $|F_{i-1}| = |P_i|$. Now, if P_i is non-empty, then, (because R_σ is valid for σ,) R_σ contains the unique edge in the set $E(F_{i-1},P_i)$. Again, since the set R_σ is valid for σ, we have the following.

(i) $R_\sigma \cap Q_i = Y_i$.

(ii) $Q_i \setminus V(R_\sigma) = (F_i \cup N_i \cup T_i)$. This, along with the fact that $P_i \cup Y_i \subseteq V(R_\sigma)$ implies that for every vertex $v \in Q_i \setminus (F_i \cup P_i \cup Y_i \cup N_i \cup T_i)$, R_σ contains an edge (from $E(Q_i)$) that is incident with v. More specifically, $R_\sigma \setminus (\hat{R} \cup P_i \cup Y_i)$ contains an edge from $E(Q_i)$ that is incident with v. Therefore, $|R_\sigma \setminus (\hat{R} \cup P_i \cup Y_i)| \geq \frac{|Q_i \setminus (F_i \cup P_i \cup Y_i \cup N_i \cup T_i)|}{2}$.

The above facts imply that

$$\begin{aligned}
M[i,\sigma] &= |R_\sigma| \\
&\geq |\hat{R}| + |P_i| + |Y_i| + \frac{|Q_i \setminus (F_i \cup P_i \cup Y_i \cup N_i \cup T_i)|}{2} \\
&\geq M[i-1,\hat{\sigma}] + |P_i| + |Y_i| + \frac{|Q_i \setminus (F_i \cup P_i \cup Y_i \cup N_i \cup T_i)|}{2} \\
&\geq \min_{\substack{\sigma'' \\ \text{compatible} \\ \text{with } \sigma}} \left\{ M[i-1,\sigma''] + |P_i| + |Y_i| + \frac{|Q_i \setminus (F_i \cup P_i \cup Y_i \cup N_i \cup T_i)|}{2} \right\}.
\end{aligned}$$

This completes the correctness proof of the algorithm.

Runtime analysis. For each fixed $i \in [q]$, for a feasible tuple at the ith stage $\sigma = (F_i \cup P_i \cup Y_i \cup N_i \cup T_i)$, we have $|F_i \cup P_i \cup Y_i \cup N_i \cup T_i| \leq 6$. Therefore, the number of feasible tuples at the ith stage is at most $|Q_i|^6 \leq n^6$. Now since $q \leq n$, there are at most $n \cdot n^6 = n^7$ pairs (i,σ). Thus there are at most n^7 entries in the DP table. Computing each DP table entry entails computing the minimum of a set of size at most n^6, which can be done in $\mathcal{O}(n^6)$ time. Hence the total running time of the algorithm can be bounded by $\mathcal{O}(n^{13})$.

The above algorithm leads to Theorem 18.

4.8 Exact exponential time algorithm for MDS

In this section, we design an exponential time algorithm that computes the size of a minimum mixed dominating set of an n-vertex graph in time $2^n n^{\mathcal{O}(1)}$. The problem is formally defined as follows.

MINIMUM MIXED DOMINATING SET (MIN MDS)
Input: An undirected graph G.
Question: The size of a minimum mixed dominating set of G.

Let G be an n-vertex graph. In light of Observation 87, in order to find a minimum-sized mds of G, we can restrict the search space to the collection of all special mixed dominating sets of G. Moreover, by Observation 85(iii) if $S = V' \cup E'$ is a special mixed dominating set of G, then for any arbitrary perfect matching E'' of $G[V(E')]$, $V' \cup E''$ is a special mixed dominating set of size $|S|$. Hence, to test whether G has an mds of size ℓ, it is enough to check the existence of a partition (V_1, V_2, V_3) of $V(G)$ such that $|V_1| + \frac{|V_2|}{2} = \ell$ and $V_1 \cup E_2$ is an mds of G, where E_2 is an arbitrary perfect matching in $G[V_2]$. Thus, our search space here is the collection of all partitions of $V(G)$ into at most three parts, which is upper bounded by 3^n. Now by checking if each partition corresponds to a special mds of G, (which can be done in polynomial time because a maximum matching can be found in polynomial time), and by finding the minimum-sized one among the special mixed dominating sets, we get the required result.

Our goal here is to design a faster exponential time algorithm. We now state the following two lemmas that form the basis of our algorithm.

Lemma 119. *Let $S = V' \cup E'$ be a minimum-sized special mixed dominating set of a graph G, where $V' \subseteq V(G)$ and $E' \subseteq E(G)$. Let G' be the subgraph of G induced by $V(G) \setminus V(E')$. Let V'' be the set of isolated vertices in G'. Then $V'' \subseteq V'$ and $V' \setminus V''$ is a minimum-sized vertex cover of G'. Moreover, for any vertex cover U of G', $U \cup V'' \cup E''$ is a mixed dominating set of G, where E'' is any perfect matching in $G[V(E')]$.*

Proof. To see that $V'' \subseteq V'$, consider $w \in V''$, i.e., w is an isolated vertex in G'. Since $V' \cup E'$ dominates w, and E' does not dominate w, we must have $w \in V'$.

Since V'' is the set of isolated vertices in G', to prove that $V' \setminus V''$ is a vertex cover of G', it is enough to prove that V' is a vertex cover of G'. In order to prove that V' is a vertex cover of G', consider $uv \in E(G')$. Then, note that $u,v \in V(G') = V(G) \setminus V(E')$. Then, since $S = V' \cup E'$ dominates the edge uv (and since E' does not dominate uv), either $u \in V'$ or $v \in V'$. This implies that V' and hence $V' \setminus V''$ is a vertex cover of G'. Now we need to prove that $V' \setminus V''$ is indeed a minimum-sized vertex cover of G'.

Claim 120. *Let $\widetilde{V}$ be a vertex cover of G'. Then $\widetilde{S} = \widetilde{V} \cup V'' \cup E'$ is a special mixed dominating set of G.*

Proof. The set E' dominates $V(E')$ and all edges incident on $V(E')$. Since $\widetilde{V}$ is a vertex cover of G', $\widetilde{V}$ dominates all edges of G with both their endpoints in $V(G) \setminus V(E')$. That is $E' \cup \widetilde{V}$ dominates $E(G)$ and $V(E')$. If $v \in V(G) \setminus V(E')$ is isolated in G', then $v \in V''$, and hence $\widetilde{S}$ dominates v. Otherwise, there is an edge, say $uv \in E(G')$, and then either $u \in \widetilde{V}$ or $v \in \widetilde{V}$, and in either case S dominates the vertex v. □ □

The above claim along with the assumption that S is a minimum-sized special mds implies that $V' \setminus V''$ is a minimum-sized vertex cover of G'. By the above claim we also know that $U \cup V'' \cup E'$ is a special mds of G, where U is a vertex cover of G'. Thus, for any perfect matching E'' of $G[V(E')]$, by Observation 85(iii), $U \cup V'' \cup E''$ is a special mds of G for any perfect matching E'' of G. □

Lemma 121. *There is an algorithm that, given an n-vertex graph G, runs in time $2^n n^{\mathcal{O}(1)}$ and outputs a minimum vertex cover of $G[U]$ for every $U \subseteq V(G)$. That is, the algorithm outputs 2^n minimum vertex covers, one for each $G[U]$, where $U \subseteq V(G)$.*

Proof. We design a simple dynamic programming algorithm, where we have a DP table entry for every $U \subseteq V(G)$. That is, the DP table entry $M[U]$ stores the size of a minimum vertex cover of G[U]. Notice that for the graph $G[U]$ and a non-isolated vertex v in $G[U]$, any vertex cover of G contains either v or $N_{G[U]}(v)$. Therefore we can design a recursive formula for computing $M[U]$ as follows.

$$M[U] = \begin{cases} \min\{M[U \setminus \{v\}] + 1, M[U \setminus N_G(v)] + |N_G(v) \cap U|\}, & \text{if } d_{G[U]}(v) \geq 1 \\ M[U \setminus \{v\}], & \text{otherwise} \end{cases}$$

Here $v \in U$ is an arbitrary vertex in U. The base case is $M[\emptyset] = 0$. The above recursive formula can be turned into a DP algorithm over the subsets of the vertices of input graph. Since the size of DP table is 2^n and computation of each entry using the above recursive formula requires only polynomial time, the algorithm will have the desired running time. $\square$

We are now ready to describe our algorithm for Min MDS.

```
Algorithm for Min MDS

Step 1. Run the algorithm in Lemma 121 on $G$ and let $M[X]$ be the value returned by
        the algorithm for $X \subseteq V(G)$.

Step 2. For each $X \subseteq V(G)$, do the following.

        • Let $G_X = G - X$, and let $I_X$ be the set of isolated vertices in $G_X$.

        • If $G[X]$ has a perfect matching, then set $val(X) := |X|/2 + |I_X| + M[X]$;
          otherwise, set $val(X) := \infty$.

Step 3. Return $\min_{X \subseteq V(G)} val(X)$.
```

The correctness of the algorithm follows from Lemma 119. As for the running time of the algorithm, note that each of step 1 and step 2 takes time $2^n n^{\mathcal{O}(1)}$. Hence our algorithm runs in time $2^n n^{\mathcal{O}(1)}$. We thus have the following result.

Theorem 19. MIN MDS *on a n-vertex graph can be solved in time* $2^n n^{\mathcal{O}(1)}$.

4.9 Lower Bounds

Now we prove a kernel lower bound for MDS. That is, we show that unless coNP $\subseteq$ NP/poly, MDS does not admit a polynomial kernel when parameterized by k. We do this by a *polynomial parameter transformation* from an appropriate parameterization of RED BLUE DOMINATING SET (RBDS).

Definition 122 ([15]). *Let P and Q be two parameterized problems. A polynomial parameter transformation (*PPT*, for short) from P to Q is a polynomial time algorithm, which given an instance, say* (x,k) *of P, produces an equivalent instance* (y,k') *of Q such that* $k' \leq p(k)$ *for some polynomial* $p(\cdot)$.

