Karlsruhe Institute of Technology

Complexity of Zero Forcing

Bachelor's Thesis of

Tim Wellenreich

At the Department of Informatics
Institute of Theoretical Informatics (ITI)

Reviewer:	T.T.-Prof. Dr. Thomas Bläsius
Second reviewer:	PD Dr. Torsten Ueckerdt
Advisor:	Max Göttlicher

June 24, 2025 – October 24, 2025

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself. I have not used any other than the aids that I have mentioned. I have marked all parts of the thesis that I have included from referenced literature, either in their original wording or paraphrasing their contents. I have followed the by-laws to implement scientific integrity at KIT.

Karlsruhe, October 24, 2025

. . . . . .

(Tim Wellenreich)

Abstract

The problem Zero Forcing asks for a minimum zero forcing set of an undirected graph $G = (V, E)$. A zero forcing set is a subset of vertices $S \subseteq V$ such that all vertices of G can be colored blue by repeatedly applying the *color-change rule*, starting with only the vertices in S colored blue. The color-change rule states that if a blue vertex has exactly one white neighbor, then this neighbor becomes blue. The size of a minimum zero forcing set is called the *zero forcing number* of G, denoted by $Z(G)$. Applying the color-change rule induces directed paths known as *forcing chains*.

In this thesis, we compare the zero forcing number with the induced path partition number and show that the zero forcing number can exceed the induced path partition number of a graph by an arbitrarily large factor. Furthermore, we present a new characterization of forcing chains that makes it possible to determine whether the paths in a given set of directed induced paths are the forcing chains of a zero forcing set. Using this characterization, we prove that for every non-isolated vertex in a graph, there exists a zero forcing set that does not contain this vertex. We also show that it is NP-hard to decide whether there exists an orientation of the undirected paths in an induced path partition such that the resulting directed paths are forcing chains of a zero forcing set. Moreover, we demonstrate that Zero Forcing remains NP-hard on undirected bipartite graphs with maximum degree three. In addition, we prove that for every undirected graph with n vertices and m edges for which a zero forcing set of size k exists, the inequality $m \leq k \cdot \left(n - \frac{k+1}{2}\right)$ must hold. We present a linear-time algorithm for solving Zero Forcing on cacti and discuss recently introduced FPT algorithms for Zero Forcing. We also extend the definition of Zero Forcing to directed graphs and prove that Zero Forcing remains NP-hard on DAGs.

Zusammenfassung

Das Problem Zero Forcing befasst sich mit der Bestimmung einer minimalen Zero-Forcing-Menge eines ungerichteten Graphen $G = (V, E)$. Eine Zero-Forcing-Menge ist eine Teilmenge der Knoten $S \subseteq V$, sodass alle Knoten von G blau gefärbt werden können, indem wiederholt die *Farbänderungsregel* angewendet wird, wobei anfänglich nur die Knoten in S blau gefärbt sind. Die Farbänderungsregel besagt, dass ein blauer Knoten, der genau einen weißen Nachbarn hat, diesen Nachbarn blau färbt. Die Größe einer minimalen Zero-Forcing-Menge wird als *Zero-Forcing-Zahl* von G bezeichnet und mit $Z(G)$ notiert. Die Anwendung der Farbänderungsregel erzeugt gerichtete Pfade, die als *Forcing-Ketten* bezeichnet werden.

In dieser Arbeit vergleichen wir die Zero-Forcing-Zahl mit der induzierten Pfad-Partitionszahl und zeigen, dass die Zero-Forcing-Zahl die induzierte Pfad-Partitionszahl eines Graphen um einen beliebig großen Faktor übersteigen kann. Außerdem stellen wir eine neue Charakterisierung von Forcing-Ketten vor, die es ermöglicht zu bestimmen, ob die Pfade in einer gegebenen Menge gerichteter induzierter Pfade die Forcing-Ketten einer Zero-Forcing-Menge sind. Mithilfe dieser Charakterisierung beweisen wir, dass es für jeden nicht-isolierten Knoten in einem Graphen eine Zero-Forcing-Menge gibt, die diesen Knoten nicht enthält. Weiterhin zeigen wir, dass es NP-schwer ist zu entscheiden, ob es eine Orientierung der ungerichteten Pfade in einer induzierten Pfad-Partition gibt, sodass die resultierenden gerichteten Pfade die Forcing-Ketten einer Zero-Forcing-Menge bilden. Zudem zeigen wir, dass Zero Forcing auch auf ungerichteten bipartiten Graphen mit maximalem Grad drei NP-schwer bleibt. Darüber hinaus beweisen wir, dass für jeden ungerichteten Graphen mit n Knoten und m Kanten, für

den eine Zero-Forcing-Menge der Größe k existiert, die Ungleichung $m \leq k \cdot \left(n - \frac{k+1}{2}\right)$ gilt. Wir präsentieren einen linearen Algorithmus zur Lösung von Zero Forcing auf Kakteen und diskutieren kürzlich eingeführte FPT-Algorithmen für Zero Forcing. Schließlich erweitern wir die Definition von Zero Forcing auf gerichtete Graphen und beweisen, dass Zero Forcing auch auf DAGs NP-schwer bleibt.

Contents

1 Introduction

The problem Zero Forcing asks for the smallest subset of vertices $S \subseteq V$ in an undirected graph $G = (V, E)$ such that all vertices can be colored blue by repeatedly applying the *color-change rule*. Initially, only the vertices in S are colored blue, while all remaining vertices in $V \setminus S$ are white. The color-change rule states that if a blue vertex has exactly one white neighbor, then this neighbor becomes blue.

1.1 Related Work

The problem Zero Forcing problem was first introduced by Burgarth et al. in the context of quantum mechanics [BG07] and independently by the AIM Minimum Rank group in graph theory as a tool to bound the minimum rank of graphs [Gro+08]. Since then, Zero Forcing has found applications across various domains. It has been used to model information diffusion in social networks [Bri+25], to study the controllability of quantum systems [Bur+13], and in linear algebra, where the zero forcing number provides an upper bound on the maximum nullity of a graph [Bar+10].

Computing the zero forcing number of a graph is known to be NP-hard for both undirected graphs [FMY16] and loop-directed graphs [TD15]. Several approaches have been proposed for characterizing graphs with a given zero forcing number or for bounding it using other graph parameters. The zero forcing number $Z(G)$ of an undirected graph $G = (V, E)$ satisfies $Z(G) = 1$ if and only if G is a path, and $Z(G) = 2$ if and only if G consists of two parallel paths [BFH19]. Graphs with $Z(G) = 3$ and maximum degree at most three have also been characterized [ARS20]. Moreover, $Z(G) = |V| - 1$ holds if and only if G is a complete graph [BFH19]. It is also known that $Z(G) \geq |V| - 2$ if and only if G is $\{P_4, \text{dart}, \ltimes, P_3 \cup K_2\}$-free [AIM08]. Liang et al. provided characterizations of graphs with maximum degree three and of triangle-free graphs satisfying $Z(G) = |V| - 3$ [LLX22].

There has also been extensive research on bounds for the zero forcing number. Amos et al. [ACDP15], Gentner et al. [GPRS16], Dávila et al. [DKS18], Brešar et al. [BCDH24], and Jing et al. [JZJ23] established several inequalities relating the zero forcing number to other graph parameters.

Applying the color-change rule to color all remaining vertices of an undirected graph blue results in a set of induced paths. Cameron et al. [CJMZ24] introduced the concept of a *chain twist*, which can be used to determine whether a set of directed induced paths are forcing chains of a zero forcing set. A related problem, Induced Path Partition, asks for the smallest set of induced paths such that every vertex lies on exactly one path. This problem is known to be NP-complete [LM+03]. The cardinality of such a minimum set, called the *induced path partition number*, provides a lower bound for the zero forcing number [Hog10]. It has also been shown that for cacti, the induced path partition number equals the zero forcing number [Row12a].

Several variants of Zero Forcing have been studied in the literature. Most of these variants start with an initial set of blue vertices, with the goal still being to color the entire graph blue. However, they differ either by modifying the color-change rule or by imposing additional

structural constraints on the zero forcing set. Since this thesis focuses on classical Zero Forcing, we briefly outline some of the most prominent variants below. Furthermore, since this work does not cover computational approaches to Zero Forcing, we also summarize several such approaches to provide broader context.

The problem Positive Semidefinite Zero Forcing asks for the smallest subset $S \subseteq V$ such that all vertices in $V \setminus S$ can be colored blue by repeatedly applying the *positive semidefinite color-change rule*. Let $W_1, \ldots, W_k$ denote the connected components of the graph $G - B$, where B is the set of currently blue vertices. If there exists a blue vertex $u \in B$ and a component $W_i \in \{W_1, \ldots, W_k\}$ such that u has exactly one neighbor $v \in W_i$, then v becomes blue. If all vertices of G can eventually be colored blue by applying the positive semidefinite color-change rule, then S is a *positive semidefinite zero forcing set* of G. The minimum size of such a set is the *positive semidefinite zero forcing number*, denoted $Z_+(G)$. For example, in any tree T, every single vertex forms a positive semidefinite zero forcing set, and hence $Z_+(T) = 1$. Since every zero forcing set is also a positive semidefinite zero forcing set, it follows that $Z_+(G) \leq Z(G)$. Moreover, $Z_+(G)$ serves as an upper bound for the maximum positive semidefinite nullity of G [Bar+10].

In Probabilistic Zero Forcing, each blue vertex $u \in V$ independently forces each of its white neighbors $v \in V$ with a certain probability in each round. The probability that u forces v is given by

$$P(u \to v) = \frac{|(N(v) \cup \{v\}) \cap B|}{\deg(v)}.$$

Starting from a non-empty set $S \subseteq V$, all vertices in $V \setminus S$ eventually become blue. The main parameter of interest is the *expected propagation time*, denoted $ept(G, S)$, which is the expected number of rounds until the entire graph is colored blue. Probabilistic zero forcing can also be modeled using Markov chains [Bre24].

The variant Leaky Forcing generalizes classical zero forcing by introducing robustness against failures. A set $S \subseteq V$ is called an ℓ-forcing set if all vertices in $V \setminus S$ can still be colored blue by repeatedly applying the color-change rule, even if up to ℓ vertices of G that could otherwise force are assumed to be *leaky*. A leaky vertex cannot force any of its neighbors. The minimum size of an ℓ-forcing set is the *ℓ-forcing number*, denoted $Z_{(\ell)}(G)$. By definition, $Z_{(0)}(G)$ coincides with the standard zero forcing number, and for every graph G,

$$Z_{(0)}(G) \leq Z_{(1)}(G) \leq Z_{(2)}(G) \leq \ldots$$

must hold [DK19].

In Skew Forcing, the goal is to find a minimum *skew forcing set*. A skew forcing set is a set of initially blue colored vertices $S \subseteq V$ such that all remaining white vertices in $V \setminus S$ can be colored blue by applying the *skew color-change rule*. This rule states that if a vertex $u \in V$ has exactly one white neighbor $v \in V$, then v becomes blue, regardless of whether u itself is blue. Unlike the standard zero forcing number, which is at least one for every nonempty graph, there exist graphs for which the skew forcing number is zero [DeA14].

The problem Connected Zero Forcing asks for a zero forcing set whose vertices induce a connected subgraph. Brimkov et al. proved that finding a minimum connected zero forcing set is NP-complete in general, while for trees and unicyclic graphs such a set can be found in $\mathcal{O}(|V|)$ time [BH17].

The problem Failed Zero Forcing considers sets that *fail* to force the entire graph. A set $S \subseteq V$ is a *failed zero forcing set* if S is not a zero forcing set. The *failed zero forcing number* $F(G)$ is defined as the maximum size of a failed zero forcing set of G. Fetcie et al. proved that $F(K_n) = n - 2$, $F(P_n) = \lceil (n-2)/2 \rceil$, and $F(C_n) = \lfloor n/2 \rfloor$ [FJS14].

The problem Power Dominating Set asks for a minimum set $S \subseteq V$ such that all vertices in V can be observed by repeatedly applying two observation rules. The first rule states that for every vertex $v \in S$, both v and all its neighbors become observed. The second rule, analogous to the zero forcing color-change rule, states that if an observed vertex has exactly one unobserved neighbor, then this neighbor becomes observed. The Power Dominating Set problem is NP-complete [HHHH02] and W[P]-complete when parameterized by $|S|$ [BG24]. Guo et al. proved that Power Dominating Set can be solved in time $\mathcal{O}(c^{k^2} \cdot n)$ for some constant c, given a graph with n vertices and a tree decomposition of width k [GNR08]. Hence, there exists an FPT algorithm for Power Dominating Set parameterized by treewidth. Bhyravarapu et al. adapted this FPT algorithm to obtain an FPT algorithm for Zero Forcing parameterized by treewidth [Bhy+25]. Since the pathwidth of a graph is a lower bound on its zero forcing number [Aaz08], this yields an FPT algorithm for Zero Forcing parameterized by the zero forcing number. We discuss these FPT algorithms for Power Dominating Set and Zero Forcing in Chapter 6.

Brimkov et al. compared several approaches for computing a minimum zero forcing set $S \subseteq V$ of a graph $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges [BFH19]. They showed that, given a subset $S \subseteq V$, one can decide in $\mathcal{O}(n + m)$ time whether S is a zero forcing set of G. This observation allows for a brute-force approach to find a minimum zero forcing set, which can be effective when the zero forcing number is known to be very small or very large. However, in practice, brute-force search is typically outperformed by more sophisticated algorithms.

The Wavefront algorithm is a dynamic programming approach that constructs minimum zero forcing sets of larger subgraphs by combining solutions for smaller subgraphs. While this method can achieve strong results, it comes at the cost of significantly higher memory consumption. Brimkov et al. also introduced a branch-and-bound algorithm for *connected zero forcing*, which systematically explores subgraphs while pruning infeasible or suboptimal branches.

In addition to the Wavefront and branch-and-bound algorithms, Brimkov et al. investigated integer linear programming formulations of the zero forcing problem. They presented the *infection perspective* and the *fort covering perspective*. In the infection perspective, an integer linear programming formulation is constructed where, for each vertex $v \in V$, a binary decision variable indicates whether v belongs to the initial zero forcing set S, and an additional integer variable represents the time step at which v becomes blue. For each edge $\{u, v\} \in E$, two binary variables are introduced to encode whether u forces v or v forces u. Although this formulation introduces only a polynomial number of variables in n, it relies on big-M constraints, which can significantly increase runtime in practice.

The fort covering perspective, in contrast, approaches the problem as a set-covering problem. Every zero forcing set must contain at least one vertex from each *fort* of the graph. Thus, finding a minimum zero forcing set can be seen as selecting a minimum subset of vertices that contains at least one vertex from every fort. While the total number of forts may be exponential in $|V|$, it is often possible to efficiently identify a violated fort if a current candidate set fails to cover all forts. Based on this observation, Brimkov et al. proposed a constraint generation approach. Starting from a relaxed master problem, they iteratively detect uncovered forts and add the corresponding covering constraints. They also discuss several strategies for efficiently identifying such violated forts. For connected zero forcing, additional constraints can be introduced to ensure that the selected vertices induce a connected subgraph.

Finally, Brimkov et al. conducted computational experiments across different graph families. Their results show that on random cubic graphs and Watts–Strogatz small-world graphs, the Wavefront algorithm achieves the best performance. In contrast, the infection perspective and the fort covering perspective tend to perform better on graphs modeling electrical power grids and other real-world networks.

1.2 Outline and Contribution

In Chapter 2, we introduce the fundamental definitions and concepts used throughout this work. We present basic notions from graph theory and formally define the Zero Forcing problem for undirected graphs, as Chapters 3, 4, 5, and 6 focus on Zero Forcing on undirected graphs. Only in Chapter 7 do we consider Zero Forcing on directed graphs.

In Chapter 3, we introduce the concept of *forcing chains*, which are directed induced paths obtained by iteratively applying the color-change rule until all vertices are colored blue. We then define the related problem Induced Path Partition and compare the induced path partition number of a graph with its zero forcing number. We provide two characterizations of forcing chains, allowing us to determine whether the directed paths in an orientation of an induced path partition are forcing chains. The first of these two characterizations is the concept of *chain twists*, presented by Cameron et al. [CJMZ24]. Secondly, we contribute a new characterization for which we define a *dependency graph* and show that the dependency graph is acyclic if and only if the directed paths are forcing chains. We then use our characterization via the dependency graph to derive results on constructing new zero forcing sets from existing ones and to prove that for every undirected graph $G = (V, E)$ and vertex $v \in V$ that is not isolated, there exists a minimum zero forcing set of G that does not contain v. Furthermore, we prove that it is NP-hard to decide whether there exists an orientation $\overrightarrow{P}$ of the paths in an induced path partition P such that the paths in $\overrightarrow{P}$ are the forcing chains of a zero forcing set. Finally, we use the forcing chains of a zero forcing set to prove that Zero Forcing remains NP-hard on undirected bipartite graphs with maximum degree three.

