Computational Complexity of Graph Orientation, Bilevel and Robust Optimization Problems

Von der Fakultät für Informatik der RWTH Aachen University
zur Erlangung des akademischen Grades einer Doktorin der Naturwissenschaften
genehmigte Dissertation

vorgelegt von

Komal Dilip Muluk, M. Sc.

Berichter: Universitätsprofessorin Dr. Britta Peis
Universitätsprofessor Dr. Christoph Buchheim
Universitätsprofessor Dr. Martin Grohe

Tag der mündlichen Prüfung: 9. Februar 2026

To my parents,
for I would not be me if it weren't for you.

Abstract

The goal of this thesis is twofold. We study two types of computational problems: first, a decision problem, which is a graph orientation problem, and second, optimization problems in the realm of bilevel optimization and robust optimization. We derive a host of hardness and tractability results for a bunch of combinatorial problems within these two settings. Our completeness results span over the first and the second level of the polynomial hierarchy; we obtain NP-complete and Σ_2^{p}-complete results.

For the graph orientation problem, we study SC-Orientation (SCO) wherein the task is to determine whether we can make an undirected graph 'singly connected' by means of edge orientations; a digraph D is said to be *singly connected* if for each (ordered) pair of vertices $s, t \in V(D)$, there is at most one path from s to t in D. Our main result concerns graphs of a fixed girth g and coloring number c. For every $g, c \geq 3$, the problem restricted to the graph class $\mathcal{G}_{g,c}$, which consists of instances of girth g and coloring number c, is either NP-complete or in P. Not only that, but when SCO is in P on $\mathcal{G}_{g,c}$, we show that $\mathcal{G}_{g,c}$ contains only yes-instances of SCO. It is noteworthy that our hardness result relies on the existence of a solitary no-instance of SCO in $\mathcal{G}_{g,c}$. The results of this nature—either NP-complete or all yes-instances—are rare in the field of computational complexity.

Further, we consider bilevel optimization problems in which the ground set is partitioned between two decision makers whose objective functions are nested into each other. We study the bilevel versions of the assignment, interval scheduling, and independent set problems, and categorize them into their respective complexity classes. One result that stands out here is that Bilevel Interval Scheduling does not increase the complexity of the problem (both the underlying problem and its bilevel version are in P), whereas Bilevel Independent Set goes from NP-hard to Σ_2^{p}-hard for the setting in which both decision makers have sum type objective functions.

Lastly, within the setting of adversarial optimization under uncertainty, we introduce a model called *recoverable robust optimization with commitment*. Ours is a recoverable robust optimization model, in which the recovered solution must contain all surviving elements chosen in the initial solution. We study our model on different combinatorial optimization problems from the algorithmic and complexity perspective.

Deutsche Zusammenfassung

Das Ziel dieser Arbeit ist es, zwei Arten von Problemen der Informatik zu untersuchen: Erstens ein Graphorientierungsproblem (ein Entscheidungsproblem) und zweitens Optimierungsprobleme aus dem Bereich der Bilevel-Optimierung und der robusten Optimierung. Wir zeigen für eine Reihe kombinatorischer Probleme in diesen beiden Bereichen, ob diese entweder schwer oder effizient lösbar sind. Unsere Schwereergebnisse erstrecken sich über die erste und zweite Stufe der polynomiellen Hierarchie; wir zeigen insbesondere, dass die dazu betrachteten Probleme vollständig für die Klassen NP bzw. Σ_2^p sind.

Für das Graphorientierungsproblem untersuchen wir die SC-Orientation (SCO), bei der es darum geht, zu bestimmen, ob ein ungerichteter Graph durch Kantenorientierungen „einfach zusammenhängend“ gemacht werden kann. Ein Digraph D heißt *einfach zusammenhängend*, wenn es für jedes (geordnete) Paar von Knoten $s, t \in V(D)$ höchstens einen Pfad von s nach t in D gibt. Unser Hauptresultat ist bezüglich Graphen mit fester Taillenweite (engl.: girth) g und fester Knotenfärbungszahl (engl. chromatic number) c: Für jedes $g, c \geq 3$ ist das Problem, eingeschränkt auf die Graphklasse $\mathcal{G}_{g,c}$, die aus Instanzen mit Taillenweite g und Färbungszahl c besteht, entweder NP-vollständig oder in P. Darüber hinaus zeigen wir, dass SCO eingeschränkt auf $\mathcal{G}_{g,c}$ in genau dann in P liegt, wenn $\mathcal{G}_{g,c}$ nur Ja-Instanzen von SCO enthält. Bemerkenswert ist, dass unser Schwereresultat auf der Existenz einer einzigen Nein-Instanz von SCO in $\mathcal{G}_{g,c}$ beruht. Ergebnisse dieser Art – also das ein Problem entweder NP-vollständig ist oder ausschließlich aus Ja-Instanzen besteht – sind im Bereich der Komplexitätstheorie selten.

Weiterhin betrachten wir zweistufige Optimierungsprobleme, bei denen die Grundmenge in genau zwei Teilmengen partitioniert ist. Beide Teilmengen sind assoziiert mit einem Entscheidungsträger und deren Zielfunktionen hängen jeweils auch von den Entscheidungen des anderen ab. Wir untersuchen die zweistufigen Versionen vom Zuordnungsproblem (engl.: assignment problem), vom Intervallplanungsproblem (engl.: interval scheduling) und vom stabile Mengenproblem (engl.: independent set) und ordnen sie in ihre jeweiligen Komplexitätsklassen ein. Ein hier herausstechendes Ergebnis ist, dass Bilevel Interval Scheduling die Komplexität des Problems nicht erhöht (sowohl die nominale als auch die Bilevel-Version liegen in P), wohingegen Bilevel Independent Set Σ_2^p-schwer wird (die nominale Version ist nur NP-schwer), wenn beide Entscheidungsträger gewichtete Summen als Zielfunktionen haben.

Schließlich stellen wir im Kontext der robusten Optimierung unter Unsicherheit ein Modell namens *Recoverable Robust Optimization with Commitment* (deutsch: Reparierbare Robuste Optimierung mit Bindung) vor. Unser Modell erfordert, dass die wiederhergestellte Lösung nach „feindlichem Eingriff“ alle in der Ausgangslösung enthaltenen Elemente enthält, die nach dem Eingriff weiter verfügbar sind. Wir untersuchen unser Modell anhand verschiedener kombinatorischer Optimierungsprobleme bezüglich Algorithmik und ihrer Komplexität.

Acknowledgment

As this roller-coaster of a PhD journey comes to an end, I would like to take a moment to give a shout-out to some incredible people who have made this experience memorable.

First, I would like to thank my late supervisor, Gerhard Woeginger, along with Joost-Pieter Katoen and all the other UnRAVeL PIs who gave me the opportunity to join this amazing research community. It has been a privilege to be a part of it.

A special thanks to my time at i1, which I shared with Gerhard, Walter, Erika, Viktor, Janosch, Tim, Dennis, Christoph, Tom, and Robin. You guys are incredibly talented, and I have gained so much from working with you.

I am also grateful to the wonderful people I met at OMS, including Britta, Dina, Andreas, Niklas, Kathi, Lennart, Philipp, Kai Van, Laura, and Niklas′. Thank you for sharing this journey with me (sometimes quite literally: by walking back home together). I will always look back on our time together fondly, especially how you introduced me to the fascinating world of German board games!

Thanks to all the UnRAVeL members, in particular to Birgit and Helen, who have been there to resolve all my bureaucratic needs promptly. To my friends in UnRAVeL, Tim and Tobias, thank you for always checking on me.

Eventually, I met some very supportive people at LSV in TU Dortmund: Christoph, Sabine, Maja, Pia, and Lowig. Thank you for your help in every possible way.

I want to thank my amazing coauthors, Gerhard, Dennis, Tim, Britta, Nicole, Felix, Saket, Soumen, Jayakrishnan, Lawqueen, Avinandan, and Nidhi. I had such a blast working with all of you. It was a pleasure collaborating with you all. Special thanks to Dennis for some helpful discussions that led to some of the results in Chapter 4 of this thesis.

I also want to express my gratitude to Dorothee Henke, Niklas Rieken, Tim Hartmann, and Felix Hommelsheim for meticulously proofreading my thesis, and all your invaluable feedback. Thanks to the members of my committee Britta Peis, Christoph Buchheim, and Martin Grohe for agreeing to further review this thesis.

To my constants, Aniruddha, Kunal, Rasika, and Vipul, for being with me through thick and thin, and to my parents, for their unwavering support, thank you all for believing in me and always having my back!

Foremost, to you, Niklas, for all your confidence in me, thank you for being my rock!

Contents

1

Introduction

„To not know math is a severe limitation to understanding the world."

—Richard P. Feynman [Fey05]

The world is driven by various challenging problems. Let it be problems related to scheduling jobs to a machine, directing a street network to enforce a one-way scheme, setting the revenue-maximizing prices for your products subject to the buyer's behaviors, or reserving a service for some customers, knowing that, when some of them cancel their reservations, we might have difficulty filling up the freed-up slots. These computational problems become more and more involved as they get more and more layered. As we encounter the diversity of various such problems, the most natural question that arises is: how realistic is it to find the best possible solution for a specific problem? Additionally, can we tell whether the problems mentioned above are equally challenging? The answer to the first question is that it might not be possible to solve every computational problem in reasonable time. However, if we cannot solve them efficiently in reasonable time, we can at least sort them according to how difficult they are to solve. This answers our second question. We can potentially show, with the help of a well-established theory, that a particular problem is at least as difficult as the other. The underlying theory is known as the *computational complexity theory*, and it allows us to differentiate problems according to their intricacies.

Evidently, the exemplary computational problems mentioned above have different structures. Some are straightforward questions that can be answered by a single decision maker, while others seem to involve some dependencies with more than one decision maker. For a combinatorial problem, its version that involves one decision maker is most likely to be easier than its version that involves two decision makers. Before we try to tackle these problems, it makes sense to estimate their intrinsic difficulty level. Then, one could use suitable tools that are developed to solve another problem of a similar difficulty level. Computational complexity theory classifies problems in different *complexity classes*. Moreover, a problem can be shown to be *complete* for a particular complexity class, which means it is as hard as any other problem that belongs to the

class, or in other words, it is one of the hardest problems of that class. The basic complexity classes are the well-known classes P and NP, which are widely believed not to be equal, although whether or not $\mathsf{P} = \mathsf{NP}$ is a longstanding open question in the field of theoretical computer science.

In this thesis, we dive into studying computational problems in different frameworks. Namely, we study a graph orientation problem, various combinatorial optimization problems in the framework of bilevel optimization, and another set of combinatorial optimization problems within the framework of robust optimization. Our study involves finding polynomial-time algorithms, if possible, or placing these problems in their respective complexity classes in the computational complexity landscape, especially in the polynomial hierarchy. We briefly discuss the above-mentioned frameworks below.

1.1 Graph Orientation to Singly Connectedness

Within the framework of *graph orientation problems*, we are given an undirected graph, and the task is to convert it into a directed graph (network) which satisfies some desirable properties.

Chapter 3 of this thesis is devoted to studying a graph orientation problem in which our goal is to convert an undirected graph into a singly connected digraph. A directed graph is called *singly connected* if, for any ordered pair of vertices s, t, there exists at most one path joining s to t in the digraph. We refer to the feasibility problem of graph orientation to singly connectedness as SC-Orientation. In Chapter 3, we give a detailed introduction to the class of graph orientation problems. We also discuss SC-Orientation along with its prospective applications. Moreover, we commit to studying SC-Orientation from a viewpoint of computational complexity, and carry out our study on different graph classes, and further classify the problem on those classes into P and NP.

1.2 Bilevel Optimization

The topic of bilevel optimization has garnered significant attention in the last two decades. The framework of bilevel optimization is used to understand the dynamics between two hierarchical decision makers, classically referred to as a leader and a follower, whose optimization problems are nested into one another. The leader is considered to be the first decision maker who decides on her set of variables, followed by the follower, who then decides on his set of variables. Since the follower, in a way, "reacts" to the leader's choice of variables, and the leader's objective value also depends on the follower's chosen set of variables, the leader has to take her decision in anticipation of how the follower will react. In case the follower has multiple optimal reactions, there are two common settings in which the follower may react. One is called the *optimistic setting*, in which the follower is altruistic towards the leader, thus he has her best interest in mind,

and the other is called the *pessimistic setting*, in which the follower is spiteful towards the leader, thus he behaves like an adversary.

In the bilevel optimization problems, the leader is often considered to have full knowledge of the follower's variables and his objective function. Despite this, these problems can be significantly challenging compared to their underlying single-level counterparts.

In this thesis, we study a particular class of bilevel optimization problems called the *partitioned-items bilevel optimization problems*, in which the variable set is partitioned into the leader's and the follower's variable sets. We give a detailed introduction of the partitioned-items bilevel optimization problems in Chapter 4. Our study involves various bilevel variants of the Assignment problem, the Interval Scheduling problem, and the Independent Set problem.

1.3 Recoverable Robust Optimization

Recoverable robust optimization is used to deal with uncertainty in adversarial optimization. Here, the goal is to opt for a solution that works the best when the worst possible scenario is revealed, and following that, we are allowed to adapt the opted solution within some recovery bounds. Often, the goal is to optimize the value of the adapted solution; however, sometimes, the recoverable robust optimization framework allows us to destroy the opted solution in the phase of adaptation. And not all unperturbed elements of the opted solution are guaranteed to survive the adaptation phase.

In Chapter 5, within the realm of *adversarial optimization under uncertainty*, we introduce and study a recoverable robust optimization model that demands *certainty*. We call this model *recoverable robust optimization with commitment* because we demand commitment to the opted solution. Our model works the following way. Given a combinatorial optimization problem, we have to opt for a feasible solution at first. Afterwards, when an adversarial scenario is revealed, in the adaptation phase, we are required to find an adapted solution that must contain the remaining elements from the opted solution. Clearly, this optimization scheme is more involved than the bilevel optimization problems discussed above. Even though here we also have only two decision makers, namely, the optimizer and the adversary, the decision procedure goes back and forth between the two decision makers multiple times. In general, a problem of recoverable robust optimization with commitment can be written as a multilevel optimization problem, which involves three stages that are hierarchically nested: the optimizer opting for the first-stage solution, the adversary revealing an uncertain scenario (by means of deletion of some elements), and finally, the optimizer adapting the partial remaining solution to an adapted solution.

In this thesis, we investigate the model of recoverable robust optimization with commitment on some combinatorial optimization problems where the nominal problems are Matroid Basis, Matching, Independent Set, and Interval Scheduling.

1.4 Publications

This thesis comprises, but is not limited to, several research topics that I worked on during my doctoral studies. Three of the articles have been peer reviewed and published; one of them is published in a conference proceedings and the other two in a journal. Additionally, the fourth article is yet to be published.

List of Publications Stated below are the publications that are content-wise significant to this thesis. The works mentioned below include a number of co-authors. Having said that, I have contributed to the different phases that led to the listed publications. In particular, I have been a part of the exploration of various ideas, further conceptualization, investigation, understanding, and analysis of them, up to the thorough write-up of the articles. Due to a standard practice in the field of theoretical computer science and discrete mathematics, all authors on these papers are listed in alphabetical order. For some topics, this thesis contains additional results apart from the contents in the following articles.

- *Make a graph singly connected by edge orientations*.
  Tim A. Hartmann and Komal Muluk
  published in the proceedings of the 34^{th} *International Workshop on Combinatorial Algorithms*[1] [HM23a]
  (preprint: [HM23b])

- *A note on the complexity of the bilevel bottleneck assignment problem*
  Dennis Fischer, Komal Muluk, and Gerhard J. Woeginger
  published in a quarterly journal of operations research *4OR* [FMW22]

- *Recoverable robust optimization with commitment*
  Felix Hommelsheim, Nicole Megow, Komal Muluk, and Britta Peis
  published in Mathematical Programming (Series A) [HMMP26]
  (preprint: [HMMP23])

- *On the Complexity of Bilevel Independent Set Problem*
  Komal Muluk
  (preprint: [Mul26])

Stated below are the publications that are not content-wise included in this thesis.

- *On the complexity of singly connected vertex deletion*
  Avinandan Das, Lawqueen Kanesh, Komal Muluk, Nidhi Purohit, Jayakrishnan Madathil, and Saket Saurabh
  published in a journal *Theoretical Computer Science* [DKM+22]

[1] This publication won the *best student paper* award of the conference.

- *Parameterized complexity of fair feedback vertex set problem*
  Lawqueen Kanesh, Soumen Maity, Komal Muluk, and Saket Saurabh
  published in a journal *Theoretical Computer Science*[2] [KMMS21]

[2]This publication builds on the results obtained during my master's thesis [Mul19].

2

Preliminaries

This chapter is devoted to introducing and revising all necessary terminologies and theoretical concepts that are required to grasp the contents of this thesis. We begin with basic set theoretic notations, then proceed with definitions of various graph theoretical concepts. Later, we discuss notions from the field of complexity theory, in particular, the polynomial hierarchy; one ought to know these concepts before diving into the technicalities of the following chapters.

2.1 Basic Notations

In set theory, a *set* is a collection of distinct elements; it is usually denoted by a capital letter and presented using curly parentheses. When the repetition of an element is allowed, we explicitly refer to it as a *multiset*. For a set S, the *cardinality* of the set is the total number of elements in the set and is denoted by $|S|$. We denote the set of *natural* numbers (including zero) by $\mathbb{N}$, the set of *integers* by $\mathbb{Z}$, and the set of *real* numbers by $\mathbb{R}$. Moreover, $\mathbb{R}_+$ represent the subsets of non-negative elements of $\mathbb{R}$. For $n \in \mathbb{N}$, define $[n] \coloneqq \{1, 2, \ldots, n\}$. Given two constants $C_1, C_2 \in \mathbb{R}_+$, we denote C_1 is *much greater than* (or *much less than*) C_2 with notation $C_1 \gg C_2$ (or $C_1 \ll C_2$, respectively). For simplification, we define the addition and subtraction of a single element from a set with the notations: $S + a \coloneqq S \cup \{a\}$ and $S - a \coloneqq S \setminus \{a\}$, respectively. The *power set* of a set S is denoted by $2^S \coloneqq \{X \mid X \subseteq S\}$. A *k-partition* of a finite set S is a grouping of elements of S into k non-empty subsets $S_1, S_2, \ldots, S_k$, $S_i \subseteq S$ for all $i \in [k]$, such that $\bigcup_{i=1}^{k} S_i = S$ and $S_i \cap S_j = \varnothing$ for all $i, j \in [k]$, $i \neq j$. We denote a k-partition of S by $S_1 \dot\cup S_2 \dot\cup \ldots \dot\cup S_k$. Given a function $f \colon S \to \mathbb{R}_+$, we define for a subset $S' \subseteq S$, $f(S') \coloneqq \sum_{e \in S'} f(e)$.

Elements in a set are not ordered, meaning the order in which they are stated is not important and does not carry any meaning. However, we do sometimes need to distinguish between two sets depending on the sequence of elements therein. For this purpose, we use the notion of *ordered sets*. An ordered set is also a collection of distinct elements. But unlike regular sets, the order in which the elements appear here matters, and they are usually presented using round parentheses. We further distinguish between

an *ordered pair* and an *unordered pair*. The former is an ordered set containing exactly two elements, e.g. (a, b), and the latter is a regular set with exactly two elements $\{a, b\}$.

Throughout the thesis, we use the notation $\leftarrow$ to define an update operation on a set (sometimes also on a graph), for example, in order to update a set A with the addition of an element b, we may write, $A \leftarrow A \cup \{b\}$.

This thesis includes a bunch of combinatorial optimization problems that are derived from graphs. So in Section 2.2, we define all relevant terminologies from the field of graph theory.

2.2 Graph Theory

In the field of discrete mathematics, a *graph* is a structure consisting of two finite sets, namely a set of *nodes* (or *vertices*) and a set of *edges* (or *arcs* for directed edges). Each edge of the graph is comprised of a pair of vertices called its *endpoints* (unless we discuss *hypergraphs*, which are a generalization of graphs wherein an edge can join any number of vertices). Depending on whether the pairs are ordered or unordered, the graph is characterized as a *directed* or an *undirected* graph. The field emerging from the study of graphs is called *graph theory*.

An *undirected graph* is usually denoted by the notation $G = (V(G), E(G))$ with $V(G)$ being the set of vertices and $E(G)$ being the set of edges. We sometimes relax the notation and only refer to the sets as V and E. Each edge $e \in E$ is *incident* to two vertices in $V(G)$. For undirected graphs, we denote an edge e incident to vertices u and v by either $\{u, v\}$ or simply uv. Vertices u and v are the *endpoints* of e. Two edges are said to be *adjacent* if they share at least one endpoint. If a graph contains multiple replicas of an edge, i.e., E is a multiset, then it is referred to as a *multigraph*; the replicated parallel edges are called *multiple edges* or *multi-edges*. A *simple graph* is an undirected graph with an edge set $E \subseteq \binom{V}{2}$; simple graphs have no *loops* (i.e., edges where both the endpoints match) and no multiple edges. Given a graph $G = (V, E)$, the deletion of a vertex $v \in V$ along with all its adjacent edges is called *vertex deletion*. Similarly, deletion of an edge $e \in E$ from the graph is called *edge deletion*. In this text, we sometimes use graphs wherein each edge has a numerical weight associated with it. A graph with a weight function $w \colon E \to \mathbb{R}_+$ is called a *weighted graph*; an *unweighted graph* has no weights associated with its edges.

A *directed graph* or *digraph*, on the other hand, is usually denoted by D with $V(D)$ as its set of vertices and $A(D)$ as its set of arcs. We sometimes refer to the vertex set and arc set simply by V and A, respectively. The arc set in a digraph is $A \subseteq \{(u, v) \mid u, v \in V\}$. Note that A is a set of ordered pairs, that means, each arc $a \in A$ has a direction; $a = (u, v) \in A$ implies the arc is *directed* from vertex u to v. Simultaneously, a is an *outgoing arc* of u and an *incoming arc* of v. When we look past the direction of the arcs in a digraph D, we get the *underlying undirected graph* of the digraph; for a digraph D we denote its underlying undirected graph by G_D. Thus $V(G_D) = V(D)$, but the edge

set $E(G_D)$ contains an edge $\{u, v\}$ for each arc $(u, v) \in A(D)$; these are the underlying undirected edges of the arcs.

For an undirected graph $G = (V, E)$ and a vertex $v \in V$, the *degree* (or *valency*) of v in G, denoted by $\delta_G(v)$, is the total number of edges incident to v in G. When it is clear, we simply write $\delta(v)$. A vertex with $\delta_G(v) = 0$ is called an *isolated* vertex. A vertex with $\delta_G(v) = 1$ is called a *pendant* vertex or a *leaf* vertex. The *minimum degree* (or *maximum degree*) of a graph G is the least (or highest) number of edges incident to a vertex in the graph, and it is denoted by the Greek letter δ (or Δ, respectively). $\delta(G) := \min\{\delta(v) \mid v \in V\}$ and $\Delta(G) := \max\{\delta(v) \mid v \in V\}$. The set $N_G(v) := \{u \in V \mid uv \in E\}$ forms the *open neighborhood* of v in G, and the set $N_G[v] := N_G(v) \cup \{v\}$ forms the *closed neighborhood* of v in G. When it is clear from the context, we omit the subscript and denote these terms simply as $N(v)$ and $N[v]$.

Similarly, for a digraph $D = (V, A)$ and $v \in V$, we distinguish between the *indegree* and *outdegree* of v. The *indegree* (or *outdegree*) of v is the total number of incoming (or outgoing) arcs incident to v and it is denoted by $\delta_D^-(v)$ (or $\delta_D^+(v)$, respectively). Analogous to the neighborhood notions in undirected graphs, here we define $N_D^-(v) := \{u \in V \mid (u, v) \in A\}$ for the set of *in-neighbors* of v and $N_D^+(v) := \{u \in V \mid (v, u) \in A\}$ for the set of *out-neighbors* of v. Additionally, we define $N_D^-[v] := N_D^-(v) \cup \{v\}$ and $N_D^+[v] := N_D^+(v) \cup \{v\}$ for the respective closed neighborhood sets. We omit the subscript D when it is clear from the context. A vertex $v \in V(D)$ is called a *source* if $\delta^-(v) = 0$, and a *sink* if $\delta^+(v) = 0$.

For a general (directed or undirected) graph $G = (V, E)$, a *subgraph* $G' = (V', E')$ of G has $V' \subseteq V$ and $E' \subseteq E$. For a subset $V' \subseteq V$, an *induced subgraph* of G on V' consists of the vertex set V' and all edges in G whose both endpoints lie in the set V'. Such an induced subgraph is denoted by $G[V']$. Further, we use the notation $G \setminus V'$ to denote the induced subgraph $G[V \setminus V']$.

A *path* (or a *directed path*) in a graph (or a digraph) is a finite or infinite sequence of distinct vertices such that each consecutive pair of vertices in it shares an edge (or an arc in a specific direction). We denote a path P going through vertices u, v, w, x, and y using the notation $uvwxy$. Sometimes, we also denote a path using edges e_1, e_2, e_3, and e_4 using the notation $e_1e_2e_3e_4$. The *length of a path* is the total number of edges in the path. A *subpath* of a path is a subsequence of consecutive vertices in the original path. For $s, t \in V$, an *s-t-path* joins vertices s and t such that they are the endpoints of the path. A *cycle* in a graph is a closed path such that the first and the last vertex match and the path contains at least one edge. We denote a cycle going through vertices u, v, w, x, and y using the notation $uvwxyu$. We use the same notation for both directed and undirected paths, but it will be clear from the context which path we have at hand. The *distance* between two vertices $u, v \in V$ of a graph is the length of the shortest path joining those two vertices. The *length of a cycle* is the total number of edges in it. The length of the shortest cycle in a graph is called the *girth* of the graph. Graphs which do not have any cycles are called *acyclic* graphs. By definition, an acyclic graph has girth ∞. A graph

which only consists of a cycle of length n is called an n-cycle, and is denoted by C_n. For a cycle C, the set of vertices in C is denoted by $V(C)$, and the set of edges in C is denoted by $E(C)$. A *chord* in a cycle C is an edge $e = uv$ such that $u, v \in V(C)$ but $e \notin E(C)$.

An undirected graph $G = (V, E)$ is *connected* if there is a u-v-path joining each pair of vertices $u, v \in V$. A *connected component*, or sometimes only referred to as a *component*, of a graph G is a maximal connected subgraph of G. A graph is called *disconnected* if it has at least two distinct connected components. A vertex $v \in V$ is called a *cut vertex* if the deletion of v along with its incident edges increases the number of connected components in the graph. A graph is *k-vertex-connected* if at least k vertices are required to be deleted to make the graph disconnected. Similarly, a *bridge* is an edge in the graph which, when deleted, renders the graph disconnected. And a graph is *k-edge-connected* if at least k edges are required to delete to make the graph disconnected. A directed graph D is said to be *weakly connected* if its underlying undirected graph is connected, and it is said to be *strongly connected* if for every pair of vertices $u, v \in V$, there are directed paths in D both from u to v and from v to u.

A *proper vertex coloring* of a graph assigns colors to the vertices of the graph such that no two adjacent vertices receive the same color. If a proper coloring uses exactly k colors for $k \in \mathbb{N}$, that is, there is a surjective mapping $f\colon V(G) \to \{1, \ldots, k\}$ such that $f(u) \neq f(v)$ when $\{u, v\} \in E(G)$, then it is called a *k-coloring*. A graph is said to be *k-colorable* if there exists a k-coloring of that graph. The *chromatic number* or the *coloring number* of a graph G is the smallest integer k such that G has a k-coloring. The chromatic number of G is denoted by $\chi(G)$.

Two important edge operations that we use in this thesis are *edge contraction* and *edge subdivision*. An edge contraction is when an edge $e = uv$ is deleted from the graph, and its endpoints u and v are identified. An edge subdivision of $e = uv$ introduces an additional vertex, say w, and modifies the edge by deleting uv and introducing two new edges uw and wv to the graph instead.

Given an undirected graph $G = (V, E)$, an *independent set* of G is a subset of vertices $S \subseteq V$ such that the induced subgraph $G[S]$ does not contain any edges. On the other hand, a *clique* is a subset of vertices $S \subseteq V$ such that $\big\{\{u, v\} \mid u, v \in S\big\} \subseteq E$. A *vertex cover* of G is a subset $S \subseteq V$ such that the vertex deletion of S from G renders the graph without any edges; the set $V \setminus S$ is an independent set when S is a vertex cover. A *feedback vertex set* of G is a subset $S \subseteq V$ such that deletion of S from G renders the graph acyclic. A *matching* in G is a subset of edges $M \subseteq E$ such that no two edges $e_1, e_2 \in M$ share an endpoint. The vertex set formed by collecting the endpoints of all edges in the matching is said to be *covered* by the matching. A matching that covers the entire vertex set of the graph is called a *perfect* matching.

Since a given graph can be drawn in various ways, an *embedding of a graph* is a specific drawing of the graph on the Euclidean plane $\mathbb{R}^2$. For a particular embedding of a graph, *crossing edges* are the set of edges that intersect with each other apart from their endpoints.

The *complement* of a graph $G = (V, E)$ is a graph $\overline{G} = (\overline{V}, \overline{E})$ where $\overline{V} = V$ and $\overline{E} = \{\{u, v\} \mid \{u, v\} \notin E\}$.

The *line graph* of a graph G contains the vertex set corresponding to $E(G)$, and there is an edge between two vertices if and only if the corresponding edges are adjacent, that is, they share an endpoint in G.

An undirected graph H is called a *minor* of G, if H is obtained from G by performing a sequence of operations: either edge deletion, vertex deletion, or edge contraction.

For further reading related to the concepts in graph theory, we refer the reader to the books by Bollobás [Bol98], Bondy and Murty [BM76], Diestel [Die12], and West [Wes01].

2.2.1 Graph Classes

Throughout this thesis, we discuss and perform our study on various graph classes. We state some of them in this section. It is noteworthy that some graph classes have a *forbidden graph characterization*, that means, one can completely characterize the given graph class using some substructures (a set of individual graphs), called *obstructions*, that are forbidden to exist in any graph belonging to that graph class. Depending on the type of the forbidden set relation, the obstructions cannot exist as a subgraph, an induced subgraph, a homeomorphic subgraph (also called topological minor), or sometimes as a graph minor. See for example [Kur30, CRST06]. For a given graph class, if its obstruction set contains graphs $G_1, G_2, \ldots, G_k$ as a forbidden induced subgraph, then we can also denote this graph class using the notation $(G_1, G_2, \ldots, G_k)$-*free*. If this obstruction set is finite, the graph class also said to have a *finite forbidden set*.

Now, we introduce a few graph classes in the undirected setting.

Complete graphs contain edge set $E = \{\{u, v\} \mid u, v \in V\}$; they have edges between every pair of vertices. A complete graph on n vertices is denoted by K_n. A *cluster* graph is a graph in which each connected component is a complete graph.

Forests are those graphs for which each connected component is cycle-free. They are also known as *acyclic graphs*. By definition, cycles of all length act as an obstruction set for forests. Every forest graph is 2-colorable.

Cactus is a class of graphs in which each edge is contained in at most one cycle. In other terms, any two cycles share at most one vertex.

The class of *bipartite graphs* contains graphs wherein the vertex set V can be bipartitioned into two sets $V = X \dot\cup Y$, and every edge $\{u, v\} \in E$ implies the vertices u and v belong to different parts of the bipartition. Since each edge of the graph go across sets X and Y, there does not exist an edge which has both the endpoints either in X or in Y. In other terms, X and Y are independent sets of the graph. Consequently, every bipartite graph is 2-colorable and vice versa. A *complete bipartite graph* $K_{m,n}$ has $|X| = m$, $|Y| = n$, and the edge set $E = \{\{u, v\} \mid u \in X, v \in Y\}$.

The class of *near-bipartite* graphs contains all graphs whose vertex set can be partitioned into two sets, one, an independent set, and the other set induces a forest. Since a forest is 2-colorable, all near-bipartite graphs are inherently 3-colorable.

The class of *split* graphs contains all graphs whose vertex set can be partitioned into two sets, one, an independent set, and the other induces a complete graph. These are exactly the graphs which are $(C_4, C_5, 2K_2)$-free, where $2K_2$ is a graph consisting of two non-adjacent edges.

A graph is called a *chordal* graph if each cycle of length at least 4 in it has a chord.

Interval graphs are undirected graphs in which each vertex corresponds to an interval on the real line and two vertices are adjacent to each other if and only if their corresponding intervals intersect. Every interval graph is chordal.

A *distance hereditary* graph has a property that they maintain the vertex distances between every pair of vertices in any connected induced subgraph that contains the pair of vertices. Moreover they can be constructed starting from a single isolated vertex with the following operations:

(A1) a *pendant* vertex to $u \in V(G)$, that is, add a vertex v and an edge $\{u, v\}$;

(A2) *true twin* on a vertex $u \in V(G)$, that is, add a vertex v with neighborhood $N(v) = N(u)$ and an edge $\{u, v\}$;

(A3) *false twin* on a vertex $u \in V(G)$, that is, add a vertex v with neighborhood $N(v) = N(u)$, but without adding the edge $\{u, v\}$.

There exists a finite forbidden set that characterized the class of distance hereditary graphs; they are exactly the (house, hole, domino, gem)-free graphs. A *hole* is a chordless cycle of length 5 or more. The other obstructions, namely house, domino, and gem, are depicted in Figure 2.1.

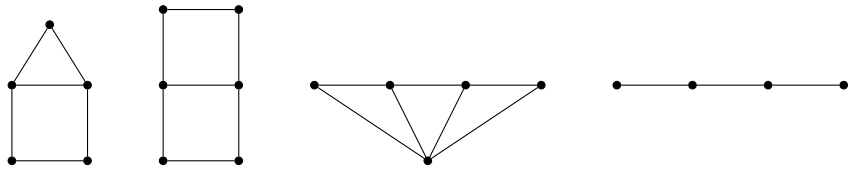


Figure 2.1: From left to right: a house, a domino, a gem, and a P_4.

Cographs do not contain a P_4 as an induced subgraphs. A P_4 is a path of length 3 and it consists of 4 vertices (refer to Figure 2.1). Moreover, cographs can be constructed starting from a single isolated vertex using the complementation and disjoint-union operations:

(B1) K_1 is a cograph;

(B2) if G is a cograph, then so is $\overline{G}$;

(B3) if G and H are cographs, then so is the disjoint union of G and H.

The class of *planar graphs* contains those graphs for which there exists a planar embedding; these graphs can be drawn on a Euclidean plane $\mathbb{R}^2$ without having any crossing edges. They are also completely characterized by forbidden minors, K_5 and $K_{3,3}$.

The class of *perfect* graphs has a property that, for each induced subgraph, the chromatic number is equal to the size of the maximum clique of the subgraph.

2.3 Complexity Theory

Computational problems consist of a set of instances such that there is a function that assigns a *solution* or *output* to each instance of the problem. There are numerous types of computational problems, for example, counting problems, decision problems, optimization problems, search problems, etc. Throughout this thesis, we mostly focus on decision problems and optimization problems. For an instance $\mathcal{I}$ of a computational problem $\mathcal{P}$, the *length of the instance*, also known as its *size*, is the total number of bits used to encode $\mathcal{I}$ under a given encoding scheme, e.g., binary encoding or unary encoding. The size of an instance $\mathcal{I}$ is denoted by $|\mathcal{I}|$. We often ask questions like whether we can find "efficient" algorithms to solve a computational problem.

An algorithm $\mathcal{A}$ is called *efficient* if there exists a polynomial p such that $\mathcal{A}$ finds a solution to every instance $\mathcal{I}$ in time at most $p(|\mathcal{I}|)$. The problems for which efficient algorithms exist are considered to be easy. When we are struggling to come up with an efficient algorithm for a problem, a natural question that arises is whether the problem is too difficult to be solved using an efficient algorithm. This leads us to the topic of *complexity theory*, which explores the inherent difficulty—the *hardness*—of computational problems.

Definition 2.1. A *decision problem* consists of tasks whose solutions are only yes or no. A decision problem $\mathcal{P}$ consists of a set $\mathfrak{I}_\mathcal{P}$ of instances which is partitioned into two subsets $(\mathcal{Y}_\mathcal{P}, \mathcal{N}_\mathcal{P})$, where the subset $\mathcal{Y}_\mathcal{P}$ contains all the instances for which the solution to the decision problem $\mathcal{P}$ is *yes*, and the subset $\mathcal{N}_\mathcal{P}$ contains all the instances for which the solution to the decision problem $\mathcal{P}$ is *no*. We refer to the instances in $\mathcal{Y}_\mathcal{P}$ and $\mathcal{N}_\mathcal{P}$ as *yes-* and *no-instances*, respectively.

Definition 2.2. A *deterministic algorithm* solves a decision problem $\mathcal{P}$, if for an instance $\mathcal{I} \in \mathfrak{I}_\mathcal{P}$ it returns the solution "yes" if and only if $\mathcal{I} \in \mathcal{Y}_\mathcal{P}$. Moreover, the algorithm halts for all instances $\mathcal{I} \in \mathfrak{I}_\mathcal{P}$, so a no-instance receives a solution "no". If the total number of steps taken by the algorithm is polynomial in the size of the input instance $\mathcal{I}$, then the algorithm is said to be a *deterministic polynomial-time algorithm*.

Definition 2.3. The complexity class P contains all decision problems for which there exists a polynomial-time deterministic algorithm.

The problems belonging to class P are also known as *tractable*.

Definition 2.4. A deterministic polynomial-time algorithm $\mathcal{A}$ is said to be a *polynomial-time verifier* for verifying the yes-instances of a decision problem $\mathcal{P}$, if for every $\mathcal{I} \in \mathcal{Y}_\mathcal{P}$, there exists a certificate c whose size is polynomially bounded in the size of $\mathcal{I}$, such that, $\mathcal{A}$ returns "yes" to the input pair $(\mathcal{I}, c)$, and for every $\mathcal{I} \in \mathcal{N}_\mathcal{P}$, no pair $(\mathcal{I}, c)$ will ever return "yes".

Definition 2.5. The complexity class NP contains all decision problems that have a polynomial-time verifier which verifies all the yes-instances.

The name NP stands for *non-deterministic polynomial-time*. An equivalent definition of the class NP can be given using *non-deterministic algorithms* that run in polynomial time. A non-deterministic algorithm, essentially, allows a perfect guess, called a *certificate* or a *witness*, that can then be verified. NP is the class of problems that can be solved using a non-deterministic algorithm that runs in polynomial time.

Indeed, if a problem can be solved deterministically in polynomial time, then it also has a polynomial-time verifier, or equivalently, a non-deterministic polynomial-time algorithm. Thus, $\mathsf{P} \subseteq \mathsf{NP}$. However, the question whether $\mathsf{P} = \mathsf{NP}$ or $\mathsf{P} \neq \mathsf{NP}$ is one of the most fundamental open questions in the field of complexity theory.

Analogously to NP, we can define the following class that requires certificates for no-instances instead of yes-instances.

Definition 2.6. The complexity class coNP contains all decision problems that have a polynomial-time verifier which verifies all the no-instances of the problem.

Given a problem $\mathcal{P}$, its *complement*, $\overline{\mathcal{P}}$, is the problem in which all yes-instances of $\mathcal{P}$ are no-instances of $\overline{\mathcal{P}}$, and vice-versa. Thus, $\mathcal{P} \in \mathsf{NP}$ if and only if $\overline{\mathcal{P}} \in \mathsf{coNP}$.

Definition 2.7. A *polynomial transformation*, a *polynomial reduction*, or a *Karp reduction* from a decision problem $\mathcal{P}_1$ to another decision problem $\mathcal{P}_2$ is a (many-one) mapping f that maps every instance $\mathcal{I}_{\mathcal{P}_1} \in \mathfrak{I}_{\mathcal{P}_1}$ to an instance $\mathcal{I}_{\mathcal{P}_2} \in \mathfrak{I}_{\mathcal{P}_2}$ such that

- the function f is deterministically computable in polynomial time, and
- for all $\mathcal{I}_{\mathcal{P}_1} \in \mathfrak{I}_{\mathcal{P}_1}$, $\mathcal{I}_{\mathcal{P}_1} \in \mathcal{Y}_{\mathcal{P}_1}$ if and only if $f(\mathcal{I}_{\mathcal{P}_1}) \in \mathcal{Y}_{\mathcal{P}_2}$.

If there exists a polynomial reduction from problem $\mathcal{P}_1$ to $\mathcal{P}_2$, then we say $\mathcal{P}_1$ is *polynomially reducible* to $\mathcal{P}_2$, and we denote this by $\mathcal{P}_1 \leq_{\mathsf{P}} \mathcal{P}_2$.

Consequently, given a polynomial reduction from $\mathcal{P}_1$ to $\mathcal{P}_2$, if problem $\mathcal{P}_2$ is easy, i.e., it can be solved using a deterministic polynomial-time algorithm, then this implies, problem $\mathcal{P}_1$ can be solved using a deterministic polynomial-time algorithm as well. In other words, if $\mathcal{P}_1 \leq_{\mathsf{P}} \mathcal{P}_2$, then we may say that the problem $\mathcal{P}_2$ is at least as hard as the problem $\mathcal{P}_1$.

Let $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ be three decision problems such that $\mathcal{P}_1 \leq_{\mathsf{P}} \mathcal{P}_2$ and $\mathcal{P}_2 \leq_{\mathsf{P}} \mathcal{P}_3$, then $\mathcal{P}_1 \leq_{\mathsf{P}} \mathcal{P}_3$. That means polynomial transformation are transitive.

Definition 2.8. A decision problem is called NP-*hard* if every problem in the class NP can be polynomially reduced to it.

For complexity classes P and NP, it is wildly believed, across mathematicians and computer scientists, that these classes are not equal. Thus unless $\mathsf{P} = \mathsf{NP}$ holds, an NP-hard problem cannot be solved using a polynomial-time algorithm. In fact, if a single NP-hard problem can be solved using a deterministic polynomial algorithm, then all problems in the class NP are polynomially solvable. Ultimately, this would then prove $\mathsf{P} = \mathsf{NP}$.

Definition 2.9. A decision problem $\mathcal{P}$ is said to be NP-*complete* if it is both NP-hard and belongs to the class NP.

All above definitions revolve around the idea of an algorithm which returns the solution in time polynomial in the length of the input instance. Although, the length of the input instance heavily depends on the encoding that one uses to store the instance. The basic schemes of encoding numbers are binary and unary. In binary, it takes up to $1 + \log n$ bits to encode the integer n, however, in unary, it takes n bits. Conversion from binary to unary encoding is an exponential blow-up, in the size of the encoding, that does not provide any additional information about the number. Consequently, the length of an instance encoded in unary is exponentially larger than the length of the same instance encoded in binary. This gives much more time for an efficient algorithm of an instance in unary, as its running time is only required to be polynomial in n instead of $\log n$. To account for this kind of oddity, we call an algorithm a *pseudopolynomial* algorithm if it is polynomial on unary encodings but not necessarily polynomial on binary encodings.

For the problems that remain hard, even when their instances are encoded in unary, we have the following definition.

Definition 2.10. If an NP-hard problem attains a pseudopolynomial algorithm, then the problem is said to be *weakly* NP-hard. Otherwise, they are said to be *strongly* NP-hard.

An instance of a computational optimization problem, a minimization or maximization problem, can be solved by asking a series of question based on its decision counterpart. For example, one can convert a task of finding a maximum value over a computational function to a problem of deciding whether there exists a function's value that exceeds a specific target value, say an integer value k. Thus a hardness result of a decision problem translates to the hardness result of its optimization counterpart. Although, in this thesis, we also discuss some optimization problems, most often we simply look into their decision variants for the study.

For further conceptual reading on complexity theory, we refer the reader to [AB09, GJ79], and [Pap03].

2.3.1 The Polynomial Hierarchy

Meyer and Stockmeyer defined the polynomial hierarchy in 1972 [MS72]. The polynomial hierarchy is a scheme of classifying multi-level decision problems into various hierarchical classes, also known as *levels*, over a complexity landscape. It consists of the classes P, NP, and coNP at its foundation. Recall that P is the class of problems that can be solved using a deterministic polynomial-time algorithm. Indeed, for a problem in P, all its yes- and no-instances can also be verified using a deterministic polynomial-time algorithm. However, NP is the class of decision problem whose yes-instances can be verified using a deterministic polynomial-time algorithm, although, its no-instances are not necessarily verifiable by any deterministic polynomial-time algorithm. Consequently, the class NP could contain decision problems that are not at all solvable using a deterministic polynomial-time algorithm. Similarly, coNP is the class of decision problem whose no-instances can be verified using a deterministic polynomial-time algorithm; its yes-instances are not necessarily verifiable using a deterministic polynomial-time algorithm. This notion of sorting decision problems, depending on how difficult it is to verify their yes- or no-instances, sits at the heart of the polynomial hierarchy. Extrapolating this idea, we get the next complexity classes in line, which are called Σ_2^P and Π_2^P (the set of complements of problems in Σ_2^P). The class Σ_2^P consists of the decision problems whose yes-instances can be verified using a deterministic polynomial-time algorithm that has access to an NP-oracle. An NP-*oracle* can be seen as a black box which solves a problem $\mathcal{P}$ in NP in one step, and for each instance in $\mathfrak{I}_\mathcal{P}$, it returns whether it is a yes-instance or a no-instance. Note that, an NP-oracle is also a coNP-*oracle*. Analogously, the class Π_2^P consists of the decision problems whose no-instances can be verified using a deterministic polynomial-time algorithm that additionally has access to an NP-oracle (or equivalently, a coNP-oracle), and so on.

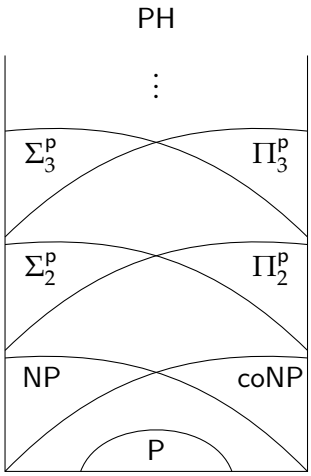


Figure 2.2: The polynomial hierarchy illustrated, assuming strict inclusions.

Each level of the polynomial hierarchy is defined inductively using its previous levels. The complexity class Σ_k^P and its complement $\text{co}\Sigma_k^P$, the class Π_k^P, sit at the k^{th} level of the polynomial hierarchy. All in all, we get, $\Sigma_0^P = \Pi_0^P = \mathsf{P}$, $\Sigma_1^P = \mathsf{NP}$, $\Pi_1^P = \text{co}\mathsf{NP}$, and $\Sigma_k^P = \mathsf{NP}^{\Sigma_{k-1}^P}$, where $\mathsf{NP}^{\Sigma_{k-1}^P}$ denotes the class of decision problems that can be solved by a non-deterministic polynomial-time algorithm that has access to a Σ_{k-1}^P-oracle. On the other hand, $\Pi_k^P = \text{co}\mathsf{NP}^{\Sigma_{k-1}^P}$.

Each class of the polynomial hierarchy is closed under polynomial transformations. Given a problem $\mathcal{P}$ in some class Σ_k^P, if there exists a polynomial transformation from $\mathcal{P}'$ to $\mathcal{P}$, then the problem $\mathcal{P}'$ also belongs to the class Σ_k^P.

Due to the recursive definition of the construction of the polynomial hierarchy, we get that each class at a specific level, contains all classes from the levels below (see Figure 2.2). We know that both NP and coNP contain the class P. Additionally, both Σ_2^P and Π_2^P contain the classes NP and coNP. Recursively, $\Sigma_k^P \supseteq \Sigma_{k-1}^P$ and $\Sigma_k^P \supseteq \Pi_{k-1}^P$. Consequently, we get,

$$\Sigma_0^P \subseteq \Sigma_1^P \subseteq \Sigma_2^P \subseteq \cdots \subseteq \Sigma_{k-1}^P \subseteq \Sigma_k^P \subseteq \cdots$$

Interestingly, it is still unknown whether any of the above inclusions are strict. However, it is strongly believed that any two consecutive complexity classes are not equal, otherwise it may invoke what is called as "the collapse of the polynomial hierarchy". That means, if $\Sigma_k^P = \Sigma_{k+1}^P$, then $\Sigma_k^P = \Sigma_i^P$ for all $i > k$.

The polynomial hierarchy, altogether, is denoted by PH, and is defined as a combined class $\mathsf{PH} = \bigcup_{k\in\mathbb{N}} \Sigma_k^P$. Equivalently, $\mathsf{PH} = \bigcup_{k\in\mathbb{N}} \Pi_k^P$. In case, $\Sigma_k^P = \Sigma_{k+1}^P$ for some k, the polynomial hierarchy is said to collapse to the k^{th}-level: $\Sigma_k^P = \mathsf{PH}$. Due to the notion of the collapse of the polynomial hierarchy, $\mathsf{P} = \mathsf{NP}$ is a stronger conjecture than $\mathsf{NP} = \Sigma_2^P$.

Moreover, if $\Sigma_k^P = \Pi_k^P$ for some $k \geq 1$, then $\Sigma_k^P = \Sigma_i^P = \Pi_i^P$ for all $i \geq k$, and again the polynomial hierarchy collapses to the k^{th}-level [MS72, Wra76].

A decision problem $\mathcal{P}$ is said to be Σ_k^{P}-hard if every problem in the class Σ_k^{P} can be polynomially reduced to it. Additionally, $\mathcal{P}$ is said to be Σ_k^{P}-complete if $\mathcal{P}$ is Σ_k^{P}-hard and it also belongs to the class Σ_k^{P}. Meyer and Stockmeyer [MS72], and Stockmeyer [Sto76] showed that each class of the polynomial hierarchy contains a problem which is complete for that class. They showed that the k-Alternating Quantifying Satisfiability problem is Σ_k^{P}-complete. An instance of the k-Alternating Quantifying Satisfiability problem consists of a Boolean expression φ over a set of boolean variables $\{x_{ij} \mid i \leq k, j \leq m\}$, and the task is to verify whether the following formula is true:

$$\exists x_{11} x_{12} \dots x_{1m} \quad \forall x_{21} x_{22} \dots x_{2m} \quad \exists x_{31} x_{32} \dots x_{3m} \quad \cdots \quad Q x_{k1} \dots x_{km} \quad \varphi$$

where the quantifier Q is either $\exists$ when k is odd or $\forall$ when k is even.

Using the polynomial hierarchy, we can pinpoint the exact hardness of a problem much better than simply restricting ourselves to P and NP. Under the assumption that the polynomial hierarchy does not collapse, the fact that a problem $\mathcal{P}$ is, say Σ_4^{P}-complete, captures much more information about its true complexity than just showing its NP-hardness.

3

Graph Orientation to Singly Connectedness

This chapter of the thesis is devoted to study a graph orientation problem called, SC-Orientation. Most of the results mentioned in this chapter is a joint work with my former colleague Tim Hartmann. Our work has been published at the 34th International Workshop on Combinatorial Algorithms (IWOCA) [HM23a]. Furthermore, a full version of the paper is also under review at the Journal of Computer and System Sciences (JCSS). Given that, this chapter contains additional results that are not part of the above mentioned publication. The notable differences are as follows. We give several examples illustrating the concepts of singly connectivity and sc-orientation in Section 3.3; this way we adhere to more detailed presentation of our work. We introduce the concepts of reverse orientation and unique sc-orientation in this chapter. Within Section 3.4, Lemma 3.10 and Lemma 3.11 are new. Additionally, Section 3.6.4 on the consequence of our dichotomy result, Section 3.7.3, and Section 3.7.4 are new. The results in Section 3.7.2 have been adapted with the help of newly added Lemma 3.11.

3.1 Graph Orientation Problems

Graph orientation problems are a well-known class of problems in which, given an undirected graph, the task is to decide whether we can assign a direction to each edge such that the resulting directed graph satisfies a particular desired property. Examples for such properties are acyclicity, strongly connectivity, existence of Eulerian circuit, etc. The graph orientation problems are important because of the emerging field of designing networks; some mathematical networks are modeled by directed graphs. Think of a rapidly developing city with an old streets layout. Due to the everyday increase in traffic demand, we need to adapt the streets existing network to accommodate the current rush. A well-established way of handling this issue is to convert all (or partial) streets into *one-way* streets by allocating a distinct direction to them, which we call an *orientation* (or *mixed-orientation*). In this particular case, the goal would be to orient the streets while maintaining the previously existing connectivity. This is a practical example of a graph orientation problem which is used in traffic control (check this application

in [Rob39]). When focused on converting all streets into one-way streets, the above mentioned problem reduces to finding a strongly connected orientation. Robbin's theorem, from 1939, states that there exists a strongly connected orientation of a graph if and only if the given undirected graph is 2-edge-connected [Rob39]. Further it is possible to find a strong-orientation of a graph in polynomial time, if one exists. Naturally, due to its practicality, the study of Robbin's theorem was generalized over the class of mixed graphs. Here, researchers focused on finding an extension of a partial orientation of a mixed graph to a full orientation such that the digraph thus obtained is strongly connected (see, for reference [BT80, CGT85]).