Proposition 123 ([15]). *If there is a* PPT *from P to Q and P has no polynomial kernel, then Q has no polynomial kernel.*

In the RBDS problem, the input is a bipartite graph G with bipartition $R \uplus B$ and a positive integer ℓ, and the question is whether there exists a set $X \subseteq R$ of size at most ℓ, which dominates the set B, i.e., $N(X) = B$. (Such a set X is called a red-blue dominating set (rbds, for short) of G). This problem when parameterized by $|R|$ is the same as SMALL UNIVERSE HITTING SET (see [32]) and thus from [32] we get the following result.

Lemma 124 ([32]). RBDS *parameterized by* $|R|$ *and* ℓ *has no polynomial kernel unless* coNP $\subseteq$ NP/poly.

Theorem 20. MDS *parameterized by the solution size has no polynomial kernel, unless* coNP $\subseteq$ NP/poly.

Proof. The proof is by a polynomial parameter transformation from RBDS parameterized by $|R|$ and ℓ. Given an instance $(G = (R \uplus B, E), \ell)$ of RBDS, we construct an equivalent

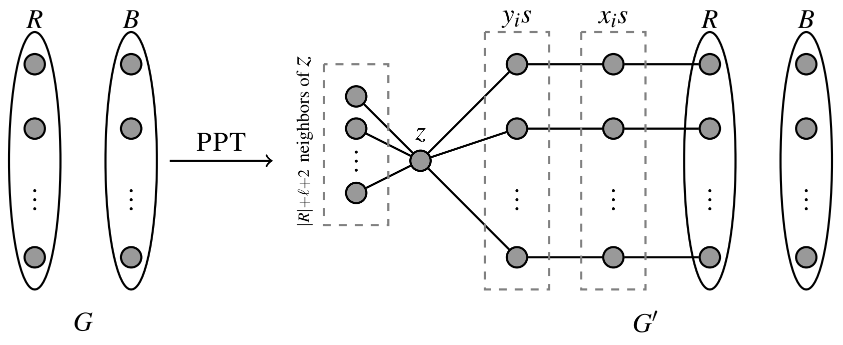


Fig 4.2: PPT from RBDS to MDS

instance $(G', |R|+\ell+1)$ of MDS. If $B \subseteq V(G)$ contains an isolated vertex, then note that G has no rbds (of any size), so take G' to be a $|R|+\ell+2$-sized matching. Otherwise, if B has no isolated vertices, then proceed as follows (see Fig. 4.2).

1. Add all vertices and all edges of G to G', i.e., $V(G') \supseteq V(G)$ and $E(G') \supseteq E(G)$.
2. Corresponding to every vertex $v_i \in R$, add vertices x_i and y_i, and add edges $v_i x_i$ and $x_i y_i$ in G'.
3. Add a vertex z and add edges zy_i, for all y_i.
4. Add $|R|+\ell+2$ additional neighbors to z.

We claim that G has a rbds of size at most ℓ if and only if G' has a mds of size at most $|R|+\ell+1$. Let $X \subseteq R$ be a rbds of G of size at most ℓ. Then $X \cup \{x_i v_i : i = 1, 2, \ldots, |R|\} \cup \{z\}$ is a mds of size at most $|R|+\ell+1$.

Conversely, assume that G does not have any rbds of size at most ℓ. Let S be a *minimum* sized mds of G'. Let S' be the set of all elements $x \in S$ such that x dominates some element(s) of B. Let $S' = S_1 \uplus S_2 \uplus S_3$, where $S_1 = S' \cap B$, $S_2 = S' \cap E(G')$ and $S_3 = S' \cap R$. Construct $S'' \subseteq R$ as follows: (i) for every $v \in S_1$, add a neighbor of v to S'', (ii) for every edge $ww' \in S_2$, where $w \in R$ and $w' \in B$, add w to S'', and (iii) add all vertices of S_3 to S''. Clearly, $|S''| \le |S'|$ and S'' is a rbds of G. By assumption, $|S''| > \ell$ which implies that $|S'| > \ell$.

Thus, S' is a subset of the minimum sized mds S and $|S'| > \ell$. Assume that $z \in S$, otherwise $|S| \geq |R| + \ell + 2$. Note that neither the elements of S' nor z can dominate any of the $|R|$ edges x_iy_i. And at least $|R|$ elements are required to dominate all of them. Therefore,

$$\begin{aligned}|S| &\geq |\{z\}| + |\{\text{the } |R| \text{ elements that dominate edges } x_iy_i\}| + |S'| \\ &> 1 + |R| + \ell.\end{aligned}$$

That is, G' does not have a mds of size at most $|R| + \ell + 1$. Hence, the theorem follows from the given reduction, Proposition 123 and Lemma 124. □ □

Now we present an improved lower bound for MDS when parameterized by the treewidth of the input graph. We can reduce an instance of SET COVER problem $(U, \complement F, \ell)$ to an equivalent instance of RBDS, $(R \uplus B, E, \ell)$, where $R = \complement F$ and $B = U$. Edge set E consists of edges between $F \in R$ and $x \in B$ if and only if $x \in F$. We now apply the reduction given in the proof of Theorem 20 to an instance of RBDS, $(R \uplus B, E, |R| + \ell)$ to get an equivalent instance of MDS, $(G, |R| + \ell + 1)$. Notice that graph G has treewidth at most $1 + |B| = 1 + |U|$. The Set Cover Conjecture [30] states that SET COVER cannot be solved in $\mathcal{O}^\star((2-\varepsilon)^{|U|})$ time for any $\varepsilon > 0$. We thus have the following theorem.

Theorem 21. *Unless the Set Cover Conjecture fails,* MDS *does not admit an algorithm with running time $\mathcal{O}^\star((2-\varepsilon)^{\mathsf{tw}(G)})$.*

4.10 Conclusion

While we studied the complexity of MIXED DOMINATING SET in details, the complexity status of MDS on interval graphs is still unknown, and is worth investigating. Another open question is to improve the size of the kernel for MDS on $K_{d,d}$ -free graphs or prove a matching lower bound.

Chapter 5

Dynamic Parameterization

In this chapter, we study the parameterized complexity of various classical graph-theoretic problems in the dynamic framework setting where the input graph is being updated by a sequence of edge insertions and deletions. The goal is to efficiently maintain a solution under these changes. In the context of parameterized algorithms, we study our problem with respect to the two natural parameters, k (the symmetric difference of the edge sets of the two graphs on n vertices) and r (the symmetric difference of the two solutions). We define the Dynamic Π-Deletion problem which is the dynamic variant of the Π-Deletion problem and then show NP-hardness, fixed-parameter tractability and kernelization results. For specific cases of Dynamic Π-Deletion problems such as DYNAMIC VERTEX COVER and DYNAMIC FEEDBACK VERTEX SET, we describe improved FPTalgorithms and give linear kernels. Specifically, we show that DYNAMIC VERTEX COVER admits algorithms with running times $1.1740^k n^{\mathcal{O}(1)}$ (polynomial space) and $1.1277^k n^{\mathcal{O}(1)}$ (exponential space). Then, we show that DYNAMIC FEEDBACK VERTEX SET admits a randomized algorithm with $1.667^k n^{\mathcal{O}(1)}$ running time. Finally, we consider DYNAMIC CONNECTED VERTEX COVER, DYNAMIC DOMINATING SET and DYNAMIC CONNECTED DOMINATING SET and describe algorithms with $2^k n^{\mathcal{O}(1)}$ running time improving over the known running time bounds for these problems. Additionally, for DYNAMIC DOMINATING SET and DYNAMIC

CONNECTED DOMINATING SET, we show that this is the optimal running time (up to polynomial factors) assuming the Set Cover Conjecture.

5.1 Introduction

Graphs are discrete mathematical structures that represent pairwise relations between objects. Due to their tremendous power to model real world systems, many problems of practical interest can be represented as problems on graphs. Consequently, the design of algorithms on graphs is of major importance in computer science. Applications that employ graph algorithms typically involve large graphs that change over time. A natural goal in this setting is to design algorithms that efficiently maintain a solution under these updates. That is, given a graph G and a solution S, one searches for a solution S' that is as close as possible to S in a graph G' that can be obtained from G by making at most k edits. In this work, we only consider instances where the graphs under consideration have the same vertex set. Formally, a dynamic version of a graph-theoretic problem is a quintuple (G, G', S, k, r) where G and G' are graphs on the same vertex set (of size n). Further, the size of the symmetric difference of the edge sets of G and G' is upper bounded by k and S is a solution (not necessarily optimal) on G. The task is to determine whether there is a solution S' (also not necessarily optimal) on G' such that the symmetric difference of S and S' is at most r.

Dynamic problems have been recently studied in the parameterized complexity framework [2, 33, 49]. Two relevant parameters for dynamic problem instances are the edit parameter k and the distance parameter r. Parameterized complexity results for the dynamic versions of various problems with these parameterizations are known [33] and [2]. In this work, we revisit several classical parameterized problems in the dynamic setting. Table 5.1 summarizes our results along with the running time bounds known for these problems.