In Chapter 4, we begin by presenting several well-established inequalities relating the zero forcing number to other graph parameters. We then apply the concept of forcing chains to prove a new inequality. We show that in every undirected graph $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges, if there exists a zero forcing set $S \subseteq V$ of size $k = |S|$, then

$$m \leq k \cdot \left(n - \frac{k+1}{2} \right).$$

We further show that edges can be added to G such that this inequality becomes tight while S remains a zero forcing set. Moreover, we prove that if the inequality is tight and $S \neq V$, then S must be a minimum zero forcing set.

Chapter 5 presents a linear-time algorithm for Zero Forcing on cacti. To the best of our knowledge, no linear-time algorithm for cacti has been established so far. Our approach builds on a theorem by D. Row stating that the induced path partition number of a cactus equals its zero forcing number [Row12a]. Based on reduction rules, we provide a linear-time algorithm for Induced Path Partition on cacti and demonstrate that one can find an orientation of the undirected induced paths in an induced path partition in linear time such that the resulting directed paths are forcing chains of a zero forcing set, yielding a linear-time algorithm for Zero Forcing on cacti.

An initial goal of this thesis was to design an FPT algorithm for Zero Forcing parameterized by the zero forcing number. However, during the course of this work, Bhyravarapu et al. developed such an algorithm. We provide a brief overview of their result in Chapter 6 and discuss how our own preliminary approach relates to their method.

Finally, in Chapter 7, we extend the definition of Zero Forcing to directed graphs and provide a proof for the NP-hardness of Zero Forcing on DAGs.

2 Preliminaries

In this chapter, we introduce the fundamental definitions and concepts that form the foundation of this thesis. We begin with essential notions from graph theory and then formally define the Zero Forcing problem for undirected graphs.

2.1 Graphs

In this section, we introduce the fundamental graph-theoretic concepts used throughout this thesis. For a comprehensive reference, we refer to Reinhard Diestel's textbook *Graph Theory* [Die25].

A *graph* is a tuple $G = (V, E)$, where V is the set of vertices and E is the set of edges. We distinguish between *directed* and *undirected* graphs. In an undirected graph, each edge is a subset of V containing exactly two vertices. Two vertices $u, v \in V$ are said to be *adjacent* if $\{u, v\} \in E$. The *neighborhood* of a vertex $v \in V$ is defined as

$$N(v) = \{u \in V \mid \{u, v\} \in E\}.$$

The *degree* of a vertex v, denoted by $\deg(v)$, is the cardinality of its neighborhood $N(v)$. A vertex $v \in V$ with $\deg(v) = 0$ is called an *isolated*, while a vertex with $\deg(v) = 1$ is called a *leaf*. The *maximum degree* of a graph G is defined as $\Delta = \max\{\deg(v) \mid v \in V\}$.

A *subgraph* $G' = (V', E')$ of an undirected graph $G = (V, E)$ satisfies $V' \subseteq V$ and $E' \subseteq E$. Removing a vertex $v \in V$ from G yields the subgraph $G' = (V', E')$ with $V' = V \setminus \{v\}$ and $E' = \{e \in E \mid v \notin e\}$, denoted by $G - v$. More generally, for a subset $W \subseteq V$, the notation $G - W$ refers to the subgraph obtained by successively deleting all vertices in W from G.

A *path* p of length ℓ in an undirected graph $G = (V, E)$ is a sequence

$$p = (v_1, v_2, \ldots, v_\ell)$$

of ℓ distinct vertices such that each pair of consecutive vertices is adjacent. The vertices v_1 and v_ℓ are called the *endpoints* of p. For undirected paths, $p = (v_1, \ldots, v_\ell)$ and $p' = (v_\ell, \ldots, v_1)$ represent the same path. Assigning an orientation to an undirected path yields a *directed path*, denoted by $\overrightarrow{p}$. For $\overrightarrow{p} = (v_1, \ldots, v_\ell)$, the vertex v_1 is the *first vertex* and v_ℓ the *last vertex*. A path p in a graph G is said to be *induced* if two vertices of p are adjacent in G if and only if they are consecutive in p.

A *cycle* C of length $\ell \geq 3$ in an undirected graph is a sequence

$$C = (v_1, \ldots, v_\ell, v_1),$$

consisting of $\ell + 1$ vertices, where only the first and last vertices coincide, and each pair of consecutive vertices is adjacent.

An undirected graph is *connected* if, for every pair of vertices $u, v \in V$, there exists a path between u and v. An undirected graph $G = (V, E)$ is *bipartite* if its vertex set can be partitioned into two disjoint subsets U and W such that no two vertices within the same subset are adjacent. A *tree* $T = (V, E)$ is a connected, acyclic, undirected graph. A *rooted tree*

is a tree with a distinguished vertex $r \in V$, called the *root*. For each vertex $v \in V$, there exists a unique path from r to v. The *parent* of a vertex $v \in V \setminus \{r\}$ is the next vertex on its path toward r, and the *children* of v are the vertices for which v is their parent. Vertices with no children are called *leaves*.

A *cactus* is a graph $G = (V, E)$ in which each edge belongs to at most one cycle. A *cut-vertex* of an undirected graph $G = (V, E)$ is a vertex $v \in V$ such that $G - v$ is disconnected. A *block* of G is a maximal connected subgraph without a cut-vertex. The *block-cutpoint tree* of G is a tree $T = (V', E')$ whose vertex set consists of the blocks and cut-vertices of G, where a block $b \in V'$ and a cut-vertex $v \in V'$ are adjacent in T if and only if v is contained in b.

A *tree decomposition* of an undirected graph $G = (V, E)$ is a pair (X, T), where $T = (I, F)$ is a tree and $X = \{X_i \mid i \in I\}$ is a family of subsets of V, called *bags*, satisfying the following conditions:

(T1) $V = \bigcup_{i \in I} X_i$.

(T2) For every edge $\{u, v\} \in E$, there exists an $i \in I$ such that $\{u, v\} \subseteq X_i$.

(T3) For all $i, j, k \in I$, if j lies on the path between i and k in T, then $X_i \cap X_k \subseteq X_j$.

The *width* of a tree decomposition is defined as

$$\max_{i \in I} |X_i| - 1.$$

The *treewidth* of an undirected graph G, denoted by $\mathrm{tw}(G)$, is the minimum integer k such that there exists a tree decomposition of G with width at most k.

It is known that a tree decomposition of an undirected graph G with minimum treewidth k can be found in time $\mathcal{O}(|V| \cdot f(k))$, where f is a computable function depending only on k [Bod93]. Hence, computing a tree decomposition of minimum width is fixed-parameter tractable (FPT) when parameterized by the treewidth of G.

A *path decomposition* of an undirected graph $G = (V, E)$ is a tree decomposition (X, T) of G where T is a path. The *width* of a path decomposition is again defined as

$$\max_{i \in I} |X_i| - 1.$$

The *pathwidth* of G, denoted by $\mathrm{pw}(G)$, is the minimum integer k such that there exists a path decomposition of G with width at most k.

In a directed graph $\overrightarrow{G} = (\overrightarrow{V}, \overrightarrow{E})$, each edge is an ordered pair of distinct vertices. For an edge $e = (u, v) \in \overrightarrow{E}$, the edge is directed from u to v, and v is called a *neighbor* of u. A cycle of length $\ell \geq 2$ in a directed graph is a sequence

$$C = (v_1, \ldots, v_\ell, v_1),$$

consisting of $\ell + 1$ vertices, where only the first and last vertices coincide. Each vertex $v_i \in \{v_2, \ldots, v_\ell\}$ of the cycle C is a neighbor of the vertex v_{i-1}, and the vertex v_1 is a neighbor of v_ℓ.

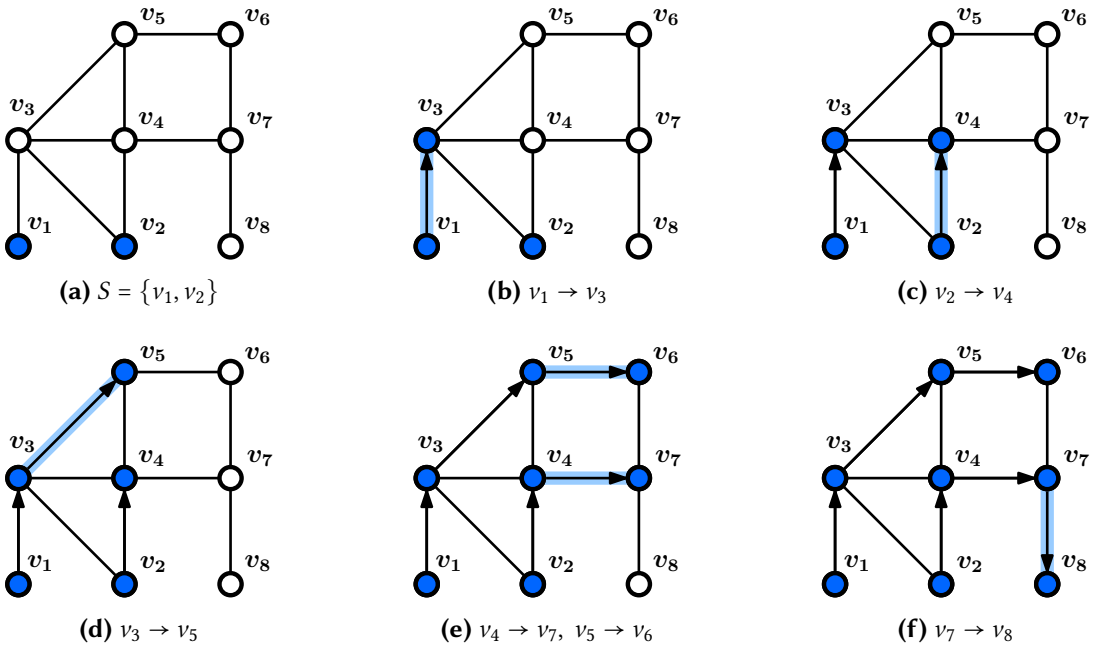
A *DAG* (directed acyclic graph) is a directed graph that contains no cycles. In a DAG, there exists an ordering of the vertices $\pi = (v_1, \ldots, v_n)$ such that for every edge $(v_i, v_j) \in \overrightarrow{E}$, it holds that $i < j$.

2.2 Zero Forcing

The Zero Forcing problem asks for the smallest subset of vertices $S \subseteq V$ of an undirected graph $G = (V, E)$ that is a *zero forcing set*. The vertex set V is partitioned into two subsets B and W, where vertices in B are colored blue and vertices in W are colored white. Initially, the vertices in S are colored blue, and all remaining vertices are colored white.

The set S is a zero forcing set of G if all vertices in $V \setminus S$ can be colored blue by repeatedly applying the *color-change rule*, defined as follows: if a blue vertex u has exactly one white neighbor v, then v becomes blue. We denote this by $u \to v$ and say that u *forces* v. A zero forcing set S is called a *minimum zero forcing set* if $|S| \leq |S'|$ for all zero forcing sets S' of G. The minimum cardinality of a zero forcing set in G is called the *zero forcing number*, denoted by $Z(G)$.

It has been shown that Zero Forcing is NP-hard [FMY16] on undirected graphs. In Section 3.4, we also prove that Zero Forcing remains NP-hard on undirected bipartite graphs in which every vertex has degree at most three.



(a) $S = \{v_1, v_2\}$ **(b)** $v_1 \to v_3$ **(c)** $v_2 \to v_4$

(d) $v_3 \to v_5$ **(e)** $v_4 \to v_7,\ v_5 \to v_6$ **(f)** $v_7 \to v_8$

Figure 2.1: Illustration of the Zero Forcing process.

Figure 2.1 illustrates Zero Forcing on an undirected graph G. The vertices $S = \{v_1, v_2\}$ is a zero forcing set of G, as all vertices can be colored blue through successive applications of the color-change rule, starting with only the vertices in S colored blue. For each application of the color-change rule $u \to v$, the edge between u and v is colored blue and drawn as a directed edge from u to v. Since no zero forcing set of size one exists in G, the set S is a minimum zero forcing set, and we conclude that $Z(G) = 2$.

3 Forcing Chains

Let $S = \{v_1, \ldots, v_k\} \subseteq V$ be a zero forcing set of an undirected graph $G = (V, E)$ of size k. The vertices in S are initially colored blue, and we apply the color-change rule until all vertices of G are colored blue. The application of the color-change rule induces a set of k directed paths $\overrightarrow{P} = \{\overrightarrow{p_1}, \ldots, \overrightarrow{p_k}\}$ in G, as shown by [Hog10]. These directed paths are called *forcing chains*.

We now describe how to construct the forcing chains $\overrightarrow{P}$. Initially, each forcing chain $\overrightarrow{p_i}$ consists of a single vertex v_i. Whenever the color-change rule $u \to v$ is applied, the vertex u must be the last vertex of some forcing chain $\overrightarrow{p} \in \overrightarrow{P}$. We then append v to $\overrightarrow{p}$ as its new endpoint. This process yields valid paths, since at each step there are exactly k blue vertices that have not yet forced another vertex. These vertices are precisely the current last vertices of the k forcing chains. Once a color-change rule $u \to v$ is applied, u cannot force again because all of its neighbors are already blue. The newly colored vertex v, however, may have white neighbors and thus becomes the new endpoint of its forcing chain.

For example, consider the graph in Figure 2.1. From the zero forcing set $S = \{v_1, v_2\}$, we obtain the forcing chains

$$\overrightarrow{p_1} = (v_1, v_3, v_5, v_6), \quad \overrightarrow{p_2} = (v_2, v_4, v_7, v_8)$$

by applying the color-change rule as illustrated.

In Section 3.1, we further characterize the forcing chains obtained from a zero forcing set. This leads to the notions of an *induced path partition* and the *induced path partition number*. We then compare the zero forcing number with the induced path partition number and show that the zero forcing number of a graph can exceed its induced path partition number by an arbitrary factor. In Section 3.2, we present both a known characterization and a new one, each of which allows us to decide whether a set of directed paths forms the forcing chains of a zero forcing set. In Section 3.3, we use our characterization of forcing chains to show how new zero forcing sets of equal size can be constructed from a given one. In Section 3.4, we prove that it is NP-hard to decide whether there exists an orientation of the induced paths in an induced path partition such that the set of first vertices forms a zero forcing set, and we provide our own proof that Zero Forcing is NP-hard on bipartite graph with maximum degree three.

The results on forcing chains from this chapter serve as an important foundation for the subsequent chapters on Zero Forcing in undirected graphs. In Chapter 4, we use the forcing chains of a zero forcing set to prove an upper bound on the number of edges in a graph with a zero forcing set of size k. Our algorithm for computing a minimum zero forcing set of a cactus in linear time, presented in Chapter 5, as well as the FPT-algorithm for zero forcing introduced in Chapter 6, both rely on induced paths in a zero forcing set. The algorithm for Zero Forcing on cacti computes a set of undirected induced paths that can be oriented in linear time so that they form forcing chains satisfying the properties described in Section 3.2. The FPT-algorithm, on the other hand, directly computes the forcing chains, which is computationally more demanding.

3.1 Induced Path Partition

Let $\overrightarrow{P}$ be a set of forcing chains. Each vertex lies on exactly one forcing chain in $\overrightarrow{P}$, because each vertex is added to a forcing chain exactly once. By construction, the vertices in S are the initial vertices of the forcing chains, and each other vertex is colored blue exactly once and thus added to a forcing chain exactly once.

Moreover, the forcing chains in $\overrightarrow{P}$ are induced paths. Suppose, for contradiction, that there exists a forcing chain $\overrightarrow{p} = (v_1, \ldots, v_n) \in \overrightarrow{P}$ with $v_1 \in S$ that is not induced. Then there must be an edge between two non-consecutive vertices $v_i \in \{v_1, \ldots, v_{n-2}\}$ and $v_j \in \{v_{i+2}, \ldots, v_n\}$. This would imply that when the color-change rule $v_i \to v_{i+1}$ was applied, v_j was still white, contradicting the requirement that v_{i+1} was the unique white neighbor of v_i at that time.