A *tournament* is a directed graph whose underlying undirected graph is a complete graph. The out-degree of a vertex in a tournament is referred to as a *score* of that vertex. Yet another graph orientation problem, from the practical point of view, is to convert a complete graph into a tournament which additionally has a score structure (also known as score sequence) with a dominance relation. If the vertices are assumed to be members of the society, then this means that there are members who dominate every other member of the society either directly or indirectly through a single intermediate member. Landau's theorem gives a complete characterization of tournaments in terms of its score sequence (see [Lan53, Lan51]).

More theoretically, consider a simple problem of converting an undirected graph to a directed acyclic graph by means of an orientation of edges. All simple graphs can be converted into directed acyclic graphs by simply considering a Depth-First search (DFS) tree on the graph and directing all edges from the ancestor to the descendant. This trivially solves the decision version of the graph orientation problem on all simple graphs for the desired property of acyclicity.

For further overview on the class of graph orientation problems, we refer the reader to a book by Bang-Jensen and Gutin [BG08, Chapter 11].

In this study, we consider 'singly-connected' as our desired property of the graph orientation problem.

Definition 3.1. A directed graph D is *singly connected* if for every *ordered* pair of vertices $s, t \in V$, there is at most one path from s to t in D.

Ironically, singly connected graphs can be disconnected. This property is motivated by the following examples.

First is that of a computer network with probabilistic behavior. Consider there are multiple paths that can lead to a destination. We desire to have a control over this model, and would like to know, deterministically, which path was executed in order to reach the destination. To accomplish this, one can convert the (undirected) network into a singly connected (directed) network to ensure if, with any positive probability, a destination is reached, then which path was taken. This way troubleshooting becomes easier as we know which node or link in the network has failed.

Another motivating example is to consider navigating a vast number of passengers at an airport. More often than not, we encounter ambiguous signs at airports. For example, a scenario in which a sign on one path says "Gates starting with C are this way", while another sign—in the exact opposite direction to the previous one—says "All gates are this way". A passenger who is seeking a route to gate C could be utterly confused by such signs. Assuming there was no 'logical' error in the signs, perhaps both paths lead to the gate C, however, one of them could also be a long detour. In order to achieve simplified passenger navigation and passenger comfort, it is desirable to have the signs in a way that a person only sees one sign at a time. So, there is a unique path if we follow the sign from point A to point B.

3.2 Associated Work

There has been a few studies around the singly connected graphs in the literature. The verification problem, which verifies whether a given directed graph is singly connected, is shown to be solvable in polynomial time [BC93, Khu99, DJ15].

In 2022, a study was conducted on the vertex deletion version of the singly connected problem [DKM+22]. Here the authors asked, given a *directed* graph D, does there exist a set of k vertices whose deletion renders a singly connected digraph. Alongside, they pointed a close link of the Singly Connected Vertex Deletion (SCVD) problem to the Feedback Vertex Set (FVS) problem. In FVS, we are given an undirected graph and the task is to find a set of at most k vertices, for a given integer k, whose removal makes the graph cycle-free. They argued how SCVD is *a* directed counterpart of FVS, which makes it theoretically important.[1] Our problem is a natural follow-up question in this direction which looks at the graph orientation version.

3.2.1 Contribution

In the upcoming sections, we prove that if a (triangle-free) graph has an sc-orientation,[2] then it also has an acyclic sc-orientation. Thus, the SC-Orientation problem could also be reformulated as: determine whether there is an acyclic sc-orientation of the graph. So, in addition to a graph being oriented to an acyclic digraph, SC-Orientation also demands the orientation to a singly connected digraph. As discussed earlier in Section 3.1, it is trivial to *find* an acyclic orientation of the graph. However, in contrast to that, we prove, the additional singly connected condition makes it harder to decide whether such an orientation even *exists* or not.

[1] The difference between Directed Feedback Vertex Set (DFVS) and SCVD is to decide whether there exist at most k vertices whose deletion renders an acyclic digraph and a digraph with at most one path between two vertices, respectively.

[2] We briefly defined "singly connected" property in Section 3.1. Sc-orientation is an orientation to achieve singly connectivity; all terms related to SC-Orientation are defined in Section 3.3.

In Section 3.6, we show dichotomy results on general graphs. As an upper bound, we show that, for any graph with girth at least twice its chromatic number, the graph is sc-orientable. Hence, the problem is tractable for such inputs. To account for this, we study SC-Orientation restricted to the graph class $\mathcal{G}_{g,c}$, the class of graphs that have girth g and chromatic number c, for some positive integers g, c. Our main result is that, for each pair $g, c \geq 3$, SC-Orientation restricted to $\mathcal{G}_{g,c}$ is either NP-complete or vacuously tractable (meaning, $\mathcal{G}_{g,c}$ consists of only yes-instances of the problem). For the record, we would like to note the rarity of such "NP-hard or trivially yes" proof. We do not pinpoint the exact boundary for the values of g and c where the transformation from P to NP-hardness occurs. The approach is to compile an NP-hardness proof that solely relies upon a single no-instance of the given girth g and chromatic number c.

Additionally, in Section 3.5, we prove that SC-Orientation is NP-complete even for planar graphs. Our hardness holds for all graphs restricted to girth at most 4. Complementing this result, we show that planar graphs of girth at least 5 are always sc-orientable; hence, the decision problem is tractable. Moreover, our NP-hardness reduction on planar graphs can be modified to show NP-hardness also on perfect graphs.

As further results, we provide polynomial-time algorithms for the SC-Orientation problem on some special graph classes such as distance hereditary graphs, chordal graphs, and cographs (see Section 3.7). For the class of distance hereditary graphs, we give a concise definition of the yes-instances by a set of finite forbidden subgraphs.

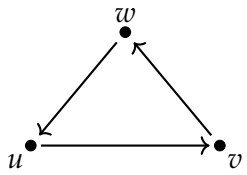
From the technical side, we provide two simplifications of the problem in Section 3.4. First, in triangle-free graphs, we restrict the search for an sc-orientation to the orientations that also avoid directed cycles. Second, we show that, in fact, we can assume that the input is triangle-free. Interestingly, we further show that triangle-free sc-orientable graphs can be characterized by a vertex ordering π, such that the corresponding orientation in which every edge is directed towards the vertex that is higher in the ordering is an sc-orientation.

3.3 Examples Illustrating Singly Connectivity

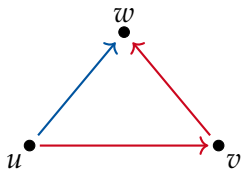
In this section we discuss, the concepts of *singly connected digraphs* and *singly connected orientations*. We give illustrations of multiple examples and build the concepts related to singly connectivity.

3.3.1 Singly Connected Digraphs

Recall the definition of singly connected digraphs: A directed graph D is singly connected if for every *ordered* pair of vertices $s, t \in V(D)$, there is at most one path from s to t in D. We illustrate the singly connected property of a digraph with the help of an example shown in the Figure 3.1. The left digraph has a cyclic orientation and contains, for any specific ordered pair of vertices, at most one path joining the ordered pair. Thus it is a singly connected digraph. Note that it is crucial to consider ordered pairs



singly connected digraph



non-singly connected digraph

Figure 3.1: Illustrating example of a singly connected and a non-singly connected digraph

of vertices for the definition of singly connected digraphs. On the other hand, the right digraph contains two disjoint paths joining the vertices u and w, namely, path uvw and path uw. Thus this digraph is not singly connected.

During a study in 2022 [DKM+22], the authors gave a characterization of the singly connected property of digraphs in the following way. They showed that, if a digraph D is *not* singly connected, then there exists a pair of vertices $u, v \in V(D)$ such that there are two paths from u to v, whose inner vertices are disjoint, that is say paths $P_1 = ux_1x_2x_3 \dots x_a v$ and $P_2 = uy_1y_2y_3 \dots y_b v$ where $x_i \neq y_j$ for all i, j. They call such paths *internally vertex disjoint paths*.

Lemma 3.2 ([DKM+22], Lemma 1). *A digraph D is not singly connected if and only if there exist two vertices $u, v \in V(D)$ such that there exist two internally vertex disjoint paths from u to v.*

We observe (from Lemma 3.2), since for each non-singly connected digraph D, there is a pair of vertices which bear two internally vertex disjoint paths P_1 and P_2 between them, these paths together form a cycle in the underlying undirected graph G_D of D. Thus, given a digraph, if one checks every cycle in the underlying undirected graph for singly connectivity, this gives a brute force algorithm to verify the singly connectivity of the digraph. Although, it is not efficient.

The problem of testing whether a digraph is singly connected or not was introduced in an exercise in the classical book "Introduction to Algorithms" by Cormen, Leiserson, Rivest and Stein [CLRS01]. Buchsbaum and Carlisle, in 1993, gave an algorithm running in time $O(n^2)$ [BC93]. Khuller, in 1999, proposed an algorithm for testing single connectedness using a similar approach [Khu99]. There have been other attempts to improve this result, for example see the paper by Dietzfelbinger and Jaberi from 2015 [DJ15]. They propose an $\mathcal{O}(\alpha \cdot \beta + m)$ algorithm, where α and β are the total number of sources and sinks, respectively, in the reduced graph obtained by first contracting each strongly connected component of the graph into one vertex and then eliminating vertices of in-degree or outdegree 1 exhaustively.

In the remainder of the chapter, we solely focus on *whether it is possible to make an undirected graph into singly connected digraph*, and if yes, then *how do we do it*?

3.3.2 Singly Connected Orientation

Let us define some notions that are relevant for the rest of the chapter.

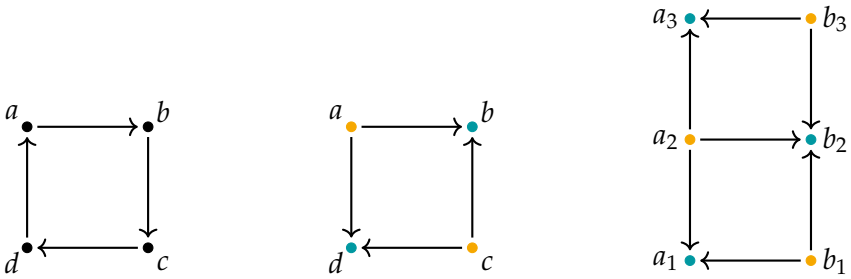
For an undirected graph G, an *orientation* is a mapping σ that maps each edge $\{u, v\} \in E(G)$ to either of its endpoints u or v, which results into a directed graph G_σ. So each edge $\{u, v\} \in E(G)$ becomes either (u, v) or (v, u), but not both, in the arc set $E(G_\sigma)$. We define $\sigma(\{u, v\}) := v$ when σ orients the edge $\{u, v\}$ towards the vertex v meaning we have the arc (u, v) in G_σ; similarly, $\sigma(\{u, v\}) = u$ implies we have the arc (v, u) in G_σ. For a path $P = u_1u_2u_3 \dots u_n$, an *alternate orientation* along P is the orientation in which for each vertex its incident arcs on the path are directed either towards the vertex or away from that vertex.

Definition 3.3. An orientation σ is an *sc-orientation* (singly connected orientation) if G_σ is singly connected. A graph G is said to be *sc-orientable* if there exists an sc-orientation of G. Moreover, we say that a graph G is *non-sc-orientable* if there does not exist any sc-orientation of G.

Let σ be an orientation which results into a digraph G_σ, then the *reverse orientation* $\widetilde{\sigma}$ of σ yields a digraph $G_{\widetilde{\sigma}}$ such that $A(G_{\widetilde{\sigma}}) = \{(u, v) \mid (v, u) \in A(G_\sigma)\}$. Similarly, given a digraph D, we denote the *reversed digraph* obtained by reversing the direction of each arc in D, by $\widetilde{D}$. Note that, D is singly connected if and only is $\widetilde{D}$ is singly connected. This is because of Lemma 3.2; D has at least two internally vertex disjoint paths from u to v if and only if $\widetilde{D}$ has at least two internally vertex disjoint paths from v to u. As a result, sc-orientation of a graph is *never* unique. However, in this text, abusing the term, we say a graph has a *unique sc-orientation* if it has a unique sc-orientation up to reverse orientation. If a graph has a unique sc-orientation, then fixing the direction of one edge fixes the orientation of all edges up to single-connectedness, that means as long as we are forcing sc-orientation, we get a unique orientation after fixing the direction of one edge.

Sc-orientation of a triangle. Consider a simple undirected graph G_1 wherein $V(G_1) = \{u, v, w\}$ and $E(G_1) = \{\{u, v\}, \{v, w\}, \{u, w\}\}$. This graph is also known as a *triangle*. Note that this is the underlying undirected graph of the digraphs mentioned in Figure 3.1. From the orientations illustrated in Figure 3.1, we know that there exists a singly connected orientation for G_1; the left orientation in Figure 3.1 is singly connected. Hence G_1 is sc-orientable. Moreover, for a triangle graph, the only sc-orientation possible is a cyclic orientation. However, we note that there are still two sc-orientations of G_1, each corresponding to the cycles uvw and uwv. But since those are reverse orientations, we safely say that a triangle has a unique sc-orientation.

Sc-orientation of a 4-cycle. A 4-cycle or C_4 is a graph consisting of $V(C_4) = \{a, b, c, d\}$ and $E(C_4) = \{\{a, b\}, \{b, c\}, \{c, d\}, \{a, d\}\}$. A C_4 realizes only two types of sc-orientations



(a) (b) (c)

Figure 3.2: Two types of sc-orientations of C_4: (a) cyclic orientation and (b) source-sink orientation. (c) contains the unique sc-orientation of a domino, also known as, a (2×3)-grid graph ($G_{2,3}$).

as given in Figure 3.2. One, σ_1, which returns $A(C_{4_{\sigma_1}}) = \{(a,b),(b,c),(c,d),(d,a)\}$ and the other, σ_2, which returns $A(C_{4_{\sigma_2}}) = \{(a,b),(c,b),(a,d),(c,d)\}$. It is worth noticing that σ_1 and σ_2 are not unique up to reverse orientations. We say σ_1 is a *cyclic orientation* and σ_2 is a *source-sink orientation*, and one is not obtained from another by means of reversing all the arcs.

Sc-orientation of a domino. The domino is obtained from two C_4 graphs by identifying exactly one edge. See Figure 3.2. Although the building blocks of a domino are C_4 graphs, there does not exists an sc-orientation of a domino wherein any induced C_4 has a cyclic orientation. As soon as we include one cyclic orientation of one of the 4-cycles, no matter what the orientation of the other 4-cycle is, we always get two disjoint paths joining an ordered pair of vertices in the graph. This is easy to verify by a trial and error method. It turns out that a domino also has a unique-sc-orientation. Moreover, each C_4 therein has a source-sink orientation. We use domino graphs as building blocks for a crucial structure required in deriving the hardness results in Section 3.5. Thus the unique-sc-orientation of a domino serves as an essential property in our main tool in the later sections.

A domino is a (2×3)-grid graph; *grid graphs* can be embedded by putting the vertices on the integer co-ordinates of a Cartesian plane, and there exists an edge between two vertices if and only if they are unit distance apart. An *$(m \times n)$-grid graph* can be embedded using m horizontal lines and n vertical lines, and is denoted by $G_{m,n}$.

Examples of non-sc-orientable graphs. In contrast to the above example, there are graphs for which no sc-orientation exists. Two such crucial examples are a *diamond* and a *house* graph (see Figure 3.3). It is easy to verify that no orientation of a diamond or a house graph is singly connected. An attempt to seek sc-orientation of a diamond will require each of its triangular subgraphs to form a directed cycle. Without loss of

(a) Diamond (b) House

Figure 3.3: Examples of non-sc-orientable graphs

generality, if the edge $\{b, d\}$ is oriented (b, d), this enforces the orientation of the rest of the arcs to be $(d, a), (a, b), (d, c)$ and (c, b). This results into two distinct paths from d to b, namely paths dab and dcb. Thus, for a diamond graph, there does not exist an sc-orientation.

Similarly, for a house graph (see for example Figure 3.3), let the chord $\{q, s\}$ be oriented (s, q). This enforces the triangular arcs to be (q, r) and (r, s). Additionally, as discussed earlier, the 4-cycle can either have a cyclic or source-sink orientation. If the 4-cycle has cyclic orientation, we have arcs $(s, q), (q, p), (p, t), (t, s), (q, r)$, and (r, s), which results into two distinct paths between vertices q and s, namely qrs and $qpts$. Otherwise, if the 4-cycle has source-sink orientation, we have arcs $(s, q), (p, q), (p, t), (s, t), (q, r)$, and (r, s), which results into two distinct paths between vertices p and t, namely pt and $pqrst$. Thus, for a house graph, there does not exist an sc-orientation.

Examples of trivial graph classes. There are numerous graph classes for which membership implies an sc-orientation. This includes the class of forests, cactus graphs, bipartite graphs, near-bipartite graphs, etc. We state the results on these graph classes below:

Forests do not contain cycles. Thus there cannot be two disjoint s-t-paths between any two vertices s and t in the graph.

Proposition 3.4. *Every orientation of a forest is an sc-orientation.*

Cactus graphs have a property that each edge in it is contained in at most one cycle, so the corresponding edge sets of all cycles are pair-wise disjoint. Thus, when each cycle is given a cyclic orientation, there cannot be two disjoint s-t-paths between any two vertices s and t in the graph.

Proposition 3.5. *For every cactus graph, giving a cyclic orientation to each cycle in the graph and any orientation to the rest of the edges results into an sc-orientation of the graph.*

For bipartite graphs, orienting all edges from one set of the bipartition to the other results into singly connected digraph.

Proposition 3.6. *Bipartite graphs are sc-orientable.*

All in all, inspiring from the notion of graph orientation problems discussed in Section 3.1, in this chapter, we indulge ourselves into orienting undirected graphs to make them singly connected. The chapter is devoted to study the following problem.

SC-Orientation (SCO)

Given: A simple undirected graph $G = (V, E)$.
Task: Decide whether G is sc-orientable.

Since the verification of singly connectivity of a digraph is polynomial-time solvable [BC93, Khu99, DJ15], for our SC-Orientation problem, this implies, that an sc-orientation of G serves as an NP-certificate. Thus we get the following result.

Lemma 3.7. SC-Orientation *is in* NP.

3.4 Preprocessing Techniques

In this section, we state a few technical observations that are useful for our later results. The upcoming lemma shows when searching for an sc-orientation of a graph, it suffices to search for an sc-orientation that additionally avoids directed cycles of length at least 4.

Lemma 3.8. *A graph G is sc-orientable if and only if there is an sc-orientation σ of G where G_σ contains no directed cycle of length at least 4.*

Proof. ($\Leftarrow$) The reverse direction is clear from the definition of sc-orientable graphs.

($\Rightarrow$) For the forward direction, since G is sc-orientable, let σ be any sc-orientation of G. If G_σ does not contain any directed cycle of size at least 4, then we are done. Otherwise let $v_1v_2 \dots v_kv_1$, $k \geq 4$ be a directed cycle C in G_σ. Let us denote the set of vertices of C by $V(C)$. Note that the induced graph $G_\sigma[V(C)]$ is exactly this cycle C and it does not contain any chords. Because a chord in the cycle gives us two paths between the endpoints of the chord, one followed through the arcs of the cycle, and the other consisting of the chord itself.

Now let the orientation σ' be same as σ except the orientation of the edges $\{v_1, v_2\}$ and $\{v_3, v_4\}$ reversed, i.e., unlike G_σ, the graph $G_{\sigma'}$ contains arcs (v_2, v_1) and (v_4, v_3). Let us refer to the set of newly oriented arcs $\{(v_2, v_1), (v_4, v_3)\}$ by A'. We claim that σ' is an sc-orientation and it does not create any new directed cycle in $G_{\sigma'}$ apart from those present in G_σ.

Let us first prove that $G_{\sigma'}$ is singly connected. Suppose the contrary, and let $s, t \in V(G_{\sigma'})$ be such that there are two vertex disjoint paths from s to t, P'_1 and P'_2, in $G_{\sigma'}$.

At least one of P_1', P_2' must use an arc from A', otherwise P_1' and P_2' are present in G_σ, contradicting to the fact that G_σ is singly connected. We consider the following cases:

- *Case 1:* Exactly one of P_1', P_2', say P_1', contains vertices from $V(C)$ and P_2' does not intersect with the vertex set $V(C)$. This implies there exists a path $P_2 = P_2'$ in G_σ. Now while tracing the path P_1' from s to t, consider the first vertex and the last vertex on P_1' that belongs to $V(C)$. Let the vertices be v_i and v_j respectively; there exists a path P joining v_i to v_j in the cycle C of G_σ. Then by replacing the path between v_i to v_j in P_1' by P, we can obtain a path P_1 from s to t in G_σ. Note that $P_1 \neq P_2$, as P_1 contains vertices from $V(C)$. Thus we get two disjoint paths in G_σ, contradicting to the fact that G_σ is singly connected.

- *Case 2:* Both P_1', P_2' intersect $V(C)$. Assume that either $s \notin V(C)$ or $t \notin V(C)$. Without loss of generality, consider $s \notin V(C)$. Tracing P_1' from s to t, let v_i be the first vertex that belongs to $V(C)$. Similarly, while tracing P_2' from s to t, let v_j be the first vertex that belongs to $V(C)$. If $v_i = v_j$, we get two paths from s to v_i in G_σ, a contradiction to σ being an sc-orientation. Thus $v_i \neq v_j$; there exists a path P from v_i to v_j in the cycle C. In this scenario, we get two paths from s to v_j in G_σ, one being the subpath P_2 of P_2' from s to v_j, and the other is obtained by concatenating the subpath $s \dots v_i$ of P_1' with P. Both are disjoint paths from s to v_j in G_σ, giving us a contradiction to σ being an sc-orientation. Similarly, if we assume that $t \notin V(C)$, then we attain a similar contradiction. Here, the argument follows by considering the last vertices on each of the paths P_1 and P_2 that belong to the vertex set $V(C)$, while tracing them from s to t.

- *Case 3:* Both $s, t \in V(C)$. Note that at least one of P_1', P_2', say P_1', contains at least one arc (u, v) such that $u \in V(C)$ and $v \notin V(C)$. This is because it is impossible to have two paths from s to t that consists of vertices only from $V(C)$, as there are no chords in $G_\sigma[V(C)]$, and thus no chords in $G_{\sigma'}[V(C)]$. Then while tracing P_1' from s to t, let v_i be the last vertex such that P_1' contains an arc (v_i, w_1), where $v_i \in V(C)$ and $w_1 \notin V(C)$. The vertex $v_j \in V(C)$ be such that P_1' contains a subpath $v_i w_1 \dots w_n v_j$ where $w_k \notin V(C)$ for $k \in [n]$. Then we obtain two paths from v_i to v_j in G_σ, namely $P_1 = v_i w_1 \dots w_n v_j$, and P_2 can be traversed on the cycle C. Thus giving two disjoints paths from v_i to v_j in G_σ, a contradiction.

So far we have proved that σ' is an sc-orientation. To prove our claim we only need to show that the newly oriented arcs from A' do not create new directed cycles in the graph $G_{\sigma'}$. Assume the contrary. If there is a new cycle C' in $G_{\sigma'}$ that was not present in G_σ, then this cycle must include an arc from A'. If C' contains only one arc from A', say (v_2, v_1), then there is a path from v_1 to v_2 in G_σ which does not contain the arc (v_1, v_2), thus we get two disjoint paths from v_1 to v_2 in G_σ, which is a contradiction to G_σ being singly connected.

Now if C' contains both the arcs from A', namely (v_2, v_1) and (v_4, v_3), then let $C' = v_2 v_1 w_1 w_2 \dots w_i v_4 v_3 w_{i+1} \dots w_j v_2$. Note that the vertices $w_1, w_2, \dots, w_j \notin \{v_1, v_2, v_3, v_4\}$.

Thus in this case, we get two disjoint paths from v_1 to v_4 in G_σ, namely $v_1v_2v_3v_4$ and $v_1w_1w_2 \dots w_iv_4$. This concludes the proof because each time we encounter a directed cycle of size at least 4, we can change the direction of two nonadjacent arcs, and thus reduce the total number of directed cycles in the graph by at least 1. □

Further, by the following lemma, we can remove triangles from the input graph unless it contains a diamond or a house as an induced subgraph. It is easy to check whether a graph G contains a diamond or a house graph as an induced subgraph. We can a priori check whether the given graph contains a diamond or a house as a subgraph, and if so, conclude that the instance is a no-instance of SC-Orientation. For each small graph H such as a diamond or a house graph, one can check whether there exists a graph homomorphism from H to G in polynomial time.

Lemma 3.9. *Let a (diamond, house)-free graph G contain a triangle uvw. Then G has an sc-orientation if and only if G', resulting from G by contracting the edges in uvw, has an sc-orientation.*

Proof. We contract the triangle uvw (by contracting edges $\{u,v\}$ $\{v,w\}$, and $\{u,w\}$) in G to a single vertex in G'. Let z be that single vertex in G'.

($\Rightarrow$) Since G is sc-orientable, let σ be an sc-orientation of G; the triangle uvw obtains a cyclic orientation in G_σ. We show that then σ' is an sc-orientation of G' where σ' is defined as follows:

$$\sigma'(\{x,y\}) = \begin{cases} \sigma(\{x,y\}) & \text{if } x,y \neq z, \\ \sigma(\{x,y'\}) & \text{if } y = z \text{ and } \{x,y'\} \in E(G) \text{ where } y' \in \{u,v,w\}. \end{cases}$$

Note that since G is a diamond-free graph, an edge $\{x,z\} \in E(G')$ implies there is a unique $y' \in \{u,v,w\}$ such that $\{x,y'\} \in E(G)$.

For the sake of contradiction, assume that σ' is not an sc-orientation. That means $G'_{\sigma'}$ is not singly connected, and due to Lemma 3.2, there are two internally vertex disjoint paths from some vertex $s \in V(G')$ to vertex $t \in V(G')$. Let these two s-t-paths in $G'_{\sigma'}$ be $P'_1 = sx_1x_2 \dots x_mt$, and $P'_2 = sy_1y_2 \dots y_nt$. Then either $z \in \{s,t\}$ or z belongs to exactly one of $\{x_1, x_2, \dots, x_m\}$ or $\{y_1, y_2, \dots, y_n\}$. If z does not belong to either of the paths, then P'_1, P'_2 are present in G_σ. This is not possible as G_σ is singly connected. If $z = s$, then there exist arcs $(a, x_1), (b, y_1)$ in G_σ where $a, b \in \{u,v,w\}$. We thus obtain two a-t-paths or two b-t-paths, depending on the orientation of the edges of triangle uvw, in G_σ. Similarly, if $z = t$, then there exist arcs $(x_m, a), (y_n, b)$ in G_σ where $a, b \in \{u,v,w\}$. We thus obtain two s-a-paths or two s-b-paths, depending on the orientation of the edges of triangle uvw in G_σ.

If z belongs to only one path, say z belongs to P'_1. Assume $z = x_i$. Then there are vertices $a, b \in \{u,v,w\}$, such that the arcs $(x_{i-1}, a), (b, x_{i+1})$ belong to G_σ. When $a = b$, $P_1 := P'_1$ is a path in G_σ. Otherwise $a \neq b$, then let P be an a-b-path obtained by traversing

the arcs of triangle uvw of G_σ. Then $P_1 = s \dots x_{i-1} P x_{i+1} \dots t$, and $P_2 := P_2'$ are disjoint s-t-paths in G_σ, hence, a contradiction.

($\Leftarrow$) Let σ' be an sc-orientation of G'. With the help of Lemma 3.8 we assume that $G'_{\sigma'}$ does not contain a directed cycle of length at least 4. We then show that σ is an sc-orientation of G where σ is obtained from σ', and for the edges of the triangle uvw in G, it orients the edges to obtain a cyclic triangle. Assume the contrary, there are two internally vertex disjoint s-t-paths P_1, P_2 in G_σ for some $s, t \in V(G)$. At least one of P_1, P_2 contains edges from triangle uvw and thus vertices from the set $\{u, v, w\}$. Then we distinguish among the following cases:

- *Case 1:* Exactly one path of P_1, P_2 goes through the vertices of the triangle uvw. Say only $P_1 = sv_1 \dots v_k t$ does so, and let the first and the last vertex in $P_1 \cap \{u, v, w\}$ be v_i and v_j, respectively. Note that $s, t \notin \{u, v, w\}$, as P_2 does not traverse through the vertices of uvw. Then let $P_2' := P_2$ and P_1' be same as P_1 but where the subpath $v_i \dots v_j$ is replaced by the vertex z in $G'_{\sigma'}$. Then P_1', P_2' are disjoint s-t-paths in $G'_{\sigma'}$, a contradiction that $G'_{\sigma'}$ is singly connected.

- *Case 2:* Both paths P_1, P_2 traverse through the vertices of the triangle uvw, but $s, t \notin \{u, v, w\}$. Let path $P_1 = sv_1 \dots v_m t$ and $P_2 = sw_1 \dots w_n t$. Then let v_i, w_i be the corresponding first vertices of P_1, P_2 that belong to $\{u, v, w\}$. Let path P_1' be the subpath of P_1 from s to $v_i (= z$ in $G')$, and let P_2' be the subpath of P_2 from s to $w_i (= z$ in $G')$. We obtain two disjoint s-z-paths in $G'_{\sigma'}$.

- *Case 3:* Paths P_1, P_2 both traverse through the triangle, and exactly one of s, t, say s, belongs to $\{u, v, w\}$. Here $t \notin \{u, v, w\}$. Let v_j be the last vertex of P_1 such that $v_j \in \{u, v, w\}$. Similarly w_j be the last vertex of P_2 such that $w_j \in \{u, v, w\}$. Then $P_1' = v_j (= z$ in $G') \dots t$, and $P_2' = w_j (= z$ in $G') \dots t$ are disjoint z-t-paths in $G'_{\sigma'}$. Thus we get a contradiction.

- *Case 4:* The remaining case is to consider $\{s, t\} \subset \{u, v, w\}$. Since the triangle uvw forms a directed cycle in G_σ, at least one of the paths P_1, P_2, say P_1, visits vertices not in $\{u, v, w\}$. Let $v_1 v_2 \dots v_i$ be a subpath of P_1 where $v_1, v_i \in \{u, v, w\}$ and $v_2, \dots, v_{i-1} \notin \{u, v, w\}$. Because G is (diamond, house)-free, $i \geq 5$. Then $G'_{\sigma'}$ contains a directed cycle $zv_2 \dots v_{i-1} z$ of length at least 4. A contradiction to our initial assumption that $G'_{\sigma'}$ does not contain any directed cycle of length at least 4.

Since all cases led to a contradiction, G_σ must be singly connected. □

As a consequence of Lemma 3.9, it is easy to either reduce an instance of SC-Orientation into an instance with girth at least 4 or return that it is a no-instance. Thus, let us now restrict ourselves to instances of girth at least 4. Under this assumption, Lemma 3.8 states that if a graph is sc-orientable then there exists an acyclic sc-orientation. From here, it is easy to work out that if we restrict the acyclic orientation σ to a cycle C in G, i.e. when considering C_σ, there are always two source vertices and

two sink vertices in the directed cycle C_σ. However, what is remarkable is the following result. Given a graph of girth ≥ 4, we can characterize an sc-orientable graph by an ordering of its vertices $\pi\colon V(G) \to \{1,\dots,|V(G)|\}$. We show that, the corresponding orientation that orients each edge to the endpoint that is higher in the ordering is in fact an sc-orientation. Moreover, the converse also holds.

Lemma 3.10. *Let G be an undirected triangle-free graph. Then G is sc-orientable if and only if there exists a vertex ordering, that is a bijective mapping, $\pi\colon V(G) \to \{1,\dots,|V(G)|\}$ such that every cycle C in G contains distinct vertices $u_1, u_2 \in V(C)$ where every $v \in N_C(u_i), i \in \{1,2\}$ has $\pi(u_i) < \pi(v)$.*

Proof. It suffices to show that there are at least two vertices in every cycle which are lower in the ordering π than their neighbors in the cycle.

($\Rightarrow$) Let σ be an sc-orientation of G. We may assume that G_σ has no directed cycle due to Lemma 3.8.

Let us construct a vertex-ordering π such that $\sigma(\{w_1,w_2\}) = w_2$ if and only if $\pi(w_1) < \pi(w_2)$. We may define an ordering for each component of G separately. So assume that G is connected. If $V(G) = \{v\}$, define $\pi(v) := 1$. Else, find a vertex v that is a sink in G_σ as follows. Pick an arbitrary vertex u. If it is not already a sink, there is at least one out-going edge (u,v), and we consider v instead and repeat this procedure. Since G_σ has no directed cycle, we may not visit a vertex twice, and hence eventually have to find a sink v. Then set $\pi(v) := |V(G)|$. Recursively define $\pi(w)$ for $w \in V(G) \setminus \{v\}$ by restricting to graph $G \setminus \{v\}$. This way, $\sigma(\{w_1,w_2\}) = w_2$ if and only if $\pi(w_1) < \pi(w_2)$, as desired.

Given such an ordering π, all it remains to prove is that each cycle C in G has two distinct vertices $u_1, u_2 \in V(C)$ where $v \in N_C(u_i), i \in \{1,2\}$ has $\pi(u_i) < \pi(v)$. Assume contrary, and let C be a cycle which does not contain two distinct vertices u_1, u_2 as desired. Let s and t be the vertices of $V(C)$ with the lowest and the highest values assigned by the mapping π in C, respectively. Let $N_C(s) = \{a_1, b_1\}$. Clearly, s satisfies $\pi(s) < \pi(a_1)$ and $\pi(s) < \pi(b_1)$. But due to our assumption no other vertex $u' \in V(C)$ satisfies $\pi(u') < \pi(v)$ for both $v \in N_C(u')$. This implies the two paths from s to t in $G[C]$, namely $P_1 = sa_1a_2\dots a_nt$ and $P_2 = sb_1b_2\dots b_mt$ have vertices which satisfy $\pi(s) < \pi(a_1) < \dots < \pi(a_n) < \pi(t)$ and $\pi(s) < \pi(b_1) < \dots < \pi(b_m) < \pi(t)$, respectively. Due to our construction of π, this implies the paths P_1 and P_2 are also the directed paths in G_σ. And we get two disjoint paths from s to t which contradicts that σ is an sc-orientation.

($\Leftarrow$) Let a permutation $\pi\colon V(G) \to \{1,\dots,|V(G)|\}$ be such that every cycle C in G contains distinct vertices u_1, u_2 where $v \in N_C(u_i), i \in \{1,2\}$ satisfies $\pi(u_i) < \pi(v)$. We define an orientation σ by setting $\sigma(\{u,v\}) = \arg\max_{w\in\{u,v\}} \pi(w)$ for every edge $\{u,v\} \in E(G)$, and claim that G_σ is singly connected. Assuming the contrary, $s,t \in V(G_\sigma)$ be such that there are internally vertex-disjoint s-t-paths P_1, P_2 in G_σ by

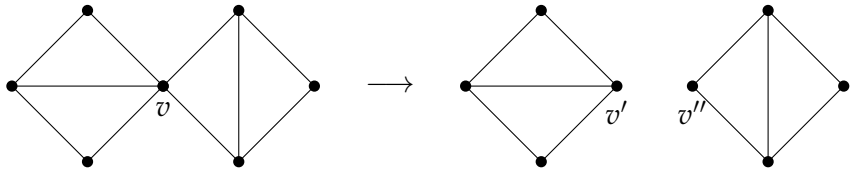


Figure 3.4: Splitting the graph across a cut-vertex and reducing it to multiple instances.

Lemma 3.2. Then P_1, P_2 together from a cycle C' in the underlying undirected graph G. This cycle C' has two distinct vertices $u'_1, u'_2 \in V(C')$ such that $v' \in N_C(u'_i), i \in \{1,2\}$, and $\pi(u'_i) < \pi(v')$. Note that, this implies both u'_1, u'_2 are source vertices of the directed paths P_1 or P_2. However, no vertex $v \in V(C') \setminus \{s\}$ will satisfy $v \in \{u'_1, u'_2\}$, because they are either the inner vertices of the directed paths, or the endpoint t which is a sink node. □

It is trivial to see that for an arbitrary instance of SC-Orientation, it is enough to solve it on each connected component of the graph. However, in what follows we show that, in fact, it is enough to solve the problem on 2-vertex-connected graphs. One can rely on an algorithm on 2-vertex-connected instances for any possible instance of SC-Orientation.

Lemma 3.11. *Each instance of* SC-Orientation *can be reduced to solving the problem on several* 2*-vertex-connected graphs.*

Proof. Recall that 2-vertex-connected graphs are those which are still connected after deletion of any single vertex from the vertex set; these are also called 2-connected graphs.

If the given instance $G = (V, E)$ is not 2-vertex-connected, then there is a vertex, also known as a *cut vertex*, $v \in V(G)$ such that deletion of v renders at least 2 connected components in $G \setminus \{v\}$. For each cut vertex, our preprocessing technique reduces the instance by splitting it across the cut vertex, that is, we consider each connected component of $G \setminus \{v\}$, add a copy of v to it along with its incident edges with the vertices of the component. For illustration see Figure 3.4.

The reduced instance is a set of k-many 2-vertex connected subgraphs of G. We claim that the G is a yes-instance of SC-Orientation if and only if all of the k reduced components are yes-instances of SC-Orientation.

($\Rightarrow$) The forward direction is easy to see because each reduced component is a subgraph of G and thus an sc-orientation of G restricted to the component gives us an sc-orientation of that component. This holds of all reduced components.

($\Leftarrow$) Note that the edge set of G is partitioned into the edge sets of the reduced components. We start with fixing an sc-orientation of each reduced component. We further exploit the fact that each edge $e \in E(G)$ belongs to exactly one of the reduced components. Thus, we construct an orientation of G by directing each edge e as per

the sc-orientation of the reduced component it belongs to. We are now left to show that the resulting orientation of G is indeed an sc-orientation. If not, then according to Lemma 3.2, there exist two vertices $u_1, u_2 \in V(G)$ such that there are two internally vertex disjoint paths from u_1 to u_2. This either implies that u_1 and u_2 belong to the same reduced component or two-different ones. In the former case, it contradicts the sc-orientation of the reduced component, and in the latter all paths must pass through the cut vertex v, thus there cannot be such two internally vertex disjoint paths. □

To conclude this section, below we give an example of a 2-vertex-connected non-sc-orientable graph with girth 4.

Example 3.12. Figure 3.5 contains an example of a non-sc-orientable graph with girth four. Here, we state two different embeddings of a Möbius ladder on 8 vertices.

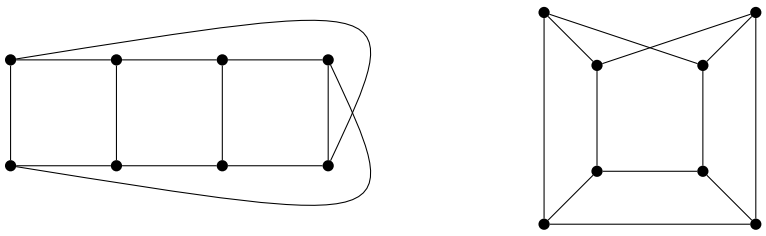


Figure 3.5: A non-sc-orientable graph with girth 4 and chromatic number 3: a Möbius ladder on 8 vertices.

3.5 NP-Hardness on Planar Graphs

This section derives NP-hardness of SC-Orientation on planar graphs. As we have previously discussed in Section 3.3.2, since verification of singly connectivity of a digraph can be done in polynomial time [BC93, Khu99, DJ15], SC-Orientation is in NP. To show hardness, we give a reduction from a variant (to be shortly defined) of the Planar 3-SAT problem [Lic82]. Given a 3-SAT formula φ, a *variable-clause graph* corresponding to φ is a graph obtained by taking the variables' and clauses' representatives as the vertex set, and the edge set contains an edge between a variable and a clause representative nodes if the variable or its negation appears in the clause. We use the variant of 3-SAT wherein there exists a planar embedding of the variable-clause graph of the given boolean formula, and there are some additional edges (only connecting clause nodes) which form a cycle traversing through all clause nodes. We denote the cycle traversing through all clause nodes as *clause-cycle*, and the problem as Clause-linked Planar 3-SAT. Below we give the formal definition of the Clause-linked Planar 3-SAT problem.

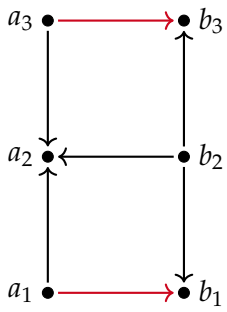


Figure 3.6: A Ladder showcasing the coupling edges. This graph is a domino; it is also known as a (2×3)-grid graph denoted by $G_{2,3}$.

Clause-linked Planar 3-SAT

Given: A boolean formula φ in conjunctive normal form (CNF) with clauses $C = \{c_1, \ldots, c_m\}$ wherein each clause contains exactly three literals on the set of variables $X = \{x_1, x_2, \ldots, x_n\}$, and a planar embedding of the variable-clause graph together with a clause-cycle.

Task: Verify whether φ is satisfiable?

Kratochvíl, Lubiw, and Nesetril proved the NP-hardness of the Clause-linked Planar 3-SAT problem; they denoted this problem by Planar Cycle 3-SAT [KLN91].

Theorem 3.13 ([KLN91], Proposition 4.1)**.** *The* Clause-linked Planar 3-SAT *problem is* NP*-complete.*

Now, we give a reduction from the above mentioned NP-hard problem, Clause-linked Planar 3-SAT, to the SC-Orientation problem in order to prove its NP-hardness.

Theorem 3.14. SC-Orientation *is* NP*-complete even for planar graphs.*

Proof. Our idea is to take the planar embedding of the given boolean formula φ, and use this as a backbone for our construction.

Let us first define η to be an ordering of clause nodes such that the clause-cycle traverses through C as per the ordering η. Now, let the given planar embedding of φ be Π, and the underlying variable-clause graph be $G_\eta(\varphi)$. For the reduction, we define a clause gadget and a variable gadget to replace the clause nodes and variable nodes in Π. Further we define how to connect the clause and variable gadgets corresponding to the edges in $G_\eta(\varphi)$. To begin with, let us start with the following interesting graph structure:

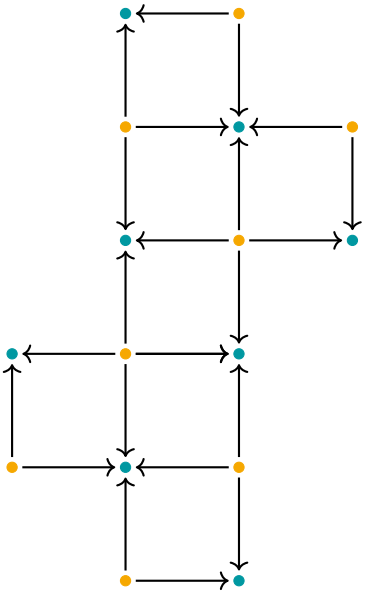


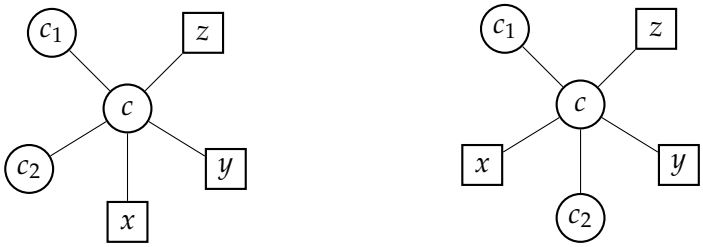
Figure 3.7: Extended Ladder ($\mathcal{L}$)

Ladder. Recall our investigation about the sc-orientation of a domino from Section 3.3.2. A domino, also known as a 2×3-grid graph and denoted by $G_{2,3}$ (see Figure 3.6), has a unique sc-orientation up to reverse orientation. As a consequence, if we fix the orientation of one edge, the orientations of all other edges is fixed up to singly-connectedness (every vertex becomes either a sink or a source of the oriented domino). We capture this compelling nature of influencing the orientation of edges in the following definition. We are specifically interested in a special pair of edges $\{a_1, b_1\}$ and $\{a_3, b_3\}$ like the bottom edge and the top edge in the Figure 3.6 which additionally satisfies the following conditions.

1. (a_1, b_1) and (a_2, b_2) are *coupled*, that is, for every sc-orientation σ, either $\sigma(\{a_i, b_i\}) = b_i$ for $i \in \{1, 2\}$, or $\sigma(\{a_i, b_i\}) = a_i$ for $i \in \{1, 2\}$, and

2. there is an sc-orientation σ without any directed path from $\{a_1, b_1\}$ to $\{a_2, b_2\}$. That is, in the directed graph G_σ, one cannot reach from the edge $\{a_1, b_1\}$ to the edge $\{a_2, b_2\}$.

Since the directions of coupling edges matter, we always refer to them in ordered pairs. Notice the embedding of domino given in Figure 3.6, the bottom edge and the top edge of the domino are coupled in orderings (a_1, b_1) and (a_3, b_3). The coupling property is transitive, in the sense, if (u_1, v_1) and (u_2, v_2) are coupled, and (u_2, v_2) and (u_3, v_3) are coupled, then so are (u_1, v_1) and (u_3, v_3). If we take two dominos and identify the top edge of one to the bottom edge of another (forming a 2×5-grid graph), the very top and the very bottom edges are also coupled. Further extrapolation can be made to form a *ladder* (a $2 \times n$-grid) for $n \geq 3$. Moreover, note that the bottom and the middle edges



(a) type 0 (b) type 1

(c) type 2 (d) type 3

Figure 3.8: Different types of clause nodes according to their neighborhoods in the planar embedding Π of $G_\eta(\varphi)$.

of a domino are in the reverse direction in every sc-orientation. Hence they are coupled in (a_1, b_1) and (b_2, a_2), or we say they are *reverse-coupled* in (a_1, b_1) and (a_2, b_2).

The coupling property of edges in dominos helps us attach (in any direction) as many 4-cycles as we want to a ladder without affecting the unique sc-orientations of the structure. Let us call the ladder with the additional 4-cycles an *extended ladder* $(\mathcal{L})$ (see Figure 3.7). For our reference, let us call the bottom of the ladder, in the embedding given in Figure 3.7, the 0^{th}-step. Let us call the set of edges which are coupled with the 0^{th}-step by *even edges*, and the set of reverse-coupled edges by *odd edges*. All in all, if we fix the orientation of the 0^{th}-step, the orientation of the whole ladder is fixed up to an sc-orientation. Let us construct a large enough extended ladder $\mathcal{L}_U$, we call this, the universal ladder.

Clause gadget. In the planar embedding Π, each clause node c is adjacent to three variable nodes, say x, y, z, and two more clause nodes, say c_1 and c_2. There are four types in which c can be surrounded by nodes x, y, z, c_1, c_2 in Π. These four types depend on the total number of variable nodes from $\{x, y, z\}$ that are embedded in the inner part of the clause cycle in the planar embedding Π. The clause c is type 0 when none of x, y, and z is embedded in the inner part of the clause cycle. Similarly, it is type 1, type 2, or type 3 when either 1, 2, or 3 variable nodes from $\{x, y, z\}$ are embedded in the inner part of the clause cycle, respectively. For reference see Figure 3.8. For a replacement to each of these four types of configurations, we define four different clause gadgets (see Figure 3.9 and Figure 3.10). For $c \in C$, the basic structure of the clause gadget

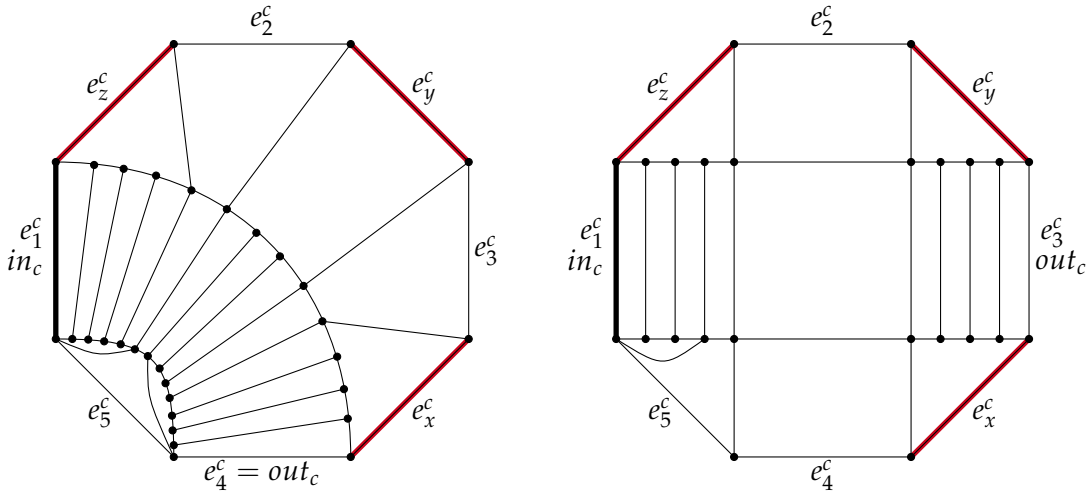


Figure 3.9: Clause gadget for type 0 (left) and type 1 (right)

G_c includes an 8-cycle which contains 3 edges e_x^c, e_y^c, and e_z^c (corresponding to x, y, and z), and five generic edges $e_1^c, e_2^c, e_3^c, e_4^c$, e_5^c. In addition, the universal ladder $\mathcal{L}_U$ passes through the gadget as shown in the Figures 3.9 and 3.10. For each of the four types of clause gadgets, we distinguish two edges of the clause gadget from where the universal ladder enters and exits by the notations (in_c) and (out_c), respectively.

Variable gadget. Each variable node x in Π is replaced by a sufficiently large *ladder-cycle* $\mathcal{LC}_x$, see Figure 3.11. The bold edges in Figure 3.11 correspond to the 0^{th}-steps of $\mathcal{LC}_x$. The orientation of all 0^{th}-steps in $\mathcal{LC}_x$ can either be clockwise or anticlockwise in an sc-orientation of the graph. This is because of the cyclic ladder formed in the gadget. The two types of orientations are in one-to-one correspondence with the two boolean assignments of the variable in the 3-SAT formula.

Construction. We make the following replacements while constructing our input graph G of SCO from the planar embedding Π given in the input of the Clause-linked Planar 3-SAT problem:

- Replace each clause node c in Π by a clause gadget G_c in G.
- Replace each variable node x in Π by a variable gadget $\mathcal{LC}_x$ in G.
- Replace each edge $\{x, c\}$ in Π by extending a ladder from the variable gadget $\mathcal{LC}_x$ to the edge e_x^c of the clause gadget G_c. If $\bar{x}$ appears in C, then we identify an odd step of $\mathcal{LC}_x$ to e_x^c. If x appears in c, then we identify an even step of $\mathcal{LC}_x$ to the corresponding edge e_x^c.
- For an edge $\{c_i, c_j\}$ in Π, where c_i, c_j are clauses and c_i appears before c_j in the clause-cycle η, add a ladder $G_{2,4}$ having bottom edge (a_1, b_1) and top edge (a_2, b_2).

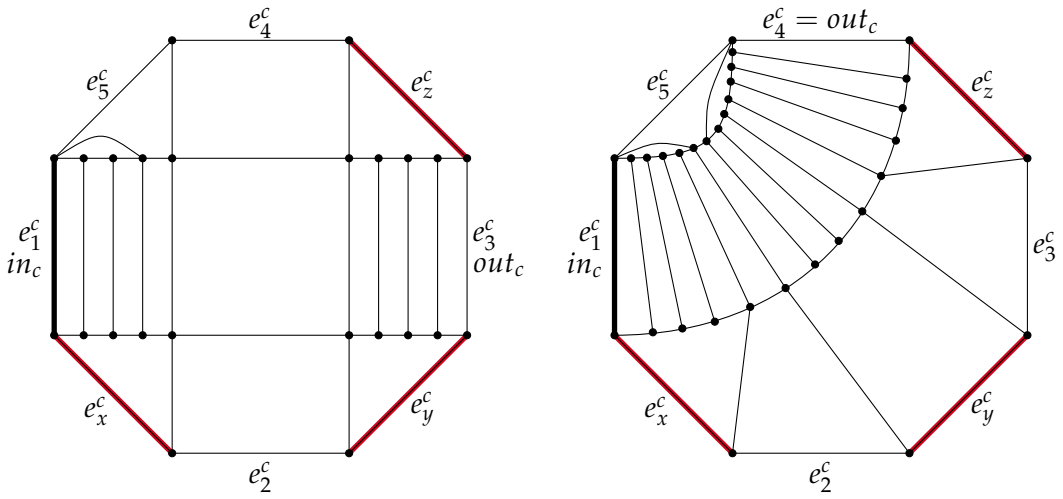


Figure 3.10: Clause gadget for type 2 (left) and type 3 (right)

Identify (a_1, b_1) with the (out_{c_i}) and (a_2, b_2) with the (in_{c_j}), while maintaining the planarity. Moreover, let us fix the 0^{th}-step of the universal ladder $\mathcal{L}_U$ to be the (in_{c_1}) edge of the clause gadget G_{c_1}.