For a fixed collection of graphs Π, given a graph G and an integer l, the Π-DELETION

Dynamic Problem	Parameter k	Parameter r
VERTEX COVER	1.0822^k $\mathcal{O}(k)$ kernel	1.2738^r $\mathcal{O}(r^2)$ kernel
CONNECTED VERTEX COVER	4^k [2], 2^k ‡ No $k^{\mathcal{O}(1)}$ size kernel	W[2]-hard [2]
FEEDBACK VERTEX SET	1.6667^k (randomized) $\mathcal{O}(k)$ kernel	3.592^r, 3^r (randomized) $\mathcal{O}(r^2)$ kernel
CONNECTED FEEDBACK VERTEX SET	$2^{\mathcal{O}(k)}$ No $k^{\mathcal{O}(1)}$ size kernel	W[2]-hard
DOMINATING SET	$2^{\mathcal{O}(k^2)}$ [33], 2^k ‡ No $k^{\mathcal{O}(1)}$ size kernel	W[2]-hard [33]
CONNECTED DOMINATING SET	4^k [2], 2^k ‡ No $k^{\mathcal{O}(1)}$ size kernel	W[2]-hard [2]

Table 5.1: Summary of known and new results for different dynamic parameterized problems. All running time bounds are specified by ignoring polynomial factors. ‡ denotes that the running time is optimal under the Set Cover Conjecture.

problem is to determine if G has a set $S \subseteq V(G)$ of vertices with $|S| \leq l$ such that $G - S \in \Pi$. Π-DELETION is an abstraction of various problems in the graph theoretic framework. Examples include VERTEX COVER and FEEDBACK VERTEX SET. Due to a generic result by Lewis and Yannakakis [69], it is known that finding a minimum solution to Π-DELETION is *NP*-hard in general for most choices of Π. Hence, it has been extensively studied in various algorithmic realms. We define the dynamic version of this problem referred to as DYNAMIC Π-DELETION and show NP-hardness, fixed-parameter tractability and kernelization results. Then, for the specific cases of Π-DELETION such as DYNAMIC VERTEX COVER and DYNAMIC FEEDBACK VERTEX SET, we describe improved FPT algorithms with respect to k as the parameter and give linear kernels. Then, for the same parameterization, we describe improved algorithms for DYNAMIC CONNECTED VERTEX COVER, DYNAMIC DOMINATING SET and DYNAMIC CONNECTED DOMINATING SET. For DYNAMIC DOMINATING SET and DYNAMIC CONNECTED DOMINATING SET, we show that this is the optimal running time (up to polynomial factors) assuming the Set Cover Conjecture [24].

5.2 Preliminaries

All graphs considered in this chapter are finite, undirected, unweighted and simple. For a graph G, $V(G)$ and $E(G)$ denote the set of vertices and edges respectively. The size of symmetric difference of two subsets $S, S' \subseteq V(G)$, denoted by $d_v(S, S')$, is defined as the size of the set $(S \setminus S') \cup (S' \setminus S)$. For two graphs G and G' on the same vertex set, $d_e(G, G')$ denotes the size of the symmetric difference of $E(G)$ and $E(G')$.

5.3 Complexity of Dynamic Π-Deletion

A graph property Π is a collection of graphs. Π is said to be (induced) hereditary if for any graph in Π, all of its (induced) subgraphs are in Π too. The membership testing problem for Π is the task of determining if a graph is in Π or not. Let I_n denote the graph on n vertices with no edges and K_n denote the complete graph on n vertices. For most natural choices of Π, the Π-DELETION problem is NP-hard [69] and interesting dichotomy results are known in the parameterized complexity framework [16, 60]. We formally define its dynamic variant referred to as DYNAMIC Π-DELETION as follows.

Dynamic Π-Deletion **Parameter:** k, r
Input: Graphs G, G' on the same vertex set, a set $S \subseteq V(G)$ such that $G - S \in \Pi$ and integers k, r with $d_e(G, G') \leq k$.
Question: Does there exist $S' \subseteq V(G')$ with $d_v(S, S') \leq r$ such that $G' - S' \in \Pi$?

Observe that if Π-DELETION is in NP then so is DYNAMIC Π-DELETION. We are now ready to state our first result.

Theorem 22. *Let Π be a graph property that includes all independent sets or all cliques. Then,* Π-DELETION *reduces to* DYNAMIC Π-DELETION *in polynomial time.*

Proof. Let (H, l) be an instance of Π-DELETION where H is a graph on n vertices. We reduce (H,l) to the instance $(G, G' = H, S = \emptyset, k, r = l)$ of DYNAMIC Π-DELETION as follows. If Π includes all independent sets, then $G = I_n$ and $k = |E(H)|$. Otherwise, $G = K_n$ and $k = \binom{|V(H)|}{2} - |E(H)|$. In both the cases, by the property of Π, $G - S \in \Pi$. Also, the vertex sets of H, G and G' are the same. Then, for a set $S' \subseteq V(H)$, we have $H - S' \in \Pi$ if and only if $G' - S' \in \Pi$ such that $d_v(S, S') = |S'|$. □

As a consequence of Theorem 22, we have the following hardness result.

Corollary 125. *The following results hold for a property* Π *that includes all independent sets or all cliques.*

- *If* Π-DELETION *is* NP*-hard, then* DYNAMIC Π-DELETION *is* NP*-hard.*
- *If* Π-DELETION *parameterized by solution size is fixed-parameter intractable then* DYNAMIC Π-DELETION *parameterized by r is fixed-parameter intractable.*
- *If* Π-DELETION *is* NP*-complete and does not admit a polynomial kernel when parameterized by solution size then* DYNAMIC Π-DELETION *parameterized by r does not admit a polynomial kernel.*

Proof. The NP-hardness and the fixed-parameter intractability results follow straightaway from Theorem 22. If Π-DELETION is NP-complete, then DYNAMIC Π-DELETION reduces to Π-DELETION in polynomial time. Therefore, if DYNAMIC Π-DELETION parameterized by r admits a polynomial kernel, such a kernel can be transformed to a polynomial kernel for Π-DELETION using this reduction and the reduction described in Theorem 22.Thus, the claimed kernelization hardness follows too. □

The following lemma shows that for many choices of Π, to solve DYNAMIC Π-DELETION, it suffices to look for a solution S' that contains S.

Lemma 126. *Let* Π *be an induced hereditary property. If S' is a solution to the* DYNAMIC Π-DELETION *instance (G, G', S, k, r) with $d_v(S, S') = r'$, then there is another solution S'' with $d_v(S, S'') \leq r'$ and $S \subseteq S''$.*

Proof. We have $d_v(S,S') = |S \setminus S'| + |S' \setminus S| = r'$. Let S'' be the set $S \cup S'$. Then, $d_v(S,S'') = |S \setminus S''| + |S'' \setminus S| = |S'' \setminus S| = |S' \setminus S| \leq r'$. Now, as $G' - S' \in \Pi$ and Π is hereditary, it follows that $G' - S'' \in \Pi$ as well. □

Now, we proceed to show certain tractable cases of DYNAMIC Π-DELETION.

Theorem 23. *Let* Π *be an induced hereditary property whose membership testing problem is polynomial-time solvable. Then,* DYNAMIC Π-DELETION *reduces to* Π-DELETION *in polynomial time.*

Proof. Consider an instance (G,G',S,k,r) of DYNAMIC Π-DELETION. The task is to determine if G' has a solution S' with $d_v(S,S') \leq r$. If $G' - S \in \Pi$, then S is the required solution S'. Otherwise, from Lemma 126, assume that the required S' contains S. Let H denote the graph $G' - S$. Then, $H - (S' \setminus S) \in \Pi$. Therefore, for a set $T \subseteq V(H)$, we have $H - T \in \Pi$ if and only if $G' - (S \cup T) \in \Pi$ such that $d_v(S,S') = |T|$. □

Now, the following claim holds.

Corollary 127. *Let* Π *be a hereditary property whose membership testing problem is polynomial-time solvable. If* Π-DELETION *is* FPT *with respect to the solution size l as the parameter, then* DYNAMIC Π-DELETION *is* FPT *with respect to both r and k as parameters.*

Proof. Consider an instance (G,G',S,k,r) of DYNAMIC Π-DELETION. Suppose Π-DELE TION admits an algorithm with $\mathcal{O}(f(l))$ running time. Then, from Theorem 23, there is an algorithm $\mathscr{A}$ solving (G,G',S,k,r) in $\mathcal{O}^*(f(r))$ time. Thus, the problem is FPT when parameterized by r. Let $\tilde{E}$ denote the set $E(G') \setminus E(G)$. Let T be a set of vertices of G' of size at most k that contains at least one endpoint of each edge in $\tilde{E}$. As Π is hereditary and $G - S \in \Pi$, it follows that $G' - (S \cup T) \in \Pi$. Now, if $r \geq k$, then $S \cup T$ is the required solution S'. Otherwise, the algorithm $\mathscr{A}$ solving DYNAMIC Π-DELETION runs in $\mathcal{O}^*(f(k))$ time. Hence, the problem is FPT when parameterized by k. □

Finally, we move on to kernelization results for the problem.

Corollary 128. *Let* Π *be a hereditary property whose membership testing problem is polynomial-time solvable. Suppose* Π-DELETION *parameterized by the solution size* l *admits a kernel with* $p(l)$ *vertices and* $q(l)$ *edges. Then the following results hold.*

- *If* Π *includes all independent sets, then* DYNAMIC Π-DELETION *admits a kernel with* $2p(r) \leq 2p(k)$ *vertices and* $q(r) \leq q(k)$ *edges.*

- *If* Π *includes all cliques, then* DYNAMIC Π-DELETION *admits a kernel with* $2p(r) \leq 2p(k)$ *vertices and* $q(r) + p(r)^2 \leq q(k) + p(k)^2$ *edges.*

Proof. Consider an instance (G, G', S, k, r) of DYNAMIC Π-DELETION. If $G' - S \in \Pi$ or $r \geq k$, the output of the kernelization algorithm is $(K_1, \emptyset, K_1, 0, 0)$ (with constant size) which is a trivial YES instance of DYNAMIC Π-DELETION. Suppose $G' - S \notin \Pi$ and $r < k$. Let (H', r') be the kernelized instance of (H, r), the instance of Π-DELETION obtained from Theorem 23. Then, $(H'', H', \emptyset, |E(H')|, r')$ is the kernelized instance of (G, G', S, k, r) where $H'' = I_{|V(H')|}$ if Π includes all independent sets and $H'' = K_{|V(H')|}$ if Π includes all cliques. Hence, the claimed bounds on the kernel size follow. □

Remark A property Π is called interesting if the number of graphs in Π and the number of graphs not in Π are unbounded. Any induced-hereditary property that is interesting either contains all independent sets or contains all cliques. Thus, all the above results hold for such properties. In particular, the results of this section hold for the dynamic variants of classical problems like VERTEX COVER, FEEDBACK VERTEX SET and ODD CYCLE TRANSVERSAL.

5.4 Dynamic Vertex Cover

A vertex cover is a set of vertices that has at least one endpoint from every edge and DYNAMIC VERTEX COVER is formally defined as follows.