Hence, every zero forcing set S of size k induces a set of k undirected induced paths such that each vertex in V lies on exactly one of them. We call such a set of paths an *induced path partition.* Moreover, we define the problem Induced Path Partition as follows:

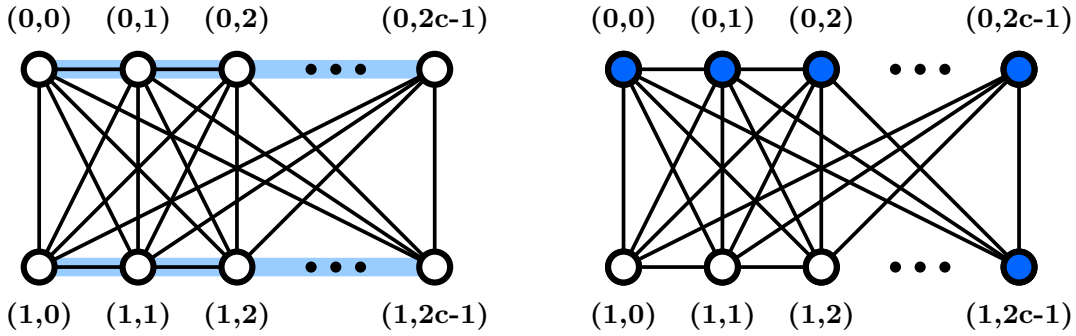
Induced Path Partition. Given an undirected graph $G = (V, E)$ and a non-negative integer k, is there an induced path partition of G of size at most k?

We define the *induced path partition number* $P(G)$ as the smallest non-negative integer k for which an induced path partition of size k exists. The Induced Path Partition problem is NP-complete. In fact, it has been shown that deciding whether there exists an induced path partition $\overrightarrow{P}$ of a graph with $|\overrightarrow{P}| = 2$ is already NP-complete [LM+03].

We now compare the induced path partition number $P(G)$ and the zero forcing number $Z(G)$ of an undirected graph G. Since we can construct an induced path partition from every zero forcing set of the same size, $P(G)$ must be a lower bound for $Z(G)$, as already observed by Hogben [Hog10].

Theorem 3.1 ([Hog10])**:** *For any undirected graph G, it holds that $P(G) \leq Z(G)$.*

In general, $P(G)$ and $Z(G)$ are not equal. In the following we show that there exist undirected graphs where $Z(G)$ exceeds $P(G)$ by an arbitrarily large factor.



(a) Induced Path Partition of size 2. **(b)** Zero Forcing Set of size $2c + 1$.

Figure 3.1: Graph for which $P(G) < \frac{1}{c} \cdot Z(G)$.

Theorem 3.2: *For any positive integer c, there exists a graph $G = (V, E)$ such that*

$$P(G) < \frac{1}{c} \cdot Z(G).$$

Proof. Let $G = (V, E)$ be an undirected graph with $V = \{0, 1\} \times \{0, 1, \ldots, 2c - 1\}$. Two vertices $(a, i), (b, j) \in V$ are adjacent if and only if $a \neq b$ or $|i - j| = 1$. The graph G is shown in Figure 3.1.

Define the undirected paths

$$p_0 = ((0, 0), (0, 1), \ldots, (0, 2c - 1)) \quad \text{and} \quad p_1 = ((1, 0), (1, 1), \ldots, (1, 2c - 1)).$$

These are induced paths since two vertices (a, i) and (a, j) on the same path are adjacent only if $|i - j| = 1$. Thus, $P = \{p_0, p_1\}$ is an induced path partition of G. Since G is not a path, $P(G) = 2$.

The set

$$S = \{(0, 0), (0, 1), \ldots, (0, 2c - 1)\} \cup \{(1, 2c - 1)\}$$

is a zero forcing set of G of size $2c + 1$, since we can color all remaining vertices blue by applying the color-change rules

$$(1, 2c - 1) \to (1, 2c - 2),\ (1, 2c - 2) \to (1, 2c - 3),\ \ldots,\ (1, 1) \to (1, 0).$$

Each vertex $(a, i) \in V$ is adjacent to all $2c$ vertices $(1 - a, j)$ for $j \in \{0, \ldots, 2c - 1\}$, and to at least one of $(a, i - 1)$ or $(a, i + 1)$. Hence, each vertex has degree at least $2c + 1$. To apply the color-change rule, a blue vertex must have exactly one white neighbor. Thus, at least $2c + 1$ vertices must be initially blue. Therefore $Z(G) = 2c + 1$.

Consequently,

$$P(G) = 2 < \frac{1}{c} \cdot (2c + 1) = \frac{1}{c} \cdot Z(G).$$

■

3.2 Characterizing Forcing Chains

As shown in Section 3.1, the undirected paths of the forcing chains of a zero forcing set always form an induced path partition of the same size as the zero forcing set. However, there does not always exist an orientation of the induced paths in an induced path partition such that the set containing the first vertex of each induced path is a zero forcing set and the induced paths are the corresponding forcing chains. This is due to the existence of graphs G for which $P(G) > Z(G)$, as established in Theorem 3.2.

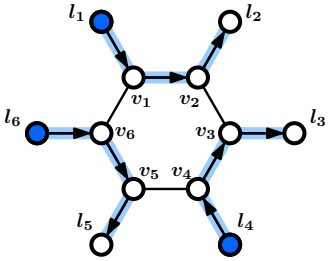
Consider, for example, the undirected graph shown in Figure 3.2a. The undirected paths $P = \{p_1, p_2, p_3\}$ with

$$p_1 = (l_1, v_1, v_2, l_2), \quad p_2 = (l_4, v_4, v_3, l_3), \quad p_3 = (l_6, v_6, v_5, l_5)$$

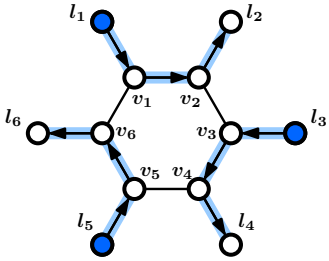
are an induced path partition of G. If we orient these undirected paths such that l_1, l_4, and l_6 are the first vertices of p_1, p_2, and p_3, respectively, then we obtain forcing chains with underlying zero forcing set $S = \{l_1, l_4, l_6\}$.

In contrast, if we orient the induced paths such that l_1, l_3, and l_5 are the first vertices of p_1, p_2, and p_3, respectively, as shown in Figure 3.2b, then $S = \{l_1, l_3, l_5\}$ is not a zero forcing set of G, and the obtained directed paths are not forcing chains.

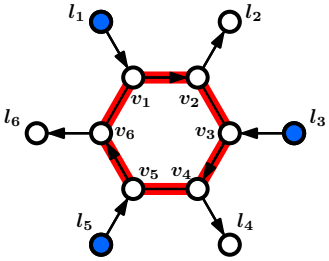
In this section, we present two characterizations of forcing chains of a zero forcing set, enabling us to decide whether the set containing the first vertices of an orientation of an induced path partition is a zero forcing set and whether the directed paths are forcing chains. These characterizations also allow us to construct alternative zero forcing sets of the same size from a given one.



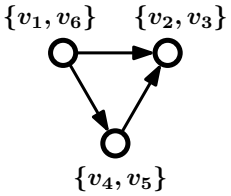
(a) Orientation $\overrightarrow{P_1}$ of an induced path partition such that the directed induced paths in $\overrightarrow{P_1}$ are forcing chains of the zero forcing set $S = \{l_1, l_4, l_6\}$.



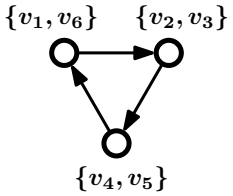
(b) Orientation $\overrightarrow{P_2}$ of the same induced path partition where $S = \{l_3, l_3, l_5\}$ is not a zero forcing set.



(c) The graph G with orientation $\overrightarrow{P_2}$ of the induced path partition, containing a chain twist.



(d) The dependency graph $G_{\overrightarrow{P_1}}$ of G with respect to $\overrightarrow{P_1}$ is acyclic.



(e) The dependency graph $G_{\overrightarrow{P_2}}$ of G with respect to $\overrightarrow{P_2}$ contains a cycle.

Figure 3.2: Illustration of forcing chains arising from a zero forcing set.

3.2.1 Characterization via Chain Twist

Cameron et al. [CJMZ24] provide a characterization of orientations $\overrightarrow{P}$ of an induced path partition P that allows one to decide whether the set containing the first vertex of each directed induced path in $\overrightarrow{P}$ is a zero forcing set and whether the directed induced paths in $\overrightarrow{P}$ are the corresponding forcing chains.

They define a *chain twist* for a graph G with respect to an orientation $\overrightarrow{P}$ of an induced path partition as a cycle

$$C = (v_0, v_1, \ldots, v_{k-1}, v_0)$$

in G such that, for each $i \in \{0, \ldots, k-1\}$, if v_i is not the successor of v_{i+1} on a directed induced path in $\overrightarrow{P}$, then v_{i-1} is the predecessor of v_i on its directed induced path in $\overrightarrow{P}$, and v_{i+2} is the successor of v_{i+1} on its directed induced path in $\overrightarrow{P}$. All indices are taken modulo k. Cameron et al. show that the set set containing the first vertex of each directed induced path set a zero forcing set if and only if G contains no chain twist.

The set containing the first vertex of each directed induced path in $\overrightarrow{P_1}$ is a zero forcing set since G contains no chain twist with respect to $\overrightarrow{P_1}$, as shown in Figure 3.2a. In contrast, for $\overrightarrow{P_2}$, the set of containing the first vertex of each directed induced path in $\overrightarrow{P_2}$ is not a zero forcing set because the cycle

$$C = (v_1, v_2, v_3, v_4, v_5, v_6, v_1)$$

is a chain twist, as illustrated in Figure 3.2c.

3.2.2 Characterization via Dependency Graph

We now provide an alternative characterization of the forcing chains $\overrightarrow{P}$ of a zero forcing set in an undirected graph $G = (V, E)$.

Consider a vertex $v \in V$ lying on a forcing chain $\overrightarrow{p} \in \overrightarrow{P}$. To color v blue, all predecessors of v on $\overrightarrow{p}$ must already be blue, as well as all their neighbors outside $\overrightarrow{p}$. Consequently, for any edge $e = \{u, v\} \in E$ not contained in a forcing chain of $\overrightarrow{P}$, both u and v must be blue before either can force their respective successors on their forcing chains in $\overrightarrow{P}$.

Moreover, for any pair of edges $e_1 = \{u, x\}$ and $e_2 = \{v, y\}$ not lying on forcing chains, if v is the successor of u on a forcing chain in $\overrightarrow{P}$, then in order for v or y to force their successors, both u and x must already be blue.

We capture these dependencies in a directed graph $G_{\overrightarrow{P}} = (V_{\overrightarrow{P}}, E_{\overrightarrow{P}})$, called the *dependency graph of G with respect to $\overrightarrow{P}$*, where $\overrightarrow{P}$ is a set of directed induced paths. Its vertices $V_{\overrightarrow{P}}$ correspond to the edges of E that do not lie on a forcing chain in $\overrightarrow{P}$. There is a directed edge from $e_1 = \{u, x\} \in V_{\overrightarrow{P}}$ to $e_2 = \{v, y\} \in V_{\overrightarrow{P}}$ if and only if u or x is the predecessor of v or y on a forcing chain in $\overrightarrow{P}$.

Theorem 3.3: *Let S be a zero forcing set of G with forcing chains $\overrightarrow{P}$. Then the dependency graph $G_{\overrightarrow{P}}$ is acyclic.*

Proof. Assign to each vertex $\{u, v\} \in V_{\overrightarrow{P}}$ the earliest timestep at which both u and v are colored blue. Consider an edge $(\{u, x\}, \{v, y\}) \in E_{\overrightarrow{P}}$ where u is a predecessor of v on a forcing chain in $\overrightarrow{P}$. To color v blue, both u and x must already be blue. Hence, the timestamp of $\{u, x\}$ is strictly smaller than that of $\{v, y\}$. Therefore, all edges in $E_{\overrightarrow{P}}$ are directed from smaller to larger timestamps, implying that $G_{\overrightarrow{P}}$ is acyclic. ∎

We can also show that the set containing the first vertex of each directed induced path in an orientation $\overrightarrow{P}$ of an induced path partition is a zero forcing set whenever $G_{\overrightarrow{P}}$ is acyclic.

Theorem 3.4: *If the dependency graph $G_{\overrightarrow{P}}$ of an orientation $\overrightarrow{P}$ of the induced paths in an induced path partition of G is acyclic, then the set S containing the first vertex of each induced path in $\overrightarrow{P}$ is a zero forcing set.*

Proof. If $G_{\overrightarrow{P}}$ is acyclic, then there exists a topological ordering $\tilde{\pi} = (\overrightarrow{v}_1, \ldots, \overrightarrow{v}_m)$ of $V_{\overrightarrow{P}}$ such that no edge points from $\overrightarrow{v}_i$ to $\overrightarrow{v}_j$ with $i > j$.

Initially, color all vertices in S blue and apply the color-change rule. Suppose there exists a vertex $\{w, y\} \in V_{\overrightarrow{P}}$ such that w is white, while all vertices corresponding to earlier edges in $\tilde{\pi}$ are already colored blue. Let u be the last blue vertex on the forcing chain $\overrightarrow{p} \in \overrightarrow{P}$ containing w, and let v be its successor on $\overrightarrow{p}$. Then v is still white. Assume, for contradiction, that u has another white neighbor $x \in V \setminus \{v\}$. Since the predecessor of u on $\overrightarrow{p}$ must be colored blue before u becomes blue, the vertex x cannot lie on $\overrightarrow{p}$. Hence, $\{u, x\} \in V_{\overrightarrow{P}}$, and there is an edge from $\{u, x\}$ to $\{w, y\}$ in $G_{\overrightarrow{P}}$. This implies that $\{u, x\}$ precedes $\{w, y\}$ in $\tilde{\pi}$, contradicting the assumption that $\{w, y\}$ is the first vertex in $\tilde{\pi}$ where w or v is still white. Therefore, u forces v.

Now, suppose that for all vertices $\{x, y\} \in V_{\overrightarrow{P}}$, both x and y are blue, yet there remains some white vertex $x \in V$. Then, on the directed induced path $\overrightarrow{p} \in \overrightarrow{P}$ containing x, there exists a vertex u that is blue with successor v still white. Since all neighbors of u outside $\overrightarrow{p}$ must already be colored blue, and the predecessor of u on $\overrightarrow{p}$ must be blue for u itself to be blue, the vertex v is the unique white neighbor of u. Hence, u forces v.

Thus, the color-change rule can be applied until all vertices of G are blue. Therefore, S is a zero forcing set of G. ∎

Combining Theorems 3.3 and 3.4, we obtain the following corollary.

Corollary 3.5: *The set $S \subseteq V$ consisting of the first vertex of each directed induced path in an orientation $\overrightarrow{P}$ of an induced path partition of an undirected graph $G = (V, E)$ is a zero forcing set of G if and only if the dependency graph $G_{\overrightarrow{P}}$ with respect to $\overrightarrow{P}$ is acyclic.*

Proof. If S is a zero forcing set of G, then the directed induced paths in $\overrightarrow{P}$ are forcing chains of G, and therefore $G_{\overrightarrow{P}}$ is acyclic by Theorem 3.3. Conversely, if $G_{\overrightarrow{P}}$ is acyclic, then the set consisting of the first vertex of each directed induced path in $\overrightarrow{P}$ is a zero forcing set of G by Theorem 3.4. ∎

3.3 Constructing New Zero Forcing Sets

The set of forcing chains $\overrightarrow{P}$ of a zero forcing set can be used to derive other zero forcing sets of the same size. If we reverse some of the forcing chains in $\overrightarrow{P}$ to obtain the directed induced paths $\overrightarrow{P'}$, and if the dependency graph of G with respect to $\overrightarrow{P'}$ is acyclic, then by Corollary 3.5, the set consisting of the first vertices of each directed induced path in $\overrightarrow{P'}$ is also a zero forcing set of G.

In this section, we prove that for every non-isolated vertex $v \in V$, there exists a zero forcing set $S \subseteq V$ of G such that $v \notin S$. We first establish the following lemma, which shows that reversing all forcing chains of a zero forcing set yields another zero forcing set.

Lemma 3.6: *Let $S \subseteq V$ be a zero forcing set with forcing chains $\overrightarrow{P}$. Then the set containing the last vertex of each forcing chain in $\overrightarrow{P}$ is also a zero forcing set of G.*

Proof. By Theorem 3.3, the dependency graph $G_{\overrightarrow{P}} = (V_{\overrightarrow{P}}, E_{\overrightarrow{P}})$ of G with respect to $\overrightarrow{P}$ is acyclic. Let $\overrightarrow{P'}$ denote the set of directed paths obtained by reversing all forcing chains in $\overrightarrow{P}$. The dependency graph $G_{\overrightarrow{P'}} = (V_{\overrightarrow{P'}}, E_{\overrightarrow{P'}})$ is also acyclic, since $V_{\overrightarrow{P'}} = V_{\overrightarrow{P}}$, and for each edge $(x, y) \in E_{\overrightarrow{P}}$, the graph $G_{\overrightarrow{P'}}$ contains the edge (y, x). By Theorem 3.4, the set containing the first vertex of each directed induced path in $\overrightarrow{P'}$ is a zero forcing set. These vertices are precisely the last vertices of the original forcing chains in $\overrightarrow{P}$. Hence, the set containing the last vertex of each directed induced path in $\overrightarrow{P}$ is a zero forcing set of G. ∎

Next, we prove that for each non-isolated vertex $v \in V$, and zero forcing set that contains v, we can color all vertices in $V \setminus S$ blue by applying the color-change rule in such an order that v lies on a forcing chain of length at least two.