Note that all clause and variable gadgets are planar, and the edge replacements by ladder also maintains the planarity of the resulting graph.

We remain to prove that the Clause-linked Planar 3-SAT instance is a yes-instance if and only if the corresponding constructed instance of the SC-Orientation problem is a yes-instance.

Correctness of the NP-hardness reduction. As per our construction of the clause gadgets, for each structure G_c, the (in_c) and (out_c) edges together are either oriented clockwise or anticlockwise in every sc-orientation of G. Moreover, all the (in_c) edges are oriented in one direction for all $c \in C$, and all the (out_c) edges are oriented in the other direction. If we fix the orientation of the 0^{th}-step of $\mathcal{L}_U$, it fixes the orientation of all edges associated with $\mathcal{L}_U$, and consequently fixes the orientation of the (in_c) and (out_c) edges for all $c \in C$. It also fixes the orientation of all generic edges in G_c for all $c \in C$. Refer to Figure 3.12 for the illustrations of the clockwise and anticlockwise orientations of (in_c) and (out_c) pairs of the clause gadget. For all clause gadgets, the orientation is either clockwise or it is anticlockwise; the orientation is fixed by the unique sc-orientation of the universal ladder $\mathcal{L}_U$ and is determined by the orientation of the 0^{th}-step of $\mathcal{L}_U$. We now prove that the Clause-linked Planar 3-SAT instance is a yes-instance if and only if the corresponding constructed instance of the SC-Orientation problem is a yes-instance.

($\Rightarrow$) Let $\Gamma\colon X \to \{0, 1\}$ be an assignment of variables in X such that φ is satisfied. Let us fix the orientation of the 0^{th}-step of $\mathcal{L}_U$ such that for $c \in C$, the (in_c, out_c) pairs are oriented anticlockwise (in this setting, each clause gadget has the anticlockwise ori-

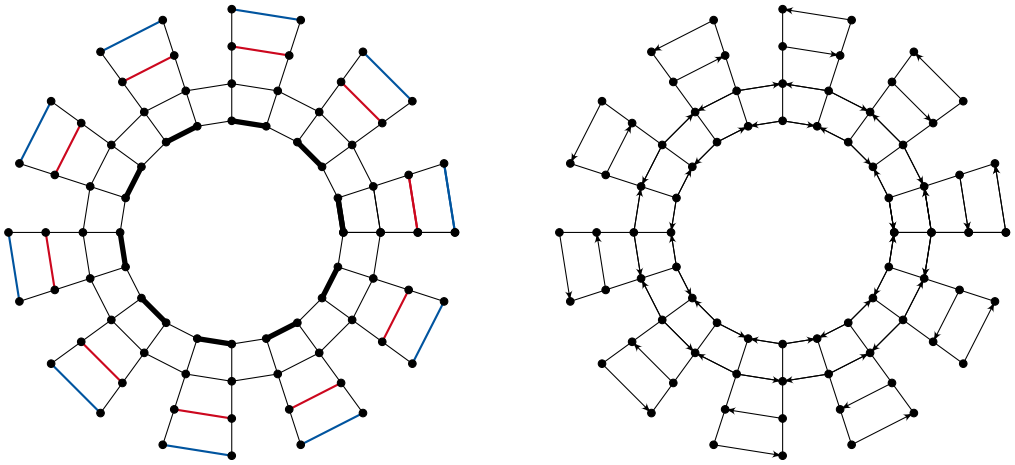


Figure 3.11: Variable gadget (left), denoted by $\mathcal{LC}_x$ for a corresponding variable x, and its sc-orientation (right). The 0^{th}-steps of the variable gadget are highlighted in bold, whereas the even steps and the odd steps are colored red and blue, respectively. For an sc-orientation, the orientation of a single edge pre-determines the orientation of all other edges. There are only two possible sc-orientations of the variable gadget, depending on the the 0^{th}-steps are oriented clockwise or anticlockwise.

entation from Figure 3.12). We orient the 0^{th}-steps of $\mathcal{LC}_x$ anticlockwise if $\Gamma(x) = 1$, and clockwise if $\Gamma(x) = 0$. We claim the resulting digraph D thus formed is singly connected. Note that, according to our construction, for an sc-orientation, the 0^{th}-step of $\mathcal{L}_U$, and the 0^{th}-steps of $\mathcal{LC}_x, x \in X$, influence the orientation of all other edges in G. Thus, the only possibility of forming two paths between a pair of vertices is in the clause gadgets: through paths $P_1 = e^c_5$ and $P_2 = e^c_1 e^c_x e^c_2 e^c_y e^c_3 e^c_z e^c_4$ (or $P_2 = e^c_4 e^c_x e^c_3 e^c_y e^c_2 e^c_z e^c_1$). To avoid path P_2, at least one edge in $\{e^c_x, e^c_y, e^c_z\}$ has to be oriented in the reverse direction to that of the generic edges in the path. This always happens because one of $\{x, y, z\}$, say x satisfies c, and hence according to the construction e^c_x gets the reverse direction. All other undirected cycles in G contain at least 2 ladder vertices each with 2 ladder edges incident on them within the cycle. Moreover, the incident ladder edges make them both either sinks or sources of the cycle. Thus, there are no two directed paths forming in this cycle. This results into an sc-orientation D of G.

($\Leftarrow$) If G is a yes-instance of SC-Orientation, then for each clause gadget, there is at least one variable edge, say e^c_x, that is oriented in the opposite direction to that of the edges $e^c_1, e^c_2, e^c_3, e^c_4$; this avoids formation of two disjoint paths between the endpoints of e^c_5 of the clause gadget. The orientation of all variable edges, e^c_x, e^c_y, e^c_z, depend on their respective variable-gadgets. In the sc-orientation of G, consider the orientation of the 0^{th}-step of $\mathcal{L}_U$, that is the orientation of (in_{c_1}, out_{c_1}) pair, and the orientations of the 0^{th}-steps of the variable-cycles. If the orientations of $\mathcal{L}_U$ and $\mathcal{LC}_x$ are the same, then assign value $\Gamma'(x) = 1$, otherwise assign $\Gamma'(x) = 0$. We claim, the resulting assignment Γ' is a satisfying assignment of φ. Note that each variable is either set to 1 or 0 in Γ'. Moreover,

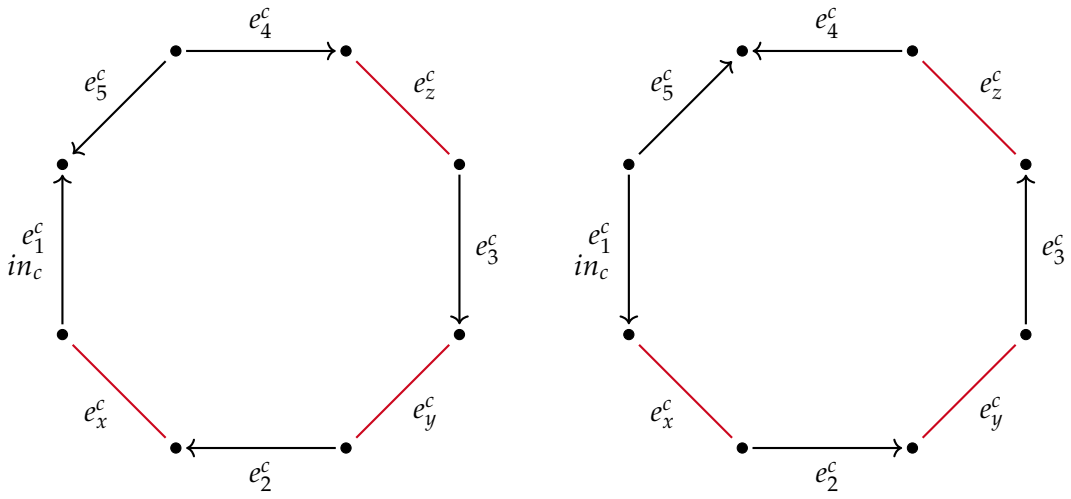


Figure 3.12: Clockwise (left) and anticlockwise (right) orientations of the in_c and out_c edges. Depending on the type 1 or type 2 clause gadget used, the out_c edge is either e_3^c or e_4^c respectively.

for each clause $c \in C$, there is at least one literal $x \in c$ which avoids the formation of paths P_1 and P_2. The corresponding literal which prevents these multi-paths is set to 1 in Γ', and hence each clause of φ is satisfied by assignment Γ'.

Since all gadgets have size $\mathcal{O}(|\varphi|)$, our reduction is polynomial. □

Observe that, the reduction used in the above theorem produces the graphs in the reduced instance to be of girth 4 and not less. Thus, in turn, we end up proving hardness for the class of graphs of girth exactly 4. However, as a consequence of Theorem 3.14, one can indeed prove that the hardness also holds for graphs of girth 3. For this, just add an extra triangle in the construction of Theorem 3.14. Since the additional triangle is an isolated component, it always gets a cyclic orientation and does not interfere in deciding whether the rest of the graph is sc-orientable or not. Consequently we get the following result as a corollary.

Corollary 3.15. SC-Orientation *is* NP*-complete for graphs of girth 3 and 4, even if planar.*

3.6 Dichotomy between the Girth and the Coloring Number

Notably, up until now the NP-hardness reduction heavily relied on the ladder structure which consists of cycles of length 4. Hence the constructed instances in Section 3.5 have girth 4. Recall the *girth* of a graph is the length of its shortest cycle. Now the question is, can we extend this result to girth 5 graphs or for graphs of even higher girth?

3.6.1 Planar Graphs With Girth at Least 5 are SC-Orientable

In this section, we prove an even stronger statement: Every *near-bipartite* graph G of girth at least 5 is sc-orientable. Recall that a graph is near-bipartite if it has an independent feedback vertex set, that is, its vertex set can be partitioned into two sets such that one is an independent set and the other induces a forest. We further show that, unless $\mathsf{P} = \mathsf{NP}$, the NP-hardness proof cannot be extended to planar graphs of girth 5, since, as we show below, such graphs are always sc-orientable.

Theorem 3.16. *Near-bipartite graphs of girth at least* 5 *are sc-orientable.*

Proof. To prove this theorem, it suffices to give a valid sc-orientation σ of any near-bipartite graph G. Let $I \dot\cup F$ be a partition of $V(G)$ such that the vertex subset I is an independent set of G, and the induced graph $G[F]$ is a forest. To obtain the digraph G_σ corresponding to an orientation σ, orient all edges that are incident to I away from I; hence all the vertices in I are sources in G_σ. Fix a coloring of $G[F]$ with colors $\{1, 2\}$ such that no two adjacent vertices have the same color. Thus all edges in $G[F]$ join a vertex with color 1 to anther vertex with color 2. Further orient the edges belonging to $G[F]$ towards the vertex with color 2; hence all vertices in color class 2 are sinks in G_σ.

We claim that G_σ is singly connected. Assuming otherwise, let the vertices $s, t \in V(G_\sigma)$ be connected by two internally vertex disjoint paths P_1, P_2. Since P_1, P_2 form a cycle in the underlying undirected graph G and G has girth at least 5, at least one of P_1, P_2, say P_1, has length at least 3. Let s, v_1, v_2, v_3 be the first 4 vertices of P_1. Note that $v_1, v_2 \notin I$, because they both have an incoming arc in G_σ and each vertex in I is a source in G_σ. Thus $v_1, v_2 \in F$. And since they are adjacent to each other, exactly one them must belong to the color class 2 in $G[F]$. Hence one of v_1, v_2 is a sink in G_σ. This contradicts to the existence of path P_1, because they both have an outgoing arc in G_σ. Thus G_σ is singly connected. □

Since planar graphs of girth at least 5 are near-bipartite [BG01], we get the following corollary from Theorem 3.16.

Corollary 3.17. *Planar graphs with girth at least 5 are sc-orientable.*

At this point, it is worth noticing that every near-bipartite graph has chromatic number at most 3. We use the ideas from this section to derive some upper bounds in the next section.

3.6.2 Upper Bounds

Extrapolating the ideas discussed in Section 3.6.1, here we show, given a general graph if the girth of the graph is at least twice its coloring number, the graph is sc-orientable.

Theorem 3.18. *Let* $\text{girth}(G) \geq 2\chi(G) - 1$. *Then* G *is sc-orientable.*

Proof. Again, to prove the theorem, it suffices to give a valid sc-orientation σ of the graph. For a graph G, consider its proper vertex-coloring given by a mapping $f\colon V(G) \to \{1, \ldots, \chi(G)\}$. Let σ be an orientation which orients each edge $\{u, v\}$ to the vertex which corresponds to the higher color class, that is, we set $\sigma(\{u, v\}) = \arg\max_{w \in \{u,v\}} f(w)$.

We claim that G_σ is singly connected. Assuming the contrary, let vertices $s, t \in V(G_\sigma)$ with two disjoint s-t-paths P_1, P_2 in G_σ. Let $P_1 = sv_1v_2 \ldots v_nt$ and $P_2 = sw_1w_2 \ldots w_mt$. Then P_1, P_2 together form a cycle $sv_1v_2 \ldots v_ntw_mw_{m-1} \ldots w_2w_1s$ of length at least $\text{girth}(G)$ in the underlying undirected graph G. Moreover, since $\text{girth}(G) \geq 2\chi(G) - 1$, at least one of the paths, say P_1, has length at least $\chi(G)$. Thus P_1 consists of more than $\chi(G)$ many vertices. By pigeonhole principle, this implies, there are at least two vertices $v_i, v_j \in V(P_1)$ with $f(v_i) = f(v_j)$. Without loss of generality say v_i appears before v_j while tracing P_1 from s to t. To form an s-t-path, we know that v_i's successor/out-neighbor v_{i+1} on path P_1 must have a higher color than v_i. With the same logic, each v_{i+k} has a strictly higher color than v_{i+k-1} in the mapping f. This contradicts $f(v_j) = f(v_i)$. Thus we cannot have two disjoint s-t-paths in G_σ. As a result, G is sc-orientable. □

As a remark, by Grötzsch's Theorem every planar graph with girth at least 4 is 3-colorable [Grö59]. Hence, Corollary 3.17 also follows from Theorem 3.18.

3.6.3 Coupling Gadgets and the Respective Hardness

As of now, we do not know whether there even exists a non-sc-orientable graph of girth at least 5. Intriguingly, however, we do not need to answer this question to follow a dichotomy between the polynomial-time decidability and NP-completeness. That is, for every $g, c \geq 3$, we show that, SC-Orientation when restricted to graphs of an arbitrary girth g and an arbitrary coloring number c is either NP-complete or trivially solvable (as in, all instances are yes-instances). Let us denote the class of graphs of girth g and chromatic number c by $\mathcal{G}_{g,c}$. Our proof does not pinpoint the exact boundary between polynomial-time decidability and NP-completeness. From Theorem 3.18, we only know, for a fixed value of the chromatic number c', there is an upper bound g' for the values of girth such that the hardness is only unclear on $\mathcal{G}_{g,c'}$ where $g < g'$; and all instances belonging to $\mathcal{G}_{g,c'}$ where $g \geq g'$ are yes-instances. Moreover, we showed in Section 3.5, the NP-hardness on $\mathcal{G}_{g,c}$ for $g = 4$; our hardness reduction reduces to graphs of girth 3 and 4 (see Theorem 3.14 and Corollary 3.15). Moreover, as discussed in Section 3.3.2, all bipartite graphs are yes-instances of SC-Orientation (see Proposition 3.6). Thus SC-Orientation contains only yes-instances for all $\mathcal{G}_{g,c}$, for $c \leq 2$ and any arbitrary value of g.

Our goal in the rest of this section is to settle the complexity of SC-Orientation on all graph classes $\mathcal{G}_{g,c}$ for $g \geq 3$ and $c \geq 3$. Now the key ingredient for our dichotomy result (between P and NP) is to compile an NP-hardness proof that solely relies on a single no-instance of a certain girth g and coloring number c.

The main building block of our construction is a *coupling gadget*. It is a graph that couples the orientation of two special edges but that does *not* enforce a coloring depending on the orientation. An sc-orientable graph H with special edges $\{a_1, b_1\}, \{a_2, b_2\} \in E(H)$ is a *coupling gadget* on $(a_1, b_1), (a_2, b_2)$ if

(1) $(a_1, b_1), (a_2, b_2)$ are *coupled*, that is, for every sc-orientation σ, either $\sigma(\{a_i, b_i\}) = b_i$ for $i \in \{1, 2\}$, or $\sigma(\{a_i, b_i\}) = a_i$ for each $i \in \{1, 2\}$,

(2) there is an sc-orientation without any directed path from $\{a_1, b_1\}$ to $\{a_2, b_2\}$, and

(3) there exist two $\chi(H)$-colorings f, f' of H such that $f(a_1) = f(a_2)$, $f(b_1) = f(b_2)$ and $f'(a_1) = f'(b_2)$, $f'(b_1) = f'(a_2)$.

For example, the $G_{2,3}$-grid graph (see Figure 3.6) is not a coupling gadget as it only satisfies properties (1) and (2), but not (3). $\chi(G_{2,3}) = 2$, and there exists a unique (up to symmetry) coloring of $G_{2,3}$ in two colors. However, if we consider the graph $G_{2,4} \cup K_3$, that is a graph with two components, one is $G_{2,4}$ and the other is a triangle, it is a coupling gadget. Here $\chi(G_{2,4} \cup K_3) = 3$, and we can easily satisfy all the above properties. Note that, in the latter example we considered $G_{2,4}$, whereas in the former one we have $G_{2,3}$. It is an easy exercise to see that $G_{2,3} \cup K_3$ is also not a coupling gadget, and still fails to fulfill (3).

Lemma 3.19. *For every $g, c \geq 3$, there is an sc-orientable graph $G \in \mathcal{G}_{g,c}$.*

Proof. Erdős gave a non-constructive proof to show that, for arbitrary large values of c and g, there exists a graph of girth at least g and chromatic number at least c (see [Erd59]). Although more explicit construction of such graphs are also given in [AKR+16, Lov68].

According to Erdős' result, for each g, c there exists at least one graph G in some $\mathcal{G}_{g',c'}$, $g' \geq g$ and $c' \geq c$. Given such a graph G, there exists a graph in $\mathcal{G}_{g'',c'}$ for all $3 \leq g'' \leq g'$; this can be obtained by adding a cycle $C_{g''}$ to G. Moreover, since $G \in \mathcal{G}_{g',c'}$, one can find a graph in $\mathcal{G}_{g',c'-1}$ by one-by-one deleting vertices of G until we find a graph G' with chromatic number exactly $c' - 1$. With each deleted vertex, the chromatic number of the graph reduces by at most one. Additionally, the girth of G' is definitely at least g', and can be further reduced to obtain a graph with girth exactly g' by adding a cycle $C_{g'}$ to G'. If we keep deleting vertices of G', we will eventually get a graph of chromatic number c. This can be further converted to get a graph in $\mathcal{G}_{g,c}$ by adjusting its girth. Hence, $\mathcal{G}_{g,c}$ is non-empty. When $g \geq 2c - 1$, Theorem 3.18 yields that $G_{g,c}$ is sc-orientable. Otherwise, $g < 2c - 1$. Then, one can consider an sc-orientable graph $G_{2c-1,c} \in \mathcal{G}_{2c-1,c}$ and, in order

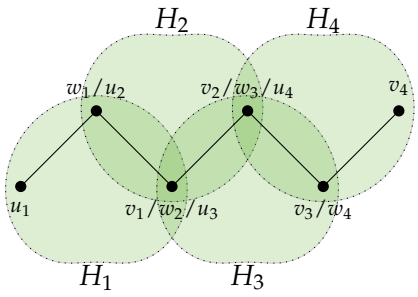


Figure 3.13: A gadget for the construction of Lemma 3.20

to control the girth of the graph, add a connected component C_g to it. Thus, the graph $G_{2c-1,c} \cup C_g \in \mathcal{G}_{g,c}$ is sc-orientable; it has girth g and chromatic number c. □

Lemma 3.20. *For $g \geq 3$, $c \geq 4$, there is a non-sc-orientable graph $G \in \mathcal{G}_{g,c}$, if and only if there is a coupling gadget $H \in \mathcal{G}_{g,c}$.*

Proof. ($\Rightarrow$) Consider a non-sc-orientable graph $G \in \mathcal{G}_{g,c}$. We want to assume that G contains at least one edge $\{u, v\}$ such that subdividing $\{u, v\}$ yields an sc-orientable graph. If that is not the case, pick an edge $\{u, v\}$ and consider the graph G' resulting from graph G where $\{u, v\}$ is subdivided into a path uwv. To assure termination, initially assume all edges unmarked, and consider subdividing an unmarked edge $\{u, v\}$ before considering any marked edges. Whenever an edge $\{u, v\}$ is subdivided, mark the new edges $\{u, w\}$ and $\{w, v\}$. The procedure terminates at the latest when all edges are marked since then the resulting graph is bipartite and hence trivially sc-orientable. Each above step does not decrease the girth and does not increase the coloring number.

Now, let H' consist of a copy of G where edge $\{u, v\}$ is subdivided resulting in a path uwv such that H' has an sc-orientation σ. Note that $\text{girth}(H') \geq \text{girth}(G)$. Crucially, σ couples arcs (u, w) and (v, w), meaning it orientates both $\{u, w\}$ and $\{w, v\}$ either both away from w or both towards w. To see this, assume an sc-orientation σ' that, without loss of generality, has $\sigma'(\{u, w\}) = w$ and $\sigma'(\{w, v\}) = v$. As a consequence, we get an sc-orientation of G by simply adapting σ' by deleting arcs $(u, w), (w, v)$, and adding arc (u, v). This contradicts to G not being sc-orientable. Thus, indeed, arcs $(u, w), (v, w)$ are coupled.

To construct H, we introduce four copies H_1, H_2, H_3, H_4 of H' naming their vertices with subscripts $1, 2, 3, 4$ respectively. We identify the vertices w_1, u_2 and vertices v_1, w_2, u_3 and vertices v_2, w_3, u_4 and vertices v_3, w_4; see the gadget in Figure 3.13. (Eventually our graph H results from adding another sc-orientable graph from $\mathcal{G}_{g,c}$, which we can ignore for now.) By the transitivity of the coupled edges in H_1, H_2, H_3, H_4, in the end, we get that edges (u_1, w_1) and (w_4, v_4) are coupled. This shows that H satisfies Property (1). In particular, combining sc-orientations of H_1, H_2, H_3, H_4 that agree on the orientation of

the edges on the path $P = u_1u_2u_3u_4w_4v_4$ (which have to be alternately orientated), yields existence of an sc-orientation of H. To show Property (2), consider any sc-orientation σ. Up to reverse orientation, σ has the edges on P oriented towards w_1 and w_3. Then H_σ contains no directed path from u_1 to v_1, since this would imply two distinct u_1-w_1-paths. Similarly, there is no directed path between u_2, v_2. Additionally, there won't be any path from u_2/w_1 to u_3/w_2, because if there is then it has to lie completely in H_1 or H_2, and this is not possible as w_1 and w_2 have degree 2 in H_1 and H_2, respectively. This concludes that H satisfies Property (2).

Finally, we state the colorings f and f' with colors $\{1, \dots, \chi(H')\}$ that show Property (3). Note that since H' is derived from G by subdividing several edges, H' is $\chi(G)$-colorable. Moreover since $\{u, v\}$ is an edge in the original graph G, there exists a $\chi(G)$-coloring of H' where the vertices u and v have different colors. Additionally in this coloring all three vertices u, v, and w have pairwise-distinct colors. Note that the vertices $V(H_i) \setminus \{u_i, w_i, v_i\}$ do not share edges with the vertices $V(H_j) \setminus \{u_j, w_j, v_j\}$ for any i, j. At this point we simply give the two colorings f and f' on the vertices of P and this can be extrapolated to a valid coloring of H_i for each i. To obtain a coloring f, let the vertices of P be colored with $f(u_1) = 1, f(u_2) = 2, f(u_3) = 3, f(u_4) = 4, f(w_4) = 1, f(v_4) = 2$, and f' be the coloring in which vertices of P are colored with $f'(u_1) = 1, f'(u_2) = 2, f'(u_3) = 3, f'(u_4) = 4, f'(w_4) = 2, f'(v_4) = 1$. Here we use that $c \geq 4$.

So far, the coloring number did not increase as the colorings f and f' demonstrate. Also the girth does not decrease since every chordless cycle is contained in a copy of H'. In order to ensure that $H \in \mathcal{G}_{g,c}$, we update the graph by appending another graph as an extra component to it. Let G' be an sc-orientable graph in $\mathcal{G}_{g,c}$. Note that such a graph always exists from Lemma 3.19. We update H by doing a disjoint union of H with graph G'. So, $H \leftarrow H \cup G'$. Since G' forms its own component, it does not interfere with the properties (1)-(3) observed so far. Thus H is a coupling gadget on (u_1, w_1) and (w_4, v_4).

($\Leftarrow$) Let $H \in \mathcal{G}_{g,c}$ be a coupling gadget on (a_1, b_1) and (a_2, b_2). Let $g' = g$. We construct a non-sc-orientable graph $G \in \mathcal{G}_{g,c}$ from g' copies of H, denoted as $H^1, \dots, H^{g'}$. In each H^i, let (a_1^i, b_1^i) and (a_2^i, b_2^i) be the corresponding pair of coupling edges. Essentially we glue the coupling gadgets together at their coupled edges in a cycle while introducing a final 'twist' of the coupling edges. The graph G results from identifying a_2^i, a_1^{i+1} and identifying b_2^i, b_1^{i+1} for $i \in \{1, \dots, g' - 1\}$, and identifying $a_2^{g'}, b_1^1$ and identifying $b_2^{g'}, a_1^1$. We further add an sc-orientable graph $G_{g,c} \in \mathcal{G}_{g,c}$ to our construction (such a graph always exists because of Lemma 3.19). As a result of the additional final twist, we force the edge (a_1^1, b_1^1) to reverse couple with itself. Which is not possible. Thus G is not sc-orientable.

The constructed graph G has the same girth as H, since any chordless cycle either has length at least g' or is contained in a copy of H. Also we can color G with c colors using the c-colorings f, f' of Property (3). We use f to color H^i with $f(a_1^i) = f(a_2^i)$ and $f(b_1^i) = f(b_2^i)$, for $i \in \{1, \dots, g' - 1\}$. Finally, we use the c-coloring f' to color $H^{g'}$ which then agrees on the coloring of the vertices shared with H^1 and shared with $H^{g'-1}$. □

Theorem 3.21. *For $g, c \geq 3$,* SC-Orientation *restricted to the graphs in $\mathcal{G}_{g,c}$ is either* NP*-complete or contains only yes-instances and is hence trivially polynomial-time solvable.*

Proof. Consider fixed $g, c \geq 3$. If all graphs in $\mathcal{G}_{g,c}$ are sc-orientable, then an algorithm may always answer 'yes' to decide whether they are sc-orientable. If not, there exists at least one no-instance of SC-Orientation in $\mathcal{G}_{g,c}$. In this case, we claim, it is NP-hard to decide whether an arbitrary graph in $\mathcal{G}_{g,c}$ is sc-orientable. We use the coupling gadget $H \in \mathcal{G}_{g,c}$ derived in Lemma 3.20, when the problem contains at least one no-instance on $\mathcal{G}_{g,c}$. However, Lemma 3.20, is applicable only for $c \geq 4$. Thus towards the end of this proof, we consider the case of $c = 3$ separately.

In order to prove the NP-hardness claim, we draw ideas from the NP-hardness construction given in Theorem 3.14. Recall, in Theorem 3.14, each clause gadget consists of a cycle which has generic edges and variable edges (see Figure 3.12). Moreover, the orientation of generic-edges and variable-edges are fixed once we fix the orientation of the 0^{th}-steps of the universal ladder and the variable-gadgets respectively. Let σ be such a fixed orientation. Note that, σ is not an sc-orientation if and only if there is a clause gadget whose all variable-edges are oriented in the same direction as that of edge e_1^c, and thus forming two disjoint paths in the cycle. We exploit this property of the clause gadget.

Adaptation to the NP-hardness Construction given in Theorem 3.14:

- Let us call the cycles shown in Figure 3.12 as the *outer cycle* of the clause gadget. We construct a clause-gadget where the outer cycle of the gadget has an even length. Let $g' \in \{g, g+1\}$ be odd. Update the construction of clause-gadget in Theorem 3.14, and replace the edge e_2^c by a path consisting of a set of edges $\{e_{2_1}^c, e_{2_2}^c, \ldots, e_{2_{g'}}^c\}$. This way, the length of the outer cycle in the clause-gadget is at least g. Moreover, since the outer cycle has even length, there exists a 2-coloring of the vertices of each outer cycle.

- The universal ladder and the variable-gadgets (in this case, the variable-ladders) each consists of plenty of copies of the coupling gadgets $H \in \mathcal{G}_{g,c}$ stacked on each other such that each pair of consecutive coupling gadgets share a unique coupled edge. We refer to these unique edges as the steps of the ladder.

- According to Property (2), the coupling-edges (a_1, b_1) and (a_2, b_2) of a coupling-gadget are disjoint. Thus one can introduce a coupling gadget between a pair of disjoint edges to make them coupled. Note that for an arbitrary edge $\{a, b\}$, we can couple (a', b') with (a, b), or (b, a). The latter is simply done by introducing a twist in the graphical representation of the structure. Moreover, to avoid short cycles of length less than g, we can do the coupling by introducing a stack of g many coupling gadgets instead of one coupling gadget between a pair of disjoint

edges. Using this idea, we connect all the generic edges of the clause-gadgets to the universal ladder. Just like we did in Theorem 3.14, we couple with e_1^c, e_3^c, e_4^c and $e_{2_1}^c, e_{2_2}^c, \ldots, e_{2_{g'}}^c$, and by adding the final twist, reverse-couple with the edge e_5^c.

- Similarly, we couple variable-edges with the steps of their respective variable ladders. We couple e_x^c with the variable ladder if $x \in c$, else we reverse-couple e_x^c with the variable ladder if $\bar{x} \in c$.

Correctness of the Construction: The constructed instance of the SC-Orientation problem indeed has girth g. This is because, since the coupling gadget H belongs to class $\mathcal{G}_{g,c}$, the universal ladder and the variable gadget have girth g, and the outer cycle of each clause has length at least g. In addition we maintain distance at least g between an edge of the clause-gadget and an edge of a (universal/variable) ladder, as between each pair of coupling edges, we allow a stack of g many coupling gadgets to avoid short cycles. Of course, to enforce the girth g on the constructed instance, one can add an sc-orientable graph $G \in \mathcal{G}_{g,c}$ as an additional component. We claim that the constructed instance has chromatic number c. Each outer cycle of the clause-gadget can be colored with colors $\{1,2\}$. We color each step of the ladders also with colors $\{1,2\}$. It remains to extend this coloring to the internal vertices of each coupling-gadget. This is possible because of Property (3) of the coupling gadgets: There exist two $\chi(H)$-colorings f, f' of the coupling gadget H such that $f(a_1) = f(a_2)$, $f(b_1) = f(b_2)$ and $f'(a_1) = f'(b_2)$, $f'(b_1) = f'(a_2)$. Hence each coupling-gadget has a coloring using f or f' that respects the prescribed coloring. Also, note that the coupling gadget H itself cannot be colored with less than c colors as $H \in \mathcal{G}_{g,c}$, hence the chromatic number of the entire constructed instance is c.

At last, we perform polynomial many replacements as the gadgets have constant size. Thus our reduction is polynomial in the size of the input instance.

The proof of Theorem 3.14 can be adapted in this settings in a way that the rest of the arguments about the correctness of the construction follow as it is.

Now, for the case of $c = 3$, we claim that the NP-hardness proof given in Section 3.5 is on planar graphs of girth 4 and coloring number at most 3. Observe that, each gadget (clause gadget or variable gadget) used in the construction of Theorem 3.14 is bipartite. Thus there exists a 2-coloring for each of these gadgets individually. The odd cycles occur because we identify different types of steps (odd or even steps) of the variable gadget to the edges of the clause gadgets. Due to the presence of these odd cycles, the reduced instances are not 2-colorable, but they can be colored using 3 colors as given in the following. We start with a 2-coloring of the nodes which are part of the universal ladder (this covers all clause nodes). Then, start with a 2-coloring for each variable gadget, and whenever, while trying to identify the variable edges with clause edges, if the coloring of the endpoints does not match, then we adapt the coloring of the variable gadget locally using a third color such that, in the end, the coloring of the identified

endpoints match. Additionally, for the case, when $c = 3$ and girth at least 5, we know from Theorem 3.18 that all such graphs are sc-orientable.

Moreover, the NP-hardness result in Theorem 3.14 can be adapted to girth 3 graphs by simply adding a triangle as an additional component to the graph. □

3.6.4 Consequence of Our Dichotomy Result

Due to our dichotomy result from Theorem 3.21, we know that, for each graph class $\mathcal{G}_{g,c}$, SC-Orientation is either NP-complete or consists only of yes-instances, and hence, is trivially polynomial-time solvable. Moreover, due to Theorem 3.18, we get that for all g, c such that $g \geq 2c - 1$, $\mathcal{G}_{g,c}$ consists of only yes-instances of SCO. Finally, due to Theorem 3.14, we get NP-completeness on $\mathcal{G}_{4,3}$ and $\mathcal{G}_{3,3}$. In this section, we give a scheme to extrapolate a tractability result, as well as a hardness result over the graph classes $\mathcal{G}_{g,c}$ for relevant values of g and c. We give the scheme as follows:

Theorem 3.22. *If* SC-Orientation *is* NP*-complete on* $\mathcal{G}_{g,c}$*, then it is also* NP*-complete on* $\mathcal{G}_{g',c'}$ *for all* $g' \leq g$ *and* $c' \geq c$*.*

Proof. It follows because each instance in $\mathcal{G}_{g,c}$ can be converted into an equivalent instance in $\mathcal{G}_{g',c}$ where $g' \leq g$ by simply adding a cycle of length g' as an additional connected component. Similarly, each instance in $\mathcal{G}_{g,c}$ can be converted into an equivalent instance $\mathcal{G}_{g,c'}$, $c' \geq c$ by adding an sc-orientable graph of chromatic number c' to it. Such an sc-orientable graph always exists due to Lemma 3.19. Thus the NP-hardness of $\mathcal{G}_{g,c}$ shows NP-hardness on $\mathcal{G}_{g',c}$, $g' \leq g$ and on $\mathcal{G}_{g,c'}$, $c' \geq c$. Inductively, it also shows NP-hardness on $\mathcal{G}_{g',c'}$, for all $g' \leq g$ and $c' \geq c$ (see Figure 3.14, top part). □

Theorem 3.23. *If* $\mathcal{G}_{g,c}$ *consists of only yes-instances of* SC-Orientation*, then so does* $\mathcal{G}_{g',c'}$ *for all* $g' \geq g$ *and* $c' \leq c$*.*

Proof. This holds because, an instance of SC-Orientation in $\mathcal{G}_{g',c'}$ can be converted to an equivalent instance in $\mathcal{G}_{g,c}$, for $g' \geq g$ and $c' \leq c$. Additionally, due to Theorem 3.21, SCO is NP-hard even if there is a single no-instance in a graph class $\mathcal{G}_{g,c}$. As a result if the class $\mathcal{G}_{g,c}$ contains only yes-instances, then all graph classes $\mathcal{G}_{g',c'}$, for $g' \geq g$ and $c' \leq c$, also consist of only yes-instances (see Figure 3.14, bottom part). □

As a remark, we point out that, although the cascading effects of a complexity result (Theorem 3.22) and an algorithmic result (Theorem 3.23) hold for any graph class $\mathcal{G}_{g,c}$, we do not know whether SC-Orientation is NP-hard or polynomial-time solvable for any graph class $\mathcal{G}_{g,c}$ where $g < 2c - 1$ and $g \geq 5$.

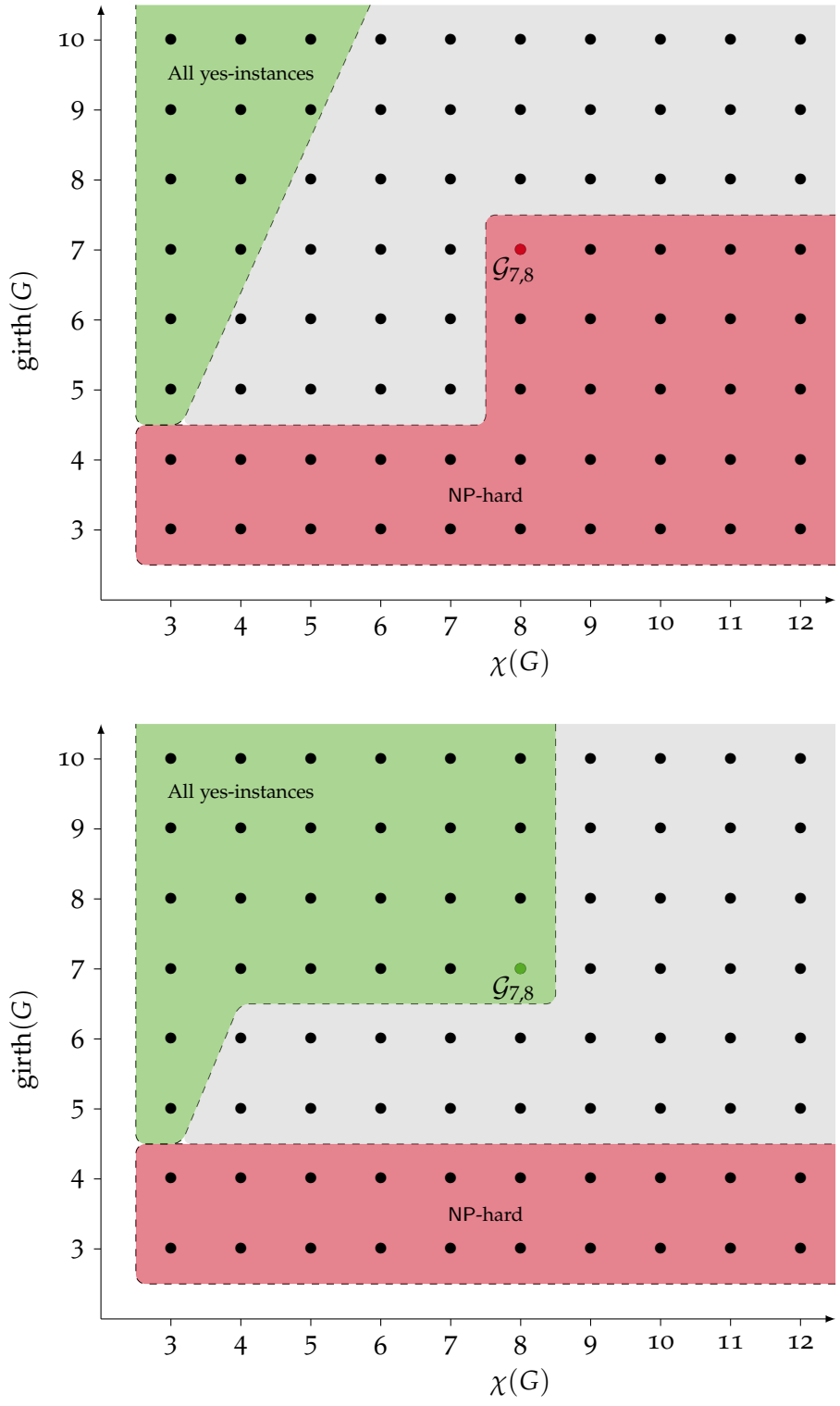


Figure 3.14: The overview of the cascading effect when some $\mathcal{G}_{g,c}$ is shown to be NP-hard (see Theorem 3.22) or consists of only yes-instances (see Theorem 3.23). Note that this illustration is merely hypothetical and it does not settle the complexity of the case $\mathcal{G}_{7,8}$.

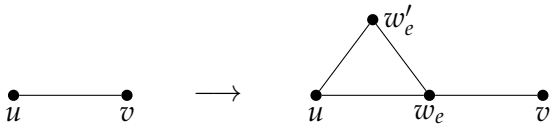


Figure 3.15: Illustrating the modification of edges from Theorem 3.14 to obtain NP-hardness on perfect graphs.

3.7 Complexity on Special Graph Classes

In this section, we consider further more variety of special graph classes and settle the complexity of SC-Orientation on these graph classes. We first start by considering the class of perfect graphs. After that we study the distance hereditary graphs as a special case.

3.7.1 NP-Hardness on Perfect Graphs

Perfects graphs are characterized by a property that, for every induced subgraph of the graph, the size of the maximum clique therein is equal to its chromatic number. First we show that the NP-hardness proof from Theorem 3.14 can be adapted to perfect graphs. Consequently we prove the following result.

Theorem 3.24. SC-Orientation *is* NP*-hard even for perfect graphs.*

Proof. We modify our NP-hardness reduction from Theorem 3.14 as follows: In the reduced instance, for each edge $e = \{u, v\}$, introduce two new vertices w_e and w'_e. Delete edge e, and instead add edges such that there is a path uw_ev, and a triangle $uw_ew'_e$. See Figure 3.15.

Then according to Lemma 3.9, the modified instance is equivalent to the original instance in the sense that the modified graph is sc-orientable if and only if the originally constructed graph is sc-orientable. Further, it can be checked that the modified graph is 3-colorable. In the reduced instance, if one deletes all newly added w'_e, then we are left with a bipartite graph, because each w_e subdivides the edge e. Thus there exists a 2-coloring of the remaining graph when all w'_e's are deleted. Moreover, each w'_e is adjacent to exactly two vertices which also share an edge. We can introduce a third color for the set of all w'_e, and the resulting 3-coloring is a proper coloring of the graph. Since each edge in the original graph has been modified, it is easy to check that the modified graph has a maximum clique of size 3. Thus all in all, it is perfect. □

3.7.2 Polynomial-Time Solvable on Distance Hereditary Graphs

Now we derive two classifications of the distance hereditary graphs that are sc-orientable. One way is to simply restrict the recursive definition of distance hereditary

graphs to (locally) avoid a diamond subgraph. The second is a concise classification by forbidden induced subgraphs. In what follows, we will see both of these approaches.

In 1986 [BM86], Bandelt and Mulder showed a graph is *distance hereditary* if it can be constructed starting from a single isolated vertex with the following operations:

(A1) a *pendant* vertex to $u \in V(G)$, that is, add a vertex v and edge $\{u, v\}$;

(A2) *true twin* on a vertex $u \in V(G)$, that is, add a vertex v with neighborhood $N(v) = N(u)$ and edge $\{u, v\}$;

(A3) *false twin* on a vertex $u \in V(G)$, that is, add a vertex v with neighborhood $N(v) = N(u)$, but without adding the edge $\{u, v\}$.

We tweak the above definition and call a graph *strongly distance hereditary* if it can be constructed from a single isolated vertex with the following restricted operations:

(S1) a pendant vertex to $u \in V(G)$;

(S2) true twin on a vertex $u \in V(G)$, restricted to u where $|N(u)| = 1$; and

(S3) false twin on a vertex $u \in V(G)$, restricted to u such that the neighbors of u do not share an edge between them, that is, when $E(G) \cap \{\{a, b\} \mid a, b \in N(u)\} = \varnothing$. Let us define $N(u)^2 := \{\{a, b\} \mid a, b \in N(u), a \neq b\}$

Observe that the forbidden operations—which we are allowed to do for the construction of distance hereditary graphs but not for the construction of strongly distance hereditary graphs—would immediately imply a diamond as a subgraph. Thus every distance hereditary graph that has an sc-orientation must be strongly distance hereditary. Now, we show the converse is also true.

Theorem 3.25. *The sc-orientable distance hereditary graphs are exactly the strongly distance hereditary graphs. They can be recognized in polynomial time.*

Proof. We show that every strongly distance hereditary graph indeed allows an sc-orientation.

Recall from Lemma 3.11, we can split any graph G across all its cut vertices to obtain a set of 2-vertex-connected components. Furthermore, G is sc-orientable if and only if all the 2-vertex-connected components thus obtained are sc-orientable. Now, we show the following property for a strongly distance hereditary graph G: Every 2-vertex-connected component is a bipartite graph or a triangle.

This property implies that G is sc-orientable, as G decomposes, according to Lemma 3.11, into bipartite graphs and triangles, which are positive base cases of our decomposition. Checking this property is possible in polynomial time. To specifically check for strongly distance hereditariness, an additional test of whether G is distance hereditary suffices; this is possible in linear time [DHP01, HM90].

It remains to show the above property. An isolated vertex is a bipartite graph. From there, we proceed inductively. Let G' be derived from G by performing one of the operations to construct a strongly distance hereditary graph. We assume G satisfies the above property and inductively prove the property for G'.

- For attaching a pendant v to vertex $u \in V(G)$: The 2-vertex-connected components of G' are those of G and the edge $\{u, v\}$. Component formed by the edge $\{u, v\}$ is bipartite, and the original components of G remain a bipartite graph or a triangle.
- For attaching a true twin v to a vertex $u \in V(G)$ with $|N(u)| = 1$: Let $N(u) = \{w\}$. Note that the vertex w is a cut vertex in G'. Thus the 2-vertex-connected components of G' are a triangle $\{u, v, w\}$ and the original components of G. Thus for G', every 2-vertex-connected component remains either a triangle or a bipartite graph.
- For attaching a false twin v to a vertex $u \in V(G)$, where $N(u)^2 \cap E(G) = \varnothing$: Let $\mathcal{C}$ be the 2-vertex-connected components of G. Let u be a part of a 2-vertex-connected component $C \in \mathcal{C}$. Note that C is not a triangle since $N(u)^2 \cap E(G) = \varnothing$. Thus C is bipartite. Since $N(u) = N(v)$, we can add vertex v to the same component C and retain its bipartite structure as the vertices u and v can be put in the same part of the bipartition. Thus all 2-vertex-connected-components of G' are again either a triangle or a bipartite graph. □

Bandelt and Mulder showed in [BM86], the distance hereditary graphs are exactly the (house, hole, domino, gem)-free graphs, where a hole is any cycle of length at least 5. We show, in the above result, replacing 'gem' with 'diamond' result into exactly the strongly distance hereditary graphs.

Lemma 3.26. *The strongly distance hereditary graphs are exactly the graphs with house, hole, domino and diamond as forbidden induced subgraph.*

Proof. We show both inclusions: Let HHDD-free graphs be those graphs with house, hole, domino, and diamond as a forbidden subgraph.

($\subseteq$) Let G be a strongly distance hereditary graph, that is distance hereditary and sc-orientable; see Theorem 3.25. Then G has house, hole, and domino as well as diamond as forbidden induced subgraphs; hence is HHDD-free.

($\supseteq$) Let G be HHDD-free. Then G is also gem-free and hence is distance hereditary. We show that every 2-vertex-connected component of G is bipartite or a triangle which, as per Theorem 3.25, implies that G is sc-orientable and thus strongly distance hereditary. Consider a 2-vertex-connected component C of G which is neither bipartite nor a triangle. Note that C contains at least 3 vertices. Moreover, since C is hole-free and not bipartite, it contains a triangle uvw. As C is not merely a triangle, there is at least one other vertex $u^\star \in C \setminus \{u, v, w\}$. Since C is a 2-vertex-connected component, $u^\star$ must have

two disjoint undirected paths to two vertices of u, v, w, say P_1 is a path from $u^\star$ to u and P_2 is a path from $u^\star$ to v. Depending on the size of P_1 and P_2, we either get a hole, a diamond, or a house. Thus forming a contradiction to G being HHDD-free. Therefore every 2-vertex-connected component of the graph is either bipartite or a triangle. □

3.7.3 Polynomial-Time Solvable on Chordal Graphs

In this section, we prove the following result on the class of chordal graphs.

Theorem 3.27. SC-Orientation *is polynomial-time solvable on chordal graphs.*

Proof. We preprocess an instance of chordal graphs using Lemma 3.11. Note that according to Lemma 3.11, a chordal graph splits into several 2-vertex-connected components which are individually chordal. This is because each cycle is preserved under the preprocessing due to Lemma 3.11. Thus we restrict ourselves only to 2-vertex-connected chordal graphs. Consequently, this instance of SC-Orientation must contain a cycle.

Within the set of 2-vertex-connected chordal graphs, we now distinguish the yes-instances of SCO from the no-instances.

According to the definition of chordal graphs, each cycle of length at least 4 in the graph has a chord. Thus, if G has a cycle of length at least 4, it contains diamond as a subgraph. This implies G is a no-instance of SCO. Otherwise, G contains cycles of length at most 3. Since we restrict to simple graphs, this implies, G contains a cycle of length exactly 3. Let this cycle be $xyzx$. We claim that G is either just a triangle or it is a no-instance of SCO. If $V(G) \setminus \{x, y, z\} \neq \varnothing$, then there exists a vertex $v \in V(G)$, $v \notin \{x, y, z\}$. Without loss of generality, let v be adjacent to x. Since x is not a cut-vertex, because G is 2-vertex-connected, there exists a larger cycle of length at least 4 in G, hence creating a contradiction.

As a result, after preprocessing according to Lemma 3.11, all chordal graphs that do not contains a cycle of length at least 4 are sc-orientable. □

3.7.4 Polynomial-Time Solvable on Cographs

There exist a representation of cographs in terms of *cotrees*. Cotrees are trees in which every internal vertex is either a *join* node or a *disjoint union* node. The join nodes and disjoint union nodes corresponds to the join and disjoint union operations, respectively. All leaves of the cotree correspond to the nodes of the original cograph. A subtree rooted at a node T of the cotree corresponds to the induced subgraph on the set of vertices residing in the leaves of the subtree rooted at T. A disjoint union operation simply takes the union of the subgraphs rooted at the children node. A join operation, in addition to the disjoint union, also adds edges between every pair of nodes coming from two different children's subgraphs.

Theorem 3.28. SC-Orientation *is polynomial-time solvable on cographs.*

Proof. Given a cograph G, it suffices to check whether each connected component of G is sc-orientable. Here we assume that G is connected. Thus, the root node of the cotree of G must consist of a join operation. Let G_1, G_2, both non-empty graphs, be subgraphs of G such that the root operation of the cotree joins G_1 and G_2. If G_1 or G_2 contains a P_3, i.e. a path of length 3, then G contains a diamond as a subgraph. Since diamond is a forbidden structure for sc-orientable graphs, this join operation results into a no-instance of SC-Orientation. Hence, a cograph is a yes-instance if G_1 is a collection of singleton edges and isolated vertices and G_2 contains one vertex (or vice versa). It is also a yes-instance if G_1 and G_2 both are collections of isolated vertices. □

3.8 Concluding Remarks and Future Directions

We have given a construction of an NP-hardness reduction of the SC-Orientation problem on a graph class $\mathcal{G}_{g,c}$, which relies solely on the existence of a single no-instance of SCO in $\mathcal{G}_{g,c}$. This kind of hardness result, which develops from a single no-instance of the problem, is very uncommon in the field of computational complexity. As a consequence of our result, we have shown that SC-Orientation restricted to graphs of girth g and chromatic number c, for every $g, c \geq 3$, is either NP-complete or contains only yes-instances, and thus, is in P. While we show NP-completeness for low girth and chromatic number ($\chi(G)$), and we know that for relatively large girth compared to $\chi(G)$, the problem is trivially in P because it contains only yes-instances, it yet remains to pinpoint the exact boundary between the NP-hard and P cases.

The overview of our dichotomy result is shown in Figure 3.16. Here, a node (g, c) in the grid is associated with a graph class that contains all graphs of girth g and chromatic number c. For $\mathcal{G}_{g,c}$ in the gray region, we have that SC-Orientation is NP-hard if and only if there exists a no-instance in $\mathcal{G}_{g,c}$. As shown, to extend the NP-hardness result, one needs to simply find a no-instance of the girth and chromatic number of interest (or alternatively merely a coupling gadget). The smallest graph class for which we do not know whether a no-instance exists is the class $\mathcal{G}_{5,4}$ consisting of graphs of girth 5 and chromatic number 4. On the other hand, one might improve Theorem 3.18 to also hold for smaller values of girth g.