DYNAMIC VERTEX COVER **Parameter:** k, r

Input: Graphs G, G' on the same vertex set, a vertex cover S of G and integers k, r such that $d_e(G, G') \leq k$.

Question: Does there exist a vertex cover S' of G' such that $d_v(S, S') \leq r$?

Clearly, DYNAMIC VERTEX COVER is DYNAMIC Π-DELETION where Π is the set of all independent sets. As VERTEX COVER, the problem of determining if a graph has a vertex cover of size l, is NP-hard, its dynamic version is NP-hard too by Theorem 22. In [2], the authors claim that DYNAMIC VERTEX COVER is W[1]-hard with respect to $k+r$ as the parameter by a reduction from INDEPENDENT SET parameterized by the solution size. However, the reduction is incorrect and the fixed-parameter intractability does not follow. VERTEX COVER parameterized by the solution size l, the problem admits a kernel with at most $2l$ vertices [22] and the current best FPT algorithm runs in $\mathcal{O}^*(1.2738^l)$ time [21]. By Theorem 23 and Corollaries 127 and 128, these results extend to DYNAMIC VERTEX COVER as well. In particular, the following results hold.

- DYNAMIC VERTEX COVER can be solved in $\mathcal{O}^*(1.2738^r)$ time and in $\mathcal{O}^*(1.2738^k)$ time.
- DYNAMIC VERTEX COVER admits a kernel with at most $4r$ vertices and $\mathcal{O}(r^2)$ edges.
- DYNAMIC VERTEX COVER admits a kernel with at most $4k$ vertices and $\mathcal{O}(k^2)$ edges.

We now improve over these results by describing a linear edge kernel and a faster FPT algorithm with respect to k as the parameter. First, we describe the linear kernelization.

Theorem 24. Dynamic Vertex Cover *admits a kernel with at most* $2k$ *vertices and* k *edges.*

Proof. Consider an instance (G,G',S,k,r) of Dynamic Vertex Cover. By Lemma 126, it suffices to search for a solution S' that contains S. As $d_e(G,G') \leq k$, we have $|E(G') \setminus E(G)| \leq k$. Also, edges in $E(G) \setminus E(G')$ do not affect the solution. Let H be the graph with $V(H) = V(E(G') \setminus E(G))$ and $E(H) = E(G') \setminus E(G)$. Then, H has at most $2k$ vertices and k edges. Further, we have that (G,G',S,k,r) is a YES instance of Dynamic Vertex Cover if and only if (H,r) is a YES instance of Vertex Cover. Then, from Corollary 128, it suffices to output a linear kernel of the instance (H,r). We apply the following standard preprocessing on H.

Reduction Rule 5.4.1. *Delete isolated vertices.*

Reduction Rule 5.4.2. *If there is a vertex* v *of degree* 1*, add its neighbour* u *into the solution and decrease* r *by 1. Delete* u *and* v *from the graph.*

Let H' denote the resultant graph on which these rules are no longer applicable and r' denote the budget. As the rules are safe (i.e., they preserve minimum vertex covers), we have the following equivalence: (H,r) is a YES instance of Vertex Cover if and only if (H',r') is a YES instance of Vertex Cover. Then, as the minimum degree of H' is at least 2, we have that $2|E(H')| \geq 2|V(H')|$. As $|E(H')| \leq k$, it follows that $|V(H')| \leq k$. Thus, from Corollary 128 the kernelized instance corresponding to (G,G',S,k,r) is $(I_{|V(H')|},H',\emptyset,k=|E(H')|,r')$. □

Next, we describe an FPT algorithm (faster than $\mathcal{O}^*(1.2738^k)$) for the problem parameterized by k.

Theorem 25. Dynamic Vertex Cover *can be solved in* $\mathcal{O}^*(1.1740^k)$ *time.*

Proof. Consider an instance (G,G',S,k,r) of Dynamic Vertex Cover. By Lemma 126, it suffices to search for a solution S' that contains S. Let H be the graph with $V(H) =$

$V(E(G') \setminus E(G))$ and $E(H) = E(G') \setminus E(G)$. Then, H has at most $2k$ vertices and k edges and it suffices to solve the instance (H, r) of VERTEX COVER. We first apply Reduction Rules 5.4.1 and 5.4.2 on H as long as they are applicable. Then, $|V(H)| \leq |E(H) \leq k$. It is known that a minimum vertex cover of a graph on n vertices can be found in $\mathcal{O}^*(1.2002^n)$ time [88]. Thus, an $\mathcal{O}^*(1.2002^k)$ algorithm follows. We will describe a faster branching algorithm where the measure used to bound the number of nodes of the search tree is the number of edges in H and the leaves of the tree are instances corresponding to the empty graph or a graph with maximum degree at most 2. To this end, we apply the following additional rule exhaustively.

Reduction Rule 5.4.3. *If there exists a triangle on vertices u, v, w such that $deg(u) = 2$, then include v, w into the solution and delete u, v and w from the graph.*

We eliminate all other triangles in the graph by applying following branching strategy.

Branching Rule 5.4.1. *Let u, v, w be vertices of a triangle.*

- *Branch 1: Include vertex u into the solution and delete it from the graph.*
- *Branch 2: Include $N(u)$ into the solution and delete $N[u]$ from the graph.*

As the degree of a vertex in a triangle is at least 3, the measure drops by at least 3 in the first branch and by at least 6 in the second. When this rule is no longer applicable, we have a triangle-free graph. Now, we state our final branching rule.

Branching Rule 5.4.2. *Let u be a vertex of degree at least three.*

- *Branch 1: Include vertex u into the solution and delete it from the graph.*
- *Branch 2: Include $N(u)$ into the solution and delete $N[u]$ from the graph.*

In the first branch, the measure drops by at least 3. As the graph is triangle-free, no two neighbours of u are adjacent. Further, all vertices have degree at least 2. Therefore, the measure drops by at least $2|N(u)| \geq 6$ in the second branch. As no new edges are added to the graph in any rule, the measure never increases after the application of a reduction or branching rule. Further, all reduction rules can be applied in polynomial time. At each branching, we only spend polynomial time to find a vertex to branch on. When k is zero or the maximum degree of the graph is at most 2, finding a minimum vertex cover is polynomial-time solvable. The initial measure is upper bounded k and the worst case branching vector is $(3,6)$. This leads to the recurrence $T(k) \leq T(k-3) + T(k-6)$ whose solution is 1.1740^k. Thus, the algorithm runs in $\mathcal{O}^*(1.1740^k)$ time. □

The treewidth of a graph is a parameter that quantifies the closeness of the graph to a tree (see [25] for the precise definition). If the treewidth of the input graph is upper bounded by tw, then a minimum vertex cover can be obtained in $\mathcal{O}^*(2^{tw})$ time [25]. The following result relates the treewidth of a graph to the number of its vertices and edges.

Lemma 129. [62] *If G is a graph on n vertices and m edges, then the treewidth of G is upper bounded by $\frac{m}{5.769} + \mathcal{O}(\log n)$. Moreover, a tree decomposition of the corresponding width can be constructed in polynomial time.*

Since the graph H on which a minimum vertex cover is desired has at most k edges and k vertices, its treewidth tw is bounded by $\frac{k}{5.769} + \mathcal{O}(\log k)$. Then, we have the following result.

Theorem 26. DYNAMIC VERTEX COVER *can be solved in $\mathcal{O}^*(1.1277^k)$ time.*

Though this algorithm is faster than the branching algorithm described earlier, it requires exponential space while the algorithm in Theorem 25 requires only polynomial space.

5.5 Dynamic Connected Vertex Cover

A connected vertex cover of a graph is a vertex cover that induces a connected subgraph and the parameterized DYNAMIC CONNECTED VERTEX COVER is defined as follows.

DYNAMIC CONNECTED VERTEX COVER **Parameter:** k, r

Input: Graphs G, G' on the same vertex set, a connected vertex cover S of G and integers k, r such that $d_e(G, G') \leq k$.

Question: Does there exist a connected vertex cover S' of G' such that $d_v(S, S') \leq r$?

The problem is NP-complete, W[2]-hard when parameterized by r and admits an $\mathcal{O}^*(4^k)$ algorithm by a reduction to finding a minimum weight Steiner tree [2]. We describe an $\mathcal{O}^*(2^k)$ algorithm by a reduction to finding a group Steiner tree. Given a graph G, an integer p and a family $\mathscr{F}$ of subsets of $V(G)$, the GROUP STEINER TREE problem is the task of determining whether G contains a tree on at most p vertices that contains at least one vertex from each set in $\mathscr{F}$. This problem is known to admit an algorithm with $\mathcal{O}^*(2^{|\mathscr{F}|})$ running time [81]. First, we show a lemma on the property of a solution to an instance of DYNAMIC CONNECTED VERTEX COVER analogous to Lemma 126.

Lemma 130. *Consider an instance (G, G', S, k, r) of* DYNAMIC CONNECTED VERTEX COVER. *If S' is a connected vertex cover of G' with $d_v(S, S') = r'$, then $S' \cup S$ is also a connected vertex cover of G' with $d_v(S, S' \cup S) \leq r'$.*

Proof. Assume G' is connected, otherwise, it is a NO instance. As a set that contains a vertex cover is also a vertex cover, it follows that $T = S' \cup S$ is a vertex cover of G'. As S' is a vertex cover of G', $S \setminus S'$ is an independent set in G'. As G' is connected, every vertex in $S \setminus S'$ is adjacent to some vertex in S'. Then, as $G'[S']$ is connected and $S' \subseteq T$, it follows that $G'[T]$ is connected too. Further, as $T \setminus S = S' \setminus S$ it follows that $d_v(S, T) = |T \setminus S| + |S \setminus T| = |T \setminus S| = |S' \setminus S| \leq r'$. □

Now, we prove the main result of this section.

Theorem 27. DYNAMIC CONNECTED VERTEX COVER *admits an* FPT *algorithm that runs in $\mathcal{O}^*(2^k)$ time.*

Proof. Consider an instance (G, G', S, k, r) of DYNAMIC CONNECTED VERTEX COVER.