Lemma 3.7: *Let $G = (V, E)$ be an undirected graph with a zero forcing set $S \subseteq V$ and let $v \in S$ be a non-isolated vertex. Then it is possible to color all vertices of V blue such that the forcing chain of v has length at least two.*

Proof. Suppose, for contradiction, that all neighbors of v are contained in S. Since $\deg(v) \geq 1$, v has at least one neighbor $x \in S$. Consider the set $S' = S \setminus \{x\}$ and initially color all vertices in S' blue. Since x is the only neighbor of v not in S', and $v \in S'$, we can apply $v \to x$. This ensures that all vertices in S are colored blue, after which the color-change rule can be applied until all vertices in V are blue. Thus, S' is a zero forcing set of smaller size than S, contradicting the assumption that S is a minimum zero forcing set.

Hence, v must have at least one neighbor not in S. Starting from S, apply the color-change rule until exactly one neighbor $x \in N(v)$ remains white. Then apply $v \to x$, and continue applying the color-change rule until all vertices are colored blue. In this process, v lies on a forcing chain of length at least two. ∎

Together with Lemma 3.7, we can now show that there exists a minimum zero forcing set of G that does not contain v.

Theorem 3.8: *Let $G = (V, E)$ be an undirected graph and let $v \in V$ be a vertex with $\deg(v) \geq 1$. Then there exists a minimum zero forcing set S of G such that $v \notin S$.*

Proof. Let S be a minimum zero forcing set of G with $v \in S$. By Lemma 3.7, there exists a set of forcing chains $\overrightarrow{P}$ of S such that v lies on a forcing chain of length at least two. By Lemma 3.6, the set S' consisting of the last vertex of each forcing chain in $\overrightarrow{P}$ is also a zero forcing set of G. Since v lies on a forcing chain of length at least two, $v \notin S'$. ∎

3.4 Zero Forcing and Induced Path Orientation are NP-hard

In this section, we prove that Zero Forcing is NP-hard on undirected bipartite graphs with maximum degree three by a reduction from Monotone Not-All-Equal 3-Sat, and that Induced Path Orientation is NP-hard. We first introduce the problems Induced Path Orientation and Monotone Not-All-Equal 3-Sat. Then, we construct an undirected graph G with maximum degree three from an instance of Monotone Not-All-Equal 3-Sat and define a corresponding set of induced path partitions of G. Subsequently, we prove several lemmas that establish relationships between minimum induced path partitions of G and orientations of their induced paths such that these paths are forcing chains. Using these lemmas, we first show that Induced Path Orientation is NP-hard, and then that Zero Forcing is NP-hard on bipartite graphs with maximum degree three.

3.4.1 Introducing Induced Path Orientation

We have shown that for certain induced path partitions of a graph, there exists an orientation of the induced paths such that the set containing the first vertex of each induced path is a zero forcing set. However, for other induced path partitions, no such orientation exists. We refer to this decision problem as Induced Path Orientation.

Induced Path Orientation. Given a graph $G = (V, E)$ and an induced path partition P of G, is there an orientation $\overrightarrow{P}$ of P such that the set of first vertices of the directed induced paths in $\overrightarrow{P}$ is a zero forcing set of G?

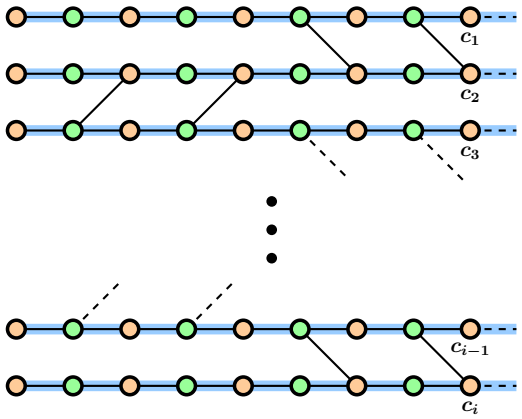
In this chapter, we show that both Zero Forcing on undirected bipartite graphs with maximum degree three and Induced Path Orientation are NP-hard by reduction from Monotone Not-All-Equal 3-Sat. The problem Monotone Not-All-Equal 3-Sat is defined as follows.

Monotone Not-All-Equal 3-Sat. Given a set of variables Var and a collection Cl of clauses over Var, where each clause in Cl contains exactly three distinct variables, is there a truth assignment such that in every clause at least one variable is true and at least one variable is false?

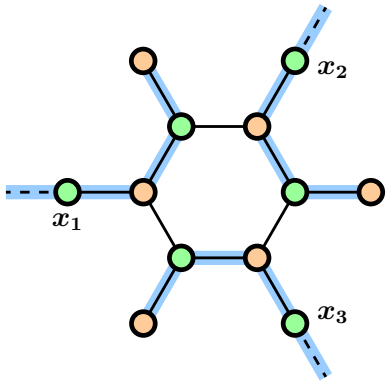
Darmann et al. proved that Monotone Not-All-Equal 3-Sat is NP-hard [DD19].

3.4.2 Constructing a graph from Monotone Not-All-Equal 3-Sat

To prove that both Zero Forcing on undirected bipartite graphs with maximum degree three and Induced Path Orientation are NP-hard, we construct a graph from an instance $I = (\text{Var}, \text{Cl})$ of Monotone Not-All-Equal 3-Sat.



(a) Variable gadget representing a variable occurring in clauses $c_1, \ldots, c_i$.



(b) Clause gadget representing a clause $c = (x_1, x_2, x_3)$.

Figure 3.3: Gadgets used in the reduction from a Monotone Not-All-Equal 3-Sat instance to construct a bipartite graph of maximum degree three.

The graph G consists of variable and clause gadgets, illustrated in Figure 3.3. Each variable gadget represents a variable $x \in \text{Var}$ that appears in i clauses $c_1, \ldots, c_i \in \text{Cl}$. For each clause c_j, the variable gadget has an interface vertex labeled c_j, which is adjacent to the corresponding interface vertex of the clause gadget representing c_j. Each clause gadget represents a clause $c = (x_1, x_2, x_3) \in \text{Cl}$ and contains three interface vertices x_1, x_2, and x_3, which are adjacent to the corresponding interface vertices of the variable gadgets representing x_1, x_2, and x_3. All other vertices of the variable and clause gadgets that are not interface vertices have no neighbors outside their respective gadget.

A variable gadget, shown in Figure 3.3a, representing a variable $x \in \text{Var}$ that appears in i clauses $c_1, \ldots, c_i$, consists of i paths $p_1, \ldots, p_i$ of length nine. The ninth vertex of each path p_j is the interface vertex c_j. Additionally, we add two edges between consecutive paths p_a and p_{a+1}. If a is odd, we add an edge between the sixth vertex of p_a and the seventh vertex of p_{a+1}, and another between the eighth vertex of p_a and the ninth vertex of p_{a+1}. If a is even, we add an edge between the third vertex of p_a and the second vertex of p_{a+1}, and another between the fifth vertex of p_a and the fourth vertex of p_{a+1}.

A clause gadget, shown in Figure 3.3b, representing a clause $c = (x_1, x_2, x_3)$, contains a cycle $C = (v_1, v_2, v_3, v_4, v_5, v_6, v_1)$ of six vertices. Each vertex v_j of the cycle has a pendant neighbor l_j that is not adjacent to any other vertex of the gadget. The vertices l_1, l_3, and l_5 serve as the interface vertices x_1, x_2, and x_3, respectively.

By construction, we obtain a bipartite graph $G = (V, E)$ with maximum degree three.

Next, we define a family of induced path partitions $\mathcal{P}(G)$ of the constructed graph. Each induced path partition $P \in \mathcal{P}(G)$ consists of $|P| = 3 \cdot |\mathrm{Cl}|$ undirected induced paths, where each induced path $p \in P$ represents the occurrence of a variable $x \in \mathrm{Var}$ in a clause $c \in \mathrm{Cl}$. One endpoint of p lies in the variable gadget of x, and the other lies in the clause gadget of c. The path p contains the interface vertex c of the variable gadget of x and the interface vertex x of the clause gadget of c, but no vertices outside these two gadgets.

The variable gadget representing x consists of paths of length nine, each having an interface vertex as one of its endpoints. The induced path p contains exactly those nine vertices that lie on the path in the variable gadget whose endpoint is the interface vertex corresponding to clause c.

The clause gadget representing the clause $c = (x_1, x_2, x_3)$, where $x \in \{x_1, x_2, x_3\}$, is covered by three induced paths in P. These paths are either of the form

$$p_1 = (l_2, v_2, v_1, l_1 = x_1, c, \dots), \quad p_2 = (l_4, v_4, v_3, l_3 = x_2, c, \dots), \quad p_3 = (l_6, v_6, v_5, l_5 = x_3, c, \dots),$$

or

$$p_1' = (l_6, v_6, v_1, l_1 = x_1, c, \dots), \quad p_2' = (l_2, v_2, v_3, l_3 = x_2, c, \dots), \quad p_3' = (l_4, v_4, v_5, l_5 = x_3, c, \dots).$$

Whether the vertices of the clause gadget are covered by the induced paths p_1, p_2, p_3 or by p_1', p_2', p_3' does not affect how the vertices of the other clause gadgets are covered. Thus, for each distinct combination determining how the vertices in the variable gadgets of G are covered, there exists a corresponding induced path partition $P \in \mathcal{P}(G)$.

3.4.3 Induced Path Orientation is NP-hard

We now show that for a Monotone Not-All-Equal 3-Sat instance $I = (\mathrm{Var}, \mathrm{Cl})$, the constructed graph $G = (V, E)$ with any induced path partition $P \in \mathcal{P}(G)$ satisfies that I is a yes-instance of Monotone Not-All-Equal 3-Sat if and only if the constructed graph with the induced path partition P is a yes-instance of Induced Path Orientation.

We use the fact that the set containing the first vertex of each directed induced path in an orientation $\overrightarrow{P}$ of the undirected induced paths in P is a zero forcing set of G if and only if G with respect to $\overrightarrow{P}$ contains no chain twist. Thus, it suffices to show that I is a yes-instance of Monotone Not-All-Equal 3-Sat if and only if there exists an orientation $\overrightarrow{P}$ of the undirected induced paths in P such that G, with respect to $\overrightarrow{P}$, contains no chain twist.

To do so, we first establish several lemmas that must hold for any orientation $\overrightarrow{P}$ of the undirected induced paths in an induced path partition $P \in \mathcal{P}(G)$. The first two lemmas will help us show that if $I = (\mathrm{Var}, \mathrm{Cl})$ is a yes-instance of Monotone Not-All-Equal 3-Sat, then there exists an orientation $\overrightarrow{P}$ of the undirected induced paths in each induced path partition $P \in \mathcal{P}(G)$ such that the directed induced paths in $\overrightarrow{P}$ are forcing chains of a zero forcing set.

Lemma 3.9: *If, for each variable gadget of G, the induced paths in $\overrightarrow{P}$ covering its vertices either all start or all end in that variable gadget, then every chain twist of G with respect to $\overrightarrow{P}$ contains no vertex of a variable gadget.*

Proof. Suppose, for contradiction, that there exists a chain twist C of G with respect to $\overrightarrow{P}$ that includes a vertex of a variable gadget. Then C must contain vertices of that variable gadget lying on at least two different directed induced paths in $\overrightarrow{P}$, where at least one of these induced paths starts in the variable gadget and another ends there, contradicting the assumption that, for each variable gadget, all induced paths covering its vertices either all start or all end in the gadget. ∎

Lemma 3.10: *If, for each clause gadget, there exists at least one directed induced path in $\overrightarrow{P}$ that starts in the clause gadget and another that ends there, then every chain twist of G with respect to $\overrightarrow{P}$ contains at least one vertex of a variable gadget.*

Proof. Suppose, for contradiction, that G contains a chain twist C with respect to $\overrightarrow{P}$ containing no vertex of a variable gadget. Then C must consist solely of vertices of a single clause gadget representing a clause $c = (x_1, x_2, x_3) \in \mathrm{Cl}$. The only directed cycles in G that contain only vertices of this clause gadget are

$$C_1 = (v_1, v_2, v_3, v_4, v_5, v_6, v_1) \quad \text{and} \quad C_2 = (v_1, v_6, v_5, v_4, v_3, v_2, v_1).$$

Thus, either $C = C_1$ or $C = C_2$ must hold. The clause gadget is covered by the three undirected induced paths

$$p_1 = (l_2, v_2, v_1, l_1 = x_1, c, \dots), \quad p_2 = (l_4, v_4, v_3, l_3 = x_2, c, \dots), \quad p_3 = (l_6, v_6, v_5, l_5 = x_3, c, \dots)$$

or

$$p_1' = (l_6, v_6, v_1, l_1 = x_1, c, \dots), \quad p_2' = (l_2, v_2, v_3, l_3 = x_2, c, \dots), \quad p_3' = (l_4, v_4, v_5, l_5 = x_3, c, \dots).$$

If the vertices of C are covered by p_1, p_2, p_3, then one of these undirected induced paths must be oriented such that it ends in the clause gadget, implying $C \neq C_2$. Since one of p_1, p_2, p_3 must also start in the clause gadget, $C \neq C_1$.

Similarly, if the vertices of C are covered by p_1', p_2', p_3', then $C \neq C_1$ because one of these paths must start in the clause gadget, and $C \neq C_2$ because one must end there.

In both cases, we have $C \neq C_1$ and $C \neq C_2$, contradicting the existence of such a chain twist. ∎

Combining Lemmas 3.9 and 3.10, we obtain the following result.

Lemma 3.11: *If $I = (Var, Cl)$ is a yes-instance of Monotone Not-All-Equal 3-Sat, then there exists an orientation $\overrightarrow{P}$ of the undirected induced paths in each induced path partition $P \in \mathcal{P}(G)$ such that the directed induced paths in $\overrightarrow{P}$ are forcing chains of a zero forcing set.*

Proof. Let $\beta : \mathrm{Var} \to \{\mathrm{true}, \mathrm{false}\}$ be a satisfying truth assignment of I. We define an orientation $\overrightarrow{P}$ of the induced path partition P as follows: each induced path $p \in P$ representing a variable $x \in \mathrm{Var}$ in a clause $c \in \mathrm{Cl}$ is oriented such that the first vertex of the directed path $\overrightarrow{p} \in \overrightarrow{P}$ lies in the variable gadget of x if $\beta(x) = \mathrm{true}$, and in the clause gadget of c if $\beta(x) = \mathrm{false}$.

Suppose, for contradiction, that there exists a chain twist C in G with respect to $\overrightarrow{P}$. As for each variable gadget, all directed induced paths in $\overrightarrow{P}$ either start or end there, Lemma 3.9 implies that C contains no vertex of a variable gadget. Since β is a satisfying assignment of I, in every clause $c \in \mathrm{Cl}$ there must exist a variable $x_a \in \mathrm{Var}$ with $\beta(x_a) = \mathrm{true}$ and another

variable $x_b \in \text{Var}$ with $\beta(x_b) = \text{false}$. Hence, the directed induced path corresponding to x_a ends in the clause gadget, while the one corresponding to x_b starts there. By Lemma 3.10, C must therefore contain at least one vertex of a variable gadget, contradicting the previous conclusion. Thus, G contains no chain twist with respect to $\overrightarrow{P}$, and the directed induced paths in $\overrightarrow{P}$ are forcing chains of a zero forcing set. ∎

We now show that if there exists an orientation $\overrightarrow{P}$ of the undirected induced paths in an induced path partition $P \in \mathcal{P}(G)$ such that the directed induced paths in $\overrightarrow{P}$ are forcing chains of a zero forcing set, then $I = (\text{Var}, \text{Cl})$ must be a yes-instance of Monotone Not-All-Equal 3-Sat. To establish this, we first prove two supporting lemmas.

Lemma 3.12: *If G contains no chain twist with respect to $\overrightarrow{P}$, then at least one directed induced path in $\overrightarrow{P}$ must start in each clause gadget, and at least one must end in each clause gadget.*

Proof. Suppose, for contradiction, that there exists a clause gadget in which no directed induced path in $\overrightarrow{P}$ ends. Then the leaves l_2, l_4, and l_6 must each be the first vertex of the directed induced path on which they lie in $\overrightarrow{P}$. If l_2, l_4, and l_6 are the first vertices of the undirected induced paths p_1, p_2, and p_3 in P, respectively, then C_2 would be a chain twist. If, instead, they lie on p'_1, p'_2, and p'_3 in P, then C_1 would be a chain twist. Both cases contradict the assumption that G contains no chain twist with respect to $\overrightarrow{P}$.