Among many interesting technical results, we showed that a triangle-free graph is sc-orientable if and only if there exists an acyclic sc-orientation of the graph. Triangles in the graph can be easily dealt with. And it suffices to solve the SC-Orientation problem on 2-vertex-connected graphs. Further, we showed that the problem is NP-hard on planar and perfect graphs. On the other hand, it is polynomial-time solvable on chordal graphs, cographs, and distance hereditary graphs. Further to the existence of sc-orientation, it

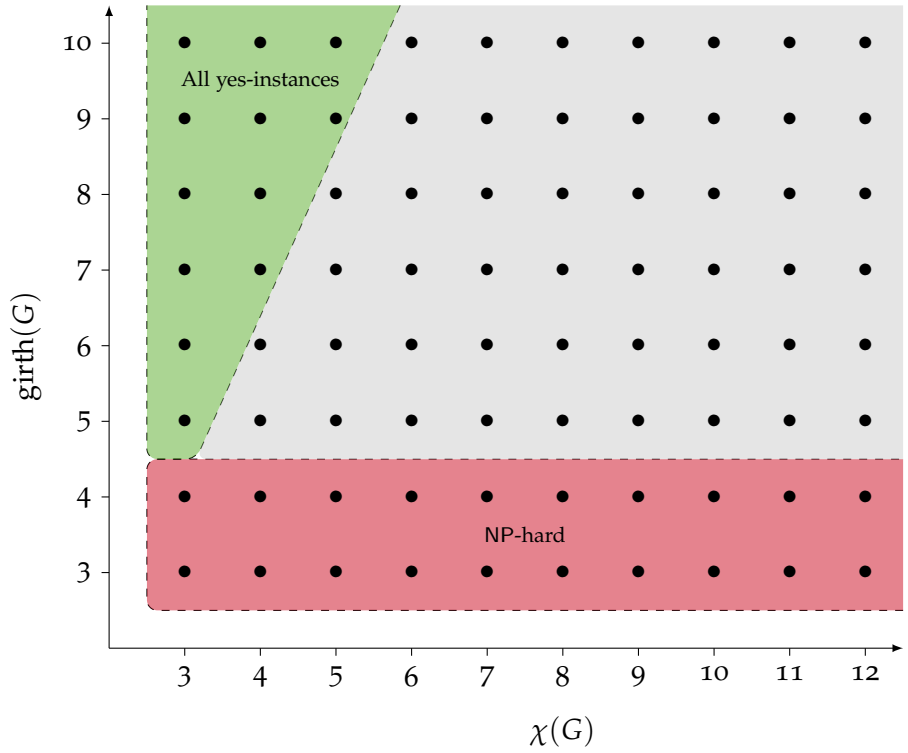


Figure 3.16: An overview of our dichotomy result.

would be interesting to know how many such sc-orientations exist; thus, a potential question would be to find the counting complexity of the problem.

For the SC-Orientation problem in this chapter, our task was to convert an undirected graph into a singly connected digraph. Another interesting setting would be to capture the essence of partial orientations by converting an undirected graph to an sc-orientable mixed graph, or by converting a mixed graph to an sc-orientable digraph.

Graph orientation problems can be used to navigate a crowd to find efficient crowd management solutions. *Crowd management* is a way of organizing a huge number of people at large events (such as concerts, football games) or public spaces (such as airports, malls), in order to ensure safety and efficiency.[3] Crowd management solutions involve a variety of disciplines, network design being one of them. Depending on the crowd management scenario, one may search for suitable properties that are desired, for the network to be oriented. This might lead to interesting problems that demand orientation to mixed graphs.

[3]Not to be confused with *crowd control*, which is rather concerned with on-site actions to prevent and contain crowd crushes or riots by adapting to (unexpected or expected) crowd occurrences.

4

Partitioned-Items Bilevel Optimization Problems

In this chapter, we take a sharp turn from decision problems and now focus on some optimization problems. More specifically, here we study bilevel optimization problems. Classically, these problems consist of two decision makers whose optimization problems are nested into each other. We study the class of partitioned-items bilevel problems where the set of decision variables is partitioned between the two decision makers. We study this class of problems on three different combinatorial problems, namely the Assignment problem, the Interval Scheduling problem, and the Independent Set problem. A part of the results mentioned in Section 4.3 is a joint work with my former colleague Dennis Fischer and my late supervisor Prof. Gerhard Woeginger [FMW22]. Section 4.3.2 is newly added and does not belong to the publication mentioned earlier. All other results, especially the study on Bilevel Interval Scheduling from Section 4.4 and Bilevel Independent Set from Section 4.5, have not been peer-reviewed before, although, they appear on arXiv [Mul26].

4.1 Introduction to Bilevel Optimization

The field of bilevel optimization is grown rapidly over the last few decades. A bilevel optimization problem consists of two interleaved optimization problems that are controlled by two non-cooperating decision makers called the *leader* and the *follower*. Both decision makers have a private objective function, a private set of decision variables, and a private set of constraints on their variables. Furthermore, there are coupling constraints that connect the decision variables of the leader and follower to each other; these constraints can appear in the constraint set of both the leader's or the follower's optimization problem. Both decision makers want to optimize their private objective function. The decision-making process is as follows. First, the leader makes her decision and fixes the values of her variables, and then the follower reacts by setting his variables. The leader has perfect knowledge of the follower's scenario (objective function,

variables, and constraints) and also of the follower's behavior. The follower observes the leader's actions, and then optimizes his private objective function subject to the decisions made by the leader (and subject to all the imposed constraints). As the leader's objective function and, possibly, the coupling constraints in the leader's problem depend on the follower's decision, the leader must carefully take the follower's reaction into account.

The concept of bilevel optimization goes back to the economic duopoly model of von Stackelberg [Sta34] from the 1930s. Jeroslow in 1985, obtained the first complexity results for bilevel optimization problems; his results imply that bilevel linear programs with continuous variables are NP-hard and that bilevel integer linear programs are even Σ_2^p-hard [Jer85]. Over the last two decades, bilevel optimization has received enormous attention in the optimization community. We refer the reader to the book by Dempe [Dem02], to the survey by Brotcorne, Marcotte & Savard [BMS08], and to the annotated bibliographies of Dempe [Dem03] and Colson, Marcotte & Savard [CMS05] for more information on this area. Further recent overview on the topic with ample references can be found in [DKPK15, Dem20]. Moreover, we highly recommend reading a survey by Woeginger, which builds intuition for the complexity class Σ_2^p, and elaborates on the troubles that might arise due to the existence of the additional quantifier [Woe21]. The article leaves us with some concrete questions and concerns in the field of bilevel optimization that need to be addressed in order to get a good hold on solving these problems belonging to the second level of the polynomial hierarchy. This is crucial because until now, a lot of literature on bilevel problems mostly contains negative results, i.e., Σ_2^p-hardness and inapproximability results for various Σ_2^p-hard problems [KL95, Uma99]. Grüne and Wulf even derived a meta reduction technique which proves Σ_2^p-completeness of a number of problems [GW25]. We lack the tools and techniques needed to tackle these problems algorithmically [Woe21].

Meyer and Stockmeyer, in their classical paper on the polynomial hierarchy [MS72], gave the first Σ_2^p-complete problem, which was the 2-Alternating Quantified Satisfiability problem ((B_2) as short). An instance of B_2 consists of a Boolean expression φ in the disjunctive normal form (DNF) over two sets of Boolean variables X and Y, and the question is to decide whether there exists a truth assignment to the variables in X such that for all truth assignments to the variables in Y, the expression φ is satisfied. We use an alternating problem called B_2^{CNF} to prove Σ_2^p-hardness of our results in Section 4.5.1.

In this chapter, we investigate a class of bilevel optimization problems called *partitioned-items bilevel problems*, which we define in the next section.

4.2 The Class of Partitioned-Items Bilevel Problems

In a very recent study by Henke and Wulf [HW25], they coined an umbrella term *partitioned-items bilevel problems* to cover a specific class of bilevel optimization problems, which involves a bipartitioned ground set of variables, such that each part of the

bipartition is controlled by a decision maker; the leader controls one set of variables, and the follower controls the other set of variables.

Depending on the combinatorial optimization problem at hand, we have different types of ground sets of variables. For example, when considering the Assignment problem, we are interested in choosing a desired subset of edges of the given bipartite graph. Thus, the edge set of the graph can be regarded as the ground set. Similarly, for the Independent Set problem, the vertex set of the graph can be regarded as the ground set.

The goal of the bilevel problem is that both decision makers, despite having potentially distinct objective functions, work together to build a feasible solution of the underlying problem.

A real-life scenario which involves a partitioned-items bilevel problem is as follows: Think of a factory with only one machine and two operators who have completely disjoint skill sets. There are several jobs (with designated time slots) that can be performed at this machine, but only one job can be run at a time. Each job has a tight time window in which it can be performed. Due to the mutually exclusive skill sets of the operators, each job can be performed by either of the operators, but not both. Each operator benefits differently from a job that gets done (they also benefit from each other's jobs). Their individual goal is to optimize their personal total payoff. In order to schedule the jobs on the machine, a specific, say more senior operator, chooses a set of her jobs that she would like to execute during their respective time slots, so she reserves the machine for those times. Then the second operator arranges a suitable subset of his jobs that he wants to perform, while making sure that the time windows of his jobs do not clash with the jobs of the other operator. This is *a* bilevel variant of the Interval Scheduling problem. We refer to this problem as simply Bilevel Interval Scheduling (BISch, as short) and discuss it thoroughly in Section 4.4. In order to view an instance of this problem, we refer the reader to Figure 4.2.

For the sake of defining the structure of partitioned-items bilevel problems that we consider in this thesis, let us denote, for the time being, the ground set of an arbitrary combinatorial optimization problem by E. Moreover, let us denote the feasibility set of the combinatorial optimization problem by $\mathcal{F}$; note that $\mathcal{F} \subseteq 2^E$. Now, we define the general problem formulation that we use throughout this chapter.

4.2.1 Our Bilevel Problem Formulation

Consider a combinatorial optimization problem with the set of feasible solutions $\mathcal{F} \subseteq 2^E$ over a ground set E. Due to the partitioned-items formulation, the set of variables, E, is partitioned into two subsets, $E = E_\ell \dot\cup E_f$ where E_ℓ is controlled by the leader and E_f is controlled by the follower. Additionally, we have weight functions $w_\ell, w_f \colon E \to \mathbb{R}_+$, w_ℓ for the leader and w_f for the follower. Moreover, we have two objective functions c and d for the leader and the follower, respectively. Where, typically, the leader's objective function, c, depends on her weight function w_ℓ, for example $\mathrm{c}(L \cup F) = \sum_{v \in L \cup F} w_\ell(v)$, whereas the follower's objective function, d, depends on his weight function w_f. Then

our bilevel optimization problem is formulated as follows:

$$\min_{L \subseteq E_\ell} \quad c(L \cup F) \tag{4.1a}$$

$$\text{where } F \subseteq E_f \text{ solves the follower's problem}$$

$$\min^*_{F \subseteq E_f} \quad d(L \cup F) \tag{4.1b}$$

$$\text{such that } L \cup F \in \mathcal{F}$$

The formulation given in the Problem (4.1) corresponds to a minimization problem; for a maximization problem, we have slightly different formulations. We explore them in the later sections when we discuss the Bilevel Interval Scheduling problem in Section 4.4 and the Bilevel Independent Set problem in Section 4.5.

Note that, in the definition above, we have two optimization problems, namely one mentioned in (4.1a) which we refer to as the *leader's optimization problem* and another mentioned in (4.1b) which we refer to as the *follower's optimization problem*. In general, the goal is to solve the leader's optimization problem subject to the condition that the follower chooses his set of variables, i.e., $F \subseteq E_f$, that optimizes his own objective function. The leader has to choose a set $L \subseteq E_\ell$ which in the end can be augmented by a suitable follower's reaction F such that $L \cup F$ belongs to the family $\mathcal{F}$ of feasible solutions. An action $L \subseteq E_\ell$ of the leader is called a *leader's feasible action* if there exists a follower's reaction F such that $L \cup F \in \mathcal{F}$. Given a leader's feasible action $L \subseteq E_\ell$, we denote the set of the follower's suitable reactions to it by $\Omega(L) := \{F \subseteq E_f \mid L \cup F \in \mathcal{F}\}$. Now, given the set L, a follower's reaction $F' \subseteq E_f$ is called a *follower's feasible reaction* if $F' \in \arg\min_{F \in \Omega(L)} d(L \cup F)$. Note that for a specific leader's action L, the follower may have multiple feasible reactions. In this case, we consider two classical settings in which the follower reacts. First, we consider the *optimistic* setting in which the follower always chooses, among his feasible reactions the one which benefits the leader the most. Second, we consider the *pessimistic* setting, in which he reacts by choosing the set F which is the worst for the leader. The asterisk $(*)$ in the follower's problem in (4.1b) encodes the follower's behavior, either optimistic $(\mathtt{o})$ or pessimistic $(\mathtt{p})$. Given a leader's action L, we denote the *follower's optimal reaction* to it (according to $\mathtt{o}$ or $\mathtt{p}$ setting) by F_L. Note that, in case the follower's optimal reaction is also not unique, all such reactions evaluate to the same leader's objective value and the same follower's objective value. So, from the optimization perspective, a specific optimal reaction of the follower does not make any difference, thus without loss of generality, we say *the* follower's optimal reaction.

Furthermore, we consider two types of objective functions, up to optimistic or pessimistic setting, namely the sum type and the bottleneck type. The leader's sum type function (c_s) and bottleneck type function (c_b) can be defined as follows:

$$c_s(S) = \sum_{v \in S} w_\ell(v), \tag{4.2}$$

$$c_b(S) = \max_{v \in S} \quad w_\ell(v), \tag{4.3}$$

where S is a subset of the ground set E. We define the follower's objective functions $d_s(S)$ and $d_b(S)$ analogously; there we use the follower's weight function w_f instead of w_ℓ.

$$d_s(S) = \sum_{v \in S} \quad w_f(v), \tag{4.4}$$

$$d_b(S) = \max_{v \in S} \quad w_f(v). \tag{4.5}$$

The bottleneck functions here are written for a minimization problem; for a maximization problem, however, they change to $c_b(S) = \min_{v \in S} \ w_\ell(v)$, and $d_b(S) = \min_{v \in S} \ w_f(v)$.

Throughout this chapter, we discuss eight different variants of bilevel problems that emerge from choosing:

(C1) the leader's objective function from c_s or c_b,

(C2) the follower's objective function from d_s or d_b, and

(C3) the behavior of the follower from o or p.

For simplicity, we represent each of these variants using a 3-tuple, each component of the tuple capturing a binary choice from the above three factors.

Thus we represent a variant as $(c, d, *)$, wherein $c \in \{c_s, c_b\}$, $d \in \{d_s, d_b\}$, and $* \in \{o, p\}$.

The eight variants were studied thoroughly in a paper by Gassner & Klinz in 2009 [GK09]. There they considered all these variants over the Assignment problem. In this chapter, we build on this study, and discuss this very problem in Section 4.3. Later in this chapter, we also study the Bilevel Interval Scheduling and Bilevel Independent Set problems in similar settings.

Finally, we describe the definitions of the underlying combinatorial optimization problems that we consider here.

Assignment

Given: A bipartite graph $G = (V, E)$ and a weight function $w\colon E \to \mathbb{R}_+$.
Task: Find a perfect matching $M \subseteq E$ that minimizes $\sum_{e \in M} \ w(e)$.

Interval Scheduling

Given: A set of intervals $I = \{i_1, i_2, \ldots, i_n\}$ on the real line and a weight function $w\colon I \to \mathbb{R}_+$.
Task: Find a subset of pairwise-disjoint intervals $I' \subseteq I$ that maximizes $\sum_{i \in I'} \ w(i)$.

Independent Set

Given: A graph $G = (V, E)$ and a weight function $w\colon V \to \mathbb{R}_+$.
Task: Find an independent set $V' \subseteq V$ that maximizes $\sum_{v \in V'} \ w(v)$.

4.2.2 Associated Work

Lately, numerous articles have been devoted to studying the partitioned-items bilevel optimization problems.

Starting from the simplest case of the Bilevel Selection problem [Hen25], the study has expanded to the bilevel versions of more complex problems. For the underlying Selection problem, given a cost function over a finite set of items and a positive integer k, the task is to select exactly k items which minimize the total cost. In addition to the bilevel version of the Selection problem, Henke also deals with the *robust* bilevel version of the problem [Hen25].

One of the generalizations of the Bilevel Selection problem is the Bilevel Assignment problem. Gassner and Klinz studied the Bilevel Assignment problem and showed them to be NP-hard for seven variants out of the eight variants we discussed in Section 4.2.1 [GK09]. The unresolved case from [GK09] consists of bottleneck objective functions for both the leader and the follower in the optimistic setting.

Yet another generalization of the Bilevel Selection problem is the Bilevel Knapsack problem, which has been shown to be Σ_2^p-hard for three different variants, one of which is a partitioned-items problem formulation [CCLW14]. Additionally, two of the three problem variants have been shown to be solvable in pseudopolynomial time, and the remaining one is strongly NP-hard [MAVH12]. The authors of the former study also derived some approximation results for the three bilevel knapsack variants studied; they showed two of their variants do not allow polynomial-time approximation algorithms with a worst-case guarantee (unless $P = NP$), however, for the third variant, they were able to derive a polynomial-time approximation scheme [CCLW13]. This is the first approximation result derived for a Σ_2^p-hard problem in the history of approximation algorithms. Carvalho, Lodi, and Marcotte gave a polynomial-time algorithm to solve a continuous bilevel knapsack problem [CLM18], which was further improved by Fischer and Woeginger [FW20]. Yet another study on robust bilevel continuous knapsack problem was done by Buchheim and Henke [BH22].

Moreover, the partitioned-items version of the Bilevel Spanning Tree problem has been studied in [BHH22]; the authors also considered the eight variants of the problem as we do, and they showed that some variants are NP-hard, while some are in P. Additionally, a few other bilevel versions of the spanning tree problem have been studied [SZP19, SPR23]. A recent addition to the list is a study on the partitioned-items Bilevel Shortest Path problem, wherein Henke and Wulf completely classified the problem variants into classes of the polynomial hierarchy [HW25].

Arguably, several robust optimization problems can be seen through the lens of bilevel optimization [GKST25]. Some robust optimization problems can also be seen as an interplay between two hierarchical decision makers where the robust uncertainty lies in the decision variables of the second decision maker who is an adversary to the first decision maker. When seeing robust optimization from a bilevel point of view, the objective functions of both decision makers are in complete clash with each other; one

Optimistic		
	d_s	d_b
c_s	Σ_2^p-complete (Thm 4.10)	NP-complete (Thm 4.15)
c_b	Σ_2^p-complete (Thm 4.10)	P (Thm 4.11)
Pessimistic		
	d_s	d_b
c_s	Σ_2^p-complete (Thm 4.10)	NP-complete (Thm 4.14)
c_b	Σ_2^p-complete (Thm 4.10)	NP-complete (Thm 4.12)

Table 4.1: Computational complexity results of BIS on simple undirected graphs for different variants emerged from $(c_s/c_b, d_s/d_b, o/p)$.

wants to minimize it, the other maximize. Buchheim, Henke, and Hommelsheim investigated the robust counterparts of bilevel problems with uncertain follower's objective; they showed that, when considered interval uncertainty, the problem can be Σ_2^p-hard even when the underlying bilevel variant is NP-equivalent and the follower's problem is tractable [BHH21].

4.2.3 Contribution

In this chapter, we provide a variety of computational complexity results ranging over polynomial-time algorithms, NP-hardness, and even Σ_2^p-hardness, for a number of different variants of bilevel optimization problems.

In Section 4.3, we present our study on the bilevel assignment problem. Recall the formulation of the bilevel problem given in (4.1), and the eight variants mentioned in Section 4.2.1. During this study, we settle the last variant (of the eight variants mentioned in Section 4.2.1) left open by Gassner & Klinz for over 10 years (see for reference [GK09]). Moreover, we give an NP-hardness reduction that unitedly works for all the variants of the bilevel assignment problem discussed in [GK09]. Our reduction shows NP-hardness in the strong sense. Moreover, as a byproduct of this result, we show that there does not exist an approximation algorithm for the bilevel assignment problem, unless P = NP. Additionally, we show that the decision versions of these variants of the problem belong to the class NP, and thus prove NP-completeness for the decision versions of all considered variants. We remark that the Bilevel Assignment problem is yet another generalization of the Bilevel Selection problem. Bilevel Selection and its other generalization, namely the Bilevel Knapsack problem, have been thoroughly studied in [Hen25] and [CCLW14], respectively. The Bilevel Selection problem is polynomial-time solvable [Hen25], whereas we prove that its generalized version, the Bilevel Assignment problem, is NP-hard.

In the next study, in Section 4.4, we consider the Bilevel Interval Scheduling problem (BISch as short). We show that BISch is tractable when the leader's objective

Optimistic		
	d_s	d_b
c_s	NP-complete (Thm 4.17)	P (Thm 4.18)
c_b	NP-complete (Thm 4.20)	P (Thm 4.11)

Pessimistic		
	d_s	d_b
c_s	NP-complete (Thm 4.17)	P (Thm 4.19)
c_b	NP-complete (Thm 4.20)	NP-complete (Thm 4.20)

Table 4.2: Computational complexity results of BIS on the class of bipartite graphs for different variants emerged from $(c_s/c_b, d_s/d_b, o/p)$.

function and the follower's objective function are of sum type, in both the optimistic and the pessimistic settings. We give a dynamic programming algorithm that runs in $\mathcal{O}(n^4 \log n)$ time, where n is the total number of intervals in the given instance.

Pursuing the close link between the Interval Scheduling problem and the Independent Set problem, next, we carry forward our study of bilevel optimization on the Bilevel Independent Set problem (BIS) in Section 4.5. Here we start with a general study on simple undirected graphs. We get the results mentioned in Table 4.1 for the eight different variants emerging from considering $(c_s/c_b, d_s/d_b, o/p)$ on general simple undirected graphs. Additionally, the results mentioned in Table 4.2 are for all variants of BIS on bipartite graphs.

4.3 Bilevel Assignment

In the field of computer science, the Assignment problem is also referred to as Bipartite Matching. The first algorithm to solve the Assignment problem was given by Kuhn in 1955 [Kuh55]. It was later refined by Munkres in 1957, see [Mun57]. Kuhn called the algorithm as the *Hungarian method* as it heavily relies on ideas developed by Hungarian scientists, especially Kőnig and Egerváry.

As mentioned earlier, we consider the bilevel assignment problem in the same setting as defined in the Problem (4.1). Gassner & Klinz [GK09] first studied the eight variants considered in Section 4.2 on the assignment problem. To revise, the eight variants emerge by choosing the types of the objective functions for the leader and the follower, i.e. whether $c \in \{c_s, c_b\}$ and $d \in \{d_s, d_b\}$, and the behavior of the follower, either o or p.

Given a bipartite graph $G = (V, E)$ wherein the edge set is partitioned into the subsets E_ℓ and E_f which are controlled by the leader and the follower, respectively, we

formally define our bilevel assignment problem analogous to (4.1) below:

$$\min_{L \subseteq E_\ell} \quad c(L \cup F) \tag{4.6a}$$

where $F \subseteq E_f$ solves the follower's problem

$$\min^*_{F \subseteq E_f} \quad d(L \cup F) \tag{4.6b}$$

such that $L \cup F$ is a perfect matching in the given bipartite graph

Out of the eight variants, Gassner & Klinz [GK09] establish the NP-hardness for seven bilevel assignment variants, and leave the complexity of the eighth variant as an open problem; the eighth variant consists of both bottleneck functions c_b and d_b along with the optimistic behavior of the follower. To accomplish all their results, they derive *three* different hardness reductions from the Vertex Cover problem: Given an undirected graph $G = (V, E)$ and a positive integer k, the task is to find a subset $S \subseteq V$ of at most k vertices such that the removal of this set renders the graph edgeless, that is the graph $G[V \setminus S]$ does not contain any edges. Here, we derive our reduction from the Balanced Complete Bipartite Subgraph (BCBS) problem, and we not only resolve the complexity of the last open variant, but also give a unified reduction which works for all eight variants.

The open problem of Gassner & Klinz. It consists of the bottleneck objective functions for both decision makers, that is for $c = c_b$ and $d = d_b$, and further contains the optimistic behavior of the follower. Thus the open case of Gassner and Klinz is (c_b, d_b, o). We refer to this as the *Bilevel Bottleneck Assignment Problem* (BiBAP). An instance of BiBAP is built around a bipartite graph $G = (V, E)$. The edge set E is partitioned into a part E_ℓ that is controlled by the leader and a part E_f that is controlled by the follower. The leader has her own private weight function $w_\ell \colon E \to \mathbb{R}_+$ on the edges, and also the follower has a private weight function $w_f \colon E \to \mathbb{R}_+$ on the edges. The BiBAP problem is formally stated below.

$$\text{minimize} \quad \max\{w_\ell(e) \mid e \in L \cup F\} \tag{4.7a}$$

$$\text{so that} \quad L \subseteq E_\ell,$$

and so that $F \subseteq E_f$ solves the optimistic follower's problem

$$\text{minimize}^* \quad \max\{w_f(e) \mid e \in L \cup F\} \tag{4.7b}$$

so that $L \cup F$ is a perfect matching

Here, the optimization process is as discussed in Section 4.2. The leader and the follower sequentially select their desired subsets $L \subseteq E_\ell$ and $F \subseteq E_f$ respectively so that

$L \cup F$ forms a perfect matching in G. The objective of the leader is to minimize her private bottleneck weight $\max\{w_\ell(e) \mid e \in L \cup F\}$, and similarly the objective of the follower is to minimize his private bottleneck weight $\max\{w_f(e) \mid e \in L \cup F\}$. For simplicity of presentation we will assume that in the case of infeasible solutions, where $L \cup F$ does not form a perfect matching, the objective values of the leader and the follower become infinitely large. As a consequence, the leader will always pick some set L that can be extended to a perfect matching by adding some edges from E_f, and the follower will always select some set F whose union with L forms a perfect matching. Whenever the minimizer F in (4.7b) is not uniquely determined, the follower acts according to the optimistic setting and among all possible minimizers selects one that (in combination with the set L selected by the leader) yields the best objective value for the leader. In (4.7), this optimistic variant of minimization is indicated by the little star in the follower's problem.

On our quest to settle the open problem of Gassner & Klinz [GK09], we prove that the eighth and last variant of the bilevel assignment problem is NP-complete. Though our proof draws some ideas from [GK09], it is technically very different and uses a number of new ideas.

4.3.1 The Hardness Proof

In this section we present the NP-hardness proof for problem BiBAP. Eventually, we point out that our hardness reduction works for all eight variants of the bilevel assignment problem mentioned in Section 4.2.1. Additionally, we show that the decision versions of all these variants belong to the class NP. Consequently, we prove our main result, Theorem 4.4, stating the NP-completeness of the decision version of all variants of the bilevel assignment problem.

For the hardness result, the argument is done through a polynomial-time reduction from the NP-hard Balanced Complete Bipartite Subgraph problem; we refer the reader to Garey & Johnson [GJ79] and to Johnson [Joh87] for more information on this problem, in particular, about its hardness result.

Balanced Complete Bipartite Subgraph (BCBS)

Given:	A bipartite graph $H = (V_H, E_H)$ with bipartition $V_H = W_1 \cup W_2$ of the vertex set and with $E_H \subseteq \{\{w, w'\} \mid w \in W_1 \text{ and } w' \in W_2\}$, and a positive integer k.
Task:	Verify whether H contains a complete bipartite subgraph with k vertices on each side of the bipartition.

Now consider an instance of BCBS consisting of the bipartite graph $H = (W_1 \cup W_2, E_H)$ and the integer k. Without loss of generality we assume that both sides W_1 and W_2 have the same size, and we denote $n = |W_1| = |W_2|$; furthermore we will assume $k \leq n$. We construct bipartite graph $G = (V, E)$ with $4n$ vertices and $(3n^2 - |E_H| + 1)$ many edges as follows (also see Figure 4.1):

- For every vertex $v \in W_1$ and for every vertex $w \in W_2$, the vertex set V contains the corresponding vertices $a(v)$ and $a'(w)$.
- Additionally, the vertex set V contains extra $2(n-k)$ vertices $b_1, \dots, b_{n-k}$ and $b'_1, \dots, b'_{n-k}$, and another $2k$ vertices $c_1, \dots, c_k$ and $c'_1, \dots, c'_k$.
- For every $v \in W_1$ and $w \in W_2$, the edge set E contains the edge $\{a(v), a'(w)\}$ if and only if $\{v, w\} \notin E_H$.
- The edge set E connects every vertex $a(v)$ to the n vertices $b'_1, \dots, b'_{n-k}$ and $c'_1, \dots, c'_k$, and it connects every vertex $a'(w)$ to the n vertices $b_1, \dots, b_{n-k}$ and $c_1, \dots, c_k$.
- Finally the edge set E contains the single edge $\{c_k, c'_k\}$.

Note that the constructed graph G is indeed bipartite, as every edge connects some unprimed vertex to some primed vertex. Next, we specify the edge weights for the leader and for the follower.

- For the leader, the edges $e = \{a(v), a'(w)\}$ all have weight $w_\ell(e) = 1$, whereas the remaining edges in E have weight 0.
- For the follower, the $2n$ edges $e = \{a(v), c'_k\}$ and $e' = \{a'(w), c_k\}$ with $v \in W_1$ and $w \in W_2$ carry the weight $w_f(e) = w_f(e') = 1$. All other edges in E have weight 0 for the follower.

Finally, we specify the edge sets E_ℓ and E_f controlled by the leader and the follower.

- The set E_ℓ consists of the edges $\{a(v), b'_j\}$ and $\{a'(w), b_j\}$ with $v \in W_1$, $w \in W_2$ and $1 \le j \le n-k$.
- The set E_f contains all the remaining edges: the edges $\{a(v), a'(w)\}$, the edges $\{a(v), c'_i\}$ and $\{a'(w), c_i\}$, and the single edge $\{c_k, c'_k\}$.

This completes the construction of the BiBAP instance. Since all edge weights w_ℓ and w_f only take the values 0 and 1, the possible objective values for the leader and the follower, in the BiBAP instance, are 0 and 1. We next state some simple observations on the behavior of the leader and the follower. Note that the matching $L \subseteq E_\ell$ selected by the leader contains at most $2(n-k)$ edges, since every edge in E_ℓ is incident to one of the $2(n-k)$ vertices $b_1, \dots, b_{n-k}$ and $b'_1, \dots, b'_{n-k}$.

Lemma 4.1. *Let $L \subseteq E_\ell$ be an arbitrary matching chosen by the leader.*

(i) If $|L| < 2(n-k)$, then no reaction $F \subseteq E_f$ of the follower can yield a perfect matching $L \cup F$ for the graph G. Hence in this case, the objective values of the leader and of the follower are infinite.

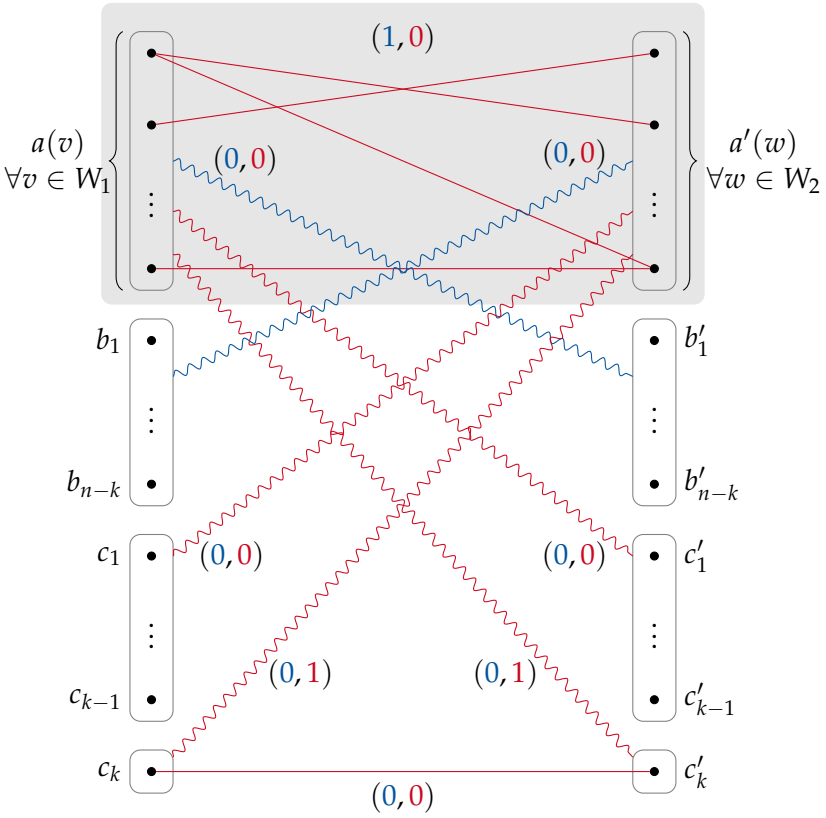


Figure 4.1: NP-hardness reduction from BCBS to BiBAP. Here, we use straight lines to show an edge between a specific pair of nodes, whereas we use a wavy line joining two sets to describe that every vertex of one set is adjacent to every vertex of the other set. The leader's set of edges are drawn in blue while the follower's set of edges are drawn in red. The edge weight for the leader and the follower is given in blue and red, respectively.

(ii) If $|L| = 2(n-k)$, then there exists a reaction $F \subseteq E_f$ of the follower that yields a perfect matching $L \cup F$ in graph G. Hence in this case, the objective values of the leader and of the follower are finite.

Proof. For (i), observe that the $2(n-k)$ vertices $b_1, \ldots, b_{n-k}$ and $b'_1, \ldots, b'_{n-k}$ are pairwise non-adjacent and are only incident to edges controlled by the leader. If $|L| < 2(n-k)$, then one of these $2(n-k)$ vertices will not be saturated by $L \cup F$.

For (ii), observe that every matching $L \subseteq E_\ell$ with $|L| = 2(n-k)$ saturates the $2(n-k)$ vertices $b_1, \ldots, b_{n-k}$ and $b'_1, \ldots, b'_{n-k}$ together with $n-k$ of the a-vertices and $n-k$ of the a'-vertices. The unsaturated vertices may be divided into two groups: The first group consists of the k unsaturated a-vertices together with $c'_1, \ldots, c'_k$, and the second group

consists of the k unsaturated a'-vertices together with $c_1, \dots, c_k$. As either group induces a balanced complete bipartite subgraph in G and as all edges in these two induced subgraph belong to set E_f, the follower easily extends L to a perfect matching $L \cup F$ for G. □

Lemma 4.2. *Let $L \subseteq E_\ell$ with $|L| = 2(n-k)$ be some matching chosen by the leader, and let $F \subseteq E_f$ be the optimal reaction of the follower to L.*

(i) If F does not contain any of the edges $\{a(v), a'(w)\}$ with $v \in W_1$ and $w \in W_2$, then the leader has objective value 0 and the follower has objective value 1.

(ii) If F contains one of the edges $\{a(v), a'(w)\}$ with $v \in W_1$ and $w \in W_2$, then the leader has objective value 1 and the follower has objective value 0.

Proof. The only edges with non-zero weight for the leader are the edges $\{a(v), a'(w)\}$, and all these edges are controlled by the follower. Hence the leader has objective value 0 in (i) and objective value 1 in (ii).

Let U denote the set of vertices $a(v)$ with $v \in W_1$ that are not saturated by L, and let U' denote the set of vertices $a'(w)$ with $w \in W_2$ that are not saturated by L. Note that $|U| = |U'| = k$. In (i) the follower must saturate the vertices in U by matching them to $c'_1, \dots, c'_k$. As the edge to vertex c'_k has weight 1 for the follower, his objective value is 1. In (ii) the follower saturates one of the vertices in U and one of the vertices in U' by picking an edge $\{a(v), a'(w)\}$ for F. The remaining $k-1$ vertices in U can be matched to $c'_1, \dots, c'_{k-1}$, the remaining $k-1$ vertices in U' can be matched to $c_1, \dots, c_{k-1}$, and finally the two vertices c_k and c'_k can be matched to each other at cost 0. This yields an objective value of 0 for the follower. □

Lemma 4.3. *In the BiBAP instance the leader can reach an objective value of 0, if and only if the bipartite graph $H = (W_1 \cup W_2, E_H)$ in the BCBS instance possesses a complete bipartite subgraph with k vertices on each side of the bipartition.*

Proof. ($\Leftarrow$) First assume that there exist subsets $X_1 \subseteq W_1$ and $X_2 \subseteq W_2$ with $|X_1| = |X_2| = k$ so that $X_1 \cup X_2$ induces a complete bipartite subgraph in H. Then the leader picks for her set L a perfect matching between $b'_1, \dots, b'_{n-k}$ and the vertices $a(v)$ with $v \in W_1 \setminus X_1$ together with a perfect matching between $b_1, \dots, b_{n-k}$ and the vertices $a'(w)$ with $w \in W_2 \setminus X_2$. Since $X_1 \cup X_2$ induces a complete bipartite subgraph in graph H, the unsaturated a-vertices and a'-vertices induce an independent set in graph G. Now Lemma 4.2.(i) completes the argument.

($\Rightarrow$) Next assume that graph H does not contain a complete bipartite subgraph with k vertices on each side of the bipartition. Then for any action L of the leader, let $Y_1 \subseteq W_1$ denote the vertices $v \in W_1$ for which L does not saturate $a(v)$, and let $Y_2 \subseteq W_2$ denote

the vertices $w \in W_2$ for which L does not saturate $a'(w)$. Then $|Y_1| = |Y_2| = k$, and by our assumption the subgraph of H induced by $Y_1 \cup Y_2$ can not be complete. Hence there exists at least one edge $e = \{a(v), a'(w)\}$ in G, so that the matching L does neither saturate $a(v)$ nor $a'(w)$. Lemma 4.2 yields the small objective value 0 for the follower in case he puts such an edge e into his set F, and a large objective value 1 for the cases where he does not use any such edge. Hence the follower will choose such an edge e for F, and will thereby yield an objective value of 1 for the leader. □

Lemma 4.3 establishes the correctness of our reduction: The optimal objective value of the leader in the constructed BiBAP instance tells us the answer to the BCBS instance. Since the edge weights introduced in the reduction are polynomially bounded in the input instance size and the encoding of the reduced instance does not decrease, the reduction actually yields NP-hardness in the strong sense.

The polynomial-time reduction from BCBS to BiBAP is simple and highly structured. In the constructed instance of BiBAP, the follower has so little decision power that behavior according to the optimistic setting and behavior according to the pessimistic setting yield the same outcome. Furthermore, it can be verified that the behavior of leader and follower also does not change, if one or both of them switch from bottleneck objective function to sum objective function. Hence, our construction yields as a by-product a unified alternative proof for the seven hardness results that have been derived in [GK09]. Finally, we remark that our reduction also implies the inapproximability of problem BiBAP: Unless P = NP, there is no polynomial-time approximation algorithm for BiBAP that approximates the optimal objective value of the leader with a worst case ratio. This holds because, according to our NP-hardness reduction, it is hard to decide whether the leader achieves an objective value 0 (see Lemma 4.2 and Lemma 4.3). Thus we claim the inapproximability, as any approximation result would contradict this hardness result.

4.3.2 Containment in NP

Having proven the NP-hardness for all 8 variants of the bilevel assignment problem in the previous section, we would like to point out that we are far from determining the exact complexity class of the decision version of those problems. Even if the decision version of a problem is proven to be NP-hard, it could still be proven harder (for example, Σ_2^p-hard), unless its containment in NP is proven as well. In particular, decision versions of bilevel problems tend to be Σ_2^p-hard because of their innate intertwined structure. As discussed at the beginning of this chapter, a bilevel optimization problem consists of two interleaved optimization problems: we are interested in finding *a* leader's feasible action such that along with a follower's optimal reaction to it, the leader's objective function evaluates within the desired bounds. For a more detailed explanation, see the survey by Woeginger titled "The trouble with the second quantifier" [Woe21].

Our goal in this section is to identify the exact complexity of decision versions of all bilevel variants considered. So far, we have shown NP-hardness. Now we show the

containment in NP and settle the complexity for these variants. It is often more involved to verify the containment in NP for a bilevel optimization problem compared to its underlying single level variant. This is because, in addition to checking the feasibility of the given certificate, we may also need to verify that the certificate, in fact, accounts for the *optimal* reaction of the second decision maker. Thus verification of a certificate to a bilevel optimization problem requires solving the lower level optimization problem. In light of the hardness results in [GK09], we want to point out that the authors of [GK09] did not comment on the completeness of their results—they mainly focused on the optimization versions of the problem than that of the decision versions—and thus did not fully resolve the exact complexity of the problems. In this section, we close this gap by showing that all eight variants of the bilevel assignment problem indeed belong to NP.

In order to show that a decision problem belongs to the class NP, one needs to show that there exists polynomial-sized certificate that is also verifiable in polynomial time. Let $G = (V, E_\ell \dot\cup E_f)$ be the given graph with weight functions w_ℓ and w_f for the leader and the follower, respectively. We consider the leader's action $L \subseteq E_\ell$ as our polynomial-sized certificate and show that it can be verified whether choosing L leads to the final objective value of at most k for the leader, where k is the bound given in the decision version of the problem.

Each of our attempts to prove containment in NP involves some sophisticated strategies. In what follows, we club different variants together to implement our strategies:

- *Case* $(\mathsf{c}_s, \mathsf{d}_s, \mathsf{o/p})$: When both the leader and the follower have sum type objective functions, and the leader chooses $L \subseteq E_\ell$, delete all vertices $V(L)$ saturated by the matching L in G. Note that if we can calculate the follower's optimal reaction F_L to a leader's action L, then we can verify whether the leader's objective function value $\mathsf{c}_s(L \cup F_L) \leq k$. So, the task boils down to solving for the follower's optimal reaction F_L in polynomial time. For this, we consider the subgraph G' of G on vertex set $V \setminus V(L)$ with only follower's edges incident on it, and further update the follower's weight function as follows:

  – for an optimistic follower, update,

  $$w_f(e) \leftarrow w_f(e) + \varepsilon \cdot w_\ell(e),$$

  – for a pessimistic follower, update,

  $$w_f(e) \leftarrow w_f(e) - \varepsilon \cdot w_\ell(e),$$

  where $\varepsilon := \frac{1}{\sum_{e \in E_f} w_f(e)} > 0$ is a very small number. Use the Hungarian method for the assignment problem as a subroutine to calculate the minimum weighted perfect matching on G' with respect to the updated weight function w_f [Kuh55]. Let M be the solution obtained using the Hungarian method. We claim that M

is equal to the follower's optimal reaction F_L. Since ε is a small positive number, the follower's weights are perturbed only slightly to capture the essence of the behavior of the follower towards the leader. Note that these tiny changes do not influence the choice of the follower's matching F. Thus, the set M belongs to the set of follower's optimal reactions to L. Further if there are multiple follower's optimal reactions available, the follower chooses the best or the worst possible outcome for the leader, according to the updated function in the optimistic or pessimistic setting, respectively.

- *Case* $(\mathsf{c}_b, \mathsf{d}_s, \mathsf{o/p})$: Here, since the leader's objective function is of bottleneck type, the task boils down to verifying whether $w_\ell(e) \leq k$ for all $e \in L \cup F_L$, for the given leader's action L and the follower's optimal reaction F_L to it. For this we again consider the graph G' without vertices covered by L along with only follower's edges $E' = \{uv \mid u, v \notin V(L) \text{ and } uv \in E_f\}$. We further color an edge $e \in E'$ with color *red* if $w_\ell(e) > k$. That means, the leader's action L results into a final assignment which evaluates the leader's objective function value to $\leq k$ if and only if the follower does not choose any of the red edges. To calculate the follower's optimal reaction F_L, we update the follower's weight function as follows:

  – for an optimistic follower, update

$$w_f(e) \leftarrow \begin{cases} w_f(e) + \varepsilon & \text{if } e \text{ is red,} \\ w_f(e) & \text{otherwise,} \end{cases}$$

  – for a pessimistic follower, update

$$w_f(e) \leftarrow \begin{cases} w_f(e) - \varepsilon & \text{if } e \text{ is red,} \\ w_f(e) & \text{otherwise,} \end{cases}$$

  where $\varepsilon > 0$ is a very small number. Then, use the Hungarian algorithm to calculate the minimum weighted perfect matching in G' according to the updated weight function w_f. Among multiple feasible reactions available, an optimistic follower would try to avoid taking red edges as they worsen the leader's objective function, whereas a pessimistic follower would rather take a red edge to be cynical towards the leader. All in all, from here, F_L can be calculated in polynomial time and consequently the leader's objective value. So one can verify a certificate in polynomial time and both these variants belong to the class NP.

- *Case* $(\mathsf{c}_s, \mathsf{d}_b, \mathsf{o/p})$: The cases where the follower's objective function is of bottleneck type are a bit involved as the leader has the power to restrict the weight of the follower's reaction. The follower's goal is to minimize the $\max\{w_f(e) \mid e \in L \cup F\}$. Note that a leader's reaction L already bounds the follower's function value from below, let us denote this lower bound by $lower_{\mathsf{d}_b} := \max_{e \in L} w_f(e)$. When the leader

has a sum type objective function, and we have given the leader's action L, the task boils down to calculating F_L in G', the remaining graph with $E' = \{uv \mid u, v \notin V(L) \text{ and } uv \in E_f\}$, and checking whether $\sum_{e \in L \cup F_L} w_\ell(e) \leq k$. We sort the follower's weights of all edges in E' in a non-descending order. Let the ordering be $\pi := (w_f^1, w_f^2, \ldots, w_f^m)$. Let $i \in [m]$ be the smallest integer such that $lower_{\mathsf{d}_b} \leq w_f^i$. Then for each $j \in \{lower_{\mathsf{d}_b}, w_f^i, w_f^{i+1}, \ldots, w_f^m\}$ in an incremental fashion, we solve for a minimum weighted perfect matching in the subgraph $G'_j = (V'_j, E'_j)$ where $V'_j := V \setminus V(L)$ and $E'_j := \{e \in E' \mid w_f(e) \leq j\}$ with respect to the following weight functions depending on the variant in hand:

- in the optimistic setting, use the weight function w_ℓ, and
- in the pessimistic setting, use the weight function $-w_\ell$.

Use the Hungarian method to solve for the minimum weighted perfect matching in G'_j according to the respective weight function: In the optimistic setting the follower chooses, if he had multiple options, the one that additionally minimizes the leader's objective function. On the other hand, a pessimistic follower chooses a set that maximizes the leader's objective function. It is clear that the resultant perfect matching obtained this way is equal to F_L. Then, we calculate the final objective value for the follower and check whether $\mathsf{c}_s(L \cup F_L) \leq k$.

- *Case* $(\mathsf{c}_b, \mathsf{d}_b, \mathsf{o/p})$: The last pair of variants have bottleneck objective functions for both the leader and the follower. Here, we use a combination of two strategies mentioned in the previous cases. First, since the leader is of bottleneck type, it is crucial to verify whether for all $e \in L \cup F_L$, $w_\ell(e) \leq k$. We again consider only the remaining graph G' with the follower's edges $E' \subseteq E_f$. Color an edge $e \in E'$ red, if $w_\ell(e) > k$. To tackle with the follower's bottleneck function, we consider the follower's weight values in an ascending order similar to what we did in the previous case. For each $j \in \{lower_{\mathsf{d}_b}, w_f^i, w_f^{i+1}, \ldots, w_f^m\}$ (where i is the smallest index such that $lower_{\mathsf{d}_b} \leq w_f^i$) in an incremental fashion, we solve for a minimum weighted perfect matching in the subgraph $G'_j = (V'_j, E'_j)$ where $V'_j := V \setminus V(L)$ and $E'_j := \{e \in E' \mid w_f(e) \leq j\}$ with respect to the following weight functions depending on the variant in hand:

  - in the optimistic setting, use the weight function

$$w(e) \leftarrow \begin{cases} \varepsilon & \text{if } e \text{ is red,} \\ 0 & \text{otherwise,} \end{cases}$$

  - in the pessimistic setting, use the weight function

$$w(e) \leftarrow \begin{cases} -\varepsilon & \text{if } e \text{ is red,} \\ 0 & \text{otherwise,} \end{cases}$$

for a very small number $\varepsilon > 0$. Solve the instance G'_j with weight function w to calculate the minimum weighted perfect matching. Consider the perfect matching obtained in G'_j for the smallest value of j. If we find a matching, then depending on whether a red edge was selected or not, we determine whether the original bilevel optimization instance is a yes-instance.

Thus, we conclude that the decision versions of all bilevel assignment variants are contained in NP. All in all, we formulate the following summarizing theorem.

Theorem 4.4. *A decision version of a variant of the bilevel assignment problem is* NP*-complete in the strong sense when the leader has bottleneck or sum objective function* (c_b/c_s)*, the follower has bottleneck or sum objective function* (d_b/d_s)*, and the follower behaves in an optimistic or pessimistic way* (o/p)*.*

4.4 Bilevel Interval Scheduling

An instance of the Interval Scheduling (ISch) problem consists of a set of intervals I on the real line and a weight function $w\colon I \to \mathbb{R}_+$. The task is to find a subset $I' \subseteq I$ of pairwise non-intersecting intervals that maximizes the total sum of weights within I'. Note that, unlike the Assignment problem in the previous section, the underlying Interval Scheduling problem is a maximization problem. In this thesis, we study various different variants of the Interval Scheduling problem according to the Problem (4.1), therefore we refer to its normal, non-bilevel version as the underlying Interval Scheduling problem. Frank, in 1976, gave a dynamic program for solving Interval Scheduling optimally [Fra76]. In fact, he proposed a polynomial-time algorithm to find the maximum weighted independent set on chordal graphs. Since interval graphs are a subclass of chordal graphs and the Interval Scheduling problem is equivalent to finding the maximum weighted independent set on interval graphs, this algorithm also works for our problem. Since we need the solution of the Interval Scheduling problem as a subroutine in our algorithm, we briefly revise the well-known dynamic program for Interval Scheduling in our words below.

4.4.1 Reviewing the Interval Scheduling Problem

Consider an instance $\mathcal{I}$ of Interval Scheduling with a set of intervals $I = \{i_1, i_2, \ldots, i_n\}$, where each interval $i_j = [a_j, b_j)$ starts at a_j and ends at b_j, and a weight function $w\colon I \to \mathbb{R}_+$ which assigns a weight $w(i_j) \geq 0$ for each interval $i_j \in I$. We denote by $\mathcal{F}(I)$ the set of feasible solutions of $\mathcal{I}$, i.e., $\mathcal{F}(I)$ is the family of all sets of pairwise-disjoint intervals in I. Let $A := \{a_1, \ldots, a_n\}$ and $B := \{b_1, \ldots, b_n\}$ be the collection of start- and end-points of all intervals, respectively. Without loss of generality, we assume that the intervals are ordered according to a non-descending order of their end-points,

i.e., we have $b_i \leq b_j$ for all $i \leq j$. We define Interval Scheduling in terms of half-open intervals because, the Interval Scheduling problem with closed intervals in which the endpoints are allowed to intersect, can be easily converted to our problem. Moreover, the setting of half-open intervals allows us to be carefree regarding the boundary cases.

Definition 4.5. Define $p(k)$ to be the largest possible index smaller than k such that, for the pair i_k and $i_{p(k)}$ of intervals, we have $b_{p(k)} \leq a_k$. That means $i_{p(k)}$ is the last interval in the ordering which completely ends before the interval i_k starts. If no such interval exists, then set $p(k) := 0$.

Now, we define an operator opt^{ISch} such that $\text{opt}^{\text{ISch}}[k]$ returns the maximum weight of a subset of pairwise-disjoint intervals within $\{i_1, i_2, \ldots, i_k\}$. Define $\text{opt}^{\text{ISch}}[0] := 0$.

We use a bottom-up approach and recursively calculate the values of $\text{opt}^{\text{ISch}}[k]$ for all $k \in [n]$. Observe that the optimal value of the solution to the given Interval Scheduling instance resides within $\text{opt}^{\text{ISch}}[n]$. In the bottom-up approach of our dynamic program, assume that all values of $\text{opt}^{\text{ISch}}[j]$ for $j \leq k-1$ are calculated. Then recursively we get,

$$\text{opt}^{\text{ISch}}[k] := \max\Big\{\text{opt}^{\text{ISch}}[k-1], \text{opt}^{\text{ISch}}[p(k)] + w(i_k)\Big\}.$$

Depending on whether the interval i_k contributes towards the max-weighted subset of intervals, we arrive to $\text{opt}^{\text{ISch}}[k]$ from different sets. Thus we maintain a predecessor, $\text{pred}(k)$ for each $k \in [n]$, to remember how we calculated the value of $\text{opt}^{\text{ISch}}[k]$.

$$\text{pred}(k) := \begin{cases} k-1 & \text{if } \text{opt}^{\text{ISch}}[k] = \text{opt}^{\text{ISch}}[k-1], \\ p(k) & \text{otherwise.} \end{cases}$$

Once the values of $\text{opt}^{\text{ISch}}[k]$ and $\text{pred}(k)$ are calculated for all $k \in [n]$, the optimal value of the Interval Scheduling problem is $\text{opt}^{\text{ISch}}[n]$, and the subset of intervals corresponding to the optimal value can be calculated by traversing back through the predecessors in a particular way. We store the corresponding subset of intervals in $\text{Sol}^{\text{ISch}}(I, w)$. For exact details about how to compute $\text{Sol}^{\text{ISch}}(I, w)$, refer to Algorithm ISch-DP.