By Lemma 130, we can assume that the required solution S' contains S. Observe that $G'[S]$ is not necessarily connected and the edges in G' that are not covered by S are those edges in $E' = (E(G') \setminus E(G)) \cap E(G' - S)$. Now, we show a reduction to finding a group Steiner tree. Contract each connected component of $G'[S]$ to a single vertex. Let H denote the resulting graph and let $X = V(H) \setminus V(G')$. Construct an instance $(H, |X|+r, \mathscr{F})$ of GROUP STEINER TREE where $\mathscr{F} = \{\{u,v\} \mid uv \in E'\} \cup \{\{x\} \mid x \in X\}$. We claim that (G, G', S, k, r) is a YES instance of DYNAMIC CONNECTED VERTEX COVER if and only if $(H, |X|+r, \mathscr{F})$ is a YES instance of GROUP STEINER TREE.

Suppose there is a connected vertex cover S' of G' such that $d_v(S, S') \leq r$ and $S \subseteq S'$. As $G'[S']$ is connected, it follows that $H[X \cup (S' \cap V(G' - S))]$ is also connected. Moreover, as $|S' \cap V(G' - S)| \leq r$, it follows that the spanning tree of $H[X \cup (S' \cap V(G' - S))]$ is of size at most $|X|+r$. Hence $(H, |X|+r, \mathscr{F})$ is a YES instance of GROUP STEINER TREE. Conversely, suppose $(H, |X|+r, \mathscr{F})$ is a YES instance of GROUP STEINER TREE. Let T denote the solution tree of H. Then, $X \subseteq V(T)$ and $|V(T) \setminus X| = |V(T) \cap V(G' - S)| \leq r$. Define $S' = S \cup (V(T) \cap V(G' - S))$. The size of S' is at most $|S|+r$. Further, $G'[S']$ is connected as S' is obtained from the vertices of T. Also, for every edge in E', T contains at least one of its endpoints. Thus, S' is the desired connected vertex cover of G'. As the sum of the number of connected components of $G'[S]$ and the size of E' is upper bounded by $k+1$, it follows that $|\mathscr{F}| \leq k+1$. Thus, the GROUP STEINER TREE algorithm of [81] runs in $\mathscr{O}^*(2^k)$ time. □

5.6 Dynamic Feedback Vertex Set

A feedback vertex set is a set of vertices whose deletion results in an acyclic graph and DYNAMIC FEEDBACK VERTEX SET is defined as follows.

DYNAMIC FEEDBACK VERTEX SET **Parameter:** k, r

Input: Graphs G, G' on the same vertex set, a feedback vertex set X of G and integers k, r such that $d_e(G, G') \leq k$.

Question: Does there exist a feedback vertex set X' of G' such that $d_v(X, X') \leq r$?

Clearly, DYNAMIC FEEDBACK VERTEX SET is DYNAMIC Π-DELETION where Π is the set of all forests. As FEEDBACK VERTEX SET, the problem of determining if a graph on n vertices has a feedback vertex set of at most l vertices, is NP-hard, its dynamic variant is NP-hard too by Theorem 22. FEEDBACK VERTEX SET is known to admit an $\mathcal{O}^*(3.592^l)$ algorithm [63] and a kernel with $\mathcal{O}(l^2)$ vertices [86]. Also, a randomized algorithm that solves the problem in $\mathcal{O}^*(3^l)$ time is known from [28]. By Theorem 23 and Corollaries 127 and 128, all these results extend to DYNAMIC FEEDBACK VERTEX SET. In particular, the following results hold.

- DYNAMIC FEEDBACK VERTEX SET can be solved in $\mathcal{O}^*(3.592^r)$ time and in $\mathcal{O}^*(3.592^k)$ time.
- DYNAMIC FEEDBACK VERTEX SET admits randomized algorithms with $\mathcal{O}^*(3^r)$ and $\mathcal{O}^*(3^k)$ running times.
- DYNAMIC FEEDBACK VERTEX SET admits an $\mathcal{O}(r^2)$ kernel and an $\mathcal{O}(k^2)$ kernel.

We now improve these bounds by describing a linear kernel and a faster randomized FPT algorithm with respect to k as the parameter. First, we describe the linear kernelization.

Theorem 28. DYNAMIC FEEDBACK VERTEX SET *admits a kernel with at most* $4k$ *vertices and* $3k$ *edges.*

Proof. Consider an instance (G, G', X, k, r) of DYNAMIC FEEDBACK VERTEX SET. Observe that if G' is obtained from G by only deleting edges, then X is feedback vertex set of G' too. Also, edges in $E(G') \setminus E(G)$ that have an endpoint in X do not affect the solution. Moreover, from Lemma 126, it suffices to search for a feedback vertex set of G'

that contains X. Let H be the subgraph of G' induced on $V(G') \setminus X$. From Theorem 23, we have that (G, G', X, k, r) is a YES instance of Dynamic Feedback Vertex Set if and only if (H, r) is a YES instance of Feedback Vertex Set. From Corollary 128, it suffices to output a linear kernel of the instance (H, r).

We primarily exploit the fact that H is obtained by adding at most k edges to a forest. This implies that $|E(H)| \leq |V(H)| + k - 1$. Let $\tilde{E}$ be the set of edges in G' whose both endpoints are in $V(G) \setminus X$ and $U = V(\tilde{E})$. We apply the following reduction rule to $G - X$. Note that $G - X$ is a subgraph of H as $E(H) = E(G - X) \cup \tilde{E}$.

Reduction Rule 5.6.1. *If there is a vertex v of degree at most 1 such that $v \notin U$, then delete v from the graph.*

This rule is safe as v has degree at most 1 in H too and no minimal feedback vertex set of H contains it. As the number of vertices with degree at least 3 is upper bounded by the number of leaves in a forest, we have the following claim on the resulting graph G'' on which this rule is not applicable.

Observation 131. *The number of vertices of degree at least 3 is at most $2k$.*

Consider the graph H'' obtained from G'' by adding $\tilde{E}$. We once again delete vertices of degree at most 1 (if any) and then apply following reduction rule exhaustively.

Reduction Rule 5.6.2. *If there is a vertex v of degree 2, then delete v and add an edge between its two neighbours.*

Once again this rule is safe as any minimal feedback vertex set of H'' that contains v can be modified into another minimal feedback vertex set of at least the same size that does not contain v. Note that the application of Reduction Rules 5.6.1 and 5.6.2 ensure that $|E(H'')| \leq |V(H'')| + k - 1$ is satisfied. The following properties now hold for H'' on which neither of the above reduction rules are applicable.

Observation 132. *The minimum degree of H'' is at least 3 and $|E(H'')| \leq |V(H'')| + k - 1$.*

This implies that $1.5|V(H'')| \leq |E(H'')| \leq |V(H'')| + k - 1$ and hence $|V(H'')| \leq 2k-2$, $|E(H'')| \leq 3k-3$. As the reductions rules are safe (i.e. they preserve minimum feedback vertex sets), we have the following equivalence: (H,r) is a YES instance of FEEDBACK VERTEX SET if and only if (H'',r) is a YES instance of FEEDBACK VERTEX SET. Thus, from Corollary 128, the kernelized instance corresponding to (G,G',S,k,r) is $(I_{|V(H'')|}, H'', \emptyset, k = |E(H'')|, r)$. $\square$

Next, we proceed to describe an improved FPT algorithm for the problem. If the treewidth of the input graph is upper bounded by tw, then a randomized $\mathcal{O}^*(3^{tw})$ time algorithm is known [28] that computes a minimum feedback vertex set. Further, for finding a minimum feedback vertex set, there is a deterministic algorithm with running time $\mathcal{O}(1.7216^n)$ and a randomized algorithm running in $\mathcal{O}(1.6667^n)$ time [39]. The following result relates the treewidth of a graph to the number of its vertices and edges.

Lemma 133. [40] *For any $\varepsilon > 0$, there exists an integer n_ε such that for every connected graph G on n vertices and m edges with $n > n_\varepsilon$ and $1.5n \leq m \leq 2n$, the treewidth of G is upper bounded by $\frac{m-n}{3} + \varepsilon n$. Moreover, a tree decomposition of the corresponding width can be constructed in polynomial time.*

This theorem along with the described linear kernelization leads to the following result.

Theorem 29. DYNAMIC FEEDBACK VERTEX SET *admits a randomized algorithm running in $\mathcal{O}^*(1.6667^k)$ time.*

Proof. Consider an instance (G,G',X,k,r) of DYNAMIC FEEDBACK VERTEX SET. Let (H'',r) be the corresponding instance of FEEDBACK VERTEX SET obtained from the linear kernelization of Theorem 28. That is, H'' is a graph (not necessarily simple) on n vertices and m edges such that $m \leq n+k-1$ and $n \leq 2k-2$. Further, every vertex of H'' has degree at least 3 and hence $m \geq 1.5n$. If $m > 2n$, then as $m \leq n+k-1$, we have $n < k-1$. Then, a minimum feedback vertex set of H'' can be obtained in $\mathcal{O}(1.6667^k)$ using the randomized exact exponential-time algorithm described in [39]. Otherwise, $1.5n \leq m \leq 2n$. Let ε be a constant (to be chosen subsequently). Then, let n_ε be the integer obtained from Theorem 133 satisfying the required properties. If $n \leq n_\varepsilon$, then a minimum feedback vertex set of H'' can be obtained in constant time as n_ε is a constant depending only on ε. Otherwise, the treewidth of H'' is at most $t = \frac{m-n}{3} + \varepsilon n = \frac{m}{3} + n(\varepsilon - \frac{1}{3})$. Then, using the randomized algorithm described in [28], a minimum feedback vertex set of H'' can be obtained in $\mathcal{O}^*(3^t)$ time. Now, by choosing ε to be a sufficiently small constant, t can be made arbitrarily close to $\frac{m-n}{3}$. For instance, if $\varepsilon = 10^{-10}$, then t is $.\overline{3}m - .333333333\overline{2}3n$. As $\frac{m-n}{3} \leq \frac{n+k-1-n}{3} = \frac{k}{3}$, the algorithm in [28] runs in $\mathcal{O}^*(1.443^k)$ time. $\square$

5.7 Dynamic Dominating Set

A dominating set of a graph G is a set D of vertices such that $D \cap N[v] \neq \emptyset$ for every $v \in V(G)$. A set $S \subseteq V(G)$ is said to dominate another set $T \subseteq V(G)$ if $T \subseteq N[S]$. The parameterized DYNAMIC DOMINATING SET is formally defined as follows.