Similarly, suppose, for contradiction, that there exists a clause gadget in which no directed induced path in $\overrightarrow{P}$ starts. Then the leaves l_2, l_4, and l_6 must each be the last vertex of the directed induced paths on which they lie. If l_2, l_4, and l_6 are the last vertices of the undirected induced paths p_1, p_2, and p_3, then C_1 would be a chain twist. If, instead, they lie on p'_1, p'_2, and p'_3, then C_2 would be a chain twist. Both cases again contradict the assumption that G contains no chain twist.

Hence, in every clause gadget, at least one directed induced path must start and at least one must end. ∎

Lemma 3.13: *If G contains no chain twist with respect to $\overrightarrow{P}$, then for each variable gadget, all directed induced paths in $\overrightarrow{P}$ covering the vertices of that variable gadget must either all start or all end within the gadget.*

Proof. Suppose, for contradiction, that there exists a variable gadget in which at least one directed induced path starts and at least one ends. Then there must exist two consecutive directed induced paths p_j and p_{j+1} in the variable gadget, where one starts in the gadget and the other ends there. Since there are two edges in the variable gadget connecting vertices of p_j and p_{j+1}, this would create a chain twist, contradicting the assumption that G contains no chain twist with respect to $\overrightarrow{P}$. ∎

Combining Lemmas 3.12 and 3.13, we can now establish the following lemma.

Lemma 3.14: *If there exists an orientation $\overrightarrow{P}$ of the undirected induced paths in an induced path partition $P \in \mathcal{P}(G)$ such that the directed induced paths in $\overrightarrow{P}$ are forcing chains of a zero forcing set, then $I = (Var, Cl)$ is a yes-instance of Monotone Not-All-Equal 3-Sat.*

Proof. Define a truth assignment β as follows:

$$\beta(x) = \begin{cases} \text{true}, & \text{if there exists a } \overrightarrow{p} \in \overrightarrow{P} \text{ that starts in the variable gadget of } x, \\ \text{false}, & \text{otherwise.} \end{cases}$$

Since the directed induced paths in $\overrightarrow{P}$ are forcing chains of a zero forcing set, the graph G contains no chain twist with respect to $\overrightarrow{P}$. By Lemma 3.13. If $\beta(x)$ = true for some variable $x \in \text{Var}$, then all directed induced paths covering vertices of the variable gadget representing x must start in that gadget; if $\beta(x)$ = false, then they must all end there.

Now let $c = (x_1, x_2, x_3) \in \text{Cl}$ be a clause. By Lemma 3.12, there exists at least one directed induced path $p_a \in \overrightarrow{P}$ that starts in the clause gadget and at least one directed induced path $p_b \in \overrightarrow{P}$ that ends in the clause gadget. The directed path p_a must end in a variable gadget representing some $x_a \in \{x_1, x_2, x_3\}$, while p_b must end in another variable gadget representing some $x_b \in \{x_1, x_2, x_3\}$. Hence, $\beta(x_a)$ = false and $\beta(x_b)$ = true, ensuring that β satisfies c under the Not-All-Equal condition. Therefore, β satisfies all clauses, and I is a yes-instance of Monotone Not-All-Equal 3-Sat. ∎

We have now established all necessary lemmas to prove that Induced Path Orientation is NP-hard.

Theorem 3.15: *Induced Path Orientation is NP-hard.*

Proof. Let $I = (\text{Var}, \text{Cl})$ be an instance of Monotone Not-All-Equal 3-Sat. Construct the undirected graph $G = (V, E)$ as described, and let $P \in \mathcal{P}(G)$ be an induced path partition of G. By Lemma 3.11, if I is a yes-instance of Monotone Not-All-Equal 3-Sat, then there exists an orientation $\overrightarrow{P}$ of the undirected induced paths in P such that the directed induced paths in $\overrightarrow{P}$ are forcing chains of a zero forcing set. Conversely, by Lemma 3.14, if there exists an orientation $\overrightarrow{P}$ of the undirected induced paths in P such that the directed induced paths in $\overrightarrow{P}$ are forcing chains of a zero forcing set, then I must be a yes-instance of Monotone Not-All-Equal 3-Sat. Therefore, Induced Path Orientation is NP-hard. ∎

3.4.4 Zero Forcing is NP-hard

We now show that Zero Forcing is NP-hard on bipartite graphs with maximum degree three. We begin by proving that $\mathcal{P}(G)$ contains exactly all minimum induced path partitions of G.

Lemma 3.16: *A set of undirected induced paths P in G is a minimum induced path partition of G if and only if $P \in \mathcal{P}(G)$.*

Proof. Each variable gadget representing a variable $x \in \text{Var}$ that appears in i clauses has i leaves. Since each clause involves three variables, the variable gadgets contribute $3 \cdot |\text{Cl}|$ leaves in total. Each clause gadget also has three leaves, so the entire graph G contains $6 \cdot |\text{Cl}|$ leaves. Therefore, every induced path partition of G must contain at least $3 \cdot |\text{Cl}|$ induced paths. As all induced path partitions $P \in \mathcal{P}(G)$ contain $|P| = 3 \cdot |\text{Cl}|$ paths, they are all minimum induced path partitions of G, implying $P(G) = 3 \cdot |\text{Cl}|$.

Now consider a minimum induced path partition P of G. Since $P(G) = 3 \cdot |\text{Cl}|$, P must contain three undirected induced paths, each with distinct leaves as endpoints. We now show that $P \in \mathcal{P}(G)$ by describing how the induced paths in P must lie.

Consider a variable gadget corresponding to some $x \in \text{Var}$ appearing in clauses $c_1, \ldots, c_i \in \text{Cl}$. The gadget consists of i paths $p_1, \ldots, p_i$ of length nine, where one endpoint of each path is an interface vertex and the other a leaf. We now show that all nine vertices of each p_j must lie on the same induced path in $\overrightarrow{P}$.

Suppose, for contradiction, that two adjacent vertices $u, v \in V$ on p_j belong to different induced paths in $\overrightarrow{P}$. Then one of them, say u, must have degree three. By construction, no pair of adjacent vertices with degree three exist on the same path in a variable gadget, so v must have degree two or be a leaf. If v is a leaf, it would lie on an induced path of length one, contradicting that both endpoints of an induced path are distinct. If v has degree two, it would be an endpoint of an induced path, contradicting that only leaves can serve as endpoints. Hence, all nine vertices of p_j must belong to the same induced path in $\overrightarrow{P}$.

Since both endpoints of an induced path must be leaves, the interface vertex c_j cannot be an endpoint. As c_j is not adjacent to any other interface vertex of the same gadget, the interface vertex corresponding to c_j must lie on the same induced path as c_j.

Now consider a clause gadget representing $c = (x_1, x_2, x_3)$. It consists of a cycle $C = (v_1, v_2, \ldots, v_6, v_1)$, where each vertex v_j has a neighbor l_j that is not connected to any other vertex in the gadget. The vertices l_1, l_3, l_5 serve as the interface vertices corresponding to x_1, x_2, x_3, respectively, while l_2, l_4, l_6 are leaves and must therefore be endpoints of induced paths.

Consider the induced path $\overrightarrow{p_1} \in \overrightarrow{P}$ containing the interface vertex x_1. Since both endpoints of $\overrightarrow{p_1}$ must be leaves, it must include either $\{v_1, v_2, l_2\}$ or $\{v_1, v_6, l_6\}$. In the first case, the induced path $\overrightarrow{p_2}$ containing x_2 must include $\{v_3, v_4, l_4\}$, and $\overrightarrow{p_3}$ containing x_3 must include $\{v_5, v_6, l_6\}$. In the second case, if v_6 lies on $\overrightarrow{p_1}$, then $\overrightarrow{p_3}$ must include $\{v_5, v_4, l_4\}$ and $\overrightarrow{p_2}$ must include $\{v_3, v_2, l_2\}$. The induced path $\overrightarrow{p_1}$ cannot contain $\{v_1, v_2, v_3\}$ or $\{v_1, v_6, v_5\}$, as in both cases one of the leaves (l_2 or l_4) would lie on an induced path of length one, contradicting the assumption that both endpoints must be distinct.

Therefore, $P \in \mathcal{P}(G)$. ∎

Lemma 3.16 allows us to conclude the main result.

Theorem 3.17: *Zero Forcing is NP-hard on bipartite graphs with maximum degree three.*

Proof. Let $I = (\mathrm{Var}, \mathrm{Cl})$ be an instance of Monotone Not-All-Equal 3-Sat. Construct the undirected graph $G = (V, E)$ accordingly, and show that there exists a zero forcing set $S \subseteq V$ of size $|S| = 3 \cdot |\mathrm{Cl}|$ if and only if I is a yes-instance of Monotone Not-All-Equal 3-Sat. ∎

First, assume that I is a yes-instance of Monotone Not-All-Equal 3-Sat. Let $P \in \mathcal{P}(G)$ be a minimum induced path partition of G with $|P| = 3 \cdot |\mathrm{Cl}|$. By Lemma 3.11, there exists an orientation $\overrightarrow{P}$ of the undirected induced paths in P such that the directed induced paths in $\overrightarrow{P}$ are forcing chains of a zero forcing set S. Since $|S| = |P| = 3 \cdot |\mathrm{Cl}|$, S is a zero forcing set of G of the desired size.

Conversely, assume there exists a zero forcing set $S \subseteq V$ of G with $|S| = 3 \cdot |\mathrm{Cl}|$. Let P denote the set of forcing chains of S without orientation. Since $|P| = 3 \cdot |\mathrm{Cl}|$, P must be a minimum induced path partition of G, hence $P \in \mathcal{P}(G)$. Because P is a yes-instance of Induced Path Orientation, I must be a yes-instance of Monotone Not-All-Equal 3-Sat, by Lemma 3.14.

4 An Upper Edge Bound for Zero Forcing Sets

Building upon the concept of forcing chains of a zero forcing set introduced in the previous chapter, we now establish an inequality that holds for any undirected graph $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges, whenever a zero forcing set $S \subseteq V$ of size k exists.

Several results concerning bounds on the zero forcing number of an undirected graph G are already known. Amos et al. used the number of vertices $n = |V|$ and the maximum degree of a vertex Δ to prove the upper bound

$$Z(G) \leq \frac{(\Delta - 2)n + 2}{\Delta - 1}$$

for connected graphs with $\Delta \geq 2$ [ACDP15]. Gentner et al. showed that this upper bound is tight only for cycle graphs C_n, complete graphs K_n, and complete bipartite graphs $K_{\Delta,\Delta}$ [GPRS16]. They further established that every triangle-free graph with minimum degree $\delta \geq 2$ satisfies the lower bound

$$Z(G) \geq 2\delta - 2$$

[GPRS16]. Furthermore, Dávila et al. employed the girth g and the minimum degree δ to prove that

$$Z(G) \geq (g - 3)(\delta - 2) + \delta$$

for graphs with $g \geq 5$ and $\delta \geq 2$ [DKS18].

Other graph parameters have also been used to derive bounds. Brešar et al. proved that

$$Z(G) + \gamma_t \leq n \quad \text{and} \quad Z(G) \leq n - \frac{\Gamma_t}{2},$$

where γ_t and Γ_t denote the total domination number and upper total domination number of G, respectively, for any graph without isolated vertices [BCDH24]. Jing et al. related the zero forcing number to the matching number α' and the cyclomatic number c, showing that

$$n - 2\alpha' \leq Z(G) \leq n - \alpha' + c - 1$$

for graphs with $n \geq 3$ [JZJ23]. Moreover, in Section 3.1 we established that the induced path partition number of a graph provides an upper bound for its zero forcing number.

To the best of our knowledge, the inequality we present here, describing the relationship between the number of vertices n, the number of edges m, and the size k of a zero forcing set, has not been previously reported.

The chapter is organized as follows. In Section 4.1, we prove that

$$m \leq k \cdot \left(n - \frac{k+1}{2}\right)$$

for any graph G with a zero forcing set S of size k. In Section 4.2, we show how edges can be added to G such that the inequality becomes tight while preserving S as a zero forcing set. Section 4.3 demonstrates that if the inequality is tight and $S \neq V$, then S must be a minimum zero forcing set. Finally, in Section 4.4, we illustrate our results on a concrete example.

4.1 Deriving the Upper Bound

We begin by deriving an upper bound on m.

Theorem 4.1: *Let S be a zero forcing set of G with size k. Then*

$$m \leq k \cdot \left(n - \frac{k+1}{2}\right).$$

Proof. Let $\overrightarrow{P}$ be the set of forcing chains obtained by iteratively applying the color-change rule. Fix an order in which the vertices of $V \setminus S$ are colored blue. We distinguish three types of edges:

1. edges between vertices lying on the same forcing chain,
2. edges between vertices both contained in S, and
3. edges between vertices on different forcing chains, where at least one of the vertices is not in S.

We derive an upper bound for each type separately and then combine them.

First, consider vertices $u, v \in V$ lying on the same forcing chain $\overrightarrow{p} \in \overrightarrow{P}$ of length ℓ. Since u and v are adjacent if and only if they are consecutive on $\overrightarrow{p}$, there are exactly $\ell - 1$ edges among the vertices on $\overrightarrow{p}$. As each vertex lies on exactly one forcing chain, and $\overrightarrow{P}$ contains exactly k induced paths, there are in total at most $n - k$ edges between vertices lying on the same forcing chain in $\overrightarrow{P}$.

Next, consider edges between pairs of vertices both contained in S. Since S contains k vertices, there are at most

$$\frac{k(k-1)}{2}$$

edges between pairs of vertices in S.

Now consider a vertex $v \in V \setminus S$ lying on a forcing chain $\overrightarrow{p_v} \in \overrightarrow{P}$. We show that v can only be adjacent to a vertex $x \in V$ on a different forcing chain $\overrightarrow{p_x} \in \overrightarrow{P} \setminus \{\overrightarrow{p_v}\}$ if x is white or the last blue vertex on $\overrightarrow{p_x}$ when x is colored blue.

Suppose, for contradiction, that v is adjacent to a vertex $x \in \overrightarrow{p_x}$ such that both x and its successor y on $\overrightarrow{p_x}$ are already blue when v is colored blue. Then the color-change rule $x \to y$ must have been applied while v was still white, contradicting that y must be the only white neighbor of x to apply the color-change rule. Hence, v can only be adjacent to vertices lying on another forcing chain that are either white or the final blue vertex of their forcing chain when v is colored blue.

Therefore, for each $v \in V \setminus S$, the only blue neighbors it can have at the time it is colored blue are the last blue vertices of the other forcing chains, of which there are at most $k - 1$. Consequently, there are at most

$$|V \setminus S| \cdot (k-1) = (n-k)(k-1)$$

edges between vertices on different forcing chains where at least one vertex is not in S.

We thus obtain the following upper bound on the total number of edges in G:

$$\begin{aligned} m &\leq (n-k) + \frac{k(k-1)}{2} + (n-k)(k-1) \\ &= k(n-k) + \frac{k(k-1)}{2} \\ &= k\left(n-k+\frac{k-1}{2}\right) \\ &= k \cdot \left(n - \frac{k+1}{2}\right). \end{aligned}$$

■

4.2 Constructing Graphs Attaining the Upper Bound

We now show that edges can be added to an undirected graph such that the upper bound for the number of edges given in Theorem 4.1 becomes tight.

Theorem 4.2: *Let S be a zero forcing set of G with size k. Then there exists a graph $G' = (V, E')$ with $E' \supseteq E$, such that S remains a zero forcing set of G and*

$$|E'| = k \cdot \left(n - \frac{k+1}{2}\right).$$

Proof. Let $\overrightarrow{P}$ be the set of forcing chains obtained by iteratively applying the color-change rule. Fix an order in which the vertices of $V \setminus S$ are colored blue. As in the proof of Theorem 4.1, we again distinguish three types of edges. We will show that edges can be added to G until G contains at least as many edges of each type as given by the upper bound in Theorem 4.1.