Due to the bottom-up construction of our dynamic program and since each step of recurrence relation can be executed in polynomial time, Algorithm ISch-DP produces an optimal solution in polynomial time. Sorting of the intervals according to their finish times takes $\mathcal{O}(n \log n)$ time. Finding the $p(k)$ values can be done using binary search in $\mathcal{O}(n \log n)$ time. Additionally, calculating the recurrence formula takes another $\mathcal{O}(n)$ time. Thus overall, the above mentioned algorithm has running time $\mathcal{O}(n \log n)$.

Algorithm ISch-DP: Dynamic program for the Interval Scheduling problem

Input: A set of intervals $I = \{i_1, i_2, \ldots, i_n\}$, such that $i_k = [a_k, b_k)$, $a_k < b_k$ for each $k \in [n]$, $b_i \leq b_j$ when $i \leq j$, and a weight function $w\colon I \to \mathbb{R}_+$.

Output: A maximum weighted subset of pairwise-disjoint intervals in I and its corresponding weight.

$\text{opt}^{\text{ISch}}[0] := 0$
for $k := 1$ **to** n **do**
  $p(k) :=$ largest index less than k such that $b_{p(k)} \leq a_k$, if no such interval exists, set $p(k) := 0$.
  $\text{opt}^{\text{ISch}}[k] := \max\left\{\text{opt}^{\text{ISch}}[k-1], \text{opt}^{\text{ISch}}[p(k)] + w(i_k)\right\}$
  $\text{pred}(k) := \begin{cases} k-1 & \text{if } \text{opt}^{\text{ISch}}[k] = \text{opt}^{\text{ISch}}[k-1], \\ p(k) & \text{otherwise.} \end{cases}$
$x := n$
$\text{Sol}^{\text{ISch}}(I, w) := \varnothing$
while $x \neq 0$ **do**
  if $\text{opt}^{\text{ISch}}[x] \neq \text{opt}^{\text{ISch}}[\text{pred}(x)]$ **then**
    $\text{Sol}^{\text{ISch}}(I, w) := \text{Sol}^{\text{ISch}}(I, w) \cup \{i_x\}$
  $x := \text{pred}(x)$
return $\text{opt}^{\text{ISch}}[n]$ and $\text{Sol}^{\text{ISch}}(I, w)$.

4.4.2 Problem Formulation

In this section, we study the bilevel version of the Interval Scheduling problem as per the formulation given in the Problem (4.1). Let us delve straight into formulating the corresponding Bilevel Interval Scheduling problem.

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of intervals in which each interval $i_j = [a_j, b_j)$ consists of a start point a_j and an end point b_j such that $a_j, b_j \in \mathbb{R}_+$, and $a_j < b_j$. Let $\pi := (i_1, i_2, \ldots, i_n)$ be the sequence of intervals in I according to the non-decreasing ordering of their end points. So $b_j \leq b_{j'}$ for all $j \leq j'$. Define $I_k := \{i_1, i_2, \ldots, i_k\}$ for any $k \in [n]$. The intervals in I are partitioned into two sets, $I = I_\ell \dot\cup I_f$, where I_ℓ is the set of intervals controlled by the leader and I_f is controlled by the follower. Both players have their own weight functions $w_\ell, w_f\colon I \to \mathbb{R}_+$. The players also have their own objective functions $\mathrm{c}, \mathrm{d}\colon 2^I \to \mathbb{R}_+$; c and d are the objective functions of the leader and the follower, respectively. Here, we consider both c and d to be of sum type, i.e., we consider $\mathrm{c}_s(L \cup F) = \sum_{i \in L \cup F} w_\ell(i)$ and $\mathrm{d}_s(L \cup F) = \sum_{i \in L \cup F} w_f(i)$, respectively. The ultimate task is for the leader to choose her subset $L \subseteq I_\ell$ of pairwise-disjoint intervals, that the follower later augments with his chosen subset $F \subseteq I_f$ of additional pairwise-disjoint intervals—all the while trying to optimize his own objective function $\mathrm{d}_s(L \cup F)$—such that, in the end, the leader achieves the best possible weight for her objective function, i.e., $\mathrm{c}_s(L \cup F)$ is maximized. Note that we consider the variants with sum type objective

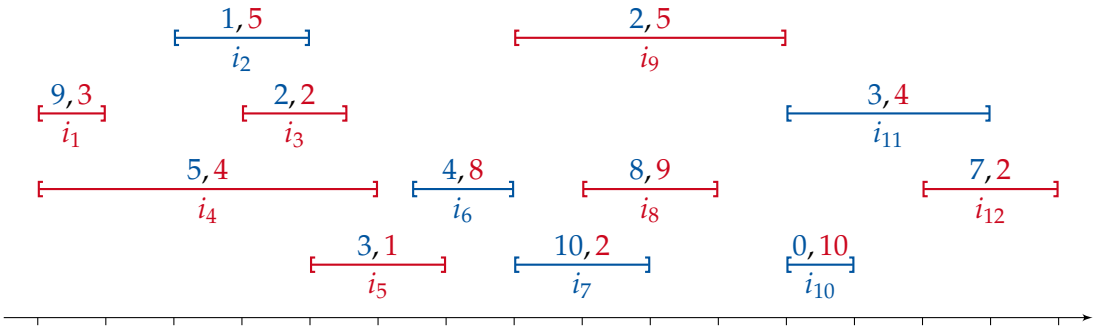


Figure 4.2: Example of an instance of the Bilevel Interval Scheduling problem. Intervals controlled by the leader and the follower are drawn in blue and red colors, respectively. The corresponding leader's and follower's weights for each interval are also shown in blue and red colors, respectively.

functions for the leader and the follower. So we only consider $(c_s, d_s, o/p)$ variants of the Bilevel Interval Scheduling problem. These are formally stated below:

$$\max_{L \subseteq I_\ell} \quad \sum_{i \in L \cup F} w_\ell(i) \tag{4.8a}$$

where $F \subseteq I_f$ solves the follower's problem

$$\max^*_{F \subseteq I_f} \quad \sum_{i \in L \cup F} w_f(i) \tag{4.8b}$$

such that $L \cup F$ is a set of pairwise-disjoint intervals in I,

where $* \in \{o, p\}$, and thus encodes the behavior or the follower, either optimistic or pessimistic. Also note that, the follower's weights are irrelevant for the objects controlled by the leader.

Refer to Figure 4.2 for an illustration of an instance of the Bilevel Interval Scheduling problem.

Let us again consider the $p(k)$ values for each $i_k \in I$ as per the Definition 4.5.

4.4.3 A Dynamic Program

We devise a dynamic programming algorithm, namely Algorithm BISch-DP, to solve the Bilevel Interval Scheduling problem for the $(c_s, d_s, o/p)$ variants. Later in Lemma 4.7, we prove that one can incorporate the follower's behavior, either optimistic or pessimistic, in the follower's weight function; In Lemma 4.7, we provide strategies to update w_f for this purpose. Thus throughout this section, we assume that, the follower always chooses according to his behavior, either optimistic or pessimistic.

We begin this section by stating a recursive formulation for our dynamic program, prove its correctness, and thus the validity of our algorithm. In Algorithm BISch-DP, $\text{opt}[n]$ returns the leader's optimal objective value, and $\text{Sol}^{\text{BISch}}(I)$ returns a corresponding leader's action which results in the optimal solution over the interval set I.

Now, we begin by giving a recursive formulation to calculate the optimal objective value of the leader in an instance restricted to the intervals $I_k = \{i_1, i_2, \ldots, i_k\}$. Let us define $\text{opt}[k]$ as the maximum value that the leader can achieve on an instance of interval set I_k. Assume that we have precalculated all $\text{opt}[k']$ for all $k' < k$, then we show that we can calculate the value $\text{opt}[k]$ in polynomial time using the precalculated values. Let $i_0 := [-\delta, 0)$ for a very small value $\delta > 0$ with weights $w_\ell(i_0) := 0$ and $w_f(i_0) := 0$. We define our base case as:

$$\text{opt}[0] := 0.$$

Further, we claim that $\text{opt}[k]$ satisfies the following recurrence relation.

$$\text{opt}[k] = \begin{cases} \max\left\{w_\ell(i_k) + \text{opt}[p(k)],\ \text{opt}[k-1]\right\} & \text{if } i_k \in I_\ell, \\ \max\limits_{\substack{j<k \\ i_j \in I_\ell \cup \{i_0\}}} \left\{\text{opt}[p(j)] + w_\ell(i_j) + \sum_{i \in \text{Sol}^{\text{ISch}}(I_f^{j,k}, w_f)} w_\ell(i)\right\} & \text{if } i_k \in I_f, \end{cases}$$

where, $I_f^{j,k} := \{i \in I_f \mid i \in I_k \setminus I_j \text{ and } i \cap i_j = \varnothing\}$, and the set $\text{Sol}^{\text{ISch}}(I_f^{j,k}, w_f)$ is obtained using the Algorithm ISch-DP on the set of intervals $I_f^{j,k}$ with the weight function w_f. The set $\text{Sol}^{\text{ISch}}(I_f^{j,k}, w_f)$ is a maximum weighted subset of pairwise-disjoint intervals in $I_f^{j,k}$ according to the follower's weight function w_f.

In the upcoming lemma, we prove the correctness of our recurrence formulation for $\text{opt}[k]$.

Lemma 4.6. *The function* $\text{opt}[k]$ *describes the leader's optimal objective value when only the subset of intervals* $\{i_1, i_2, \ldots, i_k\}$ *is considered.*

Proof. When $k = 0$, there are no intervals to choose from. Thus each set of pairwise-disjoint intervals is an empty set and the leader's objective function evaluates to 0. As we defined $\text{opt}[0] := 0$, this case holds. Otherwise $k \neq 0$, in that case either $i_k \in I_\ell$ or $i_k \in I_f$.

First, consider the case when $i_k \in I_\ell$. Let $I^\star \subseteq \{i_1, i_2, \ldots, i_k\}$ be a set of pairwise-disjoint intervals which optimizes the leader's objective function. An important remark is that, the set $I^\star$ need not be a max-weighted set of pairwise-disjoint intervals according to w_ℓ, but it is rather obtained by a systematic leader-follower interplay; it is a feasible solution, $L \cup F_L$, of the Bilevel Interval Scheduling problem. For such an $I^\star \subseteq \{i_1, i_2, \ldots, i_k\}$, exactly one of the following two holds: either $i_k \in I^\star$ or $i_k \notin I^\star$. (i) If $i_k \in I^\star$, then the rest of $I^\star$ satisfies $I^\star \setminus \{i_k\} \subseteq \{i_1, i_2, \ldots, i_{p(k)}\}$ as $i_x \cap i_k \neq \varnothing$ for all $p(k) < x \leq k$. Recall that $\{i_1, i_2, \ldots, i_{p(k)}\} =: I_{p(k)}$. The set $I^\star \setminus \{i_k\}$ maximizes the

Algorithm BISch-DP: A Dynamic program for Bilevel Interval Scheduling with sum type objective functions for both leader and follower

Input: A set of intervals $I = \{i_1, i_2, \ldots, i_n\}$ bi-partitioned into $I = I_\ell \dot\cup I_f$. Each interval $i_k := [a_k, b_k)$, $a_k, b_k \in \mathbb{R}_+$, $a_k < b_k$. For all $i \leq j$, $b_i \leq b_j$. And two weight functions $w_\ell, w_f \colon I \to \mathbb{R}_+$.

Output: The optimal objective value for the leader's objective function given in the Problem (4.8). The corresponding subset $L \subseteq I_\ell$ of pairwise-disjoint intervals such that an optimal reaction $F_L \subseteq I_f$ results into optimizing $\max \sum_{i \in L \cup F_L} w_\ell(i)$.

For the boundary cases, let $i_0 := [-\delta, 0)$ for a very small value $\delta > 0$.
$I_\ell := I_\ell \cup \{i_0\}$, define weights $w_\ell(i_0) := 0$ and $w_f(i_0) := 0$
$\text{opt}[0] := 0$
for $0 \leq r < s \leq n$ **do**
    $I_f^{r,s} := \{i \in I_f \mid i \in I_s \setminus I_r \text{ and } i \cap i_r = \varnothing\}$
    Compute $\text{Sol}^{\text{ISch}}(I_f^{r,s}, w_f)$ using the Algorithm ISch-DP.
for $k := 1$ **to** n **do**
    $p(k) :=$ largest index less than k such that $b_{p(k)} \leq a_k$, if no such interval exists, set $p(k) := 0$.
    if $i_k \in I_\ell$ **then**
        $\text{opt}[k] := \max\left\{w_\ell(i_k) + \text{opt}[p(k)],\ \text{opt}[k-1]\right\}$
        $\text{pred}(k) := \begin{cases} k-1 & \text{if } \text{opt}^{\text{ISch}}[k] = \text{opt}^{\text{ISch}}[k-1], \\ p(k) & \text{otherwise.} \end{cases}$
    else $i_k \in I_f$
        $\text{opt}[k] := \max\limits_{\substack{j<k \\ i_j \in I_\ell}} \left\{\text{opt}[p(j)] + w_\ell(i_j) + \sum_{i \in \text{Sol}^{\text{ISch}}(I_f^{j,k}, w_f)} w_\ell(i)\right\}$
        $\text{pred}(k) := \arg\max\limits_{\substack{j<k \\ i_j \in I_\ell}} \left\{\text{opt}[p(j)] + w_\ell(i_j) + \sum_{i \in \text{Sol}^{\text{ISch}}(I_f^{j,k}, w_f)} w_\ell(i)\right\}$
$x := n$ and $\text{Sol}^{\text{BISch}}(I) := \varnothing$
while $x \neq 0$ **do**
    if $i_x \in I_\ell$ **then**
        if $\text{opt}^{\text{ISch}}[k] \neq \text{opt}^{\text{ISch}}[k-1]$ **then**
            $\text{Sol}^{\text{BISch}}(I) := \text{Sol}^{\text{BISch}}(I) \cup \{i_x\}$
        $x := \text{pred}(x)$
    else $i_x \in I_f$
        $x := \text{pred}(x)$
        $\text{Sol}^{\text{BISch}}(I) := \text{Sol}^{\text{BISch}}(I) \cup \{i_x\}$
        $x := p(x)$
return $\text{opt}[n]$ and $\text{Sol}^{\text{BISch}}(I)$.

leader's objective function over the set of intervals $I_{p(k)}$, because if not, then there is yet another set $I' \subseteq I_{p(k)}$ which maximizes the leader's objective function over $I_{p(k)}$. Then we can construct a set of pairwise-disjoint intervals $I' \cup \{i_k\}$, which returns a leader's objective value greater than that of $I^\star$. Since the set $I' \cup \{i_k\}$ is also a feasible solution to the Bilevel Interval Scheduling problem over the interval set I_k, this contradicts the optimality of $I^\star$. Thus, we conclude that $I^\star \setminus \{i_k\}$ maximizes the leader's objective function over the set of intervals $I_{p(k)}$ which is captured in $\text{opt}[p(k)]$. So, $\sum_{i \in I^\star} w_\ell(i) = \text{opt}[p(k)] + w_\ell(i_k)$. (ii) If $i_k \notin I^\star$, then $I^\star \subseteq I_{k-1}$. We claim that $I^\star$ also maximizes the leader's objective function over I_{k-1}. If not, then a subset $I'' \subseteq I_{k-1}$, $I^\star \neq I''$ that optimizes the leader's objective function in I_{k-1} can replace $I^\star$ to return a better leader's valuation also for set I_k, which again contradicts the optimality of $I^\star$. Thus, we conclude that $I^\star$ maximizes the leader's objective function over I_{k-1}. Consequently, the leader's value obtained by $I^\star$ can be captured by $\text{opt}[k-1]$. Thus in this case, $\sum_{i \in I^\star} w_\ell(i) = \text{opt}[k-1]$. As a result, if $i_k \in I_\ell$, then

$$\text{opt}[k] = \max\Big\{w_\ell(i_k) + \text{opt}[p(k)],\ \text{opt}[k-1]\Big\}.$$

Now, consider the remaining case when $i_k \in I_f$. Once more, let $I^\star \subseteq \{i_1, i_2, \ldots, i_k\}$ be a set of pairwise-disjoint intervals which optimizes the leader's objective function. Since $i_k \notin I_\ell$, consider the largest index j, $j < k$, such that $i_j \in I^\star \cap I_\ell$. If no such j exists, then none of the leader's intervals are a part of the optimal solution $I^\star$, and we assume $j = 0$ with $w_\ell(i_0) := 0$, $w_f(i_0) := 0$, and $\text{opt}[0] := 0$, and $p(0) := 0$. Since, $I^\star \ni i_j$, we claim that $I^\star \subseteq I_{p(j)} \cup \{i_j\} \cup I_f^{j,k}$, where $I_{p(j)} := \{i_1, i_2, \ldots, i_{p(j)}\}$ and $I_f^{j,k} := \{i \in I_f \mid i \in I_k \setminus I_j \text{ and } i \cap i_j = \varnothing\}$ (also, define $I_f^{0,k} := I_f \cap I_k$).

The condition $I^\star \subseteq I_{p(j)} \cup \{i_j\} \cup I_f^{j,k}$ holds because, if j is the largest index such that $i_j \in I^\star \cap I_\ell$, then no other interval that intersects i_j belongs to $I^\star$. Moreover, each $i_{j'} \in I^\star$ with $j' > j$ also belongs to the set I_f. Note that the three sets $I_{p(j)}$, $\{i_j\}$, and $I_f^{j,k}$ are pairwise-disjoint,[1] in the sense that they do not share common intervals. For reference, see Figure 4.3. Additionally, each pair of intervals i and i' belonging to two distinct sets from $I_{p(j)}$, $\{i_j\}$, and $I_f^{j,k}$ is non-intersecting. Again, refer to Figure 4.3. Now, we claim that the following holds:

(a) $\sum_{i \in I^\star \cap I_{p(j)}} w_\ell(i) = \text{opt}[p(j)]$,

(b) $\sum_{i \in I^\star \cap \{i_j\}} w_\ell(i) = w_\ell(i_j)$, and

(c) $\sum_{i \in I^\star \cap I_f^{j,k}} w_\ell(i) = \sum_{i \in \text{Sol}^{\text{ISch}}(I_f^{j,k}, w_f)} w_\ell(i)$,

where the set $\text{Sol}^{\text{ISch}}(I_f^{j,k}, w_f)$ is obtained using the Algorithm ISch-DP on the set of intervals $I_f^{j,k}$ with the weight function w_f.

[1] not to be confused with the set of pairwise-disjoint intervals

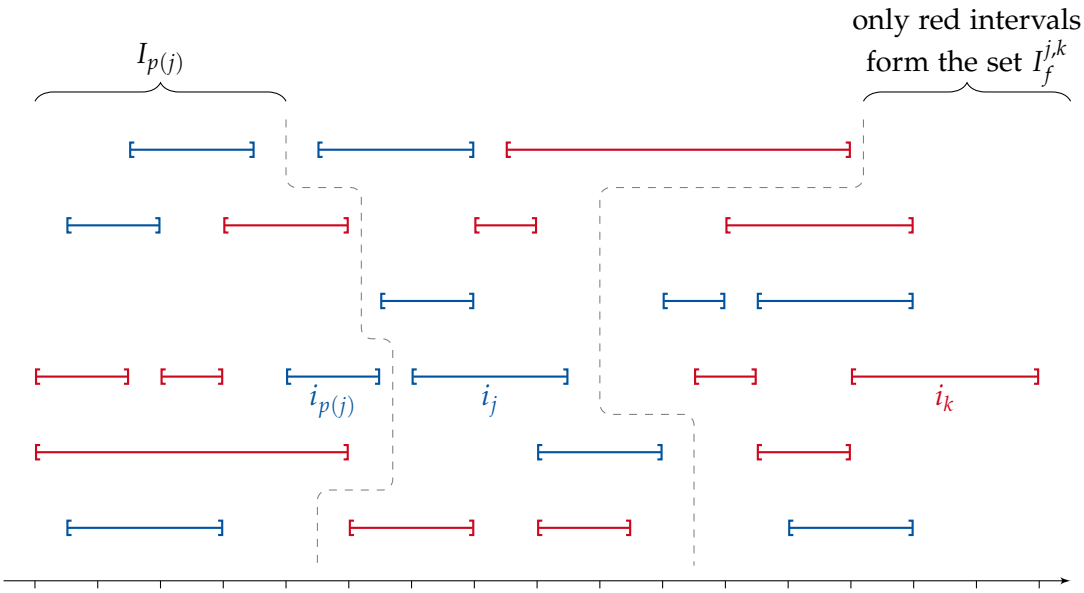


Figure 4.3: When $i_k \in I_f$: partitioning of the intervals across an interval $i_j \in I_\ell$. When i_j belongs to the leader's action $L \subseteq I_\ell$, the sets $I_{p(j)}$ and $I_f^{j,k}$ are completely disjoint. The intervals from $I_{p(j)}$ do not intersect with i_j, nor do they intersect with any interval in $I_f^{j,k}$. Similarly, i_j does not intersect with any interval from $I_f^{j,k}$.

Let us start with claim (a). Note that the set $I^\star \cap I_{p(j)}$ can be built by the leader-follower interplay of the Bilevel Interval Scheduling problem. Assume that claim (a) does not hold. Then either $\sum_{i \in I^\star \cap I_{p(j)}} w_\ell(i) < \text{opt}[p(j)]$ or $\sum_{i \in I^\star \cap I_{p(j)}} w_\ell(i) > \text{opt}[p(j)]$. For the former case, consider $(I^\star \setminus I_{p(j)}) \cup I_{\text{opt}[p(j)]}$, where $I_{\text{opt}[p(j)]}$ is the subset of pairwise-disjoint intervals in $I_{p(j)}$ that corresponds to the optimal value $\text{opt}[p(j)]$. The set $(I^\star \setminus I_{p(j)}) \cup I_{\text{opt}[p(j)]}$ is also a feasible solution for the Bilevel Interval Scheduling problem on the interval set I_k. Moreover,

$$\sum_{i \in (I^\star \setminus I_{p(j)}) \cup I_{\text{opt}[p(j)]}} w_\ell(i) \;>\; \sum_{i \in I^\star} w_\ell(i),$$

as $\sum_{i \in I^\star \cap I_{p(j)}} w_\ell(i) < \text{opt}[p(j)]$. This contradicts the optimality of $I^\star$.

As for the latter case, if $\sum_{i \in I^\star \cap I_{p(j)}} w_\ell(i) > \text{opt}[p(j)]$, this contradicts with the definition of $\text{opt}[k]$ because $\text{opt}[p(j)]$ is the maximum value that the leader can achieve on the interval set $I_{p(j)}$. Thus, we conclude that (a) always holds.

For claim (b), note that the chosen index $j < k$ is the last index that satisfies $i_j \in I^\star \cap I_\ell$. Indeed, since $i_j \in I^\star$, (b) holds trivially.

Now, we discuss claim (c). Since the leader's feasible action $(I^\star \cap I_\ell) \subseteq I^\star \setminus I_f^{j,k}$, and $i_j \in I^\star \cap I_\ell$, none of the intervals in $I_f^{j,k}$ overlaps with an interval from the leader's

action $I^\star \cap I_\ell$. Additionally, as we noticed earlier, the sets $I_f^{j,k}$ and $I_{p(j)}$ are disjoint, and the range of their intervals is non-overlapping. This allows the follower to solve his problem independently on each of these sets. Altogether, the follower's optimal reaction to $I^\star \cap I_\ell$ includes solving for the max-weighted subset of intervals, according to the weight function w_f, in $I_f^{j,k}$ and $I_{p(j)}$ which do not overlap with intervals in $I^\star \cap I_\ell$. To obtain such a subset of intervals over $I_f^{j,k}$, the follower simply solves the underlying Interval Scheduling problem on an instance with interval set $I_f^{j,k}$ and weight function w_f. We can use the Algorithm ISch-DP and obtain the solution $\text{Sol}^{\text{ISch}}(I_f^{j,k}, w_f)$. Thus, the leader's objective function in $I^\star \cap I_f^{j,k}$ must evaluate to the value $\sum_{i \in \text{Sol}^{\text{ISch}}(I_f^{j,k}, w_f)} w_\ell(i)$. Thus, claim (c) holds.

As a result of (a), (b), and (c), when $i_k \in I_f$, we enumerate over all $j < k$ such that $i_j \in I_\ell$ can be the last[2] interval of the leader's action in set I_k. Then the optimal solution of Bilevel Interval Scheduling on interval set I_k lies within one of these cases. As a result, when $i_k \in I_f$, we calculate the value of $\text{opt}[k]$ by using the following recursive formula:

$$\text{opt}[k] = \max_{\substack{j<k \\ i_j \in I_\ell}} \left\{ \text{opt}[p(j)] + w_\ell(i_j) + \sum_{i \in \text{Sol}^{\text{ISch}}(I_f^{j,k}, w_f)} w_\ell(i) \right\}. \qquad \square$$

Lemma 4.7. *Given a leader's action $L \subseteq I_\ell$, the follower's optimal reaction $F_L \subseteq I_f$ can be calculated in polynomial time both for the optimistic and the pessimistic behavior of the follower.*

Proof. In order to capture the essence of the optimistic or pessimistic behavior of the follower in (4.8), we use similar strategies as mentioned in Section 4.3.2. In particular, we update the follower's weight function w_f according to the following rules. For all $i \in I$, we do:

- if the follower behaves optimistically, update,

$$w_f(i) \leftarrow w_f(i) + \varepsilon \cdot w_\ell(i),$$

- if the follower behaves pessimistically, update,

$$w_f(i) \leftarrow w_f(i) - \varepsilon \cdot w_\ell(i),$$

where $\varepsilon := \frac{1}{\sum_{e \in E_f} w_f(e)} > 0$ is a very small number which slightly modifies the follower's weight function to capture the behavior of the follower.

[2] according to the ordering in π

Note that, for any subset $I' \subseteq I$,

$$\sum_{i \in I'} w_f(i) \leftarrow \sum_{i \in I'} \big(w_f(i) \pm \varepsilon \cdot w_\ell(i)\big).$$

Due to our choice of ε,

$$\arg\max_{I' \subseteq I} \sum_{i \in I'} \big(w_f(i) \pm \varepsilon \cdot w_\ell(i)\big) = \arg\max_{I' \subseteq I} \sum_{i \in I'} w_f(i).$$

So, the follower indeed always chooses a set $F \subseteq I_f$ which optimizes his objective function. Moreover, due to the additional $\pm \sum_{i \in I'} \varepsilon \cdot w_\ell(i)$ term—whenever the follower is faced with multiple feasible reactions to L—he additionally maximizes $\sum_{i \in I'} \varepsilon \cdot w_\ell(i)$ in the optimistic setting and minimizes $\sum_{i \in I'} \varepsilon \cdot w_\ell(i)$ in the pessimistic setting. That means, when the follower has multiple feasible reactions, in the optimistic setting, he acts in favor of the leader and in the pessimistic setting he acts against the interest of the leader. □

Theorem 4.8. *The algorithm BISch-DP returns the optimal value for the* Bilevel Interval Scheduling *problem given in* (4.8) *and it runs in polynomial time in the size of the input instance.*

Proof. We gave a recurrence formula to calculate the optimal objective value of the Bilevel Interval Scheduling problem when the instance consists of intervals only within the set $I_k = \{i_1, i_2, \ldots, i_k\}$. We store this optimal value in $\text{opt}[k]$. As shown in Lemma 4.6, our recurrence relation is correct, and it indeed maintains the leader's optimal objective value within interval set I_k. Furthermore, it is easy to verify that we implement this exact recurrence in Algorithm BISch-DP. We use the bottom-up approach in the execution of our algorithm. For each $k \in [n]$, $\text{opt}[j]$ for all $j < k$ is calculated before calculating the value of $\text{opt}[k]$. All in all, the output of the algorithm returns $\text{opt}[n]$, which is the optimal value of the Bilevel Interval Scheduling problem on the instance with interval set $\{i_1, i_2, \ldots, i_n\}$.

Now, for the running time of the Algorithm BISch-DP: It takes $\mathcal{O}(n \log n)$ time to arrange the intervals in non-decreasing order of their end points, another $\mathcal{O}(n \log n)$ to calculate the indices $p(j)$ for all $j \in [n]$. For all pairs $i < j$, it takes altogether $\binom{n}{2} \cdot n \cdot n \log n$, i.e., $\mathcal{O}(n^4 \log n)$, to calculate the set $I_f^{i,j}$ and the corresponding value of $\text{Sol}^{\text{ISch}}(I_f^{i,j}, w_f)$. Furthermore, it takes $\mathcal{O}(n^2)$ to execute the recursive formula in the bottom-up manner. Hence, Algorithm BISch-DP has time complexity $\mathcal{O}(n^4 \log n)$. □

For the forthcoming section, we generalize our study and consider the bilevel variants for the Independent Set problem over some other graph classes such as general graphs and planar bipartite graphs.

4.5 Bilevel Independent Set

In Section 4.4, we studied the Bilevel Interval Scheduling problem. The field of graph theory contains an analogous problem to Interval Scheduling which is solving the Independent Set problem on the class of interval graphs. So, in a way we can say that, in Section 4.4 we solely focused on studying the bilevel independent set problem on the class of interval graphs. There we focused on the interpretation of interval graphs in which each vertex is associated with an interval on the real line and two vertices in the graph are adjacent if and only if their corresponding intervals intersect each other. In this section, we generalize this study to different graph classes. We study the bilevel version of the Independent Set problem here. We call this problem Bilevel Independent Set (BIS). Unlike the polynomial-time results for BISch, here we show a variety of hardness results for different variants of BIS. When considering sum type objective functions for both leader and follower, the problem difficulty increases significantly from being polynomial-time solvable for the BISch to being Σ_2^p-hard for BIS.

An instance of Bilevel Independent Set (BIS) consists of a graph $G = (V_\ell \dot\cup V_f, E)$, where V_ℓ and V_f are the sets of vertices controlled by the leader and the follower, respectively. Moreover, let $w_\ell, w_f \colon V_\ell \dot\cup V_f \to \mathbb{R}_+$ be their weight functions, respectively. Then the Bilevel Independent Set problem is to optimize the following objective function.

$$\max_{L \subseteq V_\ell} \quad c(L \cup F) \tag{4.9a}$$

where $F \subseteq V_f$ solves the follower's problem

$$\max^*_{F \subseteq V_f} \quad d(L \cup F) \tag{4.9b}$$

such that $L \cup F$ is an independent set and $L \cup F \neq \varnothing$.

where $(^*)$ encodes the behavior of the follower, either optimistic or pessimistic. We again note that we are dealing with a maximization problem. Additionally, note that, we impose an extra condition that $L \cup F \neq \varnothing$ in our problem. This is because, if we allow $L \cup F = \varnothing$, then in case of bottleneck functions for both decision makers, since the minimum over an emptyset is ∞, both the leader and the follower will always choose a null set.

We study the above formulation given in (4.9) of the BIS problem on the class of simple undirected graphs and bipartite graphs. In the upcoming section, we begin the study of BIS on the simple undirected graph class.

4.5.1 Complexity on Simple Undirected Graphs

In this section, we study the 8 different variants of the Bilevel Independent Set problem and settle their computational complexity on simple undirected graphs. Please refer to Section 4.2.3 for a overview of our results given in Table 4.1.

To begin with, we study the variants of the Problem (4.9), where the leader's objective function is of sum or bottleneck type, so $c \in \{c_s, c_b\}$, the follower's objective function is of sum type, so (d_s), and the follower acts as per the optimistic or pessimistic setting, so o or p. Throughout this subsection, we work towards proving the result stated in Theorem 4.10. For all four variants $(c_s/c_b, d_s, o/p)$, we give a unified reduction from the following Σ_2^p-complete problem:

Problem: B_2^{CNF}

Given:	A boolean formula φ in CNF that contains exactly three literals per clause and consists of a variable set which is partitioned into $X \dot\cup Y$.
Task:	Does there exist a truth assignment to the variables in X such that there is no truth assignment to the variables in Y that would help satisfy the formula φ?

The problem B_2^{CNF} belongs to a class of bilevel problems called adversarial problems for which the underlying NP-complete problem is the classical Satisfiability problem. Moreover, B_2^{CNF} is equivalent to the 2-Alternating Quantified Satisfiability problem (also known as B_2), i.e., the problem whether $\exists X \forall Y\ \varphi$, where φ is a DNF formula. The problem B_2 was the very first problem that was proven to be complete for the second class of the polynomial hierarchy by Meyer and Stockmeyer [MS72]. Wrathall showed that B_2 remains Σ_2^p-complete when restricted to instances with at most three literals per clause [Wra76]. Since B_2 is Σ_2^p-complete, so is B_2^{CNF}. Berit Johannes stated, in her doctoral thesis, that B_2^{CNF} is Σ_2^p-complete [Joh11].

Construction of an instance of BIS**:** Let us consider an instance I of B_2^{CNF} which consists of a boolean formula φ with two sets of variables $X = \{x_1, x_2, \ldots, x_{n_1}\}$ and $Y = \{y_1, y_2, \ldots, y_{n_2}\}$, and the clause set $C = \{C_1, \ldots, C_m\}$, where each clause C_i contains exactly three literals. Now, we construct an instance $I' = (V_\ell \dot\cup V_f, E, w_\ell, w_f)$ of BIS as follows (also see Figure 4.4):

- For each variable $x_i \in X$, the set V_ℓ contains vertices $a_i, \bar a_i$. Denote the set of all these vertices by A.
- For each variable $y_i \in Y$, the set V_f contains vertices $b_i, \bar b_i$. Denote the set of all these vertices by B.
- For each clause $C_i = (l_i^1 \vee l_i^2 \vee l_i^3) \in C$, add vertices c_i^1, c_i^2, c_i^3, c_i to set V_f.
- For each $i \in [n_1]$, add an edge $\{a_i, \bar a_i\}$ to E. For each $i \in [n_2]$, add an edge $\{b_i, \bar b_i\}$ to E.
- For each $i \in [m]$, add edges $\{c_i^1, c_i^2\}, \{c_i^1, c_i^3\}, \{c_i^2, c_i^3\}, \{c_i^1, c_i\}, \{c_i^2, c_i\}, \{c_i^3, c_i\}$ to E.
- For all $i \in [m]$, if l_i^1 is a positive literal, say $l_i^1 = x_j$, then add edge $\{c_i^1, \bar a_j\}$ to E; add edge $\{c_i^1, \bar b_j\}$ to E when $l_i^1 = y_j$. Similarly, if l_i^1 is a negative literal, say $l_i^1 = \bar x_j$, then add edge $\{c_i^1, a_j\}$ to E. Also add $\{c_i^1, b_j\}$ to E when $l_i^1 = \bar y_j$. Add similar edges corresponding to all three literals l_i^1, l_i^2, l_i^3 of the clause C_i.

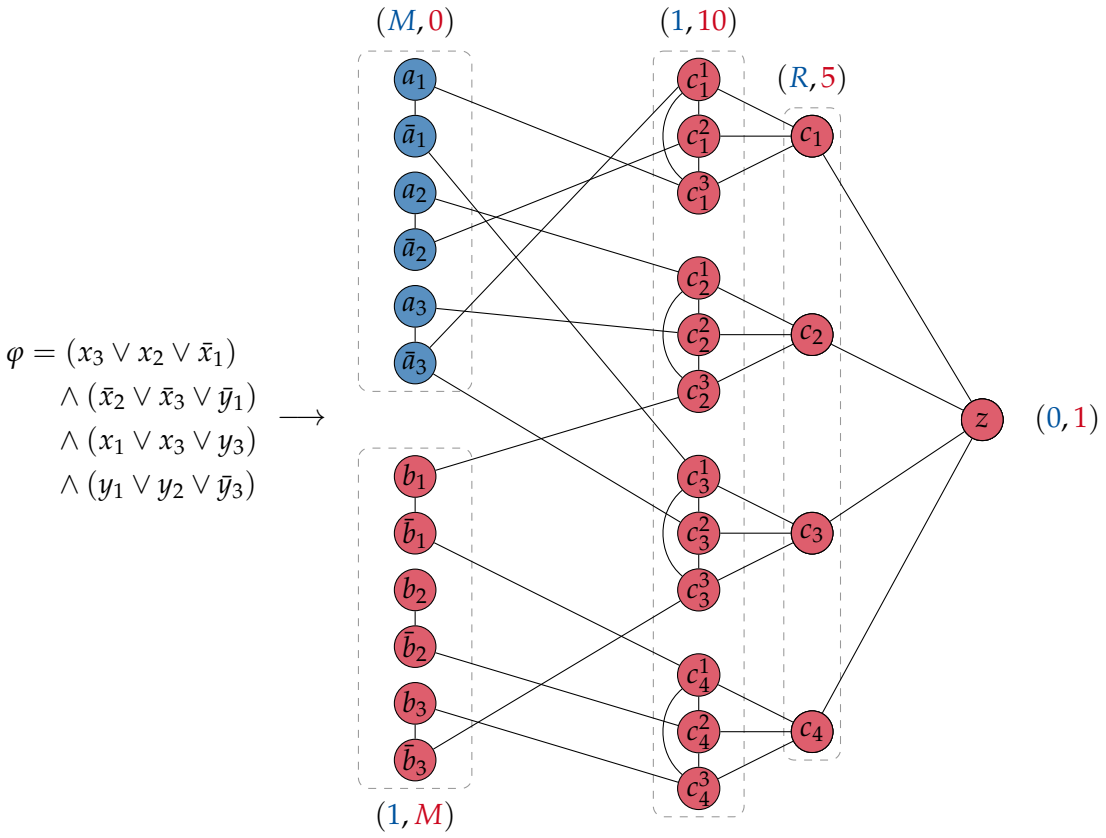


Figure 4.4: Σ_2^{p}-hardness reduction from B_2^{CNF} to Bilevel Independent Set for variants $(\mathsf{c}_s/\mathsf{c}_b, \mathsf{d}_s, \mathsf{o}/\mathsf{p})$.

- Finally, add a vertex z to set V_f and make it adjacent to c_i for all $i \in [m]$.

So far we have constructed the underlying graph of the instance I' of BIS. In Figure 4.4, the set of blue vertices is V_ℓ and is controlled by the leader, while the set of red vertices is V_f and is controlled by the follower. Let $M, R \in \mathbb{R}_+$ such that $R := 3|C| + n_2 + 1$ and $M := m \cdot R + 1$. Finally, we conclude our construction by defining the weight functions for the leader and the follower as follows:

$$(w_\ell(v), w_f(v)) = \begin{cases} (M, 0) & \text{if } v \in A, \\ (1, M) & \text{if } v \in B, \\ (1, 10) & \text{if } v \in \{c_i^1, c_i^2, c_i^3\} \text{ for all } i \in [m], \\ (R, 5) & \text{if } v = c_i \text{ for all } i \in [m], \\ (0, 1) & \text{if } v = z. \end{cases}$$

Note that the constructed graph is a simple undirected graph, and the construction itself can be done in polynomial time. Based on the above construction, we prove the following lemma.

Lemma 4.9. *In the* BIS *instance, when the follower has a sum type objective function,* d_s*, the leader achieves an objective value* $\geq M \cdot n_1 + R$ *(when considered* c_s*) and value* ≥ 1 *(when considered* c_b*), if and only if, in the* B_2^{CNF} *instance, there exists a truth assignment for the variables in* X *such that the formula* φ *is not satisfied for any truth assignment of the variables in* Y*.*

Proof. First, for the follower's optimal reaction F, observe that $z \notin F$ if and only if $c_i \in F$ for some $i \in [m]$. Now, we argue that, for the leader's objective function c_s (or c_b), her objective value is $\geq M \cdot n_1 + R$ (or ≥ 1) if and only if F does not contain the vertex z.

When the leader has a sum type objective function, c_s, her optimal action L always contains at least one vertex from each set of pairs $\{a_i, \bar{a}_i\}$ because of their associated weight M, and at most one vertex because a_i and $\bar{a}_i$ are adjacent to each other. There are n_1 many such pairs $\{a_i, \bar{a}_i\}$. Thus, the leader chooses n_1 many blue vertices in her action. Moreover, $z \notin F$ only when $c_i \in F$ for some $i \in [m]$. Since each vertex c_i carries weight R for the leader, clearly the leader's objective value, for objective function c_s, is assured to be at least $M \cdot n_1 + R$ if and only if the follower with a sum type objective function does not add vertex z to F, because it only happens if he adds at least one of the c_i's in F. When the leader has objective function c_b, irrespective of her chosen action L, her objective value is 0 if $z \in F$, and 1 otherwise.

In our construction, the vertices c_i^1, c_i^2, c_i^3, c_i form a clique, so the follower, due to his sum type function, always chooses exactly one vertex from set $\{c_i^1, c_i^2, c_i^3, c_i\}$ for each $i \in [m]$. Due to his weight function w_f, he always prefers either of c_i^1, c_i^2, or c_i^3 over c_i. As a result, vertex c_i belongs to F if and only if none of c_i^1, c_i^2, or c_i^3 can be included in F. It is crucial to note that the follower can add c_i^j to F only if its neighbor in $A \cup B$ is not in $L \cup F$. Also recall that, according to our construction, c_i^j has a neighbor in $A \cup B$ which is the negation of its corresponding literal l_i^j.

Finally, we claim that there exists a leader's action L for instance I' of BIS wherein, for all feasible reactions F of the follower, F cannot contain any of the c_i^1, c_i^2, c_i^3 for some i, if and only if for the instance I of B_2^{CNF}, there exists a truth assignment of X variables which renders the formula φ unsatisfied for any truth assignment of the Y variables.

($\Rightarrow$) In the instance I, let $f\colon X \to \{0,1\}$ be a truth assignment such that there is no truth assignment for Y which satisfies formula φ. Then consider $L = \{a_i \mid f(x_i) = 1\} \cup \{\bar{a}_i \mid f(x_i) = 0\}$. For this action L of the leader, we claim that there exists at least one $i \in [m]$ such that the follower cannot add any of c_i^1, c_i^2, c_i^3 into F. Assume otherwise, and there exists an optimal reaction F' of the follower such that for all $i \in [m]$ at least one of c_i^1, c_i^2, c_i^3 is in F'. Note that F' also contains exactly one vertex from each pair $\{b_i, \bar{b}_i\}$, since the follower's objective function is of sum type. Then consider the truth

assignment $f' \colon Y \to \{0,1\}$ where we set $f'(y_i) = 1$ if and only if $\bar{b}_i \in F'$. Note that for each set $\{c_i^1, c_i^2, c_i^3\}$, at least one of their neighbors in $A \cup B$ is not in $L \cup F'$. Since each vertex corresponding to a literal l_i^j is adjacent to a vertex corresponding to its negative counterpart in $A \cup B$, this implies the truth assignment f of the variables in X, and the truth assignment f' of the variables in Y satisfy the formula φ. We get a contradiction to the assumption.

($\Leftarrow$) Consider the leader's action L which results in an optimal objective value $\geq M \cdot n_1 + R$ (when considered c_s) and ≥ 1 (when considered c_b). Then consider the truth assignment $f_X \colon X \to \{0,1\}$ where $f_X(x_i) = 1$ if and only if $\bar{a}_i \in L$. We claim that this is a feasible solution of the B_2^{CNF} instance I, i.e., f_X cannot be extended to a satisfying assignment. Assume otherwise, and let f_Y be the truth assignment to the variables in Y such that f_X and f_Y satisfy φ. Now consider the follower's partial reaction P_F which contains the set $\{b_i \mid f_Y(y_i) = 0\} \cup \{\bar{b}_i \mid f_Y(y_i) = 1\}$ in instance I'. We claim that the partial reaction P_F of the follower can be extended to a complete reaction F such that $L \cup F$ returns an optimal leader's value $< M \cdot n_1 + R$ (when considered c_s) and < 1 (when considered c_b). This is a contradiction.

Note that, if each clause C_i is satisfied with assignment f_X, f_Y, at least one of the literals corresponding to c_i^1, c_i^2, or c_i^3 has truth value 1. Let $c_i' \in \{c_i^1, c_i^2, c_i^3\}$ be a vertex whose corresponding literal evaluates to 1. Thus, according to the construction of assignments f_X and f_Y, the neighbor of c_i' in $A \cup B$ does not belong to $L \cup P_F$. As $N(c_i') \cap (A \cup B) \notin L \cup P_F$, the set $\{c_i'\} \cup (L \cup P_F)$ is also an independent set. Extrapolating this to every clause, the follower will be able to add at least one of the vertices c_i^1, c_i^2, and c_i^3 to his partial solution P_F for all $i \in [m]$. Note that all these vertices are pairwise non-adjacent. So altogether they form an independent set. Thus none of $\{c_1, c_2, \ldots, c_m\}$ can be included into the follower's reaction anymore, as for each $i \in [m]$ at least one of c_i's neighbors from $\{c_i^1, c_i^2, c_i^3\}$ is already in the follower's reaction. As a result, the follower will include the vertex z into his reaction. Thus, the leader's objective function will evaluate to $< M \cdot n_1 + R$ (when considered c_s) and < 1 (when considered c_b). □

Theorem 4.10. *The decision version of the* Bilevel Independent Set *problem is Σ_2^{p}-complete when the leader has bottleneck or sum objective function (c_b/c_s), the follower has sum objective function (d_s), and the follower behaves in an optimistic or pessimistic way (o/p).*

Proof. Lemma 4.9 establishes the correctness of our reduction when considering the leader's objective function $c \in \{c_s, c_b\}$ and the follower's objective function d_s. Consequently, it proves that BIS is computationally at least as hard as the problem B_2^{CNF} over those cases. Moreover, note that, these results hold irrespective of the behavior of the follower. Thus, the results hold for both the optimistic and the pessimistic setting of the follower. As a consequence, we showed that BIS is Σ_2^{p}-hard when considering the leader's objective function to be c_s or c_b, the follower's objective function to be d_s, and the follower's behavior to be o or p.

We now show that, the problem belongs to the class Σ_2^{p} of the polynomial hierarchy by showing that a given certificate (L, F) consisting of a leader's action and a follower's reaction can be verified in polynomial time if one has access to an NP oracle. For a given certificate (L, F), we need to verify whether F is indeed an optimal reaction of the follower to the given leader's action L. For that one can check whether $L \cup F$ is indeed an independent set. Additionally, we also need to check whether $\mathrm{d}_s(L \cup F) \geq \mathrm{d}_s(L \cup F')$ for all $F' \subseteq V_f$ such that $L \cup F'$ is an independent set. For the follower with a sum type objective function, the items controlled by the leader are irrelevant in the objective function. Thus, we can delete all vertices in L and all their neighbors from the graph. Now, using the NP oracle, calculate the maximum weighted independent set $F^* \subseteq V_f$ in the remaining graph. Finally, verify if $\mathrm{d}_s(F) = \mathrm{d}_s(F^*)$. If yes, this implies, $\mathrm{d}_s(F) \geq \mathrm{d}_s(F')$ for all $F' \subseteq V_f$ such that $L \cup F'$ is an independent set. We additionally need to make sure that the final objective value of the leader is at least k; this is achieved by checking whether $\mathrm{c}(L \cup F) \geq k$. Moreover, the optimistic or the pessimistic behavior of the follower can be captured by updating the follower's weight function such that it also captures the essence of the leader's weight function, i.e., by assuming $w_f(v) \leftarrow w_f(v) + \varepsilon \cdot w_\ell(v)$ in the optimistic setting and $w_f(v) \leftarrow w_f(v) - \varepsilon \cdot w_\ell(v)$ in the pessimistic setting. Thus the above formulation evaluates to true if and only if, for a given instance of BIS, there exists a leader's action L which can be extended by a follower's optimal reaction F such that $\mathrm{c}(L \cup F) \geq k$. □

Moving forward, we now consider the cases where both the leader and the follower have bottleneck objective functions, i.e., we consider variants $(\mathrm{c}_b, \mathrm{d}_b, \mathsf{o}/\mathsf{p})$. The respective problem formulations are given below:

$$\max_{L \subseteq V_\ell} \quad \min\{w_\ell(v) \mid v \in L \cup F\} \tag{4.10a}$$

where $F \subseteq V_f$ solves the follower's problem

$$\max_{F \subseteq V_f}^{*} \quad \min\{w_f(v) \mid v \in L \cup F\} \tag{4.10b}$$

such that $L \cup F$ is an independent set and $L \cup F \neq \varnothing$

where $* \in \{\mathsf{o}, \mathsf{p}\}$, and it encodes the optimistic or pessimistic behavior of the follower.

Theorem 4.11. *The* Bilevel Independent Set *problem is polynomial-time solvable when both the leader and the follower have bottleneck objective functions and the follower behaves optimistically, i.e., when considering* $(\mathrm{c}_b, \mathrm{d}_b, \mathsf{o})$.

Proof. Since the follower behaves optimistically, given a leader's non-empty action $L \subseteq V_\ell, L \neq \varnothing$, the follower can be assumed to react by selecting an empty set. This is because the follower does not improve his objective function d_b by any other choice of set F but

might only reduce the leader's objective value by doing so. Thus $F_{L\neq\varnothing} = \varnothing$, where $F_{L\neq\varnothing}$ denotes the optimal reaction of the follower for the leader's action $L \neq \varnothing$. However, when $L = \varnothing$, the follower reacts by choosing a single element $v \in V_f$ which optimizes the following objective function:

$$\max_{v\in F^\star} \quad w_\ell(v)$$
$$\text{where } F^\star = \arg\max_{v\in V_f} \quad w_f(v)$$

In other words, the follower reaction $F_{L=\varnothing}$ contains a vertex which maximizes his own objective value, and when there are multiple optimal reactions, he chooses one which additionally maximizes $w_\ell(v)$.

For the case $(\mathsf{c}_b, \mathsf{d}_b, \mathsf{o})$, if $L \neq \varnothing$, then it is safe to assume $L \subseteq \left\{v' \mid v' \in \arg\max_{v\in V_\ell} w_\ell(v)\right\}$ because the follower's optimal reaction is $F_{L\neq\varnothing} = \varnothing$, and thus $\max_{v\in V_\ell} w_\ell(v)$ is the best objective value that the leader can achieve.

In consideration of the above cases, the leader strategically chooses either a vertex from $\arg\max_{v\in V_\ell} w_\ell(v)$ if $w_\ell(F_{L=\varnothing}) \leq \max_{v\in V_\ell} w_\ell(v)$, or $L = \varnothing$ if $w_\ell(F_{L=\varnothing}) > \max_{v\in V_\ell} w_\ell(v)$.

Calculating both $\arg\max_{v\in V_\ell} w_\ell(v)$ and $F_{L=\varnothing}$ can be done in polynomial time. Thus, the $(\mathsf{c}_b, \mathsf{d}_b, \mathsf{o})$ variant of BIS is tractable. □

Theorem 4.12. *The decision version of the* Bilevel Independent Set *problem is* NP*-complete when both leader and follower have bottleneck objective functions and the follower behaves pessimistically, i.e., when considering* $(\mathsf{c}_b, \mathsf{d}_b, \mathsf{p})$.

Proof. In this case, given a leader's action $L \subseteq V_\ell$, the follower's optimal reaction is as follows:

- When $L = \varnothing$, the follower's reaction is

$$F_L = \min_{v\in F^\star} w_\ell(v), \qquad \text{where } F^\star = \arg\max_{v\in V_f} w_f(v)$$

  The follower picks—among all maximum weighted vertices according to the weight function w_f—one which has the minimum weight according to the weight function w_ℓ.