DYNAMIC DOMINATING SET **Parameter:** k, r

Input: Graphs G, G', a dominating set D of G and integers k, r such that $d_e(G, G') \leq k$.

Question: Does there exist a dominating set D' of G' such that $d_v(D, D') \leq r$?

The problem is NP-complete and W[2]-hard when parameterized by r [33]. Also, it is FPT when parameterized by k but admits no polynomial kernel unless NP $\subseteq$ coNP/poly/poly. We describe a faster FPT algorithm for this parameterization. First, we show that it suffices to look for a dominating set with a specific property.

Lemma 134. *Consider an instance (G, G', D, k, r) of* DYNAMIC DOMINATING SET. *If D' is a dominating set of G' with $d_v(D, D') = r'$, then $D' \cup D$ is also a dominating set of G' with $d_v(D, D' \cup D) \leq r'$.*

Proof. As a set that contains a dominating set is also a dominating set, it follows that

$D'' = D' \cup D$ is a dominating set of G'. Further, $d_v(D, D'') = |D'' \setminus D| + |D \setminus D''| = |D'' \setminus D| = |D' \setminus D| \leq r'$. $\square$

Now, we solve DYNAMIC DOMINATING SET by reducing it to an instance of SET COVER. In the SET COVER problem, we are given a family $\mathscr{F}$ of subsets of a universe U and a positive integer ℓ. The problem is to determine whether there exists a sub family $\mathscr{F}' \subseteq \mathscr{F}$ of size at most ℓ such that $U = \bigcup_{X \in \mathscr{F}'} X$.

Theorem 30. DYNAMIC DOMINATING SET *admits an* FPT *algorithm that runs in* $\mathcal{O}^*(2^k)$ *time.*

Proof. Consider an instance (G, G', D, k, r) of DYNAMIC DOMINATING SET. If G' is obtained from G by only adding edges, then D is dominating set of G'. The only kind of edge deletions that could possibly affect the solution are those that have one endpoint in D and the other endpoint in $V(G') \setminus D$. Further, as $d_e(G, G') \leq k$, $|V(G') \setminus N_{G'}[D]| \leq k$. That is, there are at most k vertices in G' that are not dominated by D. Let H be the subgraph of G' induced on $V(G') \setminus D$. Partition $V(H)$ into two sets $C = N_{G'}(D)$ and $B = V(H') \setminus C$ where $|B| \leq k$.

We claim that (G, G', D, k, r) is a YES instance of DYNAMIC DOMINATING SET if and only if there exists a set $P \subseteq V(H)$ of cardinality at most r such that $B \subseteq N_H[P]$. If there is a set P of size at most r in $V(H)$ that dominates B, then $D' = D \cup P$ is a dominating set of G' with $d_v(D, D') \leq r$. Hence, (G, G', D, k, r) is a YES instance of DYNAMIC DOMINATING SET. Conversely, suppose there is a dominating set D' of G' with $d_v(D, D') \leq r$. Define D'' as $D' \setminus D$. Notice that $|D''| \leq r$. By construction of H, B is not dominated by D and hence $B \subseteq N_H[D'']$. This implies that D'' is the required set of vertices of H that dominates B.

The problem now reduces to finding a set of at most r vertices from $B \cup C$ that dominates B in H. We construct an instance of SET COVER with $U = B$, $\mathscr{F} = \{N_H(u) \cap B \mid u \in C\} \cup \{N_H[w] \cap B \mid w \in B\}$ and $\ell = r$. Then, there exists a set P of size at most r in H which dominates B if and only if $(U, \mathscr{F}, \ell)$ is a YES instance of SET COVER. A set $X \in \mathscr{F}$

is said to be associated with a vertex v in C if $X = N_H(v) \cap B$ or with a vertex v in B if $X = N_H[v] \cap B$. If there exists a set P with desired property, then every vertex w in B is contained in open or closed neighbourhood of some vertex in P. Consider the subfamily $\mathscr{F}'$ of $\mathscr{F}$ that are associated with vertices in P. Every element of U is contained in at least one of these sets. Thus, $\mathscr{F}'$ is the required set cover. Conversely, if there exists a set cover $\mathscr{F}'$ of size at most $\ell = r$, then let P' be the set of vertices which are associated with sets in $\mathscr{F}'$. Then, $|P'| = |\mathscr{F}'| \leq r$ and every vertex in B is either in P' or is adjacent to some vertex in P'. Hence, P' is the desired set.

As any instance $(U, \mathscr{F}, \ell)$ of SET COVER can be solved in $\mathscr{O}^*(2^{|U|})$ [41], the claimed running time bound follows. □

Finally, we show a lower bound on the running time of an algorithm that solves DYNAMIC DOMINATING SET assuming the Set Cover Conjecture which states that SET COVER cannot be solved in $\mathscr{O}^*((2-\varepsilon)^{|U|})$ for any $\varepsilon > 0$ [24]. We do so by a reduction from SET COVER to DYNAMIC DOMINATING SET.

Theorem 31. DYNAMIC DOMINATING SET *does not admit an algorithm with* $\mathscr{O}^*((2-\varepsilon)^k)$ *running time for any* $\varepsilon > 0$ *assuming the Set Cover Conjecture.*

Proof. Consider an instance $(U, \mathscr{F}, \ell)$ of SET COVER where $U = \{u_1, \cdots, u_n\}$ and $\mathscr{F} = \{S_1, \cdots, S_m\}$. Without loss of generality, assume that every u_i is in at least one set S_j. Let G be the graph with vertex set $U \cup V \cup \{x\}$ where $U = \{u_1, \cdots, u_n\}$ and $V = \{s_1, \cdots, s_m\}$. The set V is a clique and the set U is an independent set in G. Further, a vertex u_i is adjacent to s_j if and only if $u_i \in S_j$ and x is adjacent to every vertex in $U \cup V$. Clearly, $D = \{x\}$ is a dominating set of G. Let G' be the graph obtained from G by deleting edges between x and U. We claim that $(U, \mathscr{F}, \ell)$ is a YES instance of SET COVER if and only if $(G, G', D = \{x\}, k = n, r = \ell)$ is a YES instance of DYNAMIC DOMINATING SET. Suppose $\mathscr{F}'$ is a set cover of size at most ℓ. Then, $D' = D \cup \{s_i \mid S_i \in \mathscr{F}'\}$ is a dominating set of G' with $d_v(D, D') \leq \ell$. Conversely, suppose G' has a dominating set D' with $d_v(D, D') \leq \ell$.

From Lemma 134, assume that $D \subseteq D'$ and so $|D' \setminus D| \leq \ell$. For every vertex $u \in U \cap D'$, replace u by one of its neighbours in V. The resultant dominating set D'' contains D and satisfies $D'' \setminus D \subseteq V$. Now, $\{S_i \in \mathscr{F} \mid v_i \in D'' \cap V\}$ is a set cover of size at most ℓ. This leads to the claimed lower bound under the Set Cover Conjecture. $\square$

5.8 Dynamic Connected Dominating Set

A connected dominating set of a graph is a dominating set that induces a connected graph. The parameterized DYNAMIC CONNECTED DOMINATING SET is formally defined as follows.

DYNAMIC CONNECTED DOMINATING SET **Parameter:** k, r

Input: Graphs G, G' on the same vertex set, a connected dominating set D of G and integers k, r such that $d_e(G, G') \leq k$.

Question: Does there exist a connected dominating set D' of G' such that $d_v(D, D') \leq r$?

The problem is NP-complete and admits an $\mathcal{O}^*(4^k)$ algorithm by a reduction to finding a minimum weight Steiner tree [2]. We now show that it admits an $\mathcal{O}^*(2^k)$ algorithm by a reduction to finding a group Steiner tree. Analogous to the problems considered earlier, we first prove a property on the required solution.

Lemma 135. *Consider an instance (G, G', D, k, r) of* DYNAMIC CONNECTED DOMINATING SET. *If D' is a connected dominating set of G' with $d_v(D, D') = r'$, then $D' \cup D$ is also a connected dominating set of G' with $d_v(D, D' \cup D) \leq r'$.*

Proof. Assume G' is connected, otherwise, it is a NO instance. As a set that contains a dominating set is also a dominating set, it follows that $D'' = D' \cup D$ is a dominating set of G'. Now, $D'' \setminus D$ is $D' \setminus D$. As D' is a dominating set of G', every vertex in $D \setminus D'$ is adjacent to some vertex in D'. Then, as $G'[D']$ is connected and $D' \subseteq D''$, it follows that $G'[D'']$ is connected too. Further, $d_v(D, D'') = |D'' \setminus D| + |D \setminus D''| = |D'' \setminus D| = |D' \setminus D| \leq r' \leq r$. $\square$

Now, we describe an algorithm by reducing the problem to finding a Group Steiner tree.

Theorem 32. DYNAMIC CONNECTED DOMINATING SET *admits an* FPT *algorithm that runs in* $\mathcal{O}^*(2^k)$ *time.*

Proof. Consider an instance (G, G', D, k, r) of DYNAMIC CONNECTED DOMINATING SET. Assume G' is connected, otherwise, it is a NO instance. Also, edges in $E(G') \setminus E(G)$ do not affect the solution. Partition $V(G') \setminus D$ into two sets $C = N_{G'}(D)$ and $B = V(G') \setminus C$. Contract each connected component of $G'[D]$ to a single vertex. Let H denote the resulting graph and let $X = V(H) \setminus V(G')$. Construct an instance $(H, |X|+r, \mathscr{F})$ of GROUP STEINER TREE where $\mathscr{F} = \{N_{G'}[v] \mid v \in B\} \cup \{\{x\} \mid x \in X\}$. We claim that (G, G', D, k, r) is a YES instance of DYNAMIC CONNECTED DOMINATING SET if and only if $(H, |X|+r, \mathscr{F})$ is a YES instance of GROUP STEINER TREE.