First, note that there are at least $n-k$ edges in E between vertices lying on the same forcing chain. Since every vertex $v \in V \setminus S$ is colored blue by a unique color-change rule, we apply exactly $|V| - |S| = n - k$ color-change rules. Each color-change rule $u \to V$ requires u and v to be adjacent, so there are at least $n-k$ edges between pairs of vertices on the same forcing. Thus

$$|E| \geq n - k.$$

Now, consider two distinct vertices $v, x \in S$ that are not adjacent. We show that S remains a zero forcing set if we add the edge $\{v, x\}$. Assume v is not the last vertex on its forcing chain $\overrightarrow{p_v} \in \overrightarrow{P}$ and let w be its successor on $\overrightarrow{p_v}$. We can still apply the color-change rule $v \to w$ after adding the edge $\{v, x\}$. Since x is initially blue, w remains the only white neighbor of v when $v \to w$ is applied. Similarly, if x is not the last vertex on its forcing chain $\overrightarrow{p_x} \in \overrightarrow{P}$, let y be its successor on $\overrightarrow{p_x}$. We can still apply $x \to y$ because v is initially blue. Every other color-change rule $z_1 \to z_2$ is unaffected by adding $\{v, x\}$, since $z_1 \notin \{v, x\}$. Hence, S remains a zero forcing set if we add $\{v, x\}$.

Therefore, we can add all missing edges between vertices in S, obtaining the graph

$$G^{(1)} = (V, E^{(1)}) \quad \text{with} \quad E^{(1)} = E \cup (S \times S).$$

The set S remains a zero forcing set of $G^{(1)}$, and the same sequence of color-change rules can be applied to color all vertices in $V \setminus S$ blue. The graph $G^{(1)}$ then contains at least

$$|E^{(1)}| \geq (n-k) + \frac{k \cdot (k-1)}{2}$$

edges.

Next, let $v \in V \setminus S$ be a vertex on a forcing chain $\overrightarrow{p_v} \in \overrightarrow{P}$, and let x be the last blue vertex of another forcing chain $\overrightarrow{p_x} \in \overrightarrow{P} \setminus \{\overrightarrow{p_v}\}$ at the time when v is colored blue. In the following we show that S remains a zero forcing set if we add the edge $\{v, x\}$.

First, assume v is not the last vertex on $\overrightarrow{p_v}$ and w is its successor on $\overrightarrow{p_v}$. The we can still apply the color-change rule $v \to w$ because x is already colored blue when v is colored blue. If x is not the last vertex on $\overrightarrow{p_x}$ and y is its successor on $\overrightarrow{p_x}$, the we can also still apply the color-change rule $x \to y$ because the vertex x is the last blue vertex on $\overrightarrow{p_x}$ when v is colored blue. Therefore, we apply the color-change rule $x \to y$ after v is colored blue, so we can also still apply the color-change rule $x \to y$ if we add an edge between v and x. Again no other color change rule $z_1 \to z_2$ is affect by adding the edge $\{v, x\}$, since $z_1 \notin \{v, x\}$.

Thus, for each vertex $v \in V \setminus S$, we may add edges between v and the last blue vertex of every forcing chain over than its own. We obtain the graph

$$G^{(2)} = (V, E^{(2)}).$$

The set S remains a zero forcing set, and we can color all vertices in $V \setminus S$ blue by applying the same sequence of color-change rules. Since $V \setminus S$ contains $n - k$ vertices and for each vertex in $V \setminus S$ $E^{(2)}$ contains $k - 1$ edges between v and a vertex lying on another forcing chain, that is colored blue before v, we obtain

$$\begin{aligned}
|E^{(2)}| &\geq (n-k) + \frac{k \cdot (k-1)}{2} + (n-k) \cdot (k-1) \\
&= k \cdot (n-k) + \frac{k \cdot (k-1)}{2} \\
&= k \cdot \left(n - k + \frac{k}{2} - \frac{1}{2} \right) \\
&= k \cdot \left(n - \frac{k+1}{2} \right).
\end{aligned}$$

By Theorem 4.1, a graph with zero forcing set S contains at most $k \cdot \left(n - \frac{k+1}{2}\right)$ edges. Therefore, $G^{(2)}$ is a graph with $|E^{(2)}| = k \cdot \left(n - \frac{k+1}{2}\right)$, and the bound is tight. ∎

4.3 Characterizing Graphs Attaining the Upper Bound

We now prove that if the edge bound is tight and $S \neq V$, then S must be a minimum zero forcing set.

Theorem 4.3: *Let $G = (V, E)$ be an undirected graph, and let $S \subset V$ be a zero forcing set of G with $S \neq V$. If*

$$m = k \cdot \left(n - \frac{k+1}{2} \right),$$

then S is a minimum zero forcing set of G.

Proof. Since $S \neq V$, we have $k < n$. Suppose, for contradiction, that S is not a minimum zero forcing set. Then there exists a zero forcing set $S' \subset V$ with $|S'| = k - c$ for some positive integer c. Since S' is also a zero forcing set, Theorem 4.1 implies that the number of edges in G satisfies

$$\begin{aligned} m &\leq (k-c) \cdot \left(n - \frac{(k-c)+1}{2} \right) \\ &= k \cdot \left(n - \frac{(k-c)+1}{2} \right) - c \cdot \left(n - \frac{(k-c)+1}{2} \right) \\ &= \underbrace{k \cdot \left(n - \frac{k+1}{2} \right)}_{=m} + \frac{kc}{2} - c \cdot \left(n - \frac{k}{2} \right) - c \cdot \frac{c-1}{2} \\ &= m + c \cdot \underbrace{(k-n)}_{<0} + c \cdot \underbrace{\frac{1-c}{2}}_{\leq 0} \\ &< m, \end{aligned}$$

which leads to the contradiction $m < m$. Hence, our assumption is false, and S must be a minimum zero forcing set. ∎

If $S = V$ and the upper bound is tight, then S does not have to be a minimum zero forcing set. Consider the complete graph on n vertices, K_n, with $n \geq 2$. In this case, the number of edges is given by

$$m = \frac{n(n-1)}{2},$$

and $S = V$ is a zero forcing set of size n. We compute:

$$k \cdot \left(n - \frac{k+1}{2} \right) = n \cdot \left(n - \frac{n+1}{2} \right) = \frac{n(n-1)}{2} = m,$$

so the bound is tight. However, S is not a minimum zero forcing set. Since G is complete and $n \geq 2$, there exists an edge $\{u, v\} \in E$. Then, the set $S' = V \setminus \{v\}$ is also a zero forcing set of G, because u is initially colored blue and v is its only white neighbor. Therefore, we can apply the color-change rule $u \to v$, and color the only vertex in $V \setminus S'$ blue. Hence, S' is a zero forcing set of smaller size than S, and thus S is not a minimum zero forcing set.

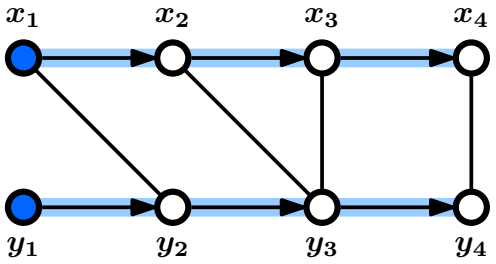
In general, if $S = V$, then S is a minimum zero forcing set if and only if $E = \varnothing$. If $E = \varnothing$, then no vertex has a neighbor, so the color-change rule cannot be applied. In this case, $S = V$ is the only zero forcing set of G and is therefore a minimum zero forcing set.

If $E \neq \varnothing$, then there exists an edge $\{u, v\} \in E$. In this case, the set $S' = V \setminus \{v\}$ is again a zero forcing set, because u is initially blue and v is its only white neighbor. Thus we can color v blue by applying $u \to v$. Therefore, S' is a smaller zero forcing set than $S = V$, and S cannot be a minimum zero forcing set.

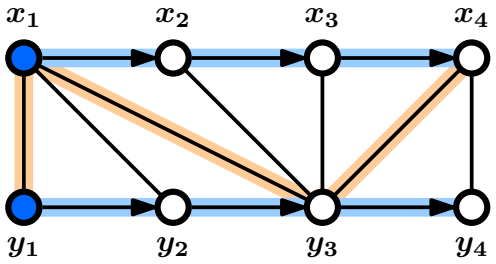
4.4 Example Graph Attaining the Upper Bound

Figure 4.1a shows a graph $G = (V, E)$ with $n = 8$ vertices and $m = 10$ edges. The set $S = \{x_1, y_1\}$ is a zero forcing set of G, since we can color all vertices blue by applying the sequence of color-change rules

$$y_1 \to y_2, \quad y_2 \to y_3, \quad x_1 \to x_2, \quad x_2 \to x_3, \quad x_3 \to x_4, \quad y_3 \to y_4.$$



(a) A graph with zero forcing set $S = \{x_1, y_1\}$ for which the upper bound on m is not tight.



(b) After adding three edges, S remains a zero forcing set and the upper bound on m becomes tight.

Figure 4.1: Illustration of adding edges to a graph to attain the upper bound.

For the graph G the upper bound on m is not tight, since

$$k \cdot \left(n - \frac{k+1}{2}\right) = 2 \cdot \left(8 - \frac{2+1}{2}\right) = 10 < 13 = m.$$

By Theorem 4.2, it is possible to add three edges to G, obtaining a graph $G' = (V, E')$ for which S remains a zero forcing set and the upper bound on the number of edges is tight. Figure 4.1b shows the graph G', which contains 13 edges. The newly added edges are highlighted in orange. Since the same sequence of color-change rules used for G can also be applied to G', the set S remains a zero forcing set of G'. Because the upper bound on the number of edges in G' with the zero forcing set S is tight, the set S is a minimum zero forcing set of G'. Therefore, by Theorem 4.3, the zero forcing number of G' is two.

5 Zero Forcing on Cacti

In this chapter, we present an algorithm for computing the zero forcing number of a cactus. Darren D. Row [Row12b] provides a linear-time algorithm for Zero Forcing on trees and unicyclic graphs. To the best of our knowledge, no linear-time algorithm has yet been established for Zero Forcing on cacti. Our approach builds upon the following theorem by Darren D. Row, which states that for any cactus, the zero forcing number equals its induced path partition number [Row12a].

Theorem 5.1: *For any cactus G, it holds that $P(G) = Z(G)$.*

Consequently, to determine whether there exist a zero forcing set of a cactus $G = (V, E)$ of size at most k, it suffices to check whether there exists an induced path partition of G of size at most k. We construct a minimum induced path partition by iteratively applying reduction rules. Each reduction rule removes at least one vertex from G and potentially decreases the number of available paths k.

This chapter is structured as follows. We begin by motivating the concepts of *mandatory* and *optional* vertices, which lead to the definition of the more general problem Induced Path Partition Extension in Section 5.1. Section 5.2 introduces the necessary reduction rules. In Section 5.3, we show how these reduction rules allow us to solve the Induced Path Partition Extension problem in linear time, thereby enabling us to decide in linear time whether a zero forcing set of a cactus of size at most k exists. Finally, Section 5.4 describes how to compute in linear time an orientation $\overrightarrow{P}$ of the induced paths in an induced path partition P such that the cactus with respect to $\overrightarrow{P}$ contains no chain twist. The directed paths in $\overrightarrow{P}$ then are forcing chains, and the set containing the first vertex of each directed induced path is a zero forcing set. This yields a linear-time algorithm for computing a minimum zero forcing set of a cactus.

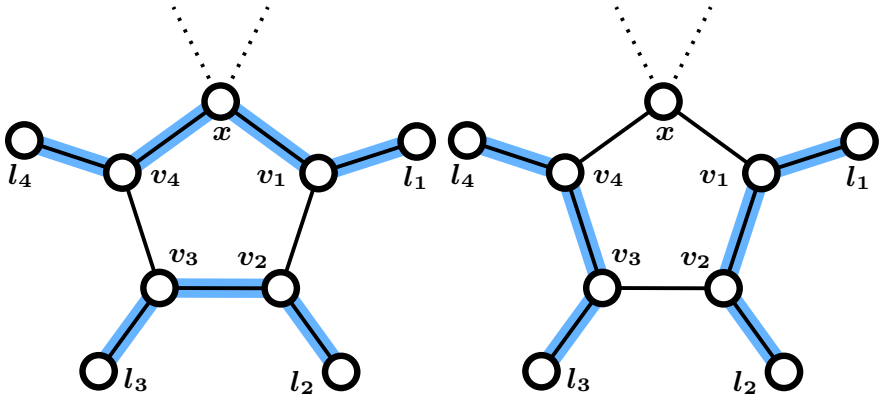
5.1 Introducing Mandatory and Optional Vertices

In some cases, we cannot greedily decide which induced paths to choose. Consider the subgraph in Figure 5.1. The vertex x is the unique attachment of this subgraph to the remainder of the graph. The subgraph contains four leaves l_1, l_2, l_3, and l_4. Any minimum induced path partition must contain at least two paths, since each l_i must appear as an endpoint of a path. Figures 5.1a and 5.1b show two valid induced path partitions for the subgraph: one in which x is included in a path, and one in which it is excluded.

Since the optimal choice depends on how the subgraph connects to the rest of the graph, we must defer the decision. For example, if x is adjacent only to v_1 and v_4, we would choose the paths including x. However, if the rest of the graph consists of two leaves y and z with x as their only neighbor, we would prefer the paths that exclude x. In this case, we could cover the remaining vertices x, y, z with a single induced path $p = (y, x, z)$.

This motivates distinguishing between *mandatory* and *optional* vertices.

Mandatory Vertex. A mandatory vertex is a vertex that must lie on an induced path of an induced path partition.



(a) Induced paths covering x. **(b)** Induced paths not covering x.

Figure 5.1: Example illustrating the need for optional vertices.

Optional Vertex. An optional vertex is a vertex that does not necessarily lie on a path of an induced path partition.

The distinction between mandatory and optional vertices allows us to propose the decision which paths to choose. We can remove all vertices of the shown subgraph except x, decrease k by two, and mark x as optional. We receive a new graph, which contains a zero forcing set of size k if and only if the original graph has a zero forcing set of size at most the initial value k. The general idea of optional vertices is, that a mandatory vertex becomes optional if there exist two sets of induced that cover the vertices of a subgraph where one includes a certain vertex and the other excludes the vertex. A minimum induced path partition of the graph must contain the induced paths of one of the sets. We thus introduce the more general problem Induced Path Partition Extension.

Induced Path Partition Extension. Given a graph $G = (V, E)$ consisting of optional and mandatory vertices and a parameter k, is there a set of k induced paths such that every mandatory vertex lies on exactly one induced path and every optional vertex lies on at most one induced path?

The problem Induced Path Partition is a special case of Induced Path Partition Extension in which all vertices are mandatory. In the following, we present an algorithm that solves the Induced Path Partition Extension problem.

5.2 Reduction Rules for Induced Path Partition

To determine whether there exists an induced path partition of a cactus of size at most k, we apply a sequence of reduction rules. Each reduction rule identifies a specific substructure in the graph, describes how it can be simplified, and specifies how this simplification affects the value of k.

Each rule identifies a specific substructure in the graph. If such a structure is found, the reduction rule can be applied to simplify the graph. A reduction rule may remove vertices, mark vertices as mandatory or optional, and update the parameter k accordingly. Crucially, there exists an induced path partition of the original graph of size at most k if and only if there

exist an induced path partition of the reduced graph of size at most the updated value of k. By exhaustively applying reduction rules, we aim to reduce the graph to a trivially solvable instance. In particular, if all vertices are removed and the final value of k is non-negative, then an induced path partition of the original graph of size at most k exists.

In this section, we present the reduction rules used by our algorithm to decide whether there exists an induced path partition of a cactus of size at most k. For each reduction rule, we provide a formal definition, a proof of correctness, and a figure illustrating the reduction rule. In our illustrations, we present mandatory vertices as white circles, optional vertices as black circles and vertices where it does not matter for the reduction rule whether the vertex is mandatory or optional as gray circles. We mark removed vertices with a red cross.

5.2.1 Reduction Rules for Trees

The first group of reduction rules applies to vertices that do not lie on any cycle.

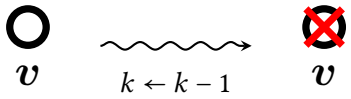


Figure 5.2: Reduction Isolated Vertex.

Reduction Isolated Vertex. Let v be an isolated mandatory vertex. Then remove v and decrease k by one. See Figure 5.2.

Proof. Since v has no neighbors, it cannot lie on an induced path with any other vertex. Therefore, $p = (v)$ must be an induced path in every induced path partition. We may thus remove v and must consequently decrease k by one. ∎

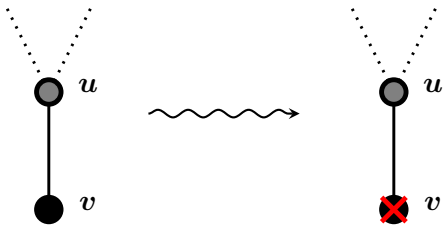


Figure 5.3: Reduction Optional Leaf.

Reduction Optional Leaf. Let v be an optional vertex with $\deg(v) \leq 1$. Then, remove v. See Figure 5.3.