- When $L \neq \varnothing$, the follower reacts by choosing

$$F_L = \arg\min_{\substack{v\in V_f\setminus N(L)\\ w_f(v)\geq \mathsf{d}_b(L)}} w_\ell(v),$$

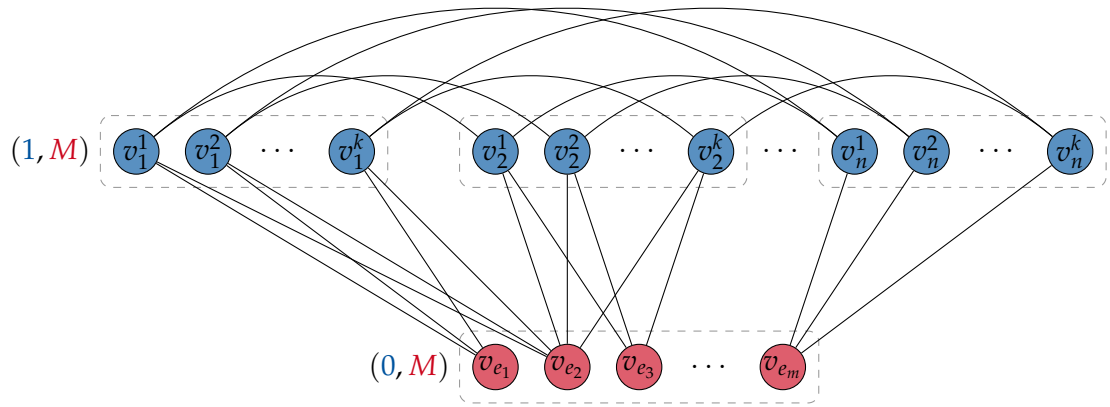


Figure 4.5: NP-hardness reduction for the Bilevel Independent Set problem when considering (c_b, d_b, p). The blue vertices form the set V_ℓ whereas the red vertices form the set V_f.

if no such $v \in V_f \setminus N(L)$ exists, then $F_L = \varnothing$ A pessimistic follower always tries to sabotage the objective function of the leader as long as he does not reduce his own objective value in the process. Given a leader's action $L \subseteq V_\ell$ as a certificate, the optimal reaction of the follower F_L can be calculated in polynomial time, and one can verify the leader's final objective value $c_b(L \cup F_L)$. Thus the problem belongs to the class NP.

In order to show hardness, we give a reduction from the Vertex Cover problem to the decision version of BIS. In Vertex Cover, given an undirected graph $G = (V, E)$ and a positive integer k, we are asked to find a subset $S \subseteq V$ of at most k vertices such that the removal of this set renders the graph edgeless, i.e., $G[V \setminus S]$ is an empty graph.

Let $I = (G = (V, E), k)$ be an instance of the Vertex Cover problem. Then we construct a new instance $I' = (G' = (V_\ell \dot\cup V_f, E'), k', w_\ell, w_f)$ of the decision version of BIS where both leader and follower have bottleneck type objective functions and the follower behaves pessimistically. Here, k' is the threshold in the decision version of BIS. We give the construction as follows (also see Figure 4.5):

- For each $v \in V$, add k copies of v to V_ℓ; let $v^1, v^2, \ldots, v^k \in V_\ell$ for $i \in [k]$. Define $V^i = \{v^i \mid v \in V\}$ for all $i \in [k]$.
- For each $e \in E$, add a vertex $v_e \in V_f$.
- If an edge e is incident to v, add edges $\{v^i, v_e\} \in E'$ for all $i \in [k]$.
- Make each $G[V^i]$ a clique, so add edges between each pair of vertices in V^i.
- Let $k' = 1$.

Finally we conclude our construction by defining the following weight functions:

$$w_\ell(v) = \begin{cases} 1 & \text{if } v \in V_\ell, \\ 0 & \text{if } v \in V_f, \end{cases} \qquad w_f(v) = \begin{cases} M & \text{if } v \in V_\ell, \\ M & \text{if } v \in V_f, \end{cases}$$

where $M \gg 0$.

Correctness: Now, we show that I is an yes-instance of the Vertex Cover problem if and only if I' is a yes-instance of BIS when considered (c_b, d_b, p).

($\Rightarrow$) In the instance I of the Vertex Cover problem, assume that there is a subset $S \subseteq V$ with $|S| \leq k$ such that $G[V \setminus S]$ does not contain any edges. Then we claim that there is a leader's action $L \subseteq V_\ell$ in I' which yields an objective value 1 for the leader. From each set V^i, $i \in [k]$, we pick one vertex corresponding to a vertex in S. Let the resultant set be the leader's action L. We have at most k vertices in S, so it can be guaranteed that L is an independent set. Moreover, there are at most k vertices in S and we have k many different sets V^i each for $i \in [k]$; we can cover one copy of each vertex in S from the sets V^i. Given this choice L of the leader, the follower cannot pick any vertex in the solution because $V_f \setminus N(L) = \varnothing$, as every edge $e \in E$ was covered by a vertex in S. Since $F_L = \varnothing$, the leader's objective function is simply $c_b(L \cup F_L) = c_b(L) = 1$.

($\Leftarrow$) Now, if there exists a leader's action $L \subseteq V_\ell$ for which she achieves the objective value at least 1, then it is clear that $V_f \subseteq N(L)$, since the follower was not able to sabotage the leader's objective function by picking a vertex from set V_f. Moreover, since L is an independent set, $|L| \leq k$ because the leader can choose at most one vertex from each of the sets V^i, $i \in [k]$. Consider the corresponding vertices in $V(G)$ whose copies were chosen in L. Let us denote this set by S; $|S| \leq k$. Note that S covers all edges in $E(G)$, since $V_f \subseteq N(L)$. Thus, S is a vertex cover of G. □

Later, we will modify the construction used in the above theorem to show further NP-hardness on bipartite graphs (see Theorem 4.20).

For the study on simple undirected graphs, we still remain to address the cases when the leader's objective function is of sum type (c_s), the follower's objective function is of bottleneck type (d_b), and the follower behaves either optimistically or pessimistically.

We first give a unified reduction which shows NP-hardness for both variants (c_s, d_b, o) and (c_s, d_b, p). Then we argue the NP containment of the decision versions of both variants (c_s, d_b, p) and (c_s, d_b, o).

Theorem 4.13. *The decision version of the* Bilevel Independent Set *problem is* NP*-hard when the leader's objective function is of sum type, the follower's objective function is of bottleneck type, and the follower behaves optimistically or pessimistically, i.e., when considering* (c_s, d_b, o) *and* (c_s, d_b, p).

Proof. We give a naive reduction from the Independent Set problem which works for both the variants. It is well-known that Independent Set is NP-complete [Kar09]. Con-

sider the underlying graph $G = (V, E)$ of an instance I of Independent Set. Take a replica of this graph to construct an instance I' of BIS. Let the vertex sets controlled by the leader and the follower be $V_\ell = V$ and $V_f = \varnothing$, respectively. Moreover, consider the weight functions of the leader and the follower to be $w_\ell, w_f \colon V \to \{1\}$.

Note that the follower always has an empty reaction irrespective of his behavior because $V_f = \varnothing$ implies he does not have a choice. So the leader does not have to care about the follower's reaction. Hence, for the variant (c_s, d_b, p) and (c_s, d_b, o), the BIS problem boils down to solving for the maximum weighted independent set in the graph according to function w_ℓ. This is equivalent to solving the original instance I of the Independent Set problem. Thus our reduction is sound. The variants (c_s, d_b, o) and (c_s, d_b, p) of BIS are NP-hard. □

Next we show that the decision version of the (c_s, d_b, p) variant of BIS is in NP and thus this variant is NP-complete as a consequence of Theorem 4.13.

Theorem 4.14. *The decision version of the* Bilevel Independent Set *problem is contained in the class* NP *when the leader's objective function is of sum type, the follower's objective function is of bottleneck type, and the follower behaves pessimistically, i.e., when considering* (c_s, d_b, p). *Additionally,* BIS *is* NP*-complete for the variant* (c_s, d_b, p).

Proof. First, we show the containment in the class NP. The certificate is just the leader's solution L. If $L = \varnothing$, then a pessimistic follower selects a vertex that is best for him and worst for the leader which can be computed using:

$$\min_{v \in F^\star} \; w_\ell(v) \qquad \text{where } F^\star = \arg\max_{v \in V_f} \; w_f(v).$$

If $L \neq \varnothing$, then any additional vertex by the follower can neither improve the follower's objective value nor make the leader's value worse, hence the follower selects nothing. Hence, for any given certificate $L \subseteq V_\ell$, one can calculate the follower's optimal reaction, and consequently verify the leader's final objective value. Thus, the problem is in NP.

Taking the NP-hardness shown in Theorem 4.13 into account, we conclude that this variant is NP-complete. □

Lemma 4.15. *The decision version of the* Bilevel Independent Set *problem is contained in the class* NP *when the leader's objective function is of sum type, the follower's objective function is of bottleneck type, and the follower behaves in an optimistic way, i.e., when considering* (c_s, d_b, o). *Additionally,* BIS *is* NP*-complete for the variant* (c_s, d_b, o).

Proof. To show containment in NP, we show that there is a valid polynomial size certificate which can be verified in polynomial time. Let the certificate be made up of both

the leader's and the follower's chosen set of variables $L \cup F$. To check whether $L \cup F$ is a valid solution of BIS, we need to check if $L \cup F$ forms an independent set. However, along with that, we also need to prove that the follower's optimal reaction F_L to the leader's action L evaluates the objective function of the leader to a value that is at most as much as the solution $L \cup F$ does. Once these conditions are verified, it is easy to calculate the objective value of the leader, so calculate $\sum_{v \in L \cup F} w_\ell(v)$ to verify whether the given instance is a yes-instance of variant (c_s, d_b, o) of BIS.

Verifying if $L \cup F$ is an independent set is trivial in polynomial time. Consider the induced subgraph over the vertex set $L \cup F$ and check whether there is an edge in it. If $L \neq \varnothing$, then the follower's objective value cannot be greater than $d_b(L)$. In this case, the follower's objective value can never improve by choosing more vertices, it is determined entirely by the leader's solution. Thus, in addition to $L \cup F$ being an independent set, if $d_b(L \cup F) = d_b(L)$ for $L \neq \varnothing$, then F belongs to the set of the follower's feasible reactions to L. Now, for the part when $L = \varnothing$, the follower's reaction F needs to satisfy the following conditions:

(1) $L \cup F$ is an independent set in the graph, which is equivalent to F being an independent set.

(2) $F \subseteq \arg\max\{w_f(v) \mid v \in V_f\}$ because the follower wants to maximize the minimum weighted vertex in the solution, thus he would only chose from the set of vertices which gives him the highest weight possible.

It is easy to check whether conditions (1) and (2) hold. Moreover, since the follower behaves optimistically towards the leader, it suffices to have one witness (for example, the set F here) to show that, the leader's objective value evaluates to at least $\sum_{v \in L \cup F} w_\ell(v)$. This is because the optimistic follower would help maximize the objective function of the leader. Thus, he will always choose a set $F \subseteq \arg\max\{w_f(v) \mid v \in V_f\}$ which works the best for the leader.

As a result, the variant (c_s, d_b, o) of BIS belongs to the class NP. □

4.5.2 Complexity on Bipartite Graphs

In this section, we study Bilevel Independent Set on bipartite graphs. The maximum weight Independent Set problem is polynomial-time solvable on the class of bipartite graphs [Edm65]. However, when considering bilevel variants of this problem, we get a variety of results on different variants. Please refer to Section 4.2.3 for an overview of our results given in Table 4.2.

We first consider the variant $(c_s, d_s, o/p)$. Moreover, we would like to point out the drastic change in the computational complexity for these variants: BIS is Σ_2^p-complete on general graphs, whereas here we prove that it is NP-complete on bipartite graphs.

4.5.2a Both decision makers have sum type objective functions

This subsection is devoted to studying BIS on bipartite graphs when both the leader and the follower have sum type objective functions. We additionally show our results on planar bipartite graphs. Note that a hardness result on planar bipartite graphs also proves hardness on the class of bipartite graphs. We also give a unified reduction for the variants where $c = c_s$, $d = d_s$, and the follower behaves optimistically or pessimistically. We show that, for both of these cases, the problem is NP-complete.

The reduction is from the planar vertex cover problem which has been shown to be NP-hard [Lic82]. Given a planar graph and a positive integer k, the planar vertex cover problem asks if there is a vertex cover of size at most k.

Let $I = (G = (V, E), k)$ be an instance of the planar vertex cover problem. Let $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{e_1, e_2, \ldots, e_m\}$. Now, we construct an instance $I' = (G' = (V_\ell \cup V_f, E'), k', w_\ell, w_f)$ of the decision version of BIS as follows (also see Figure 4.6):

- For each $v_i \in V$, add vertices a_i, a_i' in V_ℓ. Define $A := \{a_1, \ldots, a_n\}$ and $A' := \{a_1', \ldots, a_n'\}$.
- For each $e_i \in E$, add vertices b_i, b_i' in V_f. Define $B := \{b_1, \ldots, b_m\}$ and $B' := \{b_1', \ldots, b_m'\}$.
- For each $i \in [n]$, add an edge $\{a_i, a_i'\}$ to E'. For each $j \in [m]$, add an edge $\{b_j, b_j'\}$ to E'.
- Add an edge $\{a_i, b_j\}$ to E' if and only if edge e_j is incident to vertex v_i in E.

As Figure 4.6 shows, the reduced instance is clearly a bipartite graph, however, note that, the coloring depicted in the figure is not according to the bipartition. Now, we argue that the constructed instance is additionally a planar graph. This holds because we can use a planar embedding of the original graph as a backbone to obtain a planar embedding of the constructed graph: Consider a planar embedding of graph G, replace each v_i by a_i, delete each edge e_j and instead add a new vertex b_j such that b_j is embedded at the midpoint of edge e_j. Now add all edges $\{a_i, b_j\}$; due to the planar embedding of G the resulting graph is still planar. The additional vertices a_i' and b_j' along with the edges $\{a_i, a_i'\}$ and $\{b_j, b_j'\}$ can be appended to vertices a_i and b_j, respectively, while preserving the current planarity of the graph.

Finally we conclude our construction by defining the following weight functions:

$$w_\ell(v) = \begin{cases} 0 & \text{if } v \in A \cup B, \\ 1 & \text{if } v \in A', \\ M & \text{if } v \in B', \end{cases} \qquad w_f(v) = \begin{cases} 0 & \text{if } v \in A \cup A', \\ 100 & \text{if } v \in B, \\ 1 & \text{if } v \in B', \end{cases}$$

where $M \in \mathbb{N}, 100 \ll M$ is a sufficiently large number.

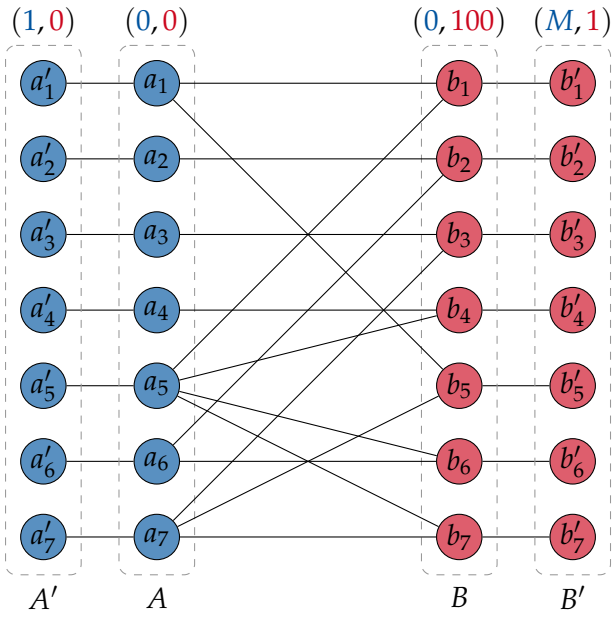


Figure 4.6: NP-hardness reduction for the Bilevel Independent Set problem on planar bipartite graphs when considering $(c_s, d_s, o/p)$.

Now it remains to argue that, for all the considered cases, I is a yes-instance of the planar vertex cover problem if and only if I' is a yes-instance of BIS.

To show the correctness of this construction, we prove the following lemma:

Lemma 4.16. *In the* BIS *instance, the leader achieves value at least* $k' := (mM + n - k)$ *if and only if the initial instance of the planar vertex cover problem has a vertex cover of size at most* k.

Proof. ($\Leftarrow$) Let $S \subseteq V$ be a vertex cover of graph $G = (V, E)$ of size at most k. Then the leader's action $L = \{a_i \in A \mid v_i \in S\} \cup \{a'_i \in A' \mid v_i \notin S\}$ results in an objective value of $\geq (mM + n - k)$ for her: Clearly, L is an independent set. Note that, for each vertex $b_i \in B$, b_i has a neighbor in L because, in the planar vertex cover instance, the set S covers all the edges of the graph. Thus, given L, the follower cannot pick a vertex from the set B in his reaction F_L. Moreover, the optimal follower's reaction is $F_L = B'$, independent of whether he is optimistic or pessimistic. Then $c_s(L \cup F_L) = mM + |\{a'_i \in A' \mid v_i \notin S\}| \geq mM + n - k$. This holds for both the optimistic and pessimistic settings, since the follower's reaction is unique.

($\Rightarrow$) Consider that BIS contains a leader's action L which results in an objective value at least $mM + n - k$. This can be achieved if and only if the follower chooses $F \supseteq B'$. Also, note that, if both b_i and b'_i are available then the follower will always pick b_i over b'_i because it gives him a better objective value. Thus, for each $b_i \in B$, L must contain a vertex which is adjacent to b_i. Let $L_A = L \cap A$ be such that $B \subseteq N(L_A)$. Note that the

corresponding vertex set $S' = \{v_i \in V \mid a_i \in L_A\}$ is a vertex cover of the original instance I of the planar vertex cover problem. Now, it only remains to prove that $|L_A| \leq k$. Suppose the contrary, $|L_A| > k$, then $|L \cap A'| < n - k$. Thus, $c_s(L \cup F_L) < (mM + n - k)$, a contradiction. Hence, $|L_A| \leq k$. □

Theorem 4.17. *Let $G = (V, E)$ be a planar bipartite graph where $V = V_\ell \dot\cup V_f$, along with two weight functions $w_\ell, w_f \colon V \to \mathbb{R}_+$ such that V_ℓ, w_ℓ correspond to the leader and V_f, w_f correspond to the follower. The decision version of the* Bilevel Independent Set *problem is* NP*-complete when the leader has a sum objective function (c_s), the follower has a sum objective function (d_s), and the follower behaves in an optimistic or pessimistic way (o/p), i.e., when considering $(c_s, d_s, o/p)$.*

Proof. For both variants of the problem considered above, the problem lies in NP because a valid certificate is the leader's solution. The follower's solution is just a weighted independent set problem on the remaining graph which is computable in polynomial time as the graph is bipartite. This consequently shows Bilevel Independent Set on planar bipartite graphs is also in NP. The optimistic or pessimistic behavior of follower can be captured by using the updated weight function of the follower, i.e., by assuming $w_f(v) \leftarrow w_f(v) + \varepsilon \cdot w_\ell(v)$ in the optimistic setting and $w_f(v) \leftarrow w_f(v) - \varepsilon \cdot w_\ell(v)$ in the pessimistic setting.

Moreover the correctness of the reduction follows from Lemma 4.16. Thus the cases $(c_s, d_s, o/p)$ are NP-complete. □

4.5.2b Leader with sum type and follower with bottleneck type objective function

In this section, we restrict to the variants (c_s, d_b, o) and (c_s, d_b, p) on bipartite graphs. We prove that BIS is polynomial-time solvable for both of these variants.

Theorem 4.18. *For a bipartite graph $G = (V, E)$,* BIS *is solvable in polynomial time when considering the leader's objective function of sum type, the follower's objective function of bottleneck type and the follower behaves in an optimistic way, i.e., case (c_s, d_b, o).*

Proof. For $L = \varnothing$, $F_{L=\varnothing}$ consists of the max-weighted independent set with respect to the leader's weight function w_ℓ, on the graph $G[F^\star]$, where $F^\star = \arg\max_{v \in V_f} w_f(v)$, which is computable in polynomial-time as we are dealing with bipartite graphs.

Observe that, given a non-empty leader's action $L \subseteq V_\ell$, the follower's objective value cannot be greater than $d_b(L)$. Moreover, since the follower is optimistic, he will always react with the maximum weighted independent set (with respect to the weight function w_ℓ) on the graph $G[F^\star]$, where $F^\star = \{v \in V_f \setminus N(L) \mid w_f(v) \geq d_b(L)\}$. In this case, the leader's optimal action can be calculated using Algorithm 1.

Algorithm 1: (c_s, d_b, o) on bipartite graphs

Input: A bipartite graph $G = (V, E)$, and the weight functions $w_\ell, w_f \colon V \to \mathbb{R}_+$.
Output: An optimal leader's action L_{opt} for the case (c_s, d_b, o).

```
1  Calculate the follower's reaction F_{L=∅}.
2  val := w_ℓ(F_{L=∅})
3  L_opt := ∅
4  R := V_ℓ
5  while R ≠ ∅ do
6      Select v* ∈ R                    // w_f(v*) sets the threshold for d_b(L) value
7      V' = {v' ∈ V | w_f(v') ≥ w_f(v*), v' ∉ N[v*]}
8      Calculate the maximum weighted independent set S_{V'} of G[V'] according to
         w_ℓ.
9      if w_ℓ(v*) + Σ_{v∈S_{V'}} w_ℓ(v) > val then
10         val := w_ℓ(v*) + Σ_{v∈S_{V'}} w_ℓ(v)
11         L_opt := {v*} ∪ (S_{V'} ∩ V_ℓ)
12     R := R \ {v*}
13 return L_opt.
```

Description of Algorithm 1: When $L \neq \varnothing$, the follower's objective value has a lower bound; it cannot be greater than $d_b(L)$. In fact, it suffices to consider that only one vertex from L actually enforces the bound $d_b(L)$. Hence, we enumerate over all possible vertices $v^\star \in V_\ell$ that can influence the value of the lower bound $d_b(L)$ for the follower. Once, $v^\star \in V_\ell$ is fixed, the leader considers the set of vertices $V' = \{v' \in V \mid w_f(v') \geq w_f(v^\star), v' \notin N[v^\star]\}$, and calculates the maximum weighted independent set $S_{V'}$ in the subgraph $G[V']$ according to the weight function w_ℓ. The maximum weighted independent set obtained this way, over all the vertices $v^\star$ considered, gives us a leader's optimal action.

Correctness of Algorithm 1: A leader's action $L \subseteq V_\ell$ sets a threshold for the optimistic follower's reaction. The follower chooses a reaction F_L such that $\min_{v \in F_L} w_f(v) \geq d_b(L)$. We go over all possible threshold values $d_b(L)$ that a leader's action can impose on the follower. And for each one of them, we calculate an optimal solution. Let $S \subseteq V$ be the set which returns the best possible value for the leader among all the enumerated cases. Then the leader's optimal action is $S \cap V_\ell$. This proves the correctness of our algorithm.

Moreover, there are at most polynomial many steps which individually take at most polynomial time. Thus, Algorithm 1 runs in polynomial time. □

Theorem 4.19. *Given a bipartite graph $G = (V, E)$,* Bilevel Independent Set *is solvable in polynomial time when considering the leader's objective function of sum type, the follower's objective function of bottleneck type, and the follower behaves in a pessimistic way, i.e., case (c_s, d_b, p).*

Proof. If $L = \varnothing$, then the follower's optimal reaction is to chose from the vertices which maximizes his objective value the one with the least w_ℓ value. Otherwise, if $L \neq \varnothing$, the pessimistic follower will not choose any vertex as it will only increase the leader's objective value and not his own. Thus, the leader accordingly chooses either $L = \varnothing$ or the maximum weighted independent set, with respect to w_ℓ, in V_ℓ. The latter can be calculated in polynomial time [Edm65]. □

The results obtained in Section 4.5.2a and Section 4.5.2b demonstrate a decrease in the computational complexity of BIS on bipartite graphs compared to the results on general graphs. Naturally, this is because the underlying Independent Set problem is easier on bipartite graphs than on general graphs. However, when considering the variant (c_b, d_b, p) on bipartite graphs in the next section, we retain that the problem is NP-complete on bipartite graphs similar to Theorem 4.12 on general graphs. We show that, with slight changes, the same construction works for proving NP-hardness for the variants $(c_b, d_s, o/p)$.

4.5.2c The remaining variants on bipartite graphs

Next we consider the cases (c_b, d_b, p) and $(c_b, d_s, o/p)$ on bipartite graphs, and prove that BIS is NP-complete for these variants. We modify the NP-hardness construction used in Theorem 4.12 such that the reduced instances map to bipartite graphs. The remaining case (c_b, d_b, o) is polynomial time solvable even on general graphs (see Theorem 4.11); this result translates on the subclass of bipartite graphs.

Theorem 4.20. *The decision version of the* Bilevel Independent Set *problem is* NP*-complete on bipartite graphs when considering any of the variants* (c_b, d_b, p) *and* $(c_b, d_s, o/p)$.

Proof. The NP-hardness proof in Theorem 4.12 is for the variant (c_b, d_b, p) on general graphs wherein the constructed graph contains odd cycles. To show NP-hardness on bipartite graphs, we only need to get rid of these odd cycles and construct equivalent bipartite instances for each instance created in Theorem 4.12. For this purpose, we introduce for each vertex $v^i \in V^i$, $i \in [k]$, an extra vertex $(v^i)'$. We include the set of all these vertices in V_ℓ and assign weight 1 for the leader and M for the follower. Additionally, for each edge $\{u^i, v^i\}$, for all $i \in [k]$ we add a vertex $(u^i v^i)'$. We include the set of all these vertices in V_f and assign weight 0 for the leader and M for the follower. Next, we replace each edge $\{u^i, v^i\}$ from the previous construction by edges $\{u^i, (u^i)'\}, \{v^i, (v^i)'\}, \{(u^i)', (u^i v^i)'\}$ and $\{(v^i)', (u^i v^i)'\}$ (see Figure 4.7).

Note that the resultant graph is bipartite with two parts being $\{v_e \mid e \in E(G)\} \cup \{(v^i)' \mid v \in V(G), i \in [k]\}$ and $\{v^i \mid v \in V(G), i \in [k]\} \cup \{(u^i v^i)' \mid u, v \in V(G), i \in [k]\}$.

We further prove that the leader achieves objective value at least 1 if and only if the original instance of the Vertex Cover problem has a vertex cover of size at most k. Note that the leader achieves objective value 0 if and only if the follower's reaction satisfies

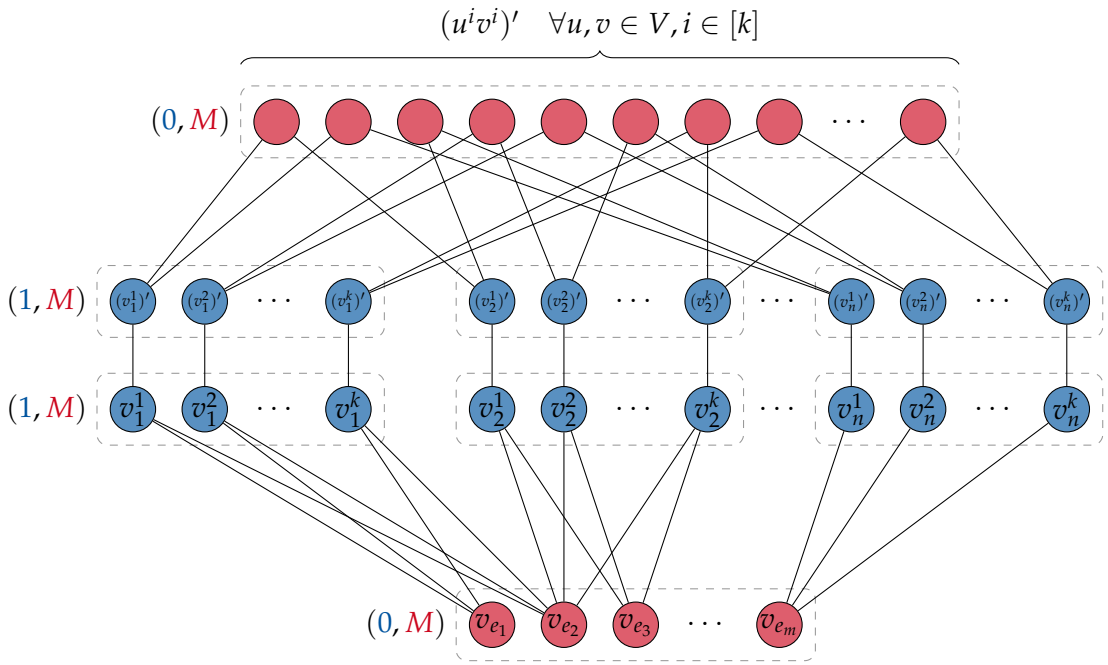


Figure 4.7: NP-hardness reduction for the BIS on bipartite graphs when considering (c_b, d_b, p) or (c_b, d_b, o). The blue vertices form the set V_ℓ whereas the red vertices form the set V_f.

$F \neq \varnothing$. In order to make sure that $F = \varnothing$, we claim that the leader has to choose at most one vertex from each V^i. Consider the contrary. If two vertices u^i and v^i in V^i belong to the leader's (optimal) action, then the leader cannot pick either of $(u^i)'$ and $(v^i)'$. Then a pessimistic follower would certainly pick $(u^iv^i)'$ in his solution set F. Moreover, an optimistic or pessimistic follower with sum type objective function would also pick the available vertex $(u^iv^i)'$ into his solution, as this will increase his objective value by M. Thus, this construction works for the variants (c_b, d_b, p) and $(c_b, d_s, o/p)$ on bipartite graphs.

Let S be a vertex cover of graph G of size at most k. Then L which includes, for each $i \in [k]$, a copy v^i of a distinct vertex $v \in S$ and all other $(v^i)'$ copies whose neighbors have not been chosen earlier, is an optimal leader's action resulting in objective value 1 for the leader.

Similarly, if the constructed instance of BIS is a yes-instance, then there exists a leader's action L, which results in her objective value being 1; here the cardinality $|L \cap \{v^i \mid v \in V(G), i \in [k]\}| \leq k$. Construct a set of the corresponding vertices in G. This set is a vertex cover of G of size at most k, since for each vertex v_e, there is at least one neighbor in $L \cap \{v^i \mid v \in V(G), i \in [k]\}$.

Thus, the above construction proves NP-hardness on bipartite graphs for the cases (c_b, d_b, p) and $(c_b, d_s, o/p)$.

Moreover, the variant (c_b, d_b, p) belongs to the class NP, as argued in Theorem 4.12.

To show the NP containment of the variant (c_b, d_s, o), we claim that $L \cup F \subseteq V_\ell \cup V_f$ serves as a valid certificate. First calculate, for the leader's action L, a maximum weighted independent set I—with respect to weight function w_f—in $G[S]$ where $S = \{v \in V_f \mid v \notin N(L)\}$; it is possible to do this in polynomial time as we are dealing with bipartite graphs. A follower's feasible reaction F to L must satisfy $\sum_{v \in I} w_f(v) = \sum_{v \in F} w_f(v)$. If this holds, further check that $L \cup F$ is an independent set and calculate the leader's objective value $\arg\min_{v \in L \cup F} w_\ell(v)$. Thus, it is possible to verify a yes-instance of the decision version of (c_b, d_s, o) in polynomial time. Hence, the (c_b, d_s, o) variant of BIS is NP-complete on bipartite graphs.

For the last variant (c_b, d_s, p), the follower behaves in a pessimistic way and has sum type objective function. So his goal is to select a maximum weighted—with respect to function w_f—independent set F from $G[S]$ that additionally minimizes the value $\min_{v \in F} w_\ell(v)$. This can be achieved by doing the following: First, consider the vertices in S in an ascending order of their w_ℓ values. Then one-by-one, check for each vertex $v \in S$, the maximum weighted independent set—with respect to function w_f—in $G[S]$ that includes the vertex v. If the follower's total weight is the same as that of set I calculated above, then this serves as the optimal reaction of the follower towards L. If not, then calculate for the next vertex v in S. This way, we can calculate the follower's optimal reaction to L in polynomial time and check for the decision bounds of the problem. Thus, the (c_b, d_s, p) variant of BIS is NP-complete on bipartite graphs. □

Recall from Theorem 4.11, Bilevel Independent Set for (c_b, d_b, o) is polynomial-time solvable on general graphs. Consequently, it is also polynomial-time solvable on bipartite graphs.

4.6 Concluding Remarks and Future Directions

Following a recent trend of studying bilevel optimization problems in which the entire variable set is partitioned into two subsets—each controlled by a distinct decision maker involved in the optimization scheme—we studied the Bilevel Assignment, Bilevel Interval Scheduling, and Bilevel Independent Set problems in the same setting. We considered eight different variants emerging from selecting either sum type (c_s, d_s) or bottleneck type (c_b, d_b) objective functions for the leader and the follower, and further considering the optimistic (o) or pessimistic (p) settings, which reflect the behavior of the follower towards the leader when faced with multiple choices for an optimal reaction.

For the Bilevel Assignment problem, we resolved the computational complexity status of the last variant (c_b, d_b, o) left open by Gassner and Klinz in the year 2009 [GK09]. On the bright side, our reduction works for all eight variants unifiedly. We remark that our reduction also implies the inapproximability of the problem: Unless

$\mathsf{P} = \mathsf{NP}$, there is no polynomial-time algorithm for the Bilevel Assignment problem that approximates the optimal objective value of the leader with any approximation guarantee.

We further studied the Bilevel Independent Set problem on the class of simple undirected graphs and also on bipartite graphs (see Table 4.1 and Table 4.2). We completely classified the complexity status of all eight variants of this problem on both of these graph classes. As anticipated, when going from simple undirected graphs to bipartite graphs, the computational complexity of most of the variants reduced by exactly one level of the polynomial hierarchy. However, interestingly, the variant $(\mathsf{c}_b, \mathsf{d}_b, \mathsf{p})$ disregarded this trend, and for this, the computational complexity of the problem remained the same.

We also studied BIS on the class of interval graphs, which we called the Bilevel Interval Scheduling problem. Here we only considered the variants $(\mathsf{c}_s, \mathsf{d}_s, \mathsf{o})$ and $(\mathsf{c}_s, \mathsf{d}_s, \mathsf{p})$, and we gave an algorithm that runs in $\mathcal{O}(n^4 \log n)$ time. It would be interesting to know what complexity results hold for the other variants.

Some of the tractability results that we have presented are evolved by exhaustively searching through all possible leader's actions and the follower's reactions to them. It would be nice to design algorithms that exploit some structure of the problem.

For the future outlook, one could deviate from the partitioned-items bilevel problems and consider the settings wherein the sets of variables controlled by the leader and the follower are not disjoint.

Other combinatorial optimization problems can be studied with respect to the eight variants of bilevel optimization problems we consider here. It would be interesting to find approximation algorithms for these problems, especially the case where the leader's and the follower's objective functions are of a sum type.

From a broader perspective, it would be interesting to study some of these "extra hard" bilevel problems within the realm of parameterized complexity or using the toolkit of approximation algorithms.

5

Recoverable Robust Optimization with Commitment

This chapter revolves around a new model that we introduced in the field of Robust Optimization, called *Recoverable Robust Optimization with Commitment*. We study this model on several combinatorial optimization problems and prove tractability and intractability results. The results mentioned in this chapter emerge from a joint work with Felix Hommelsheim, Nicole Megow, and my supervisor Britta Peis [HMMP23]. This work has been published in the Mathematical Programming journal.

5.1 The Model

Robust optimization has become a central framework in operations research which deals with uncertainty in decision-making and optimization. Unlike traditional optimization, where parameters are assumed to be known precisely, robust optimization considers *scenarios* where some parameters are uncertain but lie within a predefined uncertainty set, which may be specified implicitly. Early research in this area focused on computing static solutions that optimize the worst-case cost. For comprehensive overviews, we refer to [KY97, BEGN09, BK18]. However, it is well recognized that this approach may lead to overly conservative solutions as a single, potentially very unlikely, scenario might heavily influence the outcome.

To address this issue of conservatism in robust optimization and introduce more flexibility, various frameworks have been proposed, including two-stage adjustable robust optimization by [BGGN04], recoverable robust optimization by [LLMS09], and demand robust optimization by [DGRS05]. These frameworks share a common two-stage structure: In the first stage, a solution is determined based on the uncertainty set, while in the second stage, after the uncertain scenario has been revealed, the initial solution can be adjusted within some predefined recovery bounds. This second-stage adjustment is referred to as *recourse*.

In this work, we use the notion of *recoverable robust optimization* as introduced by Liebchen, Lübbecke, Möhring, and Stiller [LLMS09]. Initially developed in the context of timetabling, linear programming, and railway optimization, it seeks first-stage solutions that can be adapted feasibly across all scenarios within specified recovery bounds; the corresponding recovery bounds are modeled, for example, via the size of the symmetric difference of solutions. The cost is typically evaluated as the sum of the first- and second-stage costs, potentially under different cost functions. This model has been successfully applied to a variety of combinatorial optimization problems such as the shortest path problem [Büs12, JKZ24a, JKZ24b], spanning tree and more general matroid problems [HKZ17a, HKZ17b, LPT21], perfect matchings [DMP+15], scheduling problems [BG22], knapsack problems [BGKK19, BKK11a], and selection problems [LLW21] which led to insights into the computational complexity, as well as the development of exact and approximate algorithms for solving these problems.

A major drawback of recourse models, including the recoverable robustness framework, is that they do not necessarily preserve the elements of the initial solution during recourse. However, such preservation is crucial in many practical applications, particularly when commitments or promises have been made. In any reservation-based system—such as hotel bookings, rental car reservations, or airline seat reservations—the supplier guarantees services to the customers holding reservations. If a customer cancels her reservation, the supplier may modify the set of reserved customers, but must ensure that, the previously reserved customers who did not cancel remain part of the modified solution. In such scenarios, the supplier can only *augment* the existing set of reserved (and not canceled) customers with new customers, without altering the existing commitments.

This motivates us to introduce our new model *Recoverable Robust Optimization with Commitment*, where we commit to the first-stage decision, and restrict the recourse action to augmenting the partial solution which remains after the cancellation or after some other uncertain disruption has taken place. More formally, we define the problem as follows.

Recoverable Robust Optimization with Commitment. Consider a combinatorial optimization problem $\mathcal{P}$ with ground set E, a set $\mathcal{F} \subseteq 2^E$ of feasible solutions, and a weight function $w\colon E \to \mathbb{R}_+$, where the task is to find a solution $S \in \mathcal{F}$ maximizing $w(S) = \sum_{e \in S} w(e)$. In the first stage in our model, we choose a solution $S \in \mathcal{F}$ to the underlying problem $\mathcal{P}$; it is important to note that this solution need not be an optimal solution. Then, in the second stage, a set $D \subseteq E$ of *at most* k elements is deleted, followed by a step in which a set $R \subseteq E \setminus D$ of size at most ℓ elements is added to $S \setminus D$ such that the resulting set is still a solution, i.e., $S' = (S \setminus D) \cup R \in \mathcal{F}$. We call S the *first-stage* solution and S' the *second-stage* solution. The goal is to select a first-stage solution S that maximizes the weight of the second-stage solution, $w(S')$ in the worst case.

We refer to the set D that is deleted as the *interdiction* and the set R that is added as *recourse*. The interdiction can be seen as an act of an *adversary* who aims to delete a set $D \subseteq E$ that decreases the final objective value $w(S')$ as much as possible. The adversary can be assumed to have full knowledge of the set $R \subseteq E \setminus D$ that we would optimally add during the recourse phase following the deletion of D. Thus, we solve the following problem:

$$\max_{S \in \mathcal{F}} \min_{\substack{D \subseteq E \\ |D| \leq k}} \max_{\substack{R \subseteq E \setminus D,\ |R| \leq \ell \\ (S \setminus D) \cup R \in \mathcal{F}}} w((S \setminus D) \cup R)\ . \tag{k, ℓ-RP}$$

For the sake of simplicity, we update $E \leftarrow E \cup \{\varnothing\}$, i.e. the set E is updated such that $\varnothing \in E$; we define $w(\varnothing) := 0$. The inclusion of the empty element in E allows us to conveniently represent scenarios involving empty deletions or additions of the elements in E. In fact, in the later parts of this chapter, we may require multiple (at most linear in the size of the input) empty elements to realize different variables (for example in Algorithm 1, u and b both could be empty). Thus, we assume that our updated set E always contains sufficiently many empty elements at any point.

Further, throughout the chapter, we consider downward-closed set systems $\mathcal{F}$, meaning that if $X' \subseteq X$ and $X \in \mathcal{F}$, then $X' \in \mathcal{F}$. This way, we guarantee that for any solution $S \in \mathcal{F}$, its partial remaining solution $S \setminus D$ also belongs to $\mathcal{F}$. Thus in the worst case, if the recourse is not possible, then we satisfy the restriction that the second stage solution must be a feasible solution.

Additionally, for a set X and an element g, we use $X + g := X \cup \{g\}$ and $X - g := X \setminus \{g\}$.

A significant part of this chapter deals with the special case of the (k, ℓ-RP) with $k = 1$ and $\ell = 1$. Formally we define this restricted version as follows:

$$\max_{S \in \mathcal{F}} \min_{d \in E} \max_{\substack{r \in E - d \\ S - d + r \in \mathcal{F}}} w(S - d + r)\ . \tag{RP}$$

Without loss of generality, we assume that the interdiction is non-empty.

For a combinatorial optimization problem $\mathcal{P}$, we call the robust variant of $\mathcal{P}$ as in (RP) the *robust counterpart of* $\mathcal{P}$, or Robust $\mathcal{P}$ in short. We usually call the problem $\mathcal{P}$ the *underlying* or *nominal* problem. Note that the definition of any robust counterpart directly follows from the definition of the corresponding nominal problem and the definition of (RP).

As an example, consider a car-rental company with a single car, which we wish to rent out to different customers to maximize the total revenue. All customers have their own time intervals in which they want to rent the car exclusively and a price they are willing to pay. If there is no uncertainty, this is equivalent to an instance of the Interval Scheduling problem, where we are given a set of weighted intervals on the real line, and the goal is to find a maximum-weighted set of pairwise-disjoint intervals. See the

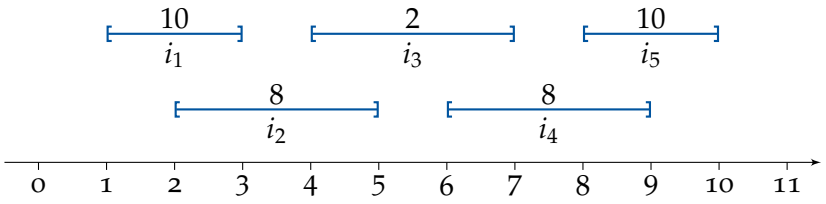


Figure 5.1: An example of an Interval Scheduling instance.

definition of the Interval Scheduling problem in Section 4.2.1 from Chapter 4. However, after the selection of the initial solution, some customers might withdraw from the contract, leading to less revenue for our company. In this case, we could still add unserved customers to our remaining initial solution. Although the additional customers should not be in conflict with our remaining initial solution, as those reservations were firm. As a concrete example, consider the five intervals $(1,3)$, $(2,5)$, $(4,7)$, $(6,9)$, $(8,10)$ with weights $10, 8, 2, 8, 10$, respectively, each representing a customer (see Figure 5.1). Picking $(1,3)$, $(4,7)$, and $(8,10)$ is an optimal solution to the underlying problem with value 22. An optimal solution to the robust counterpart, however, is to pick only $(1,3)$ and $(8,10)$, knowing that if $(1,3)$ or $(8,10)$ is interdicted, in the recourse one can add $(2,5)$ or $(6,9)$, respectively. This leads to a worst-case weight of 18 after the interdiction and recourse. On the other hand, when considering the optimal solution to the underlying problem, i.e., picking $(1,3)$, $(4,7)$, and $(8,10)$, the deletion of interval $(1,3)$ would lead to a final value of 12 in the robust version, as the interval $(4,7)$ blocks the recourse. This shows that selecting an optimal solution to the nominal problem need not be optimal for its robust counterpart.

The Adversary Problem. A closely related problem is the *adversary problem* of (k, ℓ-RP). It is the subproblem of (k, ℓ-RP) in which the set $S \in \mathcal{F}$ is already fixed and the adversary needs to find the optimal interdiction set D. That is, given some $S \in \mathcal{F}$, the task is to solve the following problem:

$$\min_{\substack{D \subseteq E \\ |D| \leq k}} \max_{\substack{R \subseteq E \setminus D,\ |R| \leq \ell \\ (S \setminus D) \cup R \in \mathcal{F}}} w((S \setminus D) \cup R)\ . \qquad (k, \ell\text{-AP})$$

Note that if $\ell \geq |E|$ for (k, ℓ-RP), it is optimal to choose an empty first-stage solution, as the recourse is essentially unbounded. Hence, for $\ell \geq |E|$, we can assume $S = \varnothing$. For this case, problem (k, ℓ-AP) is often called the *interdiction version* of the problem and is also known as the *most vital edge problem*. For many underlying problems, its interdiction version is NP-hard if k is part of the input, including spanning tree interdiction [FS99, Zen15], matching interdiction [ZRP+09, Zen10], or shortest path interdiction [BKS98]. However, for constant k the adversary problem is clearly polynomial-

time solvable by simple enumeration when the underlying nominal problem is also polynomial-time solvable.

Bounded-Regret Problem. Another closely related problem, which is both independently interesting and crucial for parts of our results, is the *bounded-regret* version of our problem. In the setting of our model, after the interdiction of an element $d \in E$, one takes the recourse action that may add an element $r \in E - d$ to the set $S - d$ which maximizes the total weight of the final solution. However, the weight of the final solution $w(S - d + r)$ could be less than the weight of the initial solution $w(S)$. This weight loss, $w(S) - w(S - d + r)$, can be seen as a *regret*; we denote it by $\Delta(d, S)$ and it is formally defined as:

$$\Delta(d, S) = w(S) - \max\{w(S - d + r) \mid r \in E - d, S - d + r \in \mathcal{F}\} .$$

The goal in the bounded-regret problem is to find a maximum-weight solution S such that, for each element $d \in E$, the interdiction of d followed by the construction of a maximum-weight second-stage solution $S - d + r$, results in a regret bounded by some given value $\lambda \in \mathbb{R}$; formally:

$$\max_{\substack{S \in \mathcal{F} \\ \Delta(d,S) \leq \lambda \colon \forall d \in E}} \quad w(S) . \tag{λ-RP}$$

We note that for some values of λ the problem (λ-RP) can be infeasible. We denote by $w_{\text{opt}}(\lambda)$ the objective value of (λ-RP) for an optimal solution, if one exists. The bounded-regret version for general values of k and ℓ of the problem is defined analogously. However, in this work, we focus on the above mentioned case with $k = \ell = 1$. Now, notice the tight connection between the bounded-regret version (λ-RP) and the robust counterpart (RP) of a problem. If we have a polynomial-time algorithm for (λ-RP), and if there are only polynomial-many different candidates for λ values, then one can enumerate over all λ values and solve (λ-RP) for each one of them. Finally, to solve (RP), we are interested in the λ value which maximizes $w_{\text{opt}}(\lambda) - \lambda$. The idea is that, if we choose a set $S \subseteq E$ corresponding to the solution $w_{\text{opt}}(\lambda)$ while selecting the first-stage solution of (RP), the maximum regret, even after adversarial interdiction and recourse, will be at most λ. Thus, the corresponding value $\max_\lambda w_{\text{opt}}(\lambda) - \lambda$ gives us an optimal solution of (RP). Now, observe that for $k = \ell = 1$, the total number of possible λ values is at most $\mathcal{O}(|E|^2)$ and hence polynomially bounded. Note that λ can also be negative and hence we also enumerate over these choices. All in all, when there are only polynomial many possible realizations of λ value, then (RP) can be polynomially reduced to (λ-RP).

5.2 Associated Work

Recoverable robust optimization is a powerful framework for decision-making under uncertainty that combines robustness with limited adaptability. Due to its generality, many (robust) combinatorial optimization problems can be naturally interpreted as recoverable robust problems. Most research in this area has focused on settings where the second-stage *costs* are uncertain, and recourse actions are constrained by a parameter that restricts the *symmetric difference* between the first-stage and second-stage solution. When facing discrete uncertainty—where a finite list of potential cost functions is explicitly given— many recoverable robust problems turn out to be NP-hard, even for problems that are polynomial-time solvable in the full-information setting, such as the shortest path problem [Büs12], basic scheduling and selection problems [KY97, BKK11b, GLW22]. To address this issue, approximation algorithms have been proposed; see for example, [CG16, GLW22].

In contrast, under interval uncertainty, these problems often reduce to finding two solutions to the nominal problem minimizing the weight for two different cost functions, while ensuring that the size of the symmetric difference is bounded. This reduction has been applied to problems such as the selection problem [LLW21], spanning trees [HKZ17a], general matroids [LPT21], the traveling salesperson problem [GLW21], and the assignment problem [FHLW21].

While most prior work focuses on cost uncertainty, some recent works have begun assuming *structural* uncertainty [DMP+15, IKK+22], as we do in this thesis. For instance, [DMP+15] studied the following related model: given a bipartite graph and a set of uncertain edges F, the task is to find a perfect matching M such that for any subset $F' \subseteq F$ with $|F'| \leq r$, the graph $G - F'$ contains a perfect matching M' where the symmetric difference of M and M' is bounded by some parameter. The authors of [DMP+15] establish hardness results for this problem and give characterizations for graphs that admit such matchings M.

Similar two-stage concepts that introduce a degree of adaptivity in robust optimization have been explored under different names. One example is *adjustable robustness*, studied by [BGGN04] in the context of robust linear programming and applications to multi-stage inventory management problem. Here, certain solution variables have to be fixed in the first stage, before the uncertain data is revealed ("here and now" decisions), whereas other variables can be adjusted in a second stage, after the uncertain data becomes known ("wait and see" decisions).

In contrast, in *demand robustness* or *two-stage adjustable robustness*, all solution variables are free to be adjusted in the second stage. However, the extent of these adjustments is regulated via a cost function, where the cost for setting a variable in the second stage (typically selecting it to be part of the solution) is substantially higher than its first-stage cost. This model has been studied extensively from both complexity and approximation perspectives for problems such as certain linear programs [BG10, ZdHS18,

HFG24], a variety of discrete covering problems including shortest paths, Steiner trees, set cover, and facility location [DGRS05, FJMM07, KKMS13, GNR14, HGS21], matching [HGHS21], and scheduling problems [CMRS15].

In all these different models, a commitment to the first-stage solution is not explicitly required. Instead, recourse is constrained by restricting the symmetric difference between first-stage and second-stage cost and/or by applying different cost functions for the two stages. In this chapter, we *explicitly require commitment* to the first-stage solution while allowing limited adaptation in the second stage. This adaptation is achieved, in the second stage, by *adding* a restricted number of additional elements to the partially remaining first-stage solution. Notably, similar fine-grained restrictions on recourse budgets have been explored in some problem-specific contexts. For example, Büsing, Koster, and Kutschka explicitly limit the total number of items that can be added or removed from the first-stage solution for the knapsack problem, in an experimental context [BKK11b]. Further, [BG22] require that a certain number of jobs must have the same scheduling position in the first and second stage in a single-machine scheduling problem. With our model, we aim to provide a general framework for recoverable robustness with commitment, unifying and extending such problem-specific approaches.

In this chapter, we focus on problems with downward-closed feasibility set, where a feasible solution remains feasible if arbitrary elements are removed. Hence, the structural uncertainty impacts only the solution quality (cost) and not its feasibility. This proves to be both interesting and challenging within our model.

A somewhat opposing setting is captured by *bulk-robustness* and *flexible network design*. In this framework, we select a superset of a solution that remains feasible even if a subset of elements from a given list is deleted. Hence, considering upward-closed problems, the goal is to find a minimum-cost solution that remains feasible in any scenario. Bulk-robustness has been studied in various contexts such as connectivity and matchings [ASZ15, AHM22, HMS21, CJ23, BCHI24]. The key difference between this model and ours is that, in this model, the structure of the solution can differ significantly from that of the nominal problem, whereas in our setting, the structure of a feasible solution remains unchanged.

For a swift introduction to robust combinatorial optimization, we refer the reader to a book by Goerigk and Hartisch [GH24].

5.2.1 Contribution

We present algorithmic and complexity results for a variety of combinatorial optimization problems in the model of Recoverable Robust Optimization with Commitment. In this chapter, we focus on some underlying problems that are combinatorial optimization problems with a downward-closed feasibility set, and which can be solved in polynomial time. As one would expect, robust counterparts of some of these problems turn out to be NP-hard, even in the setting $k = \ell = 1$. However, in some cases, the robust counterpart indeed remains polynomial-time solvable. The most prominent problem in this

	unweighted	weighted
(k, ℓ)-Robust Matroid when $k \leq \ell$	P	P
$(1, p)$-Robust Interval Scheduling for constant p	P	P
$(1, 1)$-Robust Bipartite Matching	NP-hard	NP-hard
$(1, 1)$-Robust Bipartite Independent Set	P	NP-hard

Table 5.1: Computational complexity of some combinatorial optimization problems. Note that NP-hardness results also prove NP-hardness for the general (k, ℓ)-case.

category is the Robust Matroid problem. We show that Robust Matroid is polynomial time solvable if $k \leq \ell$. In fact, we show that it suffices to solve the nominal problem for obtaining an optimal first-stage solution for the robust counterpart. However, this result does not hold as soon as $k > \ell$. For other problems, solving the robust counterpart is much more involved. Table 5.1 summarizes our results.