Suppose there exists a connected dominating set D' of G' such that $d_v(D, D') \leq r$ and $D \subseteq D'$. For every vertex u in B, there is a vertex x in $D' \cap (B \cup C)$ that is adjacent to u. As $G'[D']$ is connected, it follows that $H[X \cup (D' \cap (B \cup C))]$ is also connected. Moreover, as $|D' \cap (C \cup B)| \leq |D'| - |D| \leq r$, it follows that the spanning tree of $H[X \cup (D' \cap (C \cup B))]$ is of size at most $|X|+r$. Hence $(H, |X|+r, \mathscr{F})$ is a YES instance of GROUP STEINER TREE. Suppose $(H, |X|+r, \mathscr{F})$ is a YES instance of GROUP STEINER TREE. Let T denote the solution tree of H. Then, $X \subseteq V(T)$ and $|V(T) \setminus X| = |V(T) \cap (C \cup B)| \leq r$. Define $D' = D \cup (V(T) \cap (B \cup C))$. The size of D' is at most $|D|+r$. Now, $G'[D']$ is connected as D' is obtained from the vertices of T. Also, for every vertex u in B, T contains at least one vertex in $N_{G'}[v]$. Thus, D' is the desired connected dominating set of G'.

As $E(G) \setminus E(G')$, the sum of the number of connected components of $G'[D]$ and the size of B is upper bounded by $k+1$. That is, $|\mathscr{F}| \leq k+1$ and the GROUP STEINER TREE algorithm of [81] runs in $\mathcal{O}^*(2^k)$ time. □

Finally, by a reduction from SET COVER to DYNAMIC CONNECTED DOMINATING SET, we show the following result.

Theorem 33. DYNAMIC CONNECTED DOMINATING SET *does not admit an algorithm with* $\mathcal{O}^*((2-\varepsilon)^k)$ *running time for any* $\varepsilon > 0$ *assuming the Set Cover Conjecture.*

Proof. We observe that the reduction described in Theorem 31 produces instances of DYNAMIC CONNECTED DOMINATING SET. Thus, the claimed lower bound holds assuming the Set Cover Conjecture. □

5.9 Conclusion

We described FPT algorithms for the dynamic variants of several classical parameterized problems with respect to the edit parameter. The role of structural parameters like treewidth and pathwidth in this setting remains to be explored. Also, further exploration of the contrast between the parameterized complexity of a problem and its dynamic version is an interesting direction of research.

Chapter 6

Conclusion

In our thesis we did explore the path like clique decomposition of proper interval graphs to design algorithms for PACKING and COVERING problems. It would be interesting to see if we can find some similar properties to exploit in interval graphs and solve CYCLE PACKING on them (complexity of this problem is still open). Many PACKING problems that do not admit polynomial kernels suddenly become interesting under a slighty loose definition of kernelization known as *lossy kernelization* [74]. Basically in lossy kernelization we look for a polynomial kernel while sacrificing some accuracy/exactness in our answer. CYCLE PACKING is one of the first problems to be shown to have a lossy polynomial kernel. We believe that we can also obtain lossy polynomial kernels for many other PACKING problems such as HOLE PACKING etc.. There are also PACKING problems such as (A,ℓ)-PATH PACKING, TREE PACKING which are only recently studied in the parameterized framework and many questions on them are still open and yet to be solved.

The complexity of MIXED DOMINATING SET on interval graphs is still open. It also makes sense to close the gap between the lower bounds and our various FPTalgorithms for MIXED DOMINATING SET. Studying DYNAMIC PARAMETERIZED ALGORITHMS from the view of structural parameters such as treewidth, pathwidth, treedepth etc. might be an interesting prospect as it is yet to be explored.

References

[1] F. N. Abu-Khzam. An Improved Kernelization Algorithm for r-Set Packing. *Inf. Process. Lett.*, 110(16):621–624, 2010.

[2] F. N. Abu-Khzam, J. Egan, M. R. Fellows, F. A. Rosamond, and P. Shaw. On the parameterized complexity of dynamic problems. *Theoretical Computer Science*, 607, Part 3:426 – 434, 2015.

[3] Yousef Alavi, M. Behzad, Linda M. Lesniak-Foster, and E. A. Nordhaus. Total matchings and total coverings of graphs. *Journal of Graph Theory*, 1(2):135–140, 1977.

[4] Yousef Alavi, Jiuqiang Liu, Jianfang Wang, and Zhongfu Zhang. On total covers of graphs. *Discrete Mathematics*, 100(1-3):229–233, 1992.

[5] N. Alon, R. Yuster, and U. Zwick. Color-Coding. *Journal of the ACM*, 42(4):844–856, 1995.

[6] J. Bang-Jensen and G. Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer-Verlag London, 2009.

[7] A. A. Bertossi. Finding Hamiltonian Circuits in Proper Interval Graphs. *Information Processing Letters*, 17(2):97 – 101, 1983.

[8] S. Bessy, M. Bougeret, and J. Thiebaut. Triangle Packing in (Sparse) Tournaments: Approximation and Kernelization. In *25th Annual European Symp. on Algorithms (ESA 2017)*, volume 87, pages 14:1–14:13, 2017.

[9] S. Bessy, F. V. Fomin, S. Gaspers, C. Paul, A. Perez, S. Saurabh, and S. Thomassé. Kernels for Feedback Arc Set in Tournaments. *J. Comput. Syst. Sci*, 77(6):1071–1078, 2011.

[10] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets möbius: fast subset convolution. In *STOC*, pages 67–74, 2007.

[11] H L Bodlaender. On disjoint cycles. *Int. J. Found. Comput. Sci.*, 5(1):59–68, 1994.

[12] H. L. Bodlaender and B. M. P. Jansen. Vertex Cover Kernelization Revisited: Upper and Lower Bounds for a Refined Parameter. *Theory of Computing Systems*, 63(2):263–299, 2013.

[13] H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Kernel Bounds for Path and Cycle Problems. *Theor. Comput. Sci.*, 511:117–136, 2013.

[14] H. L. Bodlaender, S. Thomassé, and A. Yeo. Kernel Bounds for Disjoint Cycles and Disjoint Paths. *Theor. Comput. Sci.*, 412(35):4570–4578, 2011.

[15] Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theor. Comput. Sci.*, 412(35):4570–4578, 2011.

[16] L. Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171 – 176, 1996.

[17] L. Cai. Parameterized Complexity of Vertex Colouring. *Discrete Applied Mathematics*, 127(3):415 – 429, 2003.

[18] Y. Cao. Unit Interval Editing is Fixed-Parameter Tractable. *Information and Computation*, 253, Part 1:109–126, 2017.

[19] S. Chaplick, F. V. Fomin, P. A. Golovach, D. Knop, and P. Zeman. Kernelization of graph hamiltonicity: Proper h-graphs. In *Algorithms and Data Structures*, pages 296–310. Springer International Publishing, 2019.

[20] C. Chen, C. Chang, and G. J. Chang. Proper Interval Graphs and the Guard Problem. *Discrete Mathematics*, 170(1):223 – 230, 1997.

[21] J. Chen, I. A. Kanj, and W. Jia. Vertex cover: further observations and further improvements. *Journal of Algorithms*, 41(2):280–301, 2001.

[22] J. Chen, I. A. Kanj, and G. Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40-42):3736–3756, 2010.

[23] M. Chudnovsky, P. Seymour, and B. Sullivan. Cycles in Dense Digraphs. *Combinatorica*, 28(1):1–18, 2008.

[24] M. Cygan, H. Dell, D. Lokshtanov, D. Marx, J. Nederlof, Y. Okamoto, R. Paturi, S. Saurabh, and M. Wahlstrom. On problems as hard as CNF-SAT. In *Proceedings of the 2012 IEEE Conference on Computational Complexity (CCC)*, CCC '12, pages 74–84. IEEE Computer Society, 2012.

[25] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.

[26] M Cygan, F V Fomin, L Kowalik, D Lokshtanov, D Marx, M Pilipczuk, M Pilipczuk, and S Saurabh. *Parameterized algorithms*. Springer, 2015.

[27] M Cygan, J Nederlof, M Pilipczuk, M Pilipczuk, and J O Rooij, J M M Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *FOCS*, pages 150–159, 2011.

[28] M. Cygan, J. Nederlof, M. Pilipczuk, M. Pilipczuk, J. M. M. v. Rooij, and J. O. Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single

exponential time. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 150–159, 2011.

[29] M. Cygan, J. Nederlof, M. Pilipczuk, M. Pilipczuk, J. M. M. van Rooij, and J. O. Wojtaszczyk. Solving Connectivity Problems Parameterized by Treewidth in Single Exponential Time. In *IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 150–159, 2011.

[30] Marek Cygan, Holger Dell, Daniel Lokshtanov, Dániel Marx, Jesper Nederlof, Yoshio Okamoto, Ramamohan Paturi, Saket Saurabh, and Magnus Wahlström. On problems as hard as CNF-SAT. *ACM Trans. Algorithms*, 12(3):41:1–41:24, 2016.

[31] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[32] Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Kernelization lower bounds through colors and ids. *ACM Trans. Algorithms*, 11(2):13:1–13:20, 2014.

[33] R.G. Downey, J. Egan, M.R. Fellows, F.A. Rosamond, and P. Shaw. Dynamic dominating set and turbo-charging greedy heuristics. *J. Tsinghua Sci. Technol*, 19(4):329–337, 2014.

[34] Louis Dublois, Michael Lampis, and Vangelis Th. Paschos. New algorithms for mixed dominating set. *CoRR*, abs/1911.08964, 2019. URL: `http://arxiv.org/abs/1911.08964`, `arXiv:1911.08964`.