Proof. We show that there exists a minimum induced path partition of G in which v does not appear on any induced path. Let P be a minimum induced path partition of G. If v lies on an induced path $p \in P$, then p must have length at least two. Otherwise, v would be the only vertex on p and $P \setminus p$ would still be an induced path partition, contradicting the assumption that P is a minimum induced path partition. Since v has at most one neighbor, v must be an endpoint of p. Hence, we can remove v from p, obtaining a minimum induced path partition in which v does not lie on any induced path. ∎

Next, we show that from every minimum induced path partition of G, and for every mandatory leaf, we can construct a minimum induced path partition in which the leaf lies on an induced path together with its parent.

Theorem 5.2: *Let $G = (V, E)$ be a graph with mandatory and optional vertices. Let $l \in V$ be a mandatory leaf with parent $v \in V$, and let P be a minimum induced path partition of G. Then there exists a minimum induced path partition P' of G such that*

1. *l and v lie on the same induced path in P', and*
2. *for every pair of adjacent vertices $x, y \in V$ with $v \notin \{x, y\}$, the vertices x and y are consecutive on an induced path in P' if and only if they are consecutive on an induced path in P.*

Proof. Let P be a minimum induced path partition of G, and assume that l and v lie on different induced paths in P. We construct a minimum induced path partition P' from P such that l and v lie on the same induced path.

Since l has no neighbors other than v, and v lies on a different induced path, l must lie on an induced path $p_l = (l)$ of length one.

If v is optional and does not lie on any induced path in P, we can extend p_l to $p_l' = (l, v)$. Then

$$P' = P \setminus \{p_l\} \cup \{p_l'\}$$

is a minimum induced path partition of G in which l and v lie on the same induced path. Since only p_l is modified, every pair of adjacent vertices $x, y \in V$ with $v \notin \{x, y\}$ that were consecutive on an induced path in P remain so in P'.

Otherwise, let $p_v \in P$ be the induced path containing v. Because l and v lie on different induced paths and $p_l = (l)$, the vertex v cannot be an endpoint of p_v. Otherwise, we could merge p_l and p_v into a single induced path, contradicting the assumption that P is a minimum induced path partition.

Therefore, p_v must be of the form $p_v = (\dots, u, v, w, \dots)$, where v is not an endpoint. We now split p_v into two induced paths:

$$p_u = (\dots, u, v), \quad p_w = (w, \dots).$$

We then extend p_u to include l, obtaining $p_u' = (\dots, u, v, l)$. Define

$$P' = P \setminus \{p_v, p_l\} \cup \{p_u', p_w\}.$$

This is again a minimum induced path partition of G, in which l and v lie on the same induced path.

It remains to show that each pair of adjacent vertices $x, y \notin \{v\}$ that lie on the same induced path in P also lie on the same induced path in P'. Let $p_{x,y} \in P$ be the induced path containing x and y. Since x and y are adjacent and $p_{x,y}$ is an induced path, they must be consecutive vertices on $p_{x,y}$. The only pair of consecutive vertices on an induced path in P that are not consecutive on an induced path in P' are v and w. However, since $v \in \{v, w\}$, every pair of adjacent vertices $x, y \notin \{v\}$ that are consecutive on an induced path in P remain consecutive on an induced path in P'. ∎

Theorem 5.2 also shows that for every graph with a leaf, there exists a minimum induced path partition in which the leaf lies on the same induced path as its parent. We use this for the following reduction rules. We may apply the next reduction rule, when we have a mandatory leaf whose parent has at most one other neighbor besides the leaf.

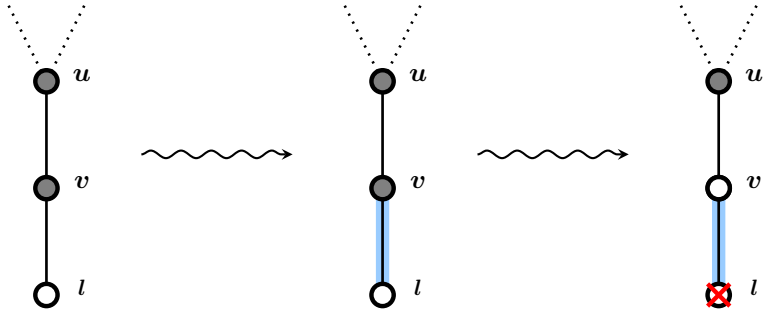


Figure 5.4: Reduction Path.

Reduction Path. Let l be a mandatory leaf and let v be its parent with $\deg(v) \leq 2$. Then, remove l and mark v as mandatory. See Figure 5.4.

Proof. By Theorem 5.2, there exists a minimum induced path partition of G in which l and v lie on the same induced path. We mark v as mandatory because there exists a minimum induced path partition in which v lies on an induced path. We can also remove l, because by marking v as mandatory we ensure that v lies on an induced path. For every induced path partition of $G - l$, the vertex v is an endpoint of an induced path to which we can append l, obtaining a minimum induced path partition of G with the same number of induced paths. ∎

Next, we present a reduction rule for a vertex with two adjacent vertices that are mandatory leaves.

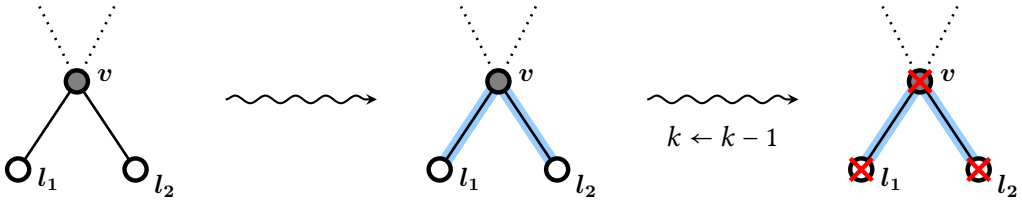


Figure 5.5: Reduction Pendant Star.

Reduction Pendant Star. Let l_1 and l_2 be mandatory leaves with the same neighbor v. Then, remove l_1, l_2, and v, and decrease k by one. See Figure 5.5.

Proof. We show that there exists a minimum induced path partition P' of G that contains the induced path $p' = (l_1, v, l_2)$. By Theorem 5.2, there exists a minimum induced path partition P of G such that l_1 and v lie on the same induced path $p \in P$. If l_2 also lies on p, then $p' = p$ and P already contains p'.

Now assume l_2 does not lie on p. Then l_2 must lie on the induced path $p_{l_2} = (l_2)$ of length one. The vertex v is not an endpoint of p, because otherwise we could merge p and p_{l_2} into a single induced path, contradicting the assumption that P is a minimum induced path partition. Therefore, p must be of the form $p = (l_1, v, w, \dots)$ with $w \neq l_2$. We split p into two induced paths

$$p^{(1)} = (l_1, v), \qquad p^{(2)} = (w, \dots).$$

We then merge $p^{(1)}$ with p_{l_2} into a single induced path, obtaining $p' = (l_1, v, l_2)$. Hence

$$P' = P \setminus \{p, p_{l_2}\} \cup \{p', p^{(2)}\}$$

is a minimum induced path partition containing $p' = (l_1, v, l_2)$. We can therefore remove l_1, l_2, and v, as we can obtain a minimum induced path partition of G by including p' into a minimum induced path partition of $G - \{l_1, l_2, v\}$. Consequently, we must decrease k by one. ∎

5.2.2 Reduction Rules for Cycles

In this subsection, we present reduction rules for cycles. Let $C = (c_1, \dots, c_n, c_1)$ be a cycle. We distinguish between different cases depending on whether the vertices $c_2, \dots, c_n$ are mandatory or optional, and whether any vertex has neighbors outside the cycle.

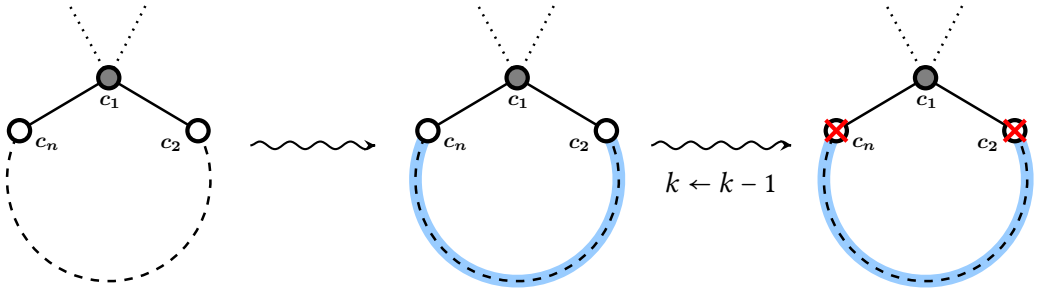


Figure 5.6: Reduction Cycle without Optional Vertices.

Reduction Cycle without Optional Vertices. Let $C = (c_1, \dots, c_n, c_1)$ be a cycle such that all vertices c_x on C with $1 < x \leq n$ are mandatory and have degree two. Then remove the vertices $c_2, \dots, c_n$ and decrease k by one. See Figure 5.6.

Proof. We show that there exists a minimum induced path partition P' of G that contains the induced path $p = (c_2, \dots, c_n)$ by constructing P' from a minimum induced path partition P without increasing its size.

First, assume that c_1 is an optional vertex and does not lie on any path of P. Then $(c_2, \dots, c_n)$ must be an induced path in P. Otherwise, there would exist a vertex $c_i \in \{c_2, \dots, c_{n-1}\}$ such that c_i lies on an induced path $p_i \in P$ and c_{i+1} lies on a different induced path $p_{i+1} \neq p_i$. In this case, we could merge p_i and p_{i+1} into a single induced path, contradicting the assumption that P is a minimum induced path partition.

Now assume that c_1 lies on an induced path $p_1 \in P$. The path p_1 cannot contain all vertices of C. Therefore, there must exist a path $p_{ab} = (c_a, \dots, c_b) \in P$ with $c_a \in \{c_2, \dots, c_{n-1}\}$ and $c_b \in \{c_{a+1}, c_n\}$.

If $c_a \neq c_2$, then c_{a-1} must be an endpoint of p_1. Otherwise, if c_a would lie on a different induced path $p_{a-1} \neq p_1$, we could again merge p_{ab} and p_{a-1} into a single induced path, contradicting the assumption that P is a minimum induced path partition. Thus, p_1 has the form $p_1 = (c_{a-1}, \dots, c_2, c_1, \dots)$. We can split p_1 into two induced paths

$$p_1^{(1)} = (c_{a-1}, \dots, c_2), \quad p_1^{(2)} = (c_1, \dots).$$

We can then merge p_{ab} and $p_1^{(1)}$ into a new induced path $p_{2b} = (c_2, \ldots, c_{a-1}, c_a, \ldots, c_b)$, and

$$P^{(1)} = P \setminus \{p_{ab}, p_1\} \cup \{p_{2b}, p_1^{(2)}\}.$$

is a minimum induced path partition of G containing an induced path of the form $(c_2, \ldots, c_{a-1}, c_a, \ldots, c_b)$.

If $c_b \neq c_n$, then c_{b+1} must lie on $p_1^{(2)}$ because $P^{(1)}$ is a minimum induced path partition. We can split $p_1^{(2)}$ into two paths

$$p_1^{(3)} = (c_{b+1}, \ldots, c_n), \quad p_1^{(4)} = (c_1, \ldots).$$

We can then merge $p_2 b$ and $p_1^{(3)}$ into a single induced path $p_{2n} = (c_2, c_2, \ldots, c_n) = p$. Hence,

$$P^{(2)} = P^{(1)} \setminus \{p_{2b}, p_1^{(2)}\} \cup \{p_{2n}, p_1^{(4)}\}$$

is a minimum induced path partition of G containing p. ∎

Next, assume that c_2 and c_n are mandatory vertices, and that there exists an optional vertex c_a on C with $2 < a < n$.

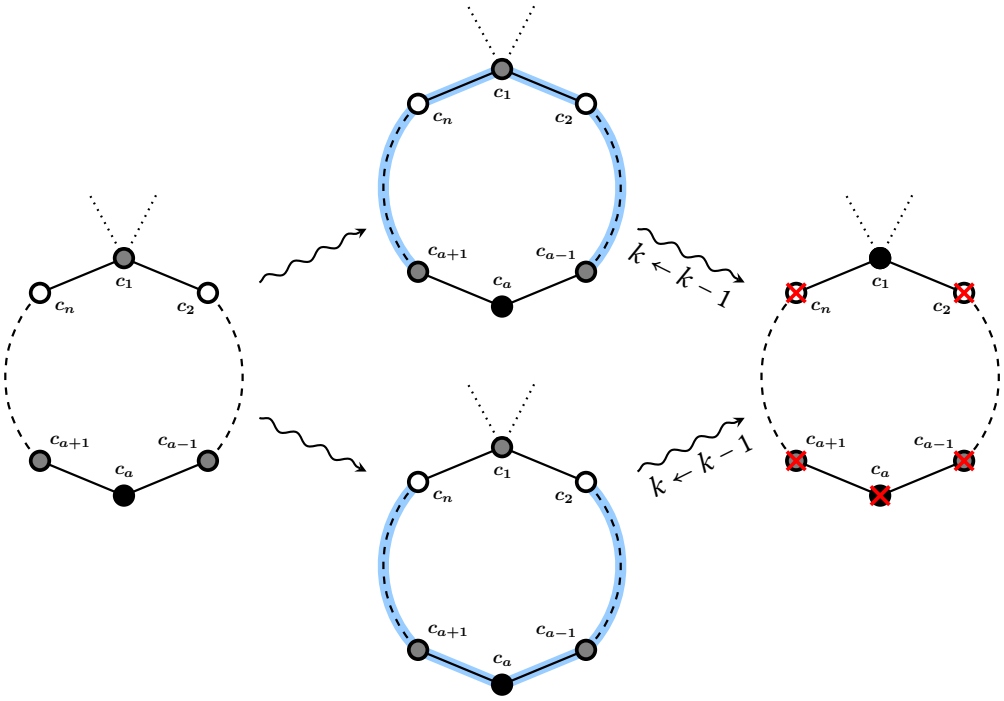


Figure 5.7: Reduction Cycle with Optional Vertex and c_2, c_n Mandatory.

Reduction Cycle with Optional Vertex and c_2, c_n Mandatory. Let $C = (c_1, \ldots, c_n, c_1)$ be a cycle such that $\deg(c_x) = 2$ for every vertex c_x on C with $1 < x \leq n$. Let c_2 and c_n be mandatory vertices, and let c_a be an optional vertex on C with $2 < a < n$. Then remove $c_2, \ldots, c_n$, mark c_1 as optional, and decrease k by one. See Figure 5.7.

Proof. We show that there exists a minimum induced path partition of G that contains either of the induced paths

$$p_1 = (c_2, c_3, \dots, c_n) \quad \text{or} \quad p_2 = (c_{a+1}, c_{a+2}, \dots, c_n, c_1, \dots, c_{a-1}).$$

Since both induced paths cover all mandatory vertices $c_2, c_3, \dots, c_n$, and since c_1 lies only on p_2, we remove $c_2, \dots, c_n$, mark c_1 as optional, and decrease k by one.

Let P be a minimum induced path partition of G, and let $p \in P$ be the induced path containing c_1. First, consider the case where c_1 is the only vertex of C lying on p. Since all remaining vertices of C can be covered by the induced path p_1, and c_2 and c_n are mandatory, P must contain p_1.

Next, suppose that both vertices c_2 and c_n lie on p. Then all mandatory vertices of C must lie on p because otherwise we could replace p and all other induced paths containing vertices of C by p_2, contradicting the assumption that P is a minimum induced path partition. Hence, we replace p by p_2 in P to obtain a minimum induced path partition which containing p_2.

If exactly one of the vertices c_2 and c_n lies on p, then p must be of the form

$$p = (\dots, c_j, c_1, v, \dots),$$

where $c_j \in \{c_2, c_n\}$ and v is a vertex not lying on C. The remaining vertex $c_\ell \in \{c_2, c_n\} \setminus \{c_j\}$ cannot lie on p, since p is induced. Let p_ℓ be the induced path containing c_ℓ.

Every mandatory vertex on C must lie either on p or on p_ℓ because otherwise, we could append all vertices of C that lie on neither p nor p_ℓ to p_ℓ. Then we could remove all other induced paths in P containing vertices of C, and obtain an induced path partition of smaller cardinality, ontradicting that P is a minimum induced path partition. Thus, all mandatory vertices of C lie either on p or p_ℓ.

We now split p into two induced paths:

$$p^{(1)} = (\dots, c_j), \quad \text{and} \quad p^{(2)} = (c_1, \dots).$$

Then

$$P' = P \cup \{p_2, p^{(2)}\} \setminus \{p, p_\ell\}$$

is a minimum induced path partition of G containing the induced path p_2. ∎

We can apply the following reduction rule if either c_2 or c_n is optional.

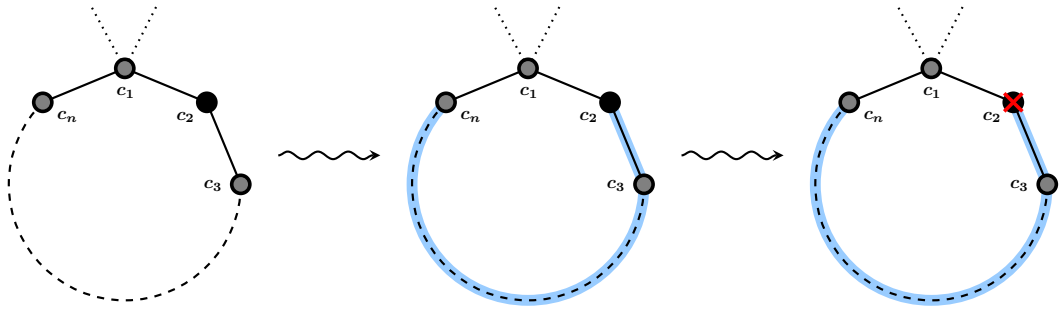


Figure 5.8: Reduction Cycle with c_2 Optional.

Reduction Cycle with c_2 Optional. Let $C = (c_1, \dots, c_n, c_1)$ be a cycle with $\deg(c_x) = 2$ for all vertices c_x on C with $1 < x \leq n$. Let c_2 be an optional vertex. Then remove c_2. See Figure 5.8.

Proof. Let P be a minimum induced path partition of G. If c_2 does not lie on any induced path of P, then P is already an induced path partition of $G - c_2$. Hence, removing c_2 does not increase the number of induced paths.

Now assume that c_2 lies on some induced path $p \in P$. We show that there exists another minimum induced path partition P' in which c_2 does not lie on any induced path. If c_2 is an endpoint of p, we can simply remove c_2 from p. Otherwise, c_2 must lie on the same induced path as c_1 and c_3. In this case, the vertices of C can be covered by a single induced path

$$p' = (c_3, \dots, c_n, c_1, \dots).$$

Since only c_1 may have a neighbor outside C, we define

$$P' = (P \setminus \{p \in P : p \text{ contains a vertex of } C\}) \cup \{p'\},$$

which is a minimum induced path partition where c_2 does not lie on any induced path. ∎

Finally, the following theorem shows that one of the reduction rules can always be applied to any cycle $C = (c_1, \dots, c_n, c_1)$ in which $c_2, \dots, c_n$ have no neighbors outside C.

Theorem 5.3: *Let $C = (c_1, \dots, c_n, c_1)$ be a cycle such that $\deg(c_x) = 2$ for every vertex c_x on C with $1 < x \leq n$. Then one of the following three reduction rules can be applied: Reduction Cycle without Optional Vertices, Reduction Cycle with Optional Vertex and c_2, c_n Mandatory, or Reduction Cycle with c_2 Optional.*

Proof. If all vertices $c_2, \dots, c_n$ are mandatory, we apply Reduction Cycle without Optional Vertices. Otherwise, suppose there exists an optional vertex among $c_3, \dots, c_{n-1}$ while both c_2 and c_n are mandatory. In this case, we apply Reduction Cycle with Optional Vertex and c_2, c_n Mandatory. In all remaining cases, at least one of c_2 or c_n is optional. We may then relabel the cycle so that c_2 is optional and apply Reduction Cycle with c_2 Optional. ∎

5.2.3 Reduction Rules for Cycles with Leaves and a Mandatory Vertex

In this subsection, we present reduction rules for a *pendant cycle* that contains at least one vertex with an adjacent leaf and a mandatory vertex that has no neighbor outside the cycle. We begin by defining a pendant cycle.

Pendant Cycle. A pendant cycle is a cycle $C = (c_1, \dots, c_n, c_1)$ such that for every vertex $c_i \in \{c_2, \dots, c_n\}$, either $\deg(c_i) = 2$, or $\deg(c_i) = 3$ and c_i has a neighbor l_i that is a mandatory leaf.

Next, we show a theorem which helps us for the reduction rules.

Theorem 5.4: *Let G be a cactus with mandatory and optional vertices. Let $C = (c_1, \dots, c_n, c_1)$ be a pendant cycle in G, and let c_i be a mandatory vertex of C with $\deg(c_i) = 2$. Let c_j be a vertex on C with $j > i$ that has an adjacent leaf l_j, and suppose that every vertex c_x with $i \leq x < j$ satisfies $\deg(c_x) = 2$. Furthermore, assume that the pendant cycle C satisfies at least one of the following conditions:*

1. *The vertex c_{j-1} is mandatory.*
2. *$j \neq n$ and the vertex c_{j+1} is optional.*
3. *There exists a vertex c_x on C with $\deg(c_x) = 3$ and $x > j$.*

Then there exists a minimum induced path partition of G that contains an induced path of the form

$$p = (\dots, c_i, c_{i+1}, \dots, c_{j-1}, c_j, l_j).$$

Proof. We construct a minimum induced path partition that contains an induced path of the form $p = (\dots, c_i, c_{i+1}, \dots, c_{j-1}, c_j, l_j)$.

Start with a minimum induced path partition P of G in which the vertices c_j and l_j lie on the same induced path $p_j \in P$, which exists by Theorem 5.2.

If $i < j-2$, we ensure that the vertices $c_{i+1}, c_{i+2}, \dots, c_{j-2}$ lie on the same induced path $p_i \in P$ as c_i. While there exists a vertex $c_x \in \{c_{i+1}, \dots, c_{j-2}\}$ that does not lie on p_i, but c_{x-1} does, we append c_x to p_i. If c_x already lies on another induced path $p_x \in P$, we first remove c_x from p_x. The path p_i remains induced, since c_j and l_j lie on the same induced path $p_j \in P$, and thus c_{x+1} does not belong to p_i.

Next, we ensure that c_{j-1} also lies on p_i if it does not already. However, we cannot simply remove c_{j-1} from its current path (if any) and append it to p_i, as the resulting path

$$p' = (c_{j-1}, c_{j-2}, \dots, c_i, \dots, c_1, c_n, c_{n-1}, \dots, c_j, l_j)$$

would not be induced, since c_{j-1} and c_j are adjacent but not consecutive on p'. Hence, we append c_{j-1} to p_i only if the resulting path remains induced. Otherwise, one of the conditions 1, 2, or 3 must hold, which allows us to modify P accordingly.

If Condition 1 holds, then c_{j+1} must lie on an induced path $p_{j+1} \in P$ of length one, because c_j and c_{j-2} both lie on the same induced path $p_i = p_j$ in P. We then define the induced paths

$$p_i' = (c_{j-1}, c_{j-2}, \dots, c_i, \dots, c_1, c_n, c_{n-1}, \dots, c_{j+1}) \quad \text{and} \quad p_j' = (l_j, c_j),$$

which together cover all vertices of $p_i = p_j$ and p_{j+1}. Replacing $p_i = p_j$ and p_{j+1} in P by p_i' and p_j' yields an induced path partition where $c_i, c_{i+1}, \dots, c_{j-1}$ lie on p_i' and l_j, c_j lie on p_j'.

If c_{j-1} is optional and Condition 2 holds, then c_{j-1} must not lie on any induced path in P. Suppose, for contradiction, that c_{j-1} lies on some induced path $p_{j-1} \in P$. Since both c_{j-2} and c_j lie on $p_i = p_j$, the induced path p_{j-1} must have length one. However, as c_{j+1} is optional, $P \setminus \{c_{j+1}\}$ would still be an induced path partition of G, contradicting the assumption that $P^{(1)}$ is a minimum induced path partition. Thus c_{j-1} lies on no induced path in P, and we can replace $p_i = p_j$ by

$$(l_j, c_j, c_{j-1}, \dots, c_i, \dots, c_1, c_n, c_{n-1}, \dots, c_{j-2}),$$

obtaining a minimum induced path partition of G containing the induced path p.

If Condition 3 holds, then there exists a vertex c_x on C with an adjacent leaf l_x and $x > j$. By Theorem 5.2, we can construct an induced path partition P' from P, where c_x and l_x lie on the same induced path, while c_j and l_j still lie on $p_j' \in P'$, and $c_i, c_{i+1}, \dots, c_{j-2}$ remain on the same induced path $p_i' \in P'$. If c_{j-1} does not already lie on p_i', then $p_i' \neq p_j'$, and we can safely append c_{j-1} to p_i', and p_i' remains an induced path.

Thus, we obtain a minimum induced path partition P of G containing the induced paths

$$p_i = (\dots, c_i, c_{i+1}, \dots, c_{j-1}) \quad \text{and} \quad p_j = (l_j, c_j, \dots).$$

If $p_i = p_j$, then P already contains an induced path of the form p. Otherwise, if $p_i \neq p_j$, then either c_{j+1} lies on p_i or c_{i-1} lies on p_j, because otherwise we could merge p_i and p_j into a single induced path, contradicting the assumption that P is a minimum induced path partition. Therefore, the vertices $c_1, \ldots, c_n$ and l_j are exactly covered by p_i and p_j. Replacing p_i and p_j by

$$p_1 = (c_i, c_{i+1}, \ldots, c_j, l_j) \quad \text{and} \quad p_2 = (c_{j+1}, \ldots, c_n, c_1, \ldots, c_{i-1}),$$

we obtain a minimum induced path partition of G containing the desired induced path p. ∎

Although Theorem 5.4 guarantees the existence of a minimum induced path partition containing an induced path of the form $p = (\ldots, c_i, c_{i+1}, \ldots, c_j, l_j)$, we cannot remove vertices without making additional assumptions about the structure of the graph. A natural idea for a reduction rule would be to remove the vertices $c_{i+1}, c_{i+2}, \ldots, c_j$ and l_j, since a minimum induced path partition exists that contains an induced path of the form $p = (\ldots, c_i, c_{i+1}, \ldots, c_j, l_j)$. For any induced path partition P' of

$$G' = G - \{c_{i+1}, \ldots, c_j, l_j\},$$

we might attempt to extend the induced path $p'_j \in P'$ containing c_i to include the removed vertices. However, the extended path may fail to be induced if c_{j+1} lies on p'_j. Therefore, distinct reduction rules are required depending on the remaining structure of C.

We now present a reduction rule for the case where there exists a vertex c_ℓ on C with $\deg(c_\ell) = 3$ and $\ell < i$.

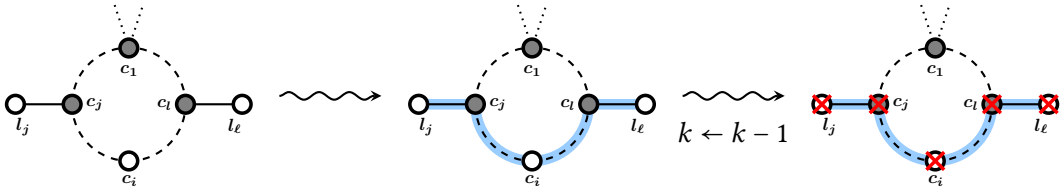


Figure 5.9: Reduction Mandatory Vertex without Leaf between Vertices with Leaf.

Reduction Mandatory Vertex without Leaf between Vertices with Leaf. Let $C = (c_1, \ldots, c_n, c_1)$ be a pendant cycle, and let c_i be a mandatory vertex on C with $\deg(c_i) = 2$. Let c_ℓ and c_j be vertices on C with adjacent leaves l_ℓ and l_j, respectively, and assume $1 < \ell < i < j \leq n$. Suppose that for all vertices c_x with $\ell < x < j$ we have $\deg(c_x) = 2$. Then remove the vertices $l_\ell, c_\ell, \ldots, c_i, \ldots, c_j, l_j$ and decrease k by one. See Figure 5.9.

Proof. By Theorem 5.4, there exists a minimum induced path partition P of G in which the vertices $c_i, c_{i+1}, \ldots, c_j, l_j$ lie on the same induced path $p_i \in P$. We now show that a minimum induced path partition can be constructed from P that contains the induced path

$$p = (l_\ell, c_\ell, \ldots, c_i, \ldots, c_j, l_j).$$

By Theorem 5.2, we can construct a minimum induced path partition P' from P such that l_ℓ and c_ℓ lie on the same induced path $p'_\ell \in P'$, while the vertices $c_i, \ldots, c_j, l_j$ remain on the same induced path $p'_i \in P'$.

Now suppose there exists a vertex c_x on C with $\ell < x < i$ that does not lie on p_i', while its neighbor c_{x+1} does. Since $\deg(c_{x+1}) = 2$, the vertex c_{x+1} must be an endpoint of p_i'. Assume, for contradiction, that c_x lies on another induced path $p_x' \in P'$. In that case, we could merge p_x' and p_i' into a single induced path, contradicting the assumption that P' is a minimum induced path partition. Hence, all vertices $c_{\ell+1}, \dots, c_{i-1}$ that initially do not lie on p_i' must be optional vertices that are not part of any induced path in P', and we can therefore append them to p_i'. Consequently, P' contains the induced path

$$p_i' = (\dots, c_{\ell-1}, c_{\ell-2}, \dots, c_i, \dots, c_j, l_j).$$

If c_ℓ already lies on p_i', then $p_i' = p_\ell'$, and p_i' already has the desired form p. Otherwise, $c_{\ell+1}$ must be an endpoint of p_i'. The vertex c_ℓ cannot be an endpoint of p_ℓ', since in that case we could merge p_i' and p_ℓ' into a single induced path, again contradicting the assumption that P' is a minimum induced path partition. Therefore, p_ℓ' must be of the form

$$p_\ell' = (l_\ell, c_\ell, c_{\ell-1}, \dots).$$

We can then split p_ℓ' into two induced paths:

$$p_\ell^{(1)} = (l_\ell, c_\ell), \qquad p_\ell^{(2)} = (c_{\ell-1}, \dots).$$

Finally, we merge p_i' and $p_\ell^{(1)}$ into a single induced path p, and define

$$P'' = P' \setminus \{p_i', p_\ell'\} \cup \{p, p_\ell^{(2)}\}.$$

Then P'' is a minimum induced path partition of G that contains the induced path p. ∎

If we cannot apply Reduction Mandatory Vertex without Leaf between Vertices with Leaf, it implies that there is no mandatory vertex c_i with $\deg(c_i) = 2$ located between two vertices that each have an adjacent leaf in a pendant cycle. However, there may still exist a mandatory vertex c_i with $\deg(c_i) = 2$ in C. In this case, all vertices c_x with $1 < x \leq i$ must then satisfy $\deg(c_x) = 2$. The following theorem helps us to derive reduction rules for this case.

Theorem 5.5: *Let G be a cactus consisting of mandatory and optional vertices. Let $C = (c_1, \dots, c_n, c_1)$ be a pendant cycle in G, and let c_i be a mandatory vertex of C with $\deg(c_i) = 2$. Let c_j be a vertex of C with $1 < i < j < n$ and $\deg(c_j) = 3$. Assume that every vertex c_x of C with $i < x < j$ satisfies $\deg(c_x) = 2$. Furthermore, assume that the pendant cycle C satisfies at least one of the following conditions:*

1. *The vertex c_{j-1} is mandatory.*
2. *The vertex c_{j+1} is optional.*
3. *There exists a vertex c_x on C with $\deg(c_x) = 3$ and $x > j$.*

Then there exists a minimum induced path partition P of G such that c_j and c_{j+1} do not lie on the same induced path.

Proof. By Theorem 5.4, there exists a minimum induced path partition P of G that contains an induced path of the form $p = (c_j, c_{j-1}, \dots, c_i)$. The vertex c_{j+1} does not lie on p, since otherwise p would not be an induced path. ∎

Theorem 5.5 motivates that we apply a reduction rule depending on c_{j+1}. The following reduction rule covers the case where c_{j+1} is an optional vertex without an adjacent leaf.

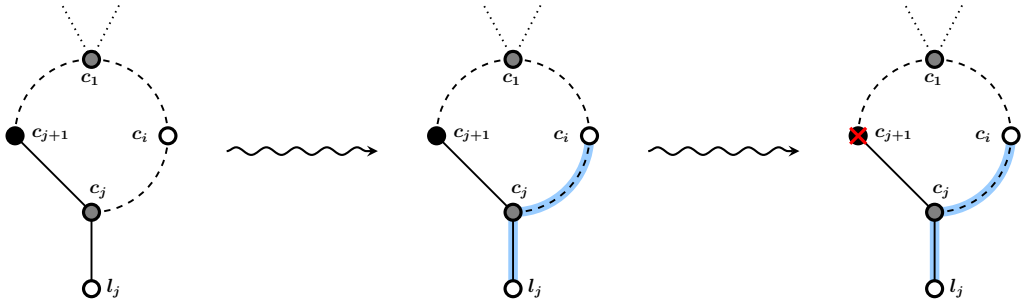


Figure 5.10: Reduction Next Vertex Optional.

Reduction Next