A major challenge in designing algorithms for the robust problem lies in the fact that a first-stage solution S must be robust against any interdiction with possible recourse taken into account. Even when $k = \ell = 1$, the problem is not easy. Intuitively, we may pre-compute for each element $f \in E$, a "backup element" for the scenario in which the element f fails. We then solve "the problem with backups". However, this increases the problem's complexity substantially, as such backups must not be selected for the solution S, and the uncertainty about the failing element (and its activated backup) causes complex dependencies.

Next, we discuss our main results for the variety of problems; definitions follow later.

Robust Matroid (RM). We study the problem of finding a maximum-weight independent set of a matroid in the robust setting. We refer to this problem as Robust Matroid (RM for short). We show that an optimal solution to the nominal problem, i.e. the problem of finding the maximum weight independent set in a matroid, is also an optimal first-stage solution for its robust counterpart, whenever $k \leq \ell$.

Hence, the classical greedy algorithm for the nominal problem implies a polynomial-time algorithm for RM. This is a bit surprising since the adversary problem of a special case of the nominal problem, Spanning Tree Interdiction, is NP-hard in general. However, this is not a contradiction since in our algorithm and proof, we do not need to actually compute the adversary's response and can simply prove that computing an optimal solution to the nominal problem is also optimal for RM.

Further, we give two cases in which this property is not true: First, we show that for $k > \ell$, the optimum solution to the nominal problem need not be an optimum solution to RM, even if $k = 2$ and $\ell = 1$. Second, we also show that for any non-matroid, this property is not true, even if $k = \ell = 1$. Therefore, for $k \leq \ell$, matroids are exactly those downward-closed structures for which the *price of robustness*, i.e., the worst-case ratio

between the weight of an optimal solution to the underlying problem and the weight of an optimal first-stage solution to its robust counterpart, is guaranteed to be 1.

Robust Bipartite Matching (RBM). Even for the unweighted case, we show that RBM is NP-hard; we give a reduction from 3-SAT. Here, the underlying problem, Bipartite Matching, can be viewed as a special case of a matroid intersection problem. Hence, our complexity result for RBM reveals a drastic increase in problem difficulty when the underlying problem involves finding a maximum-weight independent set of the *intersection of two matroids* instead of a *single matroid*. This is somewhat surprising, as for a single matroid, even an optimal solution of the nominal problem is optimal for its robust counterpart.

Robust Bipartite Independent Set (RBIS). A maximum-weight independent set in a graph is the complement of a minimum-weight vertex cover, which on bipartite graphs can be found (in polynomial time) via a primal-dual algorithm that simultaneously computes both a maximum-weight matching and a minimum-weight vertex cover. Hence, the complexity of Bipartite Independent Set and Bipartite Matching is the same, and even the algorithms are very similar (which inspired us to look into these problems more closely). However, the complexity of the robust counterpart of bipartite independent set, RBIS, depends highly on the weights. We show that, for unit weights, RBIS is polynomial-time solvable (even in more general *Kőnig-Egerváry* graphs). We prove this by reducing the problem to finding repairable maximum independent sets, which are maximum independent sets S satisfying that for each $v \in S$ there is some $u \in V \setminus S$ such that $S - v + u$ is an independent set. We then show that repairable maximum independent sets can be found in polynomial time (if they exist). Further, we show that RBIS is NP-hard in the weighted setting.

Robust Interval Scheduling (RISch). We give a polynomial-time algorithm for the $(1,1)$-Robust Interval Scheduling problem. This problem is equivalent to the Robust Independent Set problem on interval graphs. For convenience, we view this problem as selecting pairwise-disjoint intervals from a set of intervals and call it Robust Interval Scheduling (RISch). Our key ingredient is a reduction of RISch to its bounded-regret version. In general, it is not straightforward to construct a polynomial-time algorithm for the bounded-regret problem even in the case of the matroid independent set problem for which we know that the robust counterpart can be solved efficiently. As a crucial step, we show that for a given λ, the bounded-regret Robust Interval Scheduling problem can be reduced to a more general interval scheduling problem, which we call Interval Scheduling with Colors (ISC), and for which we give an efficient dynamic program. Furthermore, we outline how this result can be generalized to $(1,p)$-Robust Interval Scheduling for any constant $p \in \mathbb{N}$.

5.3 Robust Matroid

In this section, we study the Robust Matroid (RM) problem. We first define some preliminaries related to matroids and their properties. For an extensive review on matroids, we refer to the book by Oxley [Oxl06].

5.3.1 Background Notions in Matroids

In many combinatorial optimization problems, we are tasked to select a subset of elements of a ground set subject to some feasibility constraint. A matroid is a distinct combinatorial structure that provides a particularly well-behaved notion of what constitutes as a feasible set. Note however that, in the matroid literature, a "feasible set" is usually referred to as *independent*.[1] The following definition states in which way matroids are "well-behaved".

Definition 5.1. A *matroid* is a pair $\mathcal{M} = (E, \mathcal{I})$ in which E is the *ground set* and $\mathcal{I} \subseteq 2^E$ is the family of *independent sets* which satisfies the following axioms:

(I1) the empty set is independent, i.e., $\varnothing \in \mathcal{I}$,

(I2) if $I \in \mathcal{I}$, then $I' \in \mathcal{I}$ for all $I' \subseteq I$, and

(I3) if $I, I' \in \mathcal{I}$ and $|I| < |I'|$, then $I + a \in \mathcal{I}$ for some $a \in I' \setminus I$.

Axiom (I1) guarantees that there always exists an independent (i.e. feasible) set, (I2) states that each subset of an independent set is also independent, and (I3) states that an independent set can always be extended, by adding elements from another independent set that has cardinality strictly larger than that of the original independent set.

An (inclusion-wise) maximal independent set is called a *basis*. As a consequence of (I3), it follows that, all bases of a matroid have the same cardinality. A set X, that is not independent, i.e. $X \notin \mathcal{I}$, is called *dependent*. An (inclusion-wise) minimal dependent set is called a *circuit*. Given a matroid $\mathcal{M} = (E, \mathcal{I})$, we denote the set of all its bases by $\mathcal{B}$ and the set of all its circuits by $\mathcal{C}$.

An interesting feature of matroids is that, a matroid can be completely determined by its set of bases or circuits. We define this in the following two propositions.

Proposition 5.2. *$\mathcal{M} = (E, \mathcal{I})$ is a matroid and $\mathcal{B}$ is its set of bases if and only if the following two conditions hold:*

[1] This brings us in the unfortunate situation that the word "independent" is overloaded in this thesis. On one hand, we have an independent set in graph theoretic sense (see Section 4.5 and Section 5.5), whereas on the other hand, independent sets in the matroid sense. However, we assure the reader that this shall not constitute an issue; we only use the word independent in the matroid sense within this section and the other notion will not occur here.

(B1) $\mathcal{B} \neq \varnothing$ *and*

(B2) for all $B, B' \in \mathcal{B}$ *and all* $a \in B \setminus B'$*, there exists* $b \in B' \setminus B$ *such that* $B - a + b \in \mathcal{B}$*.*

Proposition 5.3. *$\mathcal{M} = (E, \mathcal{I})$ is a matroid and $\mathcal{C}$ is its set of circuits if and only if the following three conditions hold:*

(C1) $\varnothing \notin \mathcal{C}$*,*

(C2) for all $C, C' \in \mathcal{C}$ *with* $C \subseteq C'$ *it holds that* $C = C'$*, and*

(C3) for all $C, C' \in \mathcal{C}$ *with* $a \in C \cap C'$*, there exists* $C'' \in \mathcal{C}$ *with* $C'' \subseteq (C \cup C') - a$*.*

Given a set of bases $\mathcal{B}$ or circuits $\mathcal{C}$, we can derive the set of independent sets by

- $\mathcal{I} = \{I \subseteq E \mid I \subseteq B \text{ for some } B \in \mathcal{B}\}$ or
- $\mathcal{I} = \{I \subseteq E \mid I \not\supseteq C \text{ for all } C \in \mathcal{C}\}$,

respectively.

Proposition 5.2 can be strengthened to the following proposition due to Brualdi [Bru69], which we will refer to as *strong basis exchange*.

Proposition 5.4 (Theorem 2, [Bru69])**.** *A non-empty collection $\mathcal{B} \subseteq 2^E$ forms the set of bases of a matroid if and only if for all $B, B' \in \mathcal{B}$ and all $a \in B \setminus B'$ there exists $b \in B' \setminus B$ such that both $B - a + b \in \mathcal{B}$ and $B' - b + a \in \mathcal{B}$.*

Moreover, we have for every basis $B \in \mathcal{B}$ and every $a \notin B$ that, there exists a unique circuit $C(B, a) \in \mathcal{C}$ with $C(B, a) \subseteq B + a$ called the *fundamental circuit* with respect to B and a.

Furthermore, matroids are closed under two different removal operations. Let $\mathcal{M} = (E, \mathcal{I})$ be a matroid and $D \subseteq E$. The *deletion* of D yields the matroid $M \backslash D := (E \setminus D, \mathcal{I} \backslash D)$, where $\mathcal{I} \backslash D := \{I \in \mathcal{I} \mid I \cap D = \varnothing\}$. The *contraction* of an *independent* D on the other hand yields the matroid $M/D = (E \setminus D, \mathcal{I}/D)$, where $\mathcal{I}/D := \{I \in \mathcal{I} \mid I \cup D \in \mathcal{I}\}$. The latter definition of a contraction can also be extended to arbitrary contraction sets D, i.e. dependent ones, by performing the contraction with a maximal independent set $D' \subset D$ and a deletion of the remaining elements $D'' = D \setminus D'$, i.e. $\mathcal{M}/D = \mathcal{M}/D' \backslash D''$. For more details see, for example, [Sch03, Chapter 39].

5.3.2 Recoverable Robust Optimization with Commitment is Easy on Matroids

In this section, we consider the problem (k, ℓ)-Robust Matroid ((k, ℓ)-RM), for $k \leq \ell$ which is defined as follows:

(k,ℓ)-Robust Matroid ((k,ℓ)-RM)

Given: A matroid $\mathcal{M} = (E, \mathcal{I})$ with weight function $w\colon E \to \mathbb{R}_+$.
Task: Find $I \in \mathcal{I}$ such that for all $D \subseteq E, |D| \leq k$, there exists $R \subseteq E \setminus D, |R| \leq \ell$ such that $((I \setminus D) \cup R) \in \mathcal{I}$, and the weight $w((I \setminus D) \cup R)$ is maximized.

Let $\mathcal{B}$ be the collection of bases of the matroid $\mathcal{M} = (E, \mathcal{I})$. We will prove that for any weight function $w\colon E \to \mathbb{R}_+$, any maximum-weight basis in $\mathcal{B}$ is also an optimal solution to (k,ℓ)-RM if $k \leq \ell$. We use the following lemma.

Lemma 5.5. *Let B be a maximum-weight basis of $\mathcal{M}$ and let $d \in B$. Further, let $r \in \arg\max\{w(g) \mid g \in E \setminus \{d\},\ B - d + g \in \mathcal{B}\}$. Then $B - d + r$ is a maximum-weight basis of $\mathcal{M}\setminus\{d\}$.*

Proof. Let $B' = B - d + r$, and let $B^\star$ be a maximum weight independent set in $\mathcal{M}\setminus\{d\}$ with $|B^\star \cap B'|$ maximum. If $B' \neq B^\star$, take any $a \in B^\star \setminus B'$. By the strong basis exchange property from Proposition 5.4, there exists $b \in B' \setminus B^\star$ with $B^\star - a + b \in \mathcal{B}$ and $B' - b + a \in \mathcal{B}$. Therefore we have that (i) $w(a) > w(b)$, since $B^\star$ is a maximum weight basis of $\mathcal{M}\setminus\{d\}$, and by our choice of $B^\star$ being closest to B'. Hence, $B - b + a$ cannot be a basis of $\mathcal{M}$, since $w(B - b + a) > w(B)$, and B was a maximum weight basis of $\mathcal{M}$. Therefore we have that $b \notin C(B, a)$, but $b \in C(B', a)$, where $B' = B - d + r$. This implies that $d \in C(B, a)$, and thus (ii) $w(a) \leq w(r)$ by the choice of r as an element of maximum weight among those that can be feasibly exchanged with d. But then (i) and (ii) together imply that $w(b) < w(r)$. Since $d \in C(B, r) \cap C(B, a)$, by the circuit exchange property, there exists a circuit $\bar{C} \subseteq (C(B, r) \cup C(B, a)) \setminus \{d\} \subseteq B - d + r + a = B' + a$. Thus, this circuit $\bar{C}$ must be equal to the unique circuit $C(B', a)$ which contains b. It follows that $b \in C(B', a) = \bar{C} \subseteq (C(B, r) \cup C(B, a)) \setminus \{d\}$ with $b \notin C(B, a)$, implying that $b \in C(B, r)$. Summarizing, $B - b + r$ is a basis of larger weight in matroid $\mathcal{M}$ than B, a contradiction. □

Theorem 5.6. *Any maximum-weight basis of $\mathcal{M}$ is an optimal first-stage solution for (k,ℓ)-Robust Matroid when $k \leq \ell$. Thus, (k,ℓ)-Robust Matroid is polynomial-time solvable when $k \leq \ell$.*

Proof. Let S be a maximum weight basis of $\mathcal{M}$ and let $S^\star$ be an optimal first-stage solution to RM. Furthermore, let D be the set of elements that are deleted by the adversary in the worst-case for the first-stage solution S and let S' be its second-stage solution after the deletion of D and the recourse. Thus, S' is a maximum-weight basis in matroid $\mathcal{M}\setminus F / (S \setminus F)$. Additionally, let $S^\star_D$ be the second-stage solution for the first-stage solution $S^\star$ with interdiction set D. We will show that $w(S') \geq w(S^\star_D)$. Since D was the worst-case choice of the adversary for S and D could also be chosen as an interdiction set for $S^\star$, we have that then S is also an optimal first-stage solution. Hence, it remains to prove $w(S') \geq w(S^\star_D)$.

Consider the adversary and the recourse as a one-by-one process, in which the elements in D are deleted one after the other in a fixed order, and after each deletion of an element (the adversary's action) another element is added to the solution (the recourse), until all elements of D have been deleted and each recourse has been added. That is, we have a fixed order of the elements in D, i.e., $d_1, d_2, \ldots, d_k$, $d_i \in D$ for $1 \leq i \leq k$ and in step i the element d_i is deleted and an element $r_i \in E \setminus (S \cup \{d_1, d_2, \ldots, d_i\})$ is added to S such that $(S \cup \{r_1, r_2, \ldots, r_i\}) \setminus \bigcup_{j=1}^{i} \{d_j\}$ is an independent set (or an element $r_i^\star \in E \setminus (S^\star \cup \{d_1, d_2, \ldots, d_i\})$ for $S^\star$, respectively). Hence, we consider this process for S and $S^\star$. Note that we can do this process if $k \leq \ell$.

In this process, one can potentially add elements from $(E \setminus S) \cap D$ that are then removed again in a later step (and the same for $S^\star$). Furthermore, in this process, as recourse for the elements in the first-stage solution S we always add the elements from the remaining set of elements $E \setminus (S \cup \{d_1, d_2, \ldots, d_{i-1}\})$ that maximizes the weight of the resulting basis of $\mathcal{M} \setminus \bigcup_{j=1}^{i-1} \{d_j\}$. That is, $r_i \in \arg\max\{w(g) \mid ((S + g - d_i \cup \{r_1, r_2, \ldots, r_{i-1}\}) \setminus \bigcup_{j=1}^{i} \{d_j\}) \in \mathcal{I}\}$. Note that this is always possible due to the strong basis exchange property (taking $B_1 := S \cup \{r_1, \ldots, r_{i-1}\} \setminus \{d_1, \ldots, d_{i-1}\}$, $a := d_j$, and $B_2 := S'$). For the optimal first-stage solution $S^\star$, if some element is deleted during the process, we simply add some element from the final solution $S_D^\star$. Let S_i and $S_i^\star$, $0 \leq i \leq k$, be the intermediate bases obtained by this process for S and $S^\star$, respectively. Again, note that we can always add an element for each removed element since $k \leq \ell$.

Now we invoke Lemma 5.5 and have that S_1 is a maximum weight basis of $\mathcal{M} - \{d_1\}$. Furthermore, iteratively we can now invoke Lemma 5.5 to show that S_i is a maximum weight basis of $\mathcal{M} \setminus \bigcup_{j=1}^{i} \{d_j\}$. Hence, we have that $w(S_i) \geq w(S_i^\star)$ for all $1 \leq i \leq k$. In particular, since S_k is an optimal basis of $\mathcal{M} \setminus F$, and since S_k contains all elements from $S \setminus F$, it follows that $w(S') = w(S_k)$, implying $w(S') = w(S_k) \geq w(S_k^\star) = w(S_D^\star)$. As a result, by using the polynomial-time greedy algorithm for the matroid basis problem ([Kru56], see citations in Chapter 40 of Schrijver's book), we can compute an optimal first-stage solution for (k, ℓ)-Robust Matroid in polynomial time when $k \leq \ell$. □

In contrast to Theorem 5.6, we now give a simple example on graphic matroids which illustrates an optimal nominal solution need not be an optimal robust solution for our model when $k > \ell$.[2] The example is for the smallest possible case, i.e. when $k = 2$ and $\ell = 1$.

Example 5.7. Consider the diamond graph G given in Figure 5.2 with the corresponding weight function given on the edges of the graph.

Given the graph in Figure 5.2, the maximum weighted spanning tree in the graph consists of the edge set $S = \{ba, bc, bd\}$. However, if we consider this solution as the first-stage solution for our $(2, 1)$-RM problem, the worst adversarial deletion is the edge

[2] We thank Ruiyang Zhang for noticing this counterexample which he discovered during his master's thesis conducted at the Chair of Management Science at RWTH Aachen University [Zha25].

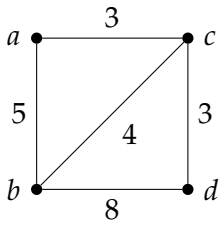


Figure 5.2: Counterexample where optimal nominal solution is not optimal for the robust counterpart for the $(2,1)$-Robust Matroid problem.

set $\{ab, bd\}$. Then followed by our recourse of edge ac, we achieve the final second-stage solution of weight 7. With simple trial and error, it is easy to verify that for the given first-stage solution, this is the optimal second-stage solution in our model. Contrary to that, we claim the nominal-optimal first-stage solution need not be optimal for the robust counterpart. Consider a suboptimal nominal solution $S' = \{ab, ac, cd\}$. Here, the worst case adversarial deletion set is $\{bd, cd\}$, which can no longer be extended by adding another edge into the partial remaining solution. Hence the second-stage solution following the set S' is $\{ab, ac\}$ with final weight 8.

Further, we show that matroids are the only downward-closed set systems for which an optimal solution to the nominal problem is also an optimal solution to its robust counterpart (provided that $k \leq \ell$).

Theorem 5.8. *For every non-empty downward-closed set system $\mathcal{F} \subseteq 2^E$ which is not a matroid, there exist weights $w\colon E \to \mathbb{R}_+$ such that an optimal solution to the nominal problem is not an optimal first-stage solution for the robust counterpart.*

Recall that a non-empty downward-closed system $\mathcal{F} \subseteq 2^E$ is not a matroid if the exchange property (I3) is violated. Such *non-matroids* can be characterized via the following lemma (see, for example, [FGH+17] for the proof).

Lemma 5.9 ([FGH+17])**.** *Let $\mathcal{F} \subseteq 2^E$ be a non-empty downward-closed system which is not a matroid, and let $\bar{\mathcal{F}} \subseteq \mathcal{F}$ be the collection of the inclusion-wise maximal sets in $\mathcal{F}$. Then there exist two sets $X, Y \in \bar{\mathcal{F}}$, and three elements $\{a, b, c\} \subseteq X \Delta Y := (X \setminus Y) \cup (Y \setminus X)$ such that for each set $Z \in \bar{\mathcal{F}}$ with $Z \subseteq X \cup Y$, either $a \in Z$, or $\{b, c\} \subseteq Z$.*

Proof of Theorem 5.8. Take $X, Y \in \bar{\mathcal{F}}$, and the three elements $\{a, b, c\} \subseteq X \Delta Y$ as in the statement of Lemma 5.9. We set the weights of all elements in $E \setminus \{a, b, c\}$ to zero, and $w(a) = 3$, $w(b) = w(c) = 2$. Then the optimal first-stage solution for the $(1,1)$-robust counterpart is to select either $S = \{b\}$, or $S = \{c\}$. Then, if the adversary interdicts, one can still add $\{a\}$ and achieve a second-stage value of 3. Otherwise, if the adversary does

not interdict, one can extend to the optimal solution of the underlying problem $\{b, c\}$ of value 4.

In contrast, if the first-stage solution S is equal to a nominal optimal solution, then S necessarily contains $\{b, c\}$, and one can only achieve a value of 2. Because if either b or c is interdicted, one cannot extend S by an element of positive value in the recourse. □

In the following greedoid[3] example, we illustrate why our approach of choosing an optimal nominal solution in the first-stage, for a non-downward-closed set system, does not result into an optimal robust solution even for the case $k = 1, \ell = 1$.

Example 5.10. Consider a greedoid $\mathcal{G} = (E, F)$ with ground set $E = \{a, b\}$ and the collection of independent sets to be $\{\varnothing, \{a\}, \{a, b\}\}$ along with weight function $w\colon E \to \{1\}$ and $w(S) = |S|$ for $S \subseteq E$. The optimal nominal solution for this greedoid is to choose the set $\{a, b\}$. However, if we do choose the first-stage solution to be $\{a, b\}$ and the adversary deletes element a, it renders the set $\{b\}$ which is not an independent set. Moreover, neither can the set $\{b\}$ be extended to an independent set using a recourse action. Thus, we even fail to achieve feasibility for the $(1, 1)$-RP. Instead, a better first-stage solution would be to only pick $\{a\}$ which is suboptimal but if nothing gets deleted, we can add b to get $\{a, b\}$ of weight 2. Moreover, if the single element a gets deleted, we cannot add anything but we are left with $\varnothing$ which is still independent.

5.4 Robust Bipartite Matching

Maximum Matching is a combinatorial optimization problem in which we are given an undirected graph G, and the task is to find the maximum cardinality matching in G. It is well known that one can find a maximum-cardinality matching in polynomial time even for a general graphs [Edm65]. In stark contrast to this, we show that the robust counterpart of Maximum Matching becomes intractable even when we restrict ourselves to bipartite graphs. In this section, we consider the problem Robust Bipartite Matching (RBM), which is the problem (RP) where $\mathcal{F}$ consists of all matchings in an underlying bipartite graph. We study the decision version of this problem. Note that the problem to find a maximum cardinality matching in a bipartite graph is a classical application for Matroid Intersection, i.e., the problem to find a maximum cardinality set which is independent in two matroids on the same ground set E. Hence, the following result illustrates a drastic increase in the complexity when considering the *intersection of two matroids* instead of a *single matroid*. Below we give a definition of the Unweighted Robust Bipartite Matching problem.

[3]Greedoids are structures similar to matroids which instead of (I2) satisfy the property that for each non-empty $I \in \mathcal{I}$, there exists $e \in I$ such that $I - e \in \mathcal{I}$.

Unweighted Robust Bipartite Matching (RBM)

Given: An undirected bipartite graph $G = (V, E)$ and a positive integer k.
Task: Decide whether there is a matching M in G such that for each $d \in E$ there exists an edge $r \in E - d$ for which $M - d + r$ is a matching of cardinality at least k.

Before we proceed, we make a general statement on Robust maximum cardinality problems. Consider the unweighted version of problem (RP), where the task is to find a set $S \in \mathcal{F}$ which maximizes $|S - d + r|$ after the interdiction of $d \in E$ and the possible recourse of element $r \in E - d$. Note that $|S - d + r| \geq |S| - 1$. Thus, it suffices to restrict our search of S to the maximum cardinality sets in $\mathcal{F}$. Moreover, it suffices to consider $d \in S$. We call a solution $S \in \mathcal{F}$ of the nominal problem *repairable*, if for each $d \in S$ there exists an element $r \in E \setminus S$ such that $S - d + r \in \mathcal{F}$.

Observation 5.11. *For the robust counterpart (RP) of a combinatorial problem $\mathcal{P}$ with unit weights, there is a maximum cardinality solution S to $\mathcal{P}$ that is optimal for (RP). Furthermore, if there is a maximum cardinality solution S of $\mathcal{P}$ that is repairable, then S is optimal.*

We are now ready to state and prove the main result of this section.

Theorem 5.12. Unweighted Robust Bipartite Matching *problem is* NP*-complete.*

Proof. Clearly, the Unweighted Robust Bipartite Matching problem is in NP, because for any given matching M, one can enumerate for each possible deletion of edge $d \in E$ the recourse $r \in E - d$, and verify whether $|M - d + r| \geq k$.

If we set k to be the size of a maximum cardinality matching, then by Observation 5.11, we need to essentially find a repairable matching in the graph. Thus, it suffices to show that deciding whether there exists a repairable matching of maximum cardinality is NP-hard. We refer to this problem as Repairable Bipartite Matching (RepBM) and give a reduction from 3-SAT: given a set of Boolean variables $X = \{x_1, x_2, \ldots, x_n\}$ and a formula φ in conjunctive normal form, where each clause has 3 variables, does there exist a truth assignment that satisfies φ?

Let I be an instance of 3-SAT with a set X of Boolean variables and a set $Y = \{C_1, C_2, \ldots C_m\}$ of clauses in which $|C_i| = 3$. We construct an instance I_{RepBM} of problem RepBM with input graph $G = (V, E)$ as follows (see also Figure 5.3):

- For each variable $x_i \in X$ in I, we add two nodes a_i and $\bar{a}_i$ in $V(G)$ corresponding to a positive and negative literal of x_i, respectively. We set $A := \{a_1, \ldots, a_n, \bar{a}_1, \ldots, \bar{a}_n\}$. Moreover, we add b_i in $V(G)$ for each $x_i \in X$ and set $B := \{b_1, \ldots, b_n\}$. Additionally, for each clause $C_j \in Y$, we add two nodes c_j, z_j in $V(G)$ and set $C := \{c_1, \ldots, c_m\}$ and $Z := \{z_1, \ldots, z_m\}$. Finally, we have $V(G) = A \cup B \cup C \cup Z$.

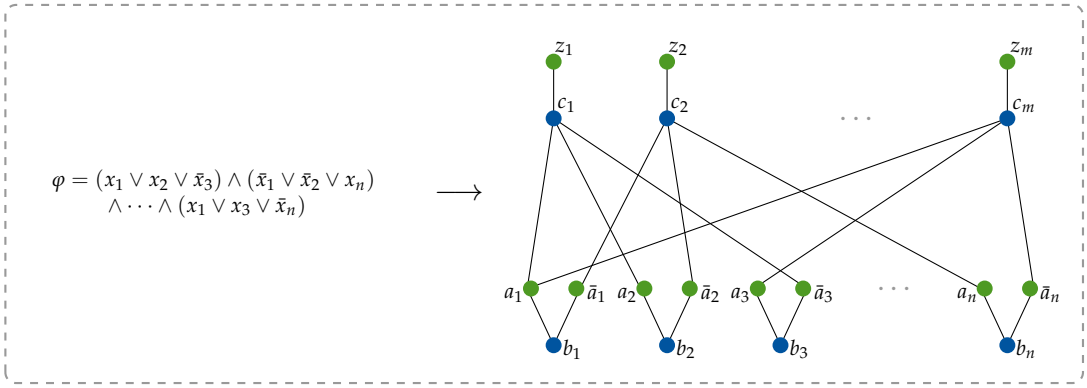


Figure 5.3: NP-hardness reduction from 3-SAT to Unweighted Robust Bipartite Matching.

- Let the edge set $E(G)$ be the union of the following set of edges.
  (a) For all $i \in [n]$, add $a_i b_i, \bar{a}_i b_i$ to $E(G)$.[4]
  (b) For all $j \in [m]$, add $c_j z_j$ to $E(G)$.
  (c) If $x_i \in C_j$, then add $a_i c_j$ to $E(G)$ and if $\bar{x}_i \in C_j$, then add $\bar{a}_i c_j$ to $E(G)$.

Note that G is bipartite as $V(G)$ can be partitioned into two independent sets, $A \cup Z$ and $B \cup C$.

Given a satisfying truth assignment $\gamma : X \to \{0, 1\}$ of I, we show that the matching $M = \{c_j z_j \mid j \in [m]\} \cup \{a_i b_i \mid \gamma(x_i) = 0\} \cup \{\bar{a}_i b_i \mid \gamma(x_i) = 1\}$ is a solution for I_{RepBM}. Notice that $|M| = m + n$ because M leaves exactly n unsaturated vertices in $V(G)$. The set of these n unsaturated vertices constitutes of one element from each $\{a_i, \bar{a}_i\}$ where $i \in [n]$. These are precisely the literals with truth value 1 assigned to them by γ. All edges $r \in M$ contain either some b_i, $i \in [n]$, or some c_j, $j \in [m]$, as an endpoint. Note that M is a matching in G of maximum cardinality.

To prove that M is a solution of I_{RepBM}, we show that for each edge $d \in M$ there is an edge $r \in E(G) \setminus M$ such that $M - d + r$ is a matching. In other words, for each edge d, there is an edge r connecting an unsaturated vertex in $V(G)$ to an endpoint of d. If d is incident to b_i, we take as r the other edge incident to b_i. If d is incident to c_j, then there is an unsaturated vertex a_i such that $a_i c_j \in E(G) \setminus M$, or an unsaturated vertex $\bar{a}_i$ such that $\bar{a}_i c_j \in E(G) \setminus M$. This is because M is derived from the satisfying truth assignment γ of I, and clause C_j contains some literal x_i or $\bar{x}_i$ with truth assignment 1.

Next, we show that if I_{RepBM} is a yes-instance of RepBM, then I also is a yes-instance of 3-SAT. If I_{RepBM} is a yes-instance of RepBM, then there is a matching M' of size $m + n$ in which each $d \in M'$ has an edge $r \in E(G) \setminus M'$ such that $M' - d + r$ is a matching.

[4] In this chapter, we do not need to worry about directed edges, so we simply denote the undirected edges by an alternating notation without any braces, e.g. we use uv instead of $\{u, v\}$.

First, we claim that we can assume without loss of generality that M' contains all edges in $\{c_j z_j \mid j \in [m]\}$. Assume this is not true. Then, since M' has to have size $m+n$, there must exist some edge $a_i c_j \in M'$ or some edge $\bar{a}_i c_j \in M'$. Without loss of generality, assume we are in the former case. Then we claim that $M'' = M' + c_j z_j - a_i c_j$ is also a repairable matching of cardinality $m+n$. Clearly, M'' is a matching of cardinality $m+n$. Hence, it remains to show that for each $d \in M''$ there exists some $r \in E \setminus M''$ such that $M'' - d + r$ is also a matching. First, consider some $d \in M'' \setminus \{c_j z_j\}$. In that case, set r to be some edge such that $M' - d + r$ is a matching (which exists since M' is repairable). Observe that r cannot be incident to c_j (since $a_i c_j \in M'$) and also not incident to z_j (since z_j has degree 1). Hence, $M'' - d + r$ is a matching. Second, consider the case $d = c_j z_j$. Now choose $r = a_i c_j$. In this case, clearly, $M'' - d + r$ is a matching since $M'' - d + r = M'$. By applying this procedure iteratively to all edges $a_i c_j \in M'$, we obtain the desired result.

Observe that all edges in M' are either of the form $c_j z_j$ for some $j \in [m]$, or of the form $a_i b_i$ or $\bar{a}_i b_i$ for some $i \in [n]$. Hence, since M' has cardinality exactly $m+n$, either a_i or $\bar{a}_i$, $i \in [n]$ is saturated by M'. Let $U \subseteq A$ be the set of vertices in A that are not saturated by M'. Let $\gamma : X \to \{0,1\}$ where $\gamma(x_i) = 1$ if $a_i \in U$, and $\gamma(x_i) = 0$ if $\bar{a}_i \in U$. We claim that this assignment satisfies the formula φ.

For the sake of contradiction, assume that γ does not satisfy φ. Then there is a clause C_j for which all three literals in C_j evaluate to 0. Now consider the corresponding vertices in the graph G; the literals evaluate to 0 because the corresponding vertices are matched to b_i in the matching M'. This implies there is no $r \in E(G) \setminus M'$ which replaces the edge $c_j z_j \in M'$. However, this contradicts the fact that M' is a solution to I_{RepBM}.

The size of the reduced instance of RBM is polynomial in the total number of variables and clauses of the 3-SAT instance. Hence our reduction implies that Unweighted Robust Bipartite Matching is NP-complete. □

5.5 Robust Independent Set on Bipartite Graphs

This section is devoted to the Robust Independent Set (RIS) problem, which is the problem (RP) where the feasibility set $\mathcal{F}$ is the set of all independent sets. In the following we consider bipartite graphs and show that the Unweighted Robust Independent Set problem admits an efficient algorithm on bipartite graphs. This result extends to *Kőnig-Egerváry* graphs which generalize bipartite graphs; they are graphs in which the size of a minimum vertex cover equals the size of a maximum matching ([Dem79]). Finally, we show that the *weighted* problem is NP-hard even on bipartite graphs.

Theorem 5.13. Unweighted Robust Independent Set *is polynomial-time solvable on bipartite graphs.*

By Observation 5.11, it suffices to show that we can find a repairable maximum cardinality independent set in a bipartite graph if it exists, or declare that none exist. Hence, Theorem 5.13 follows directly from the following lemma.

Lemma 5.14. *A bipartite graph G contains a repairable maximum cardinality independent set if and only if the underlying graph admits a perfect matching such that each edge in the matching has a degree-one endpoint.*

Note that in order to check whether a graph admits a perfect matching such that each edge in the matching has a degree-one endpoint, it suffices to compute a maximum cardinality matching M, and check whether M is perfect, and whether each edge of M has at least one endpoint of degree 1.

Proof. Let G admit a maximum independent set $S \subseteq V(G)$ which is repairable. Then by definition, for all $v \in S$, there is a vertex $w \in V \setminus S$ such that $S - v + w$ is also an independent set. Note that, if $w \notin N(v)$, then $S + w$ is an independent set in contradiction to the fact that S is an independent set of maximum cardinality in G. Hence for all $v \in S$, there is a vertex $w \in N(v)$ such that $S - v + w$ is an independent set. We call w the *backup* vertex of v. Note that $N(w) \cap S = \{v\}$. This implies that in a repairable independent set S, each vertex $v \in S$ has a distinct backup vertex w. Hence, there should be at least $|S|$ many backup vertices in the graph (one for each $v \in S$), and since the backup vertices do not belong to the independent set S, we have $|S| \leq \frac{n}{2}$. As the size of a maximum independent set in a bipartite graph is at least $\frac{n}{2}$, we have $|S| = \frac{n}{2}$. Therefore, G contains a perfect matching $M = \{vw \in E \mid v \in S, w \text{ is a backup of } v\}$. Now consider a vertex $v \in S$ and its backup vertex $w \in N(v)$; w is the only neighbor of v, otherwise S is not an independent set or some vertex $u \in V \setminus S$ is not a backup vertex. Thus each $v \in S$ is degree-one.

Now let us assume that G has the desired properties. We show that there is a repairable maximum cardinality independent set in G. If the graph contains a perfect matching and every matching edge is incident to a degree one vertex, then a collection of these degree-one vertices, $S \subseteq V(G)$, one from each edge in the perfect matching, is a solution of the instance of the problem. Observe that for each vertex $v \in S$, the unique neighbor w of v serves as a backup vertex. □

Notice that the crucial properties used to prove Lemma 5.14 and Theorem 5.13 even hold for Kőnig-Egerváry graphs and thus we get the following.

Corollary 5.15. Unweighted Robust Independent Set *is polynomial-time solvable in Kőnig-Egerváry graphs.*

This is interesting in the following sense: Theorem 5.12 yields a hardness result for Unweighted Robust Bipartite Matching. Recall, from the preliminaries Section 2.2,

that the line graph of a graph G contains the vertex set corresponding to $E(G)$, and there is an edge between two vertices if and only if the corresponding edges share an endpoint in G. In general, a maximum independent set in the line graph of a graph G corresponds to a maximum matching in G, and hence can be computed in polynomial time. However, while Theorem 5.13 gives a polynomial-time algorithm for Unweighted Robust Independent Set on bipartite graphs, this one-to-one correspondence and Theorem 5.12 imply that Unweighted Robust Independent Set is NP-hard in line graphs of bipartite graph.

Corollary 5.16. Unweighted Robust Independent Set *is* NP*-hard in line graphs of bipartite graphs.*

Next, we show that the complexity of Robust Independent Set on bipartite graphs changes with weights. We now consider the decision version of the Robust Independent Set problem.

Robust Independent Set (RIS)

Given:	An undirected graph G, a weight function $w\colon V \to \mathbb{R}_+$, and a positive integer k.
Task:	Decide whether there is an independent set S in G such that for each $v \in V$ there exists a node $v' \in V - v$ which results into another independent set $S - v + v'$ such that $w(S - v + v') \geq k$.

Theorem 5.17. Robust Independent Set *is* NP*-complete on bipartite graphs.*

Proof. We prove that the above mentioned decision variant of Robust Independent Set is NP-hard on bipartite graphs. We give a reduction from 3-SAT.

Given an instance I of 3-SAT with the set of variables $X = \{x_1, x_2, \ldots, x_n\}$ and a satisfiability formula φ with clauses $Y = \{C_1, C_2, \ldots C_m\}$, we construct an instance $I' = (G, w, k)$ of the Robust Independent Set problem as follows (also see Figure 5.4):

- For each variable $x_i \in X$, we take vertices $a_i, \bar{a}_i, b_i, b_i^1, b_i^2$ in $V(G)$. For each clause C_j in φ, we add a vertex c_j in $V(G)$. Moreover, we add 3 representative vertices, c_j^1, c_j^2, c_j^3, corresponding to each literal in C_j.
- The edge set $E(G)$ consists of the following set of edges:
  - For all $i \in [n]$, add $b_i b_i^1, b_i b_i^2, b_i^1 a_i, b_i^2 \bar{a}_i \in E(G)$.
  - For all $j \in [m]$, add $c_j c_j^1, c_j c_j^2, c_j c_j^3 \in E(G)$.
  - Add an edge between the three vertices c_j^1, c_j^2, c_j^3 and their respective corresponding literals in the clause, for example if $C_1 = (v_1 \vee \bar{v}_2 \vee v_4)$, then add edges $c_1^1 a_1, c_1^2 \bar{a}_2, c_1^3 a_4 \in E(G)$.

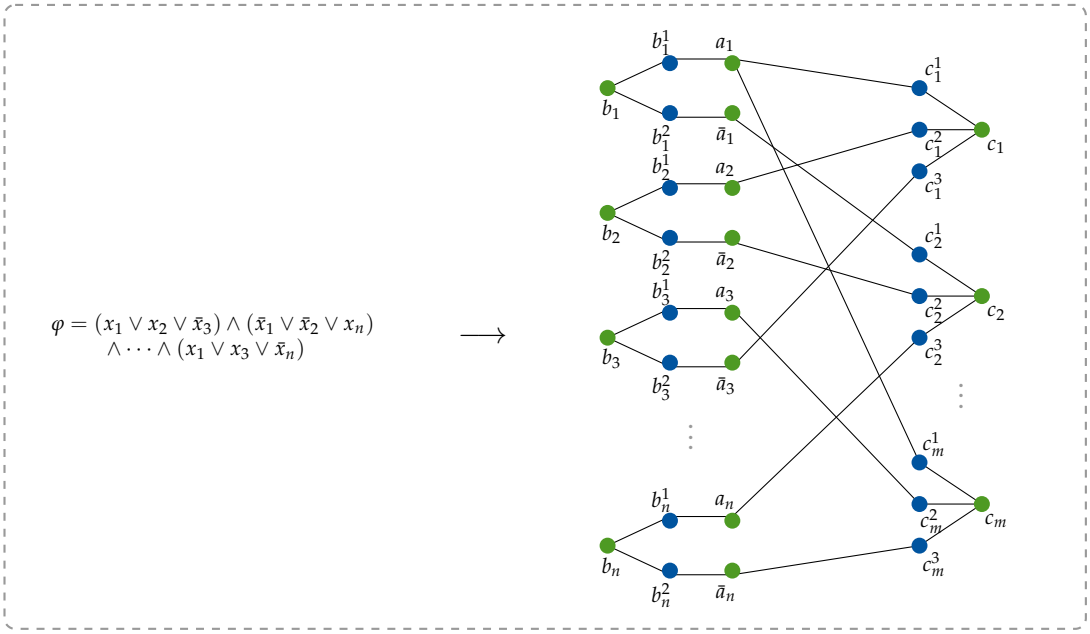


Figure 5.4: NP-hardness reduction from 3-SAT to (weighted) Robust Independent Set on bipartite graphs.

- For the weight function $w : V(G) \to \mathbb{R}_+$, consider $1 \ll r \ll s$, and $r, s \in \mathbb{R}_+$. Moreover, set the weight function as follows:

$$w(v) = \begin{cases} s & \text{if } v \in \{c_j \mid j \in [m]\} \cup \{b_i \mid i \in [n]\}, \\ r & \text{if } v \in \left\{\{c_j^1, c_j^2, c_j^3 \mid j \in [m]\} \cup \{b_i^1, b_i^2 \mid i \in [n]\}\right\}, \\ 1 & \text{if } v \in \{a_i, \bar{a}_i \mid i \in [n]\}. \end{cases}$$

- Set $k = (m + n - 1)s + r + n$.

Note that the constructed graph is bipartite.

First, let us assume that φ is a yes-instance and let $\gamma : X \to \{0, 1\}$ be the truth assignment. We claim that $S = \{c_j \mid j \in [m]\} \cup \{b_i \mid i \in [n]\} \cup \{a_i \mid \gamma(x_i) = 0\} \cup \{\bar{a}_i \mid \gamma(x_i) = 1\}$ is a solution to I' of weight at least k. First, observe that S is an independent set and that $w(S) = (m + n)s + n = k + s - r$. Moreover, note that $a_i \notin S$ if and only if $\gamma(x_i) = 1$. For all $j \in [m]$, there exists a vertex $c_j' \in \{c_j^1, c_j^2, c_j^3\}$ such that c_j' corresponds to a literal a for which $\gamma(a) = 1$, and hence $a \notin S$. Thus c_j' is only connected to c_j in S. Therefore, this vertex c_j' can be a backup vertex of c_j, meaning that $S + c_j' - c_j$ is an independent set of weight k. Also, with a similar argument, there is a backup vertex b_i^1 or b_i^2 for every $b_i \in S$. Thus, for each $v \in S$ of weight $w(v) = s$, there is a backup vertex v' of weight $w(v') = r$. Consequently, for each $v \in S$ there exists some $v' \in V \setminus S$ such that $w(S) + w(v') - w(v) \geq (m + n - 1)s + r + n = k$, as claimed.

Next, let us assume that there is an independent set S of G such that for each $v \in S$ there is some $v' \in V \setminus S$ with $w(S) + w(v') - w(v) \geq (m+n-1)s + r + n = k$. We construct a truth assignment γ for I. Observe that S must contain all $m+n$ vertices of weight s, and none of the weight r vertices since they are all neighbors of weight s vertices. Furthermore, by the property of S, for each c_j there is a vertex $c'_j \in \{c_j^1, c_j^2, c_j^3\}$, such that the corresponding literal of c'_j, say a, is not in S, i.e., $S + c'_j - c_j$ is an independent set (i). Since for all $v \in S$ there exists some $v' \in V \setminus S$ such that $w(S) - w(v) + w(v') \geq (m+n-1)s + r + n$, set S must contain n vertices from set $A = \{a_1, \ldots, a_n, \bar{a}_1, \ldots, \bar{a}_n\}$. Furthermore, since for each b_i there is a vertex v' of value r such that $S + v' - b_i$ is an independent set, not both a_i and $\bar{a}_i$ are in S. Hence, exactly one of $a_i, \bar{a}_i$ belongs to S (ii). We take the set of vertices in A which are not part of S and define the assignment $\gamma(x_i) = 1$ if and only if $a_i \notin S$. We claim that γ is a truth assignment. Clearly, by (ii), γ is an assignment; and by (i), γ satisfies exactly the definitions of a truth assignment. □

5.6 Robust Interval Scheduling

In this section we consider Robust Independent Set on interval graphs. For convenience, we use the standard interval representation of an interval graph $G = (V, E)$ where each node is represented as an interval in $\mathbb{R}$, and an independent set in G corresponds to a selection of pairwise-disjoint intervals. We refer to the task of finding a maximum-weighted subset of disjoint intervals as Interval Scheduling (ISch) and its robust counterpart as Robust Interval Scheduling (RISch). It is known that ISch is solvable in polynomial time via dynamic programming [Fra76] (or with a greedy algorithm in unweighted setting); we also revised this dynamic algorithm in Section 4.4.1. Now, we formally define the Robust Interval Scheduling problem as follows.

(1,1)-Robust Interval Scheduling (RISch)

Given: A set of intervals $I = \{i_1, i_2, \ldots, i_n\}$, where $i_j = [a_j, b_j)$, $a_j < b_j$, $a_j, b_j \in \mathbb{R}_+$, a weight function $w\colon I \to \mathbb{R}_+$, and a family $\mathcal{F} \subseteq 2^I$ of sets of pairwise-disjoint intervals in I.

Task: Calculate the maximum achievable value of the second-stage solution, that is, for an optimal pairwise-disjoint subset of intervals $S \subseteq I$, calculate the maximum value of $w(S - i_j + i_\ell)$ over the deletion of any interval i_j and the best possible recourse by another interval i_ℓ thereafter.

Recall that $\varnothing \in I$ as we redefine each of our ground sets such that it includes $\varnothing$; this way the empty set in I can be used to show the possible empty deletion and the empty recourse. Moreover, sometimes we refer to $[a_j, b_j) \cap [a_k, b_k)$ simply by $i_j \cap i_k$.

We prove the following main result in this section.

Theorem 5.18. *There is a polynomial-time algorithm for* $(1,1)$-Robust Interval Scheduling.

Later in Subsection 5.6.4, we generalize our results and layout a polynomial-time algorithm to solve the $(1,p)$-Robust Interval Scheduling problem for an arbitrary integer p.

5.6.1 Algorithm Outline and Proof Roadmap

Here, we outline our algorithmic approach for the proof of Theorem 5.18. As mentioned in the Section 5.1, one can reduce the robust counterpart of a nominal problem to its bounded-regret version (λ-RP). The bounded-regret version of RISch, which we denote as λ-Robust Interval Scheduling (λ-RISch), asks to find $S \in \mathcal{F}$ which optimizes the following objective function:

$$\max_{\substack{S\in\mathcal{F} \\ \Delta(i_j,S)\leq\lambda\colon \forall i_j\in I}} w(S), \tag{λ-RIS}$$

where the regret $\Delta(i_j,S) = w(S) - \max\{w(S-i_j+i_\ell) \mid i_\ell \in I - i_j, S-i_j+i_\ell \in \mathcal{F}\}$. The value $\Delta(i_j,S)$ defines the loss in the objective value occurred due to the change in the solution when i_j fails. Note that, if $i_j \in S$, then we simply have $\Delta(i_j,S) = w(i_j) - \max\{w(i_\ell) \mid i_\ell \in I \setminus S,\ S-i_j+i_\ell \in \mathcal{F}\}$.

Due to the established tight connection between (λ-RP) and (RP) formulations, a polynomial-time algorithm for λ-RISch implies a polynomial-time algorithm for RISch, since here the total number of different values of regrets is bounded by $\mathcal{O}(|I|^2)$. Following the definition of regret, each pair of deletion and recourse interval gives us a potential regret value, for example, deletion of i_j followed by the recourse of i_ℓ gives us a potential value of $\lambda = w(i_j) - w(i_\ell)$. As a result, we have the following proposition.

Proposition 5.19. *A polynomial-time algorithm for the problem* λ-Robust Interval Scheduling *implies a polynomial-time algorithm for* $(1,1)$-Robust Interval Scheduling.

Given Proposition 5.19, it remains to provide a polynomial-time algorithm for λ-RISch for an arbitrary fixed value of $\lambda \in \mathbb{R}$. For the bounded regret version, the regret (or the maximum loss) in the objective value, after an element has been deleted, should be at most λ. Therefore, it will be convenient to hedge against regrets larger than λ by defining for each interval i_j a backup of weight at least $w(i_j) - \lambda$; this backup can then be used in the recourse if interval i_j fails. Further, we give a formal definition of a backup of an interval.

Defining backups. Let $\mathcal{F}(I)$ be a collection of sets of pairwise-disjoint intervals in the interval set I, and $S \in \mathcal{F}(I)$. We define, for each element $i_j \in I$, a *backup element* $\beta(i_j, S)$:

$$\beta(i_j, S) \in \underset{\substack{i_\ell \in I \setminus (S + i_j) \\ S - i_j + i_\ell \in \mathcal{F}(I)}}{\arg\max} w(i_\ell) \ .$$

Informally speaking, $\beta(i_j, S)$ is a best possible recourse that can be added to the remaining solution $S - i_j$ when i_j is interdicted. Hence, $\beta(i_j, S)$ serves as a backup for i_j.

Note that, for an interval $i_j \in S$, its backup $\beta(i_j, S) = i_{j'}$ may or may not intersect with i_j. Hence, the backup interval $i_{j'}$ of i_j satisfies one of the following two conditions:

(a) $S - i_j + i_{j'} \in \mathcal{F}(I)$, but $S + i_{j'} \notin \mathcal{F}(I)$, which occurs if $i_j \cap i_{j'} \neq \varnothing$, or

(b) $S + i_{j'} \in \mathcal{F}(I)$, when $i_j \cap i_{j'} = \varnothing$.

In case (a), interval $i_{j'}$ cannot be used as a backup for any other interval $i \neq i_j$, since $i_j \in S$ and $i_j \cap i_{j'} \neq \varnothing$. Hence, we call such a backup the *private* backup of interval i_j. In case (b), interval $i_{j'}$ can be used as a backup for any interval $i \in I \setminus \{i_{j'}\}$, since $i_{j'}$ does not intersect with any interval in S, and hence we call such a backup a *universal* backup. Note that since a universal backup, say u, can be used as a backup for any interval $i \in I - u$, it is enough to have one such backup for all intervals that might not have a private backup. For a first-stage solution set S, we want a universal backup u to be in $\arg\max\{w(i_{j'}) \mid i_{j'} \in I \setminus S \text{ and } S + i_{j'} \in \mathcal{F}(I)\}$. For the case where $\lambda < 0$ and u is interdicted, if there is no recourse, the value $\Delta(u, S) = 0$ is strictly larger than λ. This contradicts the feasibility of S. To deal with this issue, we choose an *extra* backup b along with our universal backup u. In particular, we aim to have $b \in \arg\max\{w(i_{j'}) \mid i_{j'} \in I \setminus (S + u) \text{ and } S + i_{j'} \in \mathcal{F}(I)\}$. Moreover, we have $w(u) \geq w(b)$. Note that, since we have $\varnothing \in I$, if there is no candidate interval to serve as a universal backup or an extra backup, we have $u = \varnothing$ or $b = \varnothing$, respectively. All in all, we solve an instance of the λ-RISch problem by first guessing a pair of universal and extra backup, $u, b \in I$, and then solving the problem.

Later in Subsection 5.6.3, we show that for the correct guess of a universal and extra backup, u and b values, the λ-RISch problem can be reduced to solving a more general interval scheduling problem, which we call Interval Scheduling with Colors (ISC). Below we give a formal definition of Interval Scheduling with Colors (ISC): An instance $\mathcal{I} = (I, w)$ of ISC consists of a set I of intervals and a weight function $w \colon I \to \mathbb{R}_+$. Each interval $i_j \in I$ potentially has two half open intervals $[\ell_j, r_j)$ and $[a_j, b_j)$ associated with it which satisfy $\ell_j \leq a_j < b_j \leq r_j$. The inner interval $[a_j, b_j)$ is *blue* and the interval $[\ell_j, r_j)$ is *green*. A feasible solution of this Interval Scheduling with Colors instance is a subset S of intervals such that for each pair of intervals $i_j, i_k \in S$ we have that $[a_j, b_j)$ and $[\ell_k, r_k)$ do not intersect and that $[a_k, b_k)$ and $[\ell_j, r_j)$ do not intersect. That is, the blue part of any interval in the solution is not allowed to intersect with the

green parts of any other interval in the solution. The task is to pick a feasible solution of maximum weight.

Altogether, we state the pseudocode of our algorithm for $(1,1)$-Robust Interval Scheduling in Algorithm $(1,1)$-RISch.