[35] Paul Erdős and Amram Meir. On total matching numbers and total covering numbers of complementary graphs. *Discrete Mathematics*, 19(3):229–233, 1977.

[36] S. Even, A. Itai, and A. Shamir. On the Complexity of Timetable and Multicommodity Flow Problems. *SIAM J. Comput.*, 5(4):691–703, 1976.

[37] M. R. Fellows, D. Lokshtanov, N. Misra, M. Mnich, F. A. Rosamond, and S. Saurabh. The Complexity Ecology of Parameters: An Illustration Using Bounded Max Leaf Number. *Theory of Computing Systems*, 45(4):822–848, 2009.

[38] Henning Fernau. On parameterized enumeration. In *COCOON*, volume 2387 of *Lecture Notes in Computer Science*, pages 564–573. Springer, 2002.

[39] F. V. Fomin, S. Gaspers, D. Lokshtanov, and S. Saurabh. Exact algorithms via monotone local search. In *Proceedings of the* 48^{th} *Annual ACM Symposium on Theory of Computing*, STOC '16. ACM, 2016.

[40] F. V. Fomin, S. Gaspers, S. Saurabh, and A. A. Stepanov. On two techniques of combining branching and treewidth. *Algorithmica*, 54(2):181–207, 2009.

[41] F. V. Fomin, D. Kratsch, and G. J. Woeginger. Exact (exponential) algorithms for the dominating set problem. In *Graph-Theoretic Concepts in Computer Science*, pages 245–256. Springer, 2004.

[42] S. Fortune, J. Hopcroft, and J. Wyllie. The Directed Subgraph Homeomorphism Problem. *Theor. Comput. Sci.*, 10(2):111–121, 1980.

[43] M R Garey and D S Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. W.H. Freeman, New York, 1979.

[44] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs, Second Edition*. Elsevier Science B.V., 2004.

[45] M. C. Golumbic. *Algorithmic Graph Theory for Perfect Graphs*. Springer, 2004.

[46] V. Guruswami, C. Pandu Rangan, M. S. Chang, G. J. Chang, and C. K. Wong. The K_r-packing Problem. *Computing*, 66(1):79–89, 2001.

[47] G. Gutin, E. J. Kim, M. Lampis, and V. Mitsou. Vertex Cover Problem Parameterized Above and Below Tight Bounds. *Theory of Computing Systems*, 48(2):402–410, 2011.

[48] G. Gutin and A. Yeo. Constraint Satisfaction Problems Parameterized above or below Tight Bounds: A Survey. In *The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, pages 257–286, 2012.

[49] S. Hartung and R. Niedermeier. Incremental list coloring of graphs, parameterized by conservation. *Theoretical Computer Science*, 494:86–98, 2013.

[50] Pooya Hatami. An approximation algorithm for the total covering problem. *Discussiones Mathematicae Graph Theory*, 27(3):553–558, 2007.

[51] Teresa W Haynes, Stephen Hedetniemi, and Peter Slater. *Fundamentals of domination in graphs*. CRC Press, 1998.

[52] Sandra M Hedetniemi, Stephen T Hedetniemi, Renu Laskar, Alice McRae, and Aniket Majumdar. Domination, independence and irredundance in total graphs: a brief survey. In *Proceedings of the 7th Quadrennial International Conference on the Theory and Applications of Graphs*, volume 2, pages 671–683, 1995.

[53] L. Ibarra. A Simple Algorithm to Find Hamiltonian Cycles in Proper Interval Graphs. *Information Processing Letters*, 109(18):1105 – 1108, 2009.

[54] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity. *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

[55] B. M. P. Jansen. *The Power of Data Reduction: Kernels for Fundamental Graph Problems*. PhD thesis, Utrecht University, The Netherlands, 2013.

[56] B. M. P. Jansen, M. R. Fellows, and F. A. Rosamond. Towards fully Multivariate Algorithmics: Parameter Ecology and the Deconstruction of Computational Complexity. *European Journal of Combinatorics*, 34(3):541–566, 2013.

[57] B. M. P. Jansen, V. Raman, and M. Vatshelle. Parameter Ecology for Feedback Vertex Set. *Tsinghua Science and Technology*, 19(4):387–409, 2014.

[58] Y. Ke, Y. Cao, X. Ouyang, and J. Wang. Unit Interval Vertex Deletion: Fewer Vertices are Relevant, 2016. `arXiv:arXiv:1607.01162`.

[59] Yuping Ke, Yixin Cao, Xiating Ouyang, and Jianxin Wang. Unit interval vertex deletion: Fewer vertices are relevant. *arXiv preprint arXiv:1607.01162*, 2016.

[60] S. Khot and V. Raman. Parameterized complexity of finding subgraphs with hereditary properties. *Theoretical Computer Science*, 289(2):997–1008, 2002.

[61] Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994.

[62] J. Kneis, D. Mölle, S. Richter, and P. Rossmanith. A bound on the pathwidth of sparse graphs with applications to exact algorithms. *SIAM Journal on Discrete Mathematics*, 23(1):407–427, 2009.

[63] T. Kociumaka and M. Pilipczuk. Faster deterministic feedback vertex set. *Information Processing Letters*, 114(10):556–560, 2014.

[64] R. Krithika, A. Sahu, S. Saurabh, and M. Zehavi. The Parameterized Complexity of Cycle Packing: Indifference is Not an Issue. *Algorithmica*, 81(9):3803–3841, 2019.

[65] M. Krivelevich, Z. Nutov, M. R. Salavatipour, J. V. Yuster, and R. Yuster. Approximation Algorithms and Hardness Results for Cycle Packing Problems. *ACM Transactions on Algorithms*, 3(4), 2007.

[66] James K. Lan and Gerard Jennhwa Chang. On the mixed domination problem in graphs. *Theor. Comput. Sci.*, 476:84–93, 2013.

[67] T. Le, D. Lokshtanov, S. Saurabh, S. Thomassé, and M. Zehavi. Subquadratic Kernels for Implicit 3-Hitting Set and 3-Set Packing Problems. In *Proc. of the 29th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 331–342, 2018.

[68] Tien-Nam Le, Daniel Lokshtanov, Saket Saurabh, Stéphan Thomassé, and Meirav Zehavi. Subquadratic kernels for implicit 3-hitting set and 3-set packing problems. In *SODA*, 2018.

[69] J. M. Lewis and M. Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980.

[70] William Lochet, Daniel Lokshtanov, Pranabendu Misra, Saket Saurabh, Roohani Sharma, and Meirav Zehavi. Fault tolerant subgraphs with applications in kernelization. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 47:1–47:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. `doi:10.4230/LIPIcs.ITCS.2020.47`.

[71] D. Lokshtanov, A. Mouawad, S. Saurabh, and M. Zehavi. Packing Cycles Faster Than Erdös-Pósa. *To appear in ICALP*, 2017.

[72] D. Lokshtanov, N. S. Narayanaswamy, V. Raman, M. S. Ramanujan, and S. Saurabh. Faster Parameterized Algorithms Using Linear Programming. *ACM Trans. Algorithms*, 11(2):15:1–15:31, 2014.

[73] D. Lokshtanov, F. Panolan, R. Sridharan, and S. Saurabh. Lossy kernelization. *To appear in STOC*, 2017.

[74] Daniel Lokshtanov, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. Lossy kernelization. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 224–237. ACM, 2017. `doi:10.1145/3055399.3055456`.

[75] P. J. Looges and S. Olariu. Optimal Greedy Algorithms for Indifference Graphs. *Computers & Mathematics with Applications*, 25(7):15–25, 1993.

[76] Aniket Majumdar. *Neighborhood hypergraphs: a framework for covering and packing parameters in graphs*. PhD thesis, Clemson University, 1992.

[77] G. Manić and Y. Wakabayashi. Packing Triangles in Low Degree Graphs and Indifference Graphs. *Discrete Math.*, 308(8):1455–1471, 2008.

[78] David Manlove. On the algorithmic complexity of twelve covering and independence parameters of graphs. *Discrete Applied Mathematics*, 91(1-3):155–175, 1999.

[79] Amram Meir. On total covering and matching of graphs. *J. Comb. Theory, Ser. B*, 24(2):164–168, 1978.

[80] Silvio Micali and Vijay V. Vazirani. An $\mathcal{O}(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In *FOCS*, pages 17–27, 1980.

[81] N. Misra, G. Philip, V. Raman, S. Saurabh, and S. Sikdar. FPT algorithms for connected feedback vertex set. *Journal of combinatorial optimization*, 24(2):131–146, 2012.

[82] M. Naor, L. J. Schulman, and A. Srinivasan. Splitters and Near-optimal Derandomization. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 182–191, 1995.

[83] Uri N. Peled and Feng Sun. Total matchings and total coverings of threshold graphs. *Discrete Applied Mathematics*, 49(1-3):325–330, 1994.

[84] M. Rajaati, Mohammad Reza Hooshmandasl, Michael J. Dinneen, and Ali Shakiba. On fixed-parameter tractability of the mixed domination problem for graphs with bounded tree-width. *CoRR*, abs/1612.08234, 2016.

[85] A. Slivkins. Parameterized Tractability of Edge-Disjoint Paths on Directed Acyclic Graphs. *SIAM J. Discrete Math.*, 24(1):146–157, 2010.

[86] S. Thomassé. *A quadratic kernel for feedback vertex set*, chapter 13, pages 115–119. 2009.

[87] P. van ’t Hof and Y. Villanger. Proper interval vertex deletion. *Algorithmica*, 65(4):845–867, 2013.

[88] M. Xiao and H. Nagamochi. Exact algorithms for maximum independent set. In *International Symposium on Algorithms and Computation*, pages 328–338. Springer, 2013.

[89] Mihalis Yannakakis and Fanica Gavril. Edge dominating sets in graphs. *SIAM Journal on Applied Mathematics*, 38(3):364–372, 1980.

[90] Yancai Zhao, Liying Kang, and Moo Young Sohn. The algorithmic complexity of mixed domination in graphs. *Theor. Comput. Sci.*, 412(22):2387–2392, 2011.