Algorithm $(1,1)$-RISch:

Input: A set of intervals $I = \{i_1, i_2, \ldots, i_n\}$, where $i_k = [a_k, b_k)$, $a_k < b_k$ for all $k \in [n]$, and a weight function $w\colon I \to \mathbb{R}_+$.

Output: Calculate the maximum achievable value of the second-stage solution, that is, for an optimal pairwise-disjoint subset $S \subseteq I$, calculate the maximum value of $w(S - i_j + i_\ell)$ over the deletion of any interval i_j and the best possible recourse by another interval i_ℓ thereafter.

```
Let Λ be a collection of all potential λ values.
Let B := I × I be the set consisting of all pairs of universal and extra backups.
opt^RISch(I) := 0                                   // optimal value of (1,1)-RISch
for λ ∈ Λ do
    for (u, b) ∈ B do
        Create an instance I_{u,b,λ} of ISC.
        Calculate opt^ISC(I_{u,b,λ}).               // optimal value of the ISC instance
        if opt^RISch(I) < opt^ISC(I_{u,b,λ}) − λ then
            opt^RISch(I) := opt^ISC(I_{u,b,λ}) − λ
return opt^RISch(I).
```

Before we describe our reduction from λ-Robust Interval Scheduling to Interval Scheduling with Colors, we first show, in Section 5.6.2, the Interval Scheduling with Colors problem is tractable.

5.6.2 Dynamic Program for Interval Scheduling with Colors

The Interval Scheduling with Colors problem is as follows:

Interval Scheduling with Colors (ISC)

Given: A set of intervals $I_{\text{ISC}} = \{i_1, i_2, \ldots, i_n\}$, where each interval $i_k = [\ell_k, r_k)$ consists of a blue part $[a_k, b_k)$ and a green part $[\ell_k, r_k)$ with $\ell_k \leq a_k < b_k \leq r_k$, $a_k, b_k, \ell_k, r_k \in \mathbb{R}_+$, and a weight function $w\colon I \to \mathbb{R}_+$.

Task: Find the maximum possible weight of a subset $S \subseteq I_{\text{ISC}}$ such that for each pair of intervals $i_j, i_k \in S$, $[a_j, b_j) \cap [\ell_k, r_k) = \varnothing$ and $[a_k, b_k) \cap [\ell_j, r_j) = \varnothing$.

Interval Scheduling with Colors is polynomial-time solvable and we give the following dynamic programming algorithm to solve it.

Theorem 5.20. *There is a polynomial-time algorithm for* Interval Scheduling with Colors.

Proof. Note that each interval in an instance of Interval Scheduling with Colors consists of a blue part and a green part. The green part is always a superset of the blue part. The task is to calculate the maximum weight of a subset of intervals $S \subseteq I_{\text{ISC}}$ such that the blue part of each interval in S does not intersect with the blue or green part of any other interval in S. Now, we extend the classical dynamic programming algorithm of the Interval Scheduling problem (see for example [Fra76] or Section 4.4.1) to the Interval Scheduling with Colors problem. We denote by $\mathcal{F}(I_{\text{ISC}})$ the set of feasible solutions within the interval set I_{ISC}.

Let $A := \{a_1, \ldots, a_n\}$ and $B := \{b_1, \ldots, b_n\}$ be the sets of starting and ending points of blue parts of all intervals, respectively. Furthermore, let $L := \{\ell_1, \ldots, \ell_n\}$ and $R := \{r_1, \ldots, r_n\}$ be the sets of starting and ending points of green parts of all intervals, respectively. Moreover, let $E := A \cup B \cup L \cup R$.

Let us define an ordering over the elements in sets B and R. Let π_b and π_r be the non-decreasing orderings over the elements of sets B and R, respectively, such that $\pi_b(i) = b^i$ and $\pi_r(i) = r^i$ for all $i \in [n]$.[5] This gives us $b^1 \leq b^2 \leq \ldots \leq b^n$ and $r^1 \leq r^2 \leq \ldots \leq r^n$. Moreover, we define $I_{t,t'} := \{i_k \in I_{\text{ISC}} \mid b_k \leq t \text{ and } r_k \leq t'\}$ for all $t, t' \in E$.

The idea behind our dynamic program is to always maintain the maximum possible weight of a subset of intervals within the set $I_{t,t'}$, where $t, t' \in E$. We store this value in the term $\text{opt}^{\text{ISC}}[t, t']$. Further we incrementally calculate the values of $\text{opt}^{\text{ISC}}[t, t']$ for larger t and t' values. Note that $\text{opt}^{\text{ISC}}[t, t'] = \max_{b \leq t, r \leq t'} \text{opt}^{\text{ISC}}[b, r]$. Due to this relation, it suffices to calculate the values $\text{opt}^{\text{ISC}}[b, r]$ for all $b \in B$ and $r \in R$. In the end, the solution for ISC lies in the term $\text{opt}^{\text{ISC}}[b^n, r^n]$.

Without loss of generality, we assume that the first interval in the ordering π_r is i_1, and i_1 has the maximum weight among the intervals that end at $r^1 = r_1$. For the base cases, we define $\text{opt}^{\text{ISC}}[b^i, r^1]$ values for all $i \in [n]$; we claim that $\text{opt}^{\text{ISC}}[b^i, r^1] = w(i_1)$. Since r^1 is the first endpoint of a green part in the instance I_{ISC}, $\arg\max_{i \in I_{b^i, r^1}} w(i) = i_1$. Thus we get

$$\text{opt}^{\text{ISC}}[b^i, r^1] = w(i_1) \quad \forall i \in [n].$$

Additionally, for all $t, t' < r^1$, $\text{opt}^{\text{ISC}}[t, t'] := 0$ because in that case $I_{t,t'} = \varnothing$.

Now we give a recurrence relation to calculate the values of $\text{opt}^{\text{ISC}}[b^i, r^j]$ when we have a priori all values of $\text{opt}^{\text{ISC}}[t, t']$ for all $t \leq b^i$ and $t' < r^j$. For that define $R_{t,r} := \{i_k \in I_{\text{ISC}} \mid b_k \leq t \text{ and } r_k = r\}$ for all $t \in E, r \in R$.
Then we have,

$$\text{opt}^{\text{ISC}}[b^i, r^j] := \max \Big\{ \text{opt}^{\text{ISC}}[b^i, r^{j-1}], \max_{i_k \in R_{b^i, r^j}} \big\{ \text{opt}^{\text{ISC}}[\ell_k, a_k] + w(i_k) \big\} \Big\}.$$

[5] Not to be confused with the notations b_i and r_i; the little i in the subscript corresponds to the interval $i \in I$, while the little i in the superscript corresponds to the index according to the orderings π_b and π_r.

This recurrence relation is true because the maximum weighted subset either picks an interval from the newly available intervals with green endpoint exactly r or it does not consist of any such intervals. Additionally, a feasible solution can take at most one interval from the set R_{b^i,r^j} because for all intervals $i_k \in R_{b^i,r^j}$, $r_k = r^j$. As a result, for each pair in R_{b^i,r^j}, the blue part of one interval must intersect with the green part of the other. Hence no two intervals in R_{b^i,r^j} are compatible. Since all these cases are considered in the recurrence relation, it is correct.

Further, we calculate the values starting from $b^i = b^1$ to $b^i = b^n$. As a result we get our final solution $\text{opt}^{\text{ISC}}[b^n, r^n]$. The pseudocode for this dynamic program can be found in Algorithm ISCDP.

It is straightforward to verify that the above dynamic programming algorithm returns the optimum objective value in polynomial time. □

In the next subsection, we provide a reduction from an instance of the λ-Robust Interval Scheduling problem to polynomial many instances of the Interval Scheduling with Colors problem. We further show that, we can find the optimum solution value for a given instance of λ-RISch by solving all the ISC instances obtained by choosing u and b.

5.6.3 The Algorithm for Bounded-Regret Interval Scheduling

We present an efficient algorithm for λ-Robust Interval Scheduling for a fixed value of λ. As discussed in Section 5.1, if we have a polynomial-time algorithm for the (λ-RP) problem, and if there are, in total, polynomial-many different λ values, then we have a polynomial-time algorithm for our original robust problem (RP). We use the dynamic programming algorithm that we obtained for Interval Scheduling with Colors in Subsection 5.6.2 as a subroutine in the algorithm for the λ-RISch problem.

Theorem 5.21. *Given a polynomial-time algorithm for* Interval Scheduling with Colors, *there is a polynomial-time algorithm for* λ-Robust Interval Scheduling.

Proof. To prove the theorem, we first describe the reduction from λ-Robust Interval Scheduling to Interval Scheduling with Colors (ISC) and then prove its correctness.

Let $\mathcal{I} = (I, w, \lambda)$ be an instance of λ-RISch. In the first step, we enumerate over all pairs of intervals $u, b \in I$ such that $w(u) \geq w(b) \geq -\lambda$, and corresponding to each such pair, we construct an instance $\mathcal{I}'_{u,b}(\lambda) = (I'_{u,b}(\lambda), w')$ of the Interval Scheduling with Colors problem (see Figure 5.5 for reference).

First, fix a pair $u, b \in I$ of universal backup and the extra backup. Note that there are three cases: (i) there are no universal and extra backups, (ii) there is no extra backup, but some universal backup, or (iii) there are both universal and extra backups.

For each interval $i_j \in I$, we add the following set of intervals in $I'_{u,b}$ with weights w', which cover all possible recourse actions in case i_j is deleted. We further define a

Algorithm ISCDP: Dynamic program for Interval Scheduling with Colors

Input: A set of intervals $I_{\text{ISC}} = \{i_1, i_2, \ldots, i_n\}$, where each $i_k = [\ell_k, r_k)$ consists of a blue part $[a_k, b_k)$ and a green part $[\ell_k, r_k)$ with $\ell_k \leq a_k < b_k \leq r_k$, and a weight function $w\colon I_{\text{ISC}} \to \mathbb{R}_+$.

Output: Find the total weight of maximum weighted subset $S \subseteq I_{\text{ISC}}$ such that for each distinct pair of intervals $i_j, i_k \in S$, $[a_j, b_j) \cap [\ell_k, r_k) = \varnothing$ and $[a_k, b_k) \cap [\ell_j, r_j) = \varnothing$.

$B \coloneqq \{b_1, b_2, \ldots, b_n\}$

$R \coloneqq \{r_1, r_2, \ldots, r_n\}$

$E \coloneqq B \cup R \cup \{a_1, a_2, \ldots, a_n\} \cup \{\ell_1, \ell_2, \ldots, \ell_n\}$

$R_{t,r} \coloneqq \{i_k \in I_{\text{ISC}} \mid b_k \leq t \text{ and } r_k = r\}$ for all $t \in E$, $r \in R$.

$I_{t,t'} \coloneqq \{i_k \in I_{\text{ISC}} \mid b_k \leq t \text{ and } r_k \leq t'\}$ for all $t, t' \in E$.

Let $\text{opt}^{\text{ISC}}[t, t']$ be the maximum weighted set in $\mathcal{F}(I_{t,t'})$ where $I_{t,t'} \subseteq I_{\text{ISC}}$.

Let π_b and π_r be non-decreasing orderings of all points in B and R, respectively.

$\pi_b(i) \coloneqq b^i$ and $\pi_r(j) \coloneqq r^j$

Let $b^0 \coloneqq 0$.

Let i_1 be the maximum weighted instance with $r_1 = r^1$.

$\text{opt}^{\text{ISC}}[b^i, r^1] \coloneqq w(i_1)$ for all $i \in [n]$. `// base cases`

For all $t, t' \in E$, $t' < r^1$, $\text{opt}^{\text{ISC}}[t, t'] \coloneqq 0$.

for $i \coloneqq 1$ **to** n **do**

  for $j \coloneqq 2$ **to** n **do**

    For all $t, t' \in E, b^{i-1} \leq t < b^i, r^{j-1} \leq t' < r^j, \text{opt}^{\text{ISC}}[t, t'] \coloneqq \text{opt}^{\text{ISC}}[b^{i-1}, r^{j-1}]$.

    For all $t' \in E, r^{j-1} \leq t' < r^j, \text{opt}^{\text{ISC}}[b^i, t'] \coloneqq \text{opt}^{\text{ISC}}[b^i, r^{j-1}]$.

    $$\text{opt}^{\text{ISC}}[b^i, r^j] \coloneqq \max\left\{ \text{opt}^{\text{ISC}}[b^i, r^{j-1}], \max_{i_k \in R_{b^i, r^j}} \left\{ \text{opt}^{\text{ISC}}[\ell_k, a_k] + w(i_k) \right\} \right\}$$

return $\text{opt}^{\text{ISC}}[b^n, r^n]$.

function $\lambda_{u,b}$ to model the regret and use this function to prune $I'_{u,b}$ in order to obtain $I'_{u,b}(\lambda)$.

- The first set of intervals covers the possibility of using the universal backup u as recourse. Thus, for each interval $i_j \in I \setminus \{u, b\}$ in λ-RISch, we add an interval i'_j to $I'_{u,b}$ in ISC with $w'(i'_j) = w(i_j)$. Moreover, the blue part of i'_j is set to $[a_j, b_j)$, which is the original interval i_j. Additionally, the green part of i'_j is also set to $[a_j, b_j)$. We set $\lambda_{u,b}(i'_j) = w(i_j) - w(u)$.

- The second set of intervals corresponds to the possibility of using a private backup. Thus, for each ordered pair of intervals (i_j, i_k) from set $I \setminus \{u, b\}$, $i_j \neq i_k$, such that i_j and i_k intersect, we add an interval i^k_j to $I'_{u,b}$. We set $w'(i^k_j) = w(i_j)$ and $\lambda_{u,b}(i^k_j) = w(i_j) - w(i_k)$. Let $\ell_{j,k} := \min\{a_j, a_k\}$ and $r_{j,k} := \max\{b_j, b_k\}$. Then we define the blue part of i^k_j to be $[a_j, b_j)$ and the green part to be $[\ell_{j,k}, r_{j,k})$.

This finishes the construction of the set $I'_{u,b}$.

Pruning: Finally we want to make sure that the blue part of an interval does not intersect with the intervals corresponding to the universal and extra backups. For that, we simply delete all those intervals from $I'_{u,b}$ which intersects with u and b. Further, we define $I'_{u,b}(\lambda) := \{i'_j \in I'_{u,b} \mid \lambda_{u,b}(i'_j) \leq \lambda\}$ to include all intervals of $I'_{u,b}$ that have a $\lambda_{u,b}$-value of at most λ. As a result we get our instance $\mathcal{I}'_{u,b}(\lambda) = (I'_{u,b}(\lambda), w')$, here w' is restricted to the set $I'_{u,b}(\lambda)$. See Figure 5.5 for reference. Let $\mathcal{F}(I'_{u,b}(\lambda))$ be the set of feasible solutions of $\mathcal{I}'_{u,b}(\lambda)$.

In the remainder of the proof, we show that an optimal solution $S^\star$ of $\mathcal{I}$ is in one-to-one correspondence with an optimal solution S of $\mathcal{I}'_{u,b}(\lambda)$ which maximizes the value $w'(S)$ over all $u, b \in I$. Moreover, the set S has the same weight and regret value as that of $S^\star$. To get an intuition, first observe that all intervals in $I'_{u,b}(\lambda)$ correspond to the intervals in I; we simply add the green parts to them. Green parts of intervals correspond to their respective backup intervals, and hence, they are allowed to intersect in a chosen solution of ISC. Furthermore, note for all intervals $i'_j \in I'_{u,b}(\lambda)$ we have $\lambda_{u,b}(i'_j) \leq \lambda$, which encodes that the regret we suffer if the corresponding interval i_j is interdicted is indeed at most λ. Now, it remains to show that from an optimum solution $S^\star$ of $\mathcal{I}$, we can compute an optimum solution S of $\mathcal{I}'_{u,b}(\lambda)$ of the same weight, and from an optimum solution S to $\mathcal{I}'_{u,b}(\lambda)$ we can compute an optimum solution $S^\star$ to $\mathcal{I}$ of the same weight and regret λ.

Let $S^\star$ be an optimum solution to $\mathcal{I}$. We now construct, in polynomial time, a solution to the corresponding instance $\mathcal{I}'_{u,b}(\lambda)$ of ISC of the same weight. By definition we have that $w(i_j) - w(\beta(i_j, S^\star)) \leq \lambda$. Without loss of generality, we assume that $S^\star$ does not contain u and b.

By construction of $\mathcal{I}'_{u,b}(\lambda)$, for each $i_m \in S^\star$ that uses the universal backup u when i_m fails, there is a new interval i'_m in $I'_{u,b}(\lambda)$ with $w'(i'_m) = w(i_m)$ and $\lambda_{u,b}(i'_m) = w(i_m) - w(u)$. Since $w(i_m) - w(\beta(i_m, S^\star)) \leq \lambda$, we have that $\lambda_{u,b}(i'_m) \leq \lambda$, and hence $i'_m \in I'_{u,b}(\lambda)$.

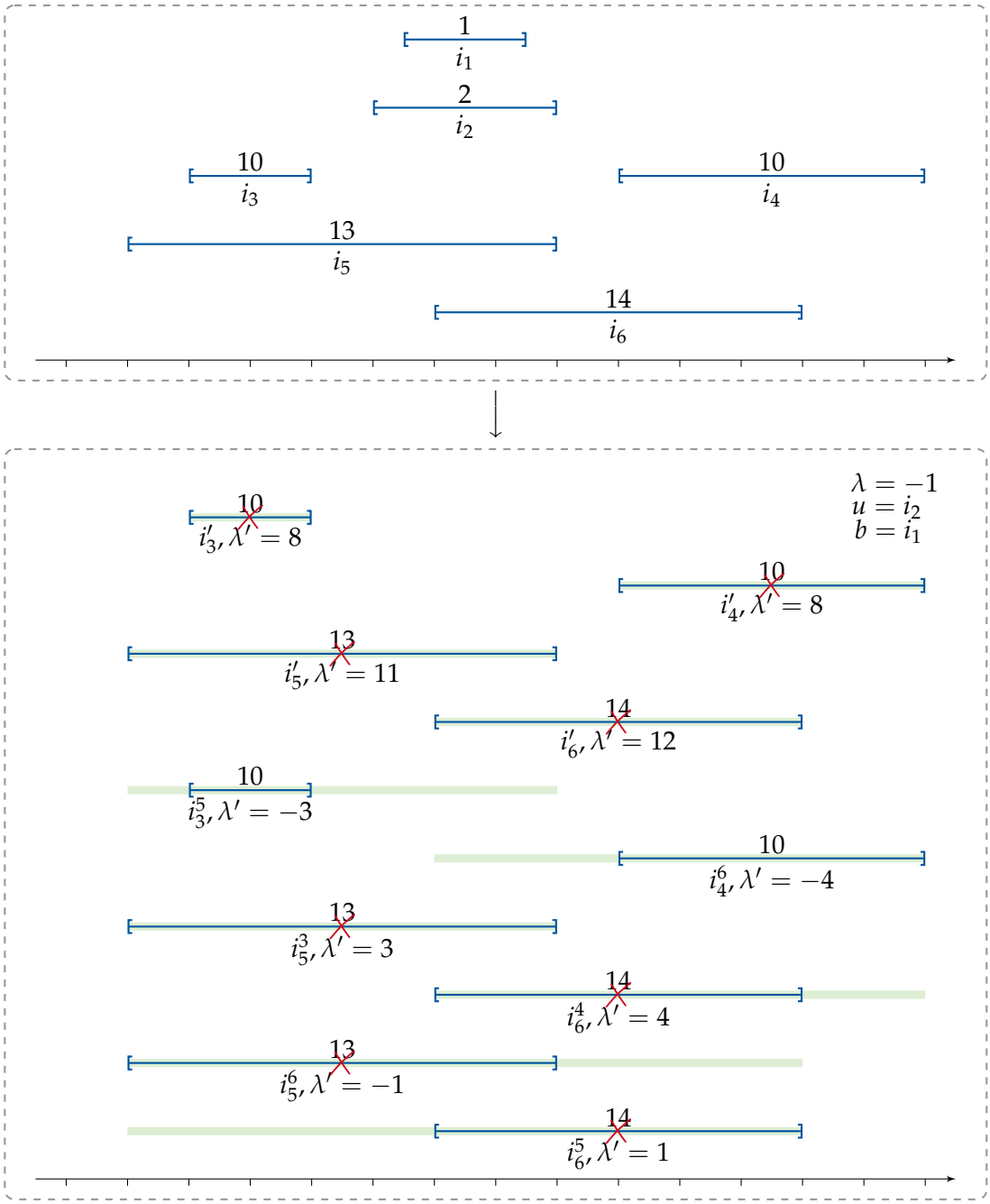


Figure 5.5: Reduction from $(1,1)$-Robust Interval Scheduling to Interval Scheduling with Colors. This example illustrates the reduction corresponding to the guessed values $u = i_2, b = i_1$, and $\lambda = -1$. The λ' values corresponds to the $\lambda_{u,b}$ values for each interval. The crossed intervals in the instance of ISC are the pruned intervals. The intervals $i'_5, i'_6, i^3_5, i^4_6, i^6_5$, and i^5_6 are pruned because their blue part overlaps with the universal backup. Additionally, the intervals i'_3 and i'_4 are pruned because their λ' values are greater than the current $\lambda = -1$ value.

We let

$$S_U \coloneqq \{i'_m \mid i_m \in S^\star \text{ and } i_m \text{ has universal backup}\} .$$

Now consider any interval $i_m \in S^\star$ that has a private backup $i_k \coloneqq \beta(i_m, S^\star)$. By construction of $I'_{u,b}(\lambda)$, there is an interval $i^k_m \in \mathcal{I}'_{u,b}(\lambda)$ with $w'(i^k_m) = w(i_m)$ and $\lambda_{u,b}(i^k_m) \leq w(i_m) - w(i_k) \leq \lambda$. Hence, i^k_m is contained in $I'_{u,b}(\lambda)$. We let

$$S_P \coloneqq \{i^k_m \mid i_m \in S^\star \text{ and } i_m \text{ has a private backup } i_k \coloneqq \beta(i_m, S^\star)\} .$$

Let $S = S_P \cup S_U$. We claim that S is a feasible solution to $\mathcal{I}'_{u,b}(\lambda)$. This directly follows from the definition of green and blue parts of the intervals: First, observe that all intervals in S are indeed contained in $I'_{u,b}(\lambda)$. Second, note that no two blue parts of intervals in S intersect, since $S^\star$ is a feasible solution before the failure of any interval. Third, observe that no green part intersects with a blue part of any other interval in S, as $S^\star - i_m + \beta(i_m, S^\star)$ is feasible for any interval $i_m \in S^\star$ and its backup $\beta(i_m, S^\star)$. Hence, S is indeed a feasible solution to $\mathcal{I}'_{u,b}(\lambda)$. Finally, observe that $w(S^\star) = w(S)$.

Next, we show how to construct, in polynomial time, a feasible solution to $\mathcal{I}$ from an optimum solution to $\mathcal{I}'_{u,b}(\lambda)$ with the same weight and regret λ. Recall that $u, b \in I$ are the universal and extra backups that maximize the optimum solution value of $\mathcal{I}'_{u,b}(\lambda)$, where $w(u) \geq w(b) \geq -\lambda$. Let S be an optimal solution to $\mathcal{I}'_{u,b}(\lambda)$. Similar to before, let

$$S'_U \coloneqq \{i'_m \in I'_{u,b} \mid i'_m \in S\}$$

be the set of all intervals in S that correspond to intervals without private backups. Furthermore, let

$$S'_P \coloneqq \{i^k_m \in I'_{u,b} \mid i^k_m \in S\}$$

be all intervals in S that do have a private backup.

We now construct a solution $S^\star$ to $\mathcal{I}$ as follows. First, for each $i'_m \in S'_U$ we add i_m to $S^\star$, where the (universal) backup of i_m is u. Note that by the feasibility of S we have $w(i_m) - w(u) \leq \lambda$. Second, for each $i^k_m \in S'_P$ we add i_m to $S^\star$, where the (private) backup of i_m is i_k. Again, note that by the feasibility of S we have $w(i_m) - w(i_k) \leq \lambda$. Now observe that $S^\star$ is a feasible solution to $\mathcal{I}$ since S is a feasible solution, and hence no blue parts intersect. Furthermore, since S was feasible to $\mathcal{I}'_{u,b}(\lambda)$, the universal or private backup of any interval can be added without violating feasibility.

Finally, observe that by construction of $\mathcal{I}'_{u,b}(\lambda)$ we have that

$$w(S) = w(S^\star) \text{ and } w(i_m) - w(\beta(i_m, S^\star) \leq \lambda \text{ for all } i_m \in S^\star,$$

since for all $i_m \in S'_U \cup S'_P$ we have $w(i_m) - w(\beta(i_m, S)) \leq \lambda$. Hence, the regret is at most λ if some element $i_m \in S^\star$ is interdicted. In case no element in $S^\star$ is interdicted, note that since $w(u) \geq w(b) \geq -\lambda$, we have that either u or b is not interdicted and hence can be added for recourse. Hence, in this case as well, we have a regret of at most λ. □

Finally, combining Proposition 5.19 with Theorem 5.21 and Theorem 5.20 directly implies Theorem 5.18.

5.6.4 Generalization to $(1, p)$-Robust Interval Scheduling

So far we have given an efficient algorithm for $(1,1)$-Robust Interval Scheduling problem. Our algorithm relies on solving the bounded regret version of the problem; we exploit the fact that for $(1,1)$-RISch, there are only $\mathcal{O}(n^2)$ many different values that λ can possibly realize. In this section, we extrapolate the same idea to the $(1,p)$-Robust Interval Scheduling problem for any constant p. Here, since the total number of different values that λ can realize is $\mathcal{O}(n^{p+1})$, we can again exploit the strong connection between robust version of a problem with its bounded-regret variant. For a particular λ value, however, we need to solve the bounded regret version of $(1,p)$-RISch problem which we refer to as λ-$(1,p)$-RISch. Recall from Theorem 5.21, in order to solve λ-RISch, we enumerated the total number of different ISC instances that we need to solve, which depends on the total number of different universal and extra backups. When the total number of ISC instances are polynomial many and ISC is polynomial-time solvable, we have an efficient algorithm for λ-RISch.

For an instance $\mathcal{I} = (I, w)$ of the generalized $(1,p)$-RISch problem, we now enumerate the total number of ISC instances that we need to generate in order to solve an instance of λ-$(1,p)$-RISch. Since we are allowed to add at most p intervals after the deletion of at most 1 interval, we have to be careful while defining the backups for an interval. Note that for the deletion of an interval, its p many backups could be a combination of universal backups and private backups. To capture this we define even more types of backups, namely *q-private backup*, *q-universal backup*, and *q-extra backup* for each $q \in [p]$; these new backups consist of q many pairwise-disjoint intervals.

Let us first define a private backup for each $i_j \in I$. Recall that, by definition, a private backup of an interval intersects the interval. Thus a q-private backup of interval i_j must be derived from the set $P^{i_j} := \{i_k \in I \mid [a_k, b_k) \cap [a_j, b_j) \neq \varnothing\}$ by selecting q many pairwise disjoint intervals from $P^{i_j} \setminus \{i_j\}$. Each P^{i_j}, $i_j \in I$, can be calculated in $\mathcal{O}(n)$ time. Further, each q-private backup of i_j can be calculated using the standard dynamic program for the interval scheduling problem which additionally keeps a track of the total number of intervals in the solution. Finally, for the interval representing a q-private backup of i_j in an instance of ISC, we can extract the endpoints of its green part in $\mathcal{O}(n)$ time. The total number of intervals corresponding to private backups is $n \cdot \sum_{q=1}^{p} \binom{n}{q}$, which is polynomial in the size of the input instance.

Now regarding the universal backups, for an interval with $(p-q)$-private backup we need a q-universal backup. Although, a unique q-universal backup does not work for all intervals with $(p-q)$-private backup. This is because it might happen that a q-universal backup intersects a $(p-q)$-private backup, and hence are not compatible to form a full p-sized backup together. However, each q-universal backup could only intersect with the private backups of at most $2q$ other intervals. Thus we have to reserve, in total, $2q+1$

many q-universal backups for each $q \in [p-1]$. Although, it suffices to have a unique p-universal backup because by definition it does not intersect with any interval with 0-private backup. All in all, for an instance of λ-$(1,p)$-RISch, we need to enumerate over $1+\sum_{q=1}^{p-1}(2q+1) \in \mathcal{O}(p^2)$ many total universal backups. Since a universal backup consists of at most p intervals, the total number of intervals we need to enumerate is $\mathcal{O}((n^p)^{p^2})$, which is polynomial for constant p.

Finally, we enumerate the total number of extra backups. Note that extra backups are used when an interval from the universal backup is deleted. Since we are dealing with deletion of at most 1 interval, for each interval that gets deleted from the unique p-universal backup, we need a p-extra backup.

Altogether, enumerating over all different kind of $p+1+\sum_{q=1}^{p-1}(2q+1) \in \mathcal{O}(p^2)$ many universal and extra backups results into an instance of ISC which can be solved in polynomial time. Moreover, since there are polynomial many such reduced instances of ISC, we get the following result.

Theorem 5.22. *There is a polynomial-time algorithm for* $(1,p)$-Robust Interval Scheduling *for an arbitrary constant* p.

5.7 Concluding Remarks and Future Directions

We have introduced a new model called *Recoverable Robust Optimization with Commitment* and initiated the first study on this model. We started by investigating the model on various combinatorial optimization problems; we settle the computational complexity of our considered problems (both tractability and intractability). While our focus was on polynomial-time solvable problems with a downward-closed feasibility set, it would be interesting to extend the study to a broader class of problems, including NP-hard problems. We note that problems whose feasibility set is not downward-closed pose an additional challenge that the adversarial deletion might destroy the feasibility of the remaining partial solution, which is then required to be repaired by the recourse action.

Our results only scratched the surface of our model. We started off considering the basic $(1,1)$-RP variants of several problems and proved them to be hard even for this simple case. However, it would be interesting to see how the computation complexity unravels for the general variant of the problem with arbitrary valued k and ℓ. We speculate the complexity, for the general variant of a problem, could shoot up to Σ_2^{p}-hard or even to a higher level of the polynomial hierarchy.

In the general variants, we gave an efficient algorithm for (k,ℓ)-Robust Matroid problem when $k \leq \ell$. But as our results show, this algorithm does not work when $k > \ell$. It still remains to show what happens to (k,ℓ)-RM when $k > \ell$, whether the problem remains polynomial-time solvable or turns hard. Similarly, even though we have efficient algorithms for $(1,p)$-Robust Interval Scheduling for a constant value p,

the complexity status of (k, ℓ)-Robust Interval Scheduling and also the (k, ℓ)-Robust Independent Set remains open.

Finally, the bounded-regret version was crucial in order to solve the Robust Interval Scheduling problem. So, naturally, one could look into whether the bounded-regret version of other problems can be solved in polynomial time as well.

With an additional twist, one could study yet another variant of this model in which cancellations/deleted elements are charged. For example, given two weight functions $w, w' : E \to \mathbb{R}_+$, solve the following variant:

$$\max_{S \in \mathcal{F}} \min_{\substack{D \subseteq E \\ |D| \leq k}} \max_{\substack{R \subseteq E \setminus D,\ |R| \leq \ell \\ (S \setminus D) \cup R \in \mathcal{F}}} w((S \setminus D) \cup R) + w'(D) \ .$$

This variant is applicable in scenarios where every reservation comes with a base price for cancellations.

List of Figures

List of Tables

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, USA, 1st edition, 2009.

[AHM22] David Adjiashvili, Felix Hommelsheim, and Moritz Mühlenthaler. Flexible graph connectivity: approximating network design problems between 1-and 2-connectivity. *Mathematical Programming*, 192(1-2):409–441, 2022.

[AKR+16] Noga Alon, Alexandr Kostochka, Benjamin Reiniger, Douglas B. West, and Xuding Zhu. Coloring, sparseness and girth. *Israel Journal of Mathematics*, 214(1):315–331, 2016.

[ASZ15] David Adjiashvili, Sebastian Stiller, and Rico Zenklusen. Bulk-robust combinatorial optimization. *Mathematical Programming*, 149:361–390, 2015.

[BC93] Adam L. Buchsbaum and Martin C. Carlisle. Determining uni-connectivity in directed graphs. *Information Processing Letters*, 48(1):9–12, 1993.

[BCHI24] Sylvia C. Boyd, Joseph Cheriyan, Arash Haddadan, and Sharat Ibrahimpur. Approximation algorithms for flexible graph connectivity. *Mathematical Programming*, 204(1):493–516, 2024.

[BEGN09] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*. Princeton Series in Applied Mathematics. Princeton University Press, 2009.

[BG01] Oleg Veniaminovich Borodin and Aleksei Nikolaevich Glebov. On the partition of a planar graph of girth 5 into an empty and an acyclic subgraph. *Diskretnyi Analiz i Issledovanie Operatsii*, 8(4):34–53, 2001.

[BG08] Jørgen Bang-Jensen and Gregory Z. Gutin. *Digraphs: theory, algorithms and applications*. Springer Science & Business Media, 2008.

[BG10] Dimitris Bertsimas and Vineet Goyal. On the power of robust solutions in two-stage stochastic and adaptive optimization problems. *Mathematics of Operations Research*, 35(2):284–305, 2010.

[BG22] Matthew Bold and Marc Goerigk. Investigating the recoverable robust single machine scheduling problem under interval uncertainty. *Discrete Applied Mathematics*, 313:99–114, 2022.

[BGGN04] Aharon Ben-Tal, Alexander Goryashko, Elana Guslitzer, and Arkadi Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376, 2004.

[BGKK19] Christina Büsing, Sebastian Goderbauer, Arie M. C. A. Koster, and Manuel Kutschka. Formulations and algorithms for the recoverable Γ-robust knapsack problem. *EURO J. Comput. Optim.*, 7(1):15–45, 2019.

[BH22] Christoph Buchheim and Dorothee Henke. The robust bilevel continuous knapsack problem with uncertain coefficients in the follower's objective. *Journal of Global Optimization*, 83(4):803–824, 2022.

[BHH21] Christoph Buchheim, Dorothee Henke, and Felix Hommelsheim. On the complexity of robust bilevel optimization with uncertain follower's objective. *Operations Research Letters*, 49(5):703–707, 2021.

[BHH22] Christoph Buchheim, Dorothee Henke, and Felix Hommelsheim. On the complexity of the bilevel minimum spanning tree problem. *Networks*, 80(3):338–355, 2022.

[BK18] Christoph Buchheim and Jannis Kurtz. Robust combinatorial optimization under convex and discrete cost uncertainty. *EURO Journal on Computational Optimization*, 6(3):211–238, 2018.

[BKK11a] Christina Büsing, Arie M. C. A. Koster, and Manuel Kutschka. Recoverable robust knapsacks: Γ-scenarios. In *inoc*, volume 6701 of *Lecture Notes in Computer Science*, pages 583–588. Springer, 2011.

[BKK11b] Christina Büsing, Arie M. C. A. Koster, and Manuel Kutschka. Recoverable robust knapsacks: the discrete scenario case. *Optim. Lett.*, 5(3):379–392, 2011.

[BKS98] Amotz Bar-Noy, Samir Khuller, and Baruch Schieber. *The complexity of finding most vital arcs and nodes*. Digital Repository at the University of Maryland, 1998.

[BM76] John Adrian Bondy and Uppaluri Siva Ramachandra Murty. *Graph theory with applications*, volume 290. Macmillan London, 1976.

[BM86] Hans-Jürgen Bandelt and Henry M. Mulder. Distance-hereditary graphs. *The Journal of Combinatorial Theory, Series B*, 41(2):182–208, 1986.

[BMS08] Luce Brotcorne, Patrice Marcotte, and Gilles Savard. Bilevel programming: The Montreal school. *INFOR: Information Systems and Operational Research*, 46(4):231–246, 2008.

[Bol98] Béla Bollobás. *Modern graph theory*, volume 184. Springer Science & Business Media, 1998.

[Bru69] Richard A. Brualdi. Comments on bases in dependence structures. *Bulletin of the Australian Mathematical Society*, 1(2):161–167, 1969.

[BT80] Frank Boesch and Ralph Tindell. Robbins's theorem for mixed multigraphs. *The American Mathematical Monthly*, 87(9):716–719, 1980.

[Büs12] Christina Büsing. Recoverable robust shortest path problems. *Networks*, 59(1):181–189, 2012.

[CCLW13] Alberto Caprara, Margarida Carvalho, Andrea Lodi, and Gerhard J. Woeginger. A complexity and approximability study of the bilevel knapsack problem. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 98–109. Springer, 2013.

[CCLW14] Alberto Caprara, Margarida Carvalho, Andrea Lodi, and Gerhard J. Woeginger. A study on the computational complexity of the bilevel knapsack problem. *SIAM Journal on Optimization*, 24(2):823–838, 2014.

[CG16] André B. Chassein and Marc Goerigk. On the recoverable robust traveling salesman problem. *Optimization letters*, 10(7):1479–1492, 2016.

[CGT85] Fan R.K. Chung, Michael R. Garey, and Robert E. Tarjan. Strongly connected orientations of mixed multigraphs. *Networks*, 15(4):477–484, 1985.

[CJ23] Chandra Chekuri and Rhea Jain. Approximation algorithms for network design in non-uniform fault models. In *ICALP*, volume 261 of *LIPIcs*, pages 36:1–36:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[CLM18] Margarida Carvalho, Andrea Lodi, and Patrice Marcotte. A polynomial algorithm for a continuous bilevel knapsack problem. *Operations Research Letters*, 46(2):185–188, 2018.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. The MIT Press and McGraw-Hill Book Company, 2001.

[CMRS15] Lin Chen, Nicole Megow, Roman Rischke, and Leen Stougie. Stochastic and robust scheduling in the cloud. In *approx-random*, volume 40 of *LIPIcs*, pages 175–186. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[CMS05] Benoît Colson, Patrice Marcotte, and Gilles Savard. Bilevel programming: A survey. *4OR*, 3:87–107, 06 2005.

[CRST06] Maria Chudnovsky, Neil Robertson, Paul Seymour, and Robin Thomas. The strong perfect graph theorem. *Annals of mathematics*, pages 51–229, 2006.

[Dem79] Robert W. Deming. Independence numbers of graphs-an extension of the Kőnig-Egerváry theorem. *Discrete Mathematics*, 27(1):23–33, 1979.

[Dem02] Stephan Dempe. Foundations of bilevel programming. 2002.

[Dem03] Stephan Dempe. Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints. *Optimization*, 52(3):333–359, 2003.

[Dem20] Stephan Dempe. Bilevel optimization: theory, algorithms, applications and a bibliography. In *Bilevel optimization: advances and next challenges*, pages 581–672. Springer, 2020.

[DGRS05] Kedar Dhamdhere, Vineet Goyal, Ramamoorthi Ravi, and Mohit Singh. How to pay, come what may: approximation algorithms for demand-robust covering problems. In *focs 2005*, pages 367–376. IEEE, 2005.

[DHP01] Guillaume Damiand, Michel Habib, and Christophe Paul. A simple paradigm for graph recognition: application to cographs and distance hereditary graphs. *Theoretical Computer Science*, 263(1-2):99–111, 2001.

[Die12] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[DJ15] Martin Dietzfelbinger and Raed Jaberi. On testing single connectedness in directed graphs and some related problems. *Information Processing Letters*, 115(9):684–688, 2015.

[DKM+22] Avinandan Das, Lawqueen Kanesh, Jayakrishnan Madathil, Komal Muluk, Nidhi Purohit, and Saket Saurabh. On the complexity of singly connected vertex deletion. *Theoretical Computer Science*, 934:47–64, 2022.

[DKPK15] Stephan Dempe, Vyacheslav Kalashnikov, Gerardo A Pérez-Valdés, and Nataliya Kalashnykova. Bilevel programming problems. *Energy Systems. Springer, Berlin*, 10(978-3):53–56, 2015.

[DMP+15] Mitre C. Dourado, Dirk Meierling, Lucia D. Penso, Dieter Rautenbach, Fabio Protti, and Aline Ribeiro de Almeida. Robust recoverable perfect matchings. *Networks*, 66(3):210–213, 2015.

[Edm65] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[Erd59] Paul Erdős. Graph theory and probability. *Canadian Journal of Mathematics*, 11:34–38, 1959.

[Fey05] Richard P. Feynman. *The pleasure of finding things out: The best short works of Richard P. Feynman*. Basic Books, 2005.

[FGH+17] Satoru Fujishige, Michel X. Goemans, Tobias Harks, Britta Peis, and Rico Zenklusen. Matroids are immune to braess' paradox. *Mathematics of Operations Research*, 42(3):745–761, 2017.

[FHLW21] Dennis Fischer, Tim A. Hartmann, Stefan Lendl, and Gerhard J. Woeginger. An investigation of the recoverable robust assignment problem. In *IPEC 2021*, volume 214 of *LIPIcs*, pages 19:1–19:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[FJMM07] Uriel Feige, Kamal Jain, Mohammad Mahdian, and Vahab Mirrokni. Robust combinatorial optimization with exponential scenarios. In *ipco 2007*, pages 439–453. Springer, 2007.

[FMW22] Dennis Fischer, Komal Muluk, and Gerhard J. Woeginger. A note on the complexity of the bilevel bottleneck assignment problem. *4OR*, 20(4):713–718, 2022.

[Fra76] Andras Frank. Some polynomial algorithms for certain graphs and hypergraphs. Proc. 5th Br. comb. Conf., Aberdeen 1975, 211-226 (1976)., 1976.

[FS99] Greg N. Frederickson and Roberto Solis-Oba. Increasing the weight of minimum spanning trees. *Journal of Algorithms*, 33(2):244–266, 1999.

[FW20] Dennis Fischer and Gerhard J. Woeginger. A faster algorithm for the continuous bilevel knapsack problem. *Operations Research Letters*, 48(6):784–786, 2020.

[GH24] Marc Goerigk and Michael Hartisch. An introduction to robust combinatorial optimization. *International Series in Operations Research and Management Science*, 2024.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.

[GK09] Elisabeth Gassner and Bettina Klinz. The computational complexity of bilevel assignment problems. *4OR*, 7(4):379–394, 2009.

[GKST25] Marc Goerigk, Jannis Kurtz, Martin Schmidt, and Johannes Thürauf. Connections between Robust and Bilevel Optimization. *Open Journal of Mathematical Optimization*, 6:1–17, 2025.

[GLW21] Marc Goerigk, Stefan Lendl, and Lasse Wulf. On the recoverable traveling salesman problem. *arXiv preprint arXiv:2111.09691*, 2021.

[GLW22] Marc Goerigk, Stefan Lendl, and Lasse Wulf. Recoverable robust representatives selection problems with discrete budgeted uncertainty. *European Journal of Operational Research*, 303(2):567–580, 2022.

[GNR14] Anupam Gupta, Viswanath Nagarajan, and Ramamoorthi Ravi. Thresholded covering algorithms for robust and max–min optimization. *Mathematical Programming*, 146(1-2):583–615, 2014.

[Grö59] Herbert Grötzsch. Ein dreifarbensatz für dreikreisfreie netze auf der kugel. *Wiss. Z. Martin Luther Univ. Halle-Wittenberg, Math. Nat. Reihe*, 8:109–120, 1959.

[GW25] Christoph Grüne and Lasse Wulf. Completeness in the polynomial hierarchy for many natural problems in bilevel and robust optimization. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 256–269. Springer, 2025.

[Hen25] Dorothee Henke. The robust bilevel selection problem. *Open Journal of Mathematical Optimization*, 6:1–35, 2025.

[HFG24] Omar El Housni, Ayoub Foussoul, and Vineet Goyal. lp-based approximations for disjoint bilinear and two-stage adjustable robust optimization. *Mathematical Programming*, 206(1):239–281, 2024.

[HGHS21] Omar El Housni, Vineet Goyal, Oussama Hanguir, and Clifford Stein. Matching drivers to riders: a two-stage robust approach. In *approx-random*, volume 207 of *LIPIcs*, pages 12:1–12:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[HGS21] Omar El Housni, Vineet Goyal, and David B. Shmoys. On the power of static assignment policies for robust facility location problems. In *ipco*, volume 12707 of *Lecture Notes in Computer Science*, pages 252–267. Springer, 2021.

[HKZ17a] Mikita Hradovich, Adam Kasperski, and Paweł Zieliński. Recoverable robust spanning tree problem under interval uncertainty representations. *Journal of Combinatorial Optimization*, 34:554–573, 2017.

[HKZ17b] Mikita Hradovich, Adam Kasperski, and Pawel Zielinski. The recoverable robust spanning tree problem with interval costs is polynomially solvable. *Optim. Lett.*, 11(1):17–30, 2017.

[HM90] Peter L. Hammer and Frédéric Maffray. Completely separable graphs. *Discrete Applied Mathematics*, 27(1-2):85–99, 1990.

[HM23a] Tim A. Hartmann and Komal Muluk. Make a graph singly connected by edge orientations. In *International Workshop on Combinatorial Algorithms*, pages 221–232. Springer, 2023.

[HM23b] Tim A. Hartmann and Komal Muluk. Make a graph singly connected by edge orientations, 2023.

[HMMP23] Felix Hommelsheim, Nicole Megow, Komal Muluk, and Britta Peis. Recoverable robust optimization with commitment. *arXiv preprint arXiv:2306.08546*, 2023.

[HMMP26] Felix Hommelsheim, Nicole Megow, Komal Muluk, and Britta Peis. Recoverable robust optimization with commitment. *Mathematical Programming*, 2026.

[HMS21] Felix Hommelsheim, Moritz Mühlenthaler, and Oliver Schaudt. How to secure matchings against edge failures. *SIAM Journal on Discrete Mathematics*, 35(3):2265–2292, 2021.

[HW25] Dorothee Henke and Lasse Wulf. On the complexity of the bilevel shortest path problem. *Networks*, 86(4):428–445, 2025.

[IKK+22] Takehiro Ito, Naonori Kakimura, Naoyuki Kamiyama, Yusuke Kobayashi, and Yoshio Okamoto. A parameterized view to the robust recoverable base problem of matroids under structural uncertainty. *Operations Research Letters*, 50(3):370–375, 2022.

[Jer85] Robert G. Jeroslow. The polynomial hierarchy and a simple model for competitive analysis. *Mathematical programming*, 32(2):146–164, 1985.

[JKZ24a] Marcel Jackiewicz, Adam Kasperski, and Pawel Zielinski. Computational complexity of the recoverable robust shortest path problem with discrete recourse. *CoRR*, abs/2403.20000, 2024.

[JKZ24b] Marcel Jackiewicz, Adam Kasperski, and Pawel Zielinski. Recoverable robust shortest path problem under interval uncertainty representations. *Networks*, 85(1):127–141, 2024.

[Joh87] David S. Johnson. The NP-completeness column: An ongoing guide. *Journal of Algorithms*, 8(3):438–448, 1987.

[Joh11] Berit Johannes. *New classes of complete problems for the second level of the polynomial hierarchy*. PhD thesis, Technische Universität Berlin, 2011.

[Kar09] Richard M. Karp. Reducibility among combinatorial problems. In *50 Years of Integer Programming 1958-2008: from the Early Years to the State-of-the-Art*, pages 219–241. Springer, 2009.

[Khu99] Samir Khuller. An $\mathcal{O}(|V|^2)$ algorithm for single connectedness. *Information Processing Letters*, 72(3):105–107, 1999.

[KKMS13] Rohit Khandekar, Guy Kortsarz, Vahab Mirrokni, and Mohammad R Salavatipour. Two-stage robust network design with exponential scenarios. *Algorithmica*, 65:391–408, 2013.

[KL95] Ker-I Ko and Chih-Long Lin. On the complexity of min-max optimization problems and their approximation. In *Minimax and Applications*, pages 219–239. Springer, 1995.

[KLN91] Jan Kratochvíl, Anna Lubiw, and Jaroslav Nesetril. Noncrossing subgraphs in topological layouts. *SIAM Journal on Discrete Mathematics*, 4(2):223–244, 1991.

[KMMS21] Lawqueen Kanesh, Soumen Maity, Komal Muluk, and Saket Saurabh. Parameterized complexity of fair feedback vertex set problem. *Theoretical Computer Science*, 867:1–12, 2021.

[Kru56] Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[Kuh55] Harold W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[Kur30] Casimir Kuratowski. Sur le probleme des courbes gauches en topologie. *Fundamenta mathematicae*, 15(1):271–283, 1930.

[KY97] Panos Kouvelis and Gang Yu. *Robust discrete optimization and its applications*. Nonconvex optimization and its applications, v. 14. Kluwer Academic Publishers, Dordrecht, 1997.

[Lan51] Hyman G. Landau. On dominance relations and the structure of animal societies: I. effect of inherent characteristics. *The bulletin of mathematical biophysics*, 13(1):1–19, 1951.

[Lan53] Hyman G. Landau. On dominance relations and the structure of animal societies. iii. the condition for a score structure. *The bulletin of mathematical biophysics*, 15(2):143–148, 1953.

[Lic82] David Lichtenstein. Planar formulae and their uses. *SIAM journal on computing*, 11(2):329–343, 1982.

[LLMS09] Christian Liebchen, Marco Lübbecke, Rolf Möhring, and Sebastian Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. *Robust and online large-scale optimization: Models and techniques for transportation systems*, pages 1–27, 2009.

[LLW21] Thomas Lachmann, Stefan Lendl, and Gerhard J. Woeginger. A linear time algorithm for the robust recoverable selection problem. *Discrete Applied Mathematics*, 303:94–107, 2021.

[Lov68] László Lovász. On chromatic number of finite set-systems. *Acta Mathematica Academiae Scientiarum Hungarica*, 19(1):59–67, 1968.

[LPT21] Stefan Lendl, Britta Peis, and Veerle Timmermans. Matroid bases with cardinality constraints on the intersection. *Mathematical Programming*, pages 1–24, 2021.

[MAVH12] Raid Mansi, Claudio Alves, JM Valério de Carvalho, and Saïd Hanafi. An exact algorithm for bilevel 0-1 knapsack problems. *Mathematical Problems in Engineering*, 2012(1):504713, 2012.

[MS72] Albert R. Meyer and Larry J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *SWAT*, volume 72, pages 125–129, 1972.

[Mul19] Komal Muluk. Parameterized complexity of fair feedback vertex set problem. Master's thesis, Indian Institute of Science Education and Research, Pune, 2019.

[Mul26] Komal Muluk. On the complexity of bilevel independent set problem. *arXiv preprint arXiv:2605.25927*, 2026.

[Mun57] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

[Oxl06] James G Oxley. *Matroid theory*, volume 3. Oxford University Press, USA, 2006.

[Pap03] Christos H. Papadimitriou. *Computational complexity*, page 260–265. John Wiley and Sons Ltd., GBR, 2003.

[Rob39] Herbert Ellis Robbins. A theorem on graphs, with an application to a problem of traffic control. *The American Mathematical Monthly*, 46(5):281–283, 1939.

[Sch03] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*. Springer, 2003.

[SPR23] Xueyu Shi, Oleg A. Prokopyev, and Ted K. Ralphs. Mixed integer bilevel optimization with a *k*-optimal follower: a hierarchy of bounds. *Mathematical Programming Computation*, 15(1):1–51, 2023.

[Sta34] Heinrich von Stackelberg. Marktform und Gleichgewicht. *Springer, Berlin [English translation:* The theory of market economy*, Oxford University Press, 1952]*, 1934.

[Sto76] Larry J Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.

[SZP19] Xueyu Shi, Bo Zeng, and Oleg A. Prokopyev. On bilevel minimum and bottleneck spanning tree problems. *Networks*, 74(3):251–273, 2019.

[Uma99] Christopher Umans. Hardness of approximating/spl sigma//sub 2//sup p/minimization problems. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 465–474. IEEE, 1999.

[Wes01] Douglas Brent West. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.

[Woe21] Gerhard J. Woeginger. The trouble with the second quantifier. *4OR*, 19(2):157–181, 2021.

[Wra76] Celia Wrathall. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):23–33, 1976.

[ZdHS18] Jianzhe Zhen, Dick den Hertog, and Melvyn Sim. Adjustable robust optimization via fourier-motzkin elimination. *Operations Research*, 66(4):1086–1100, 2018.

[Zen10] Rico Zenklusen. Matching interdiction. *Discrete Applied Mathematics*, 158(15):1676–1690, 2010.

[Zen15] Rico Zenklusen. An $\mathcal{O}(1)$-approximation for minimum spanning tree interdiction. In *2015 IEEE 56th annual symposium on foundations of computer science*, pages 709–728. IEEE, 2015.

[Zha25] Ruiyang Zhang. Recoverable robust maximum weighted forest problem with commitment. Master's thesis, RWTH Aachen University, 2025.

[ZRP+09] Rico Zenklusen, Bernard Ries, Christophe Picouleau, Dominique De Werra, Marie-Christine Costa, and Cédric Bentz. Blockers and transversals. *Discrete Mathematics*, 309(13):4306–4314, 2009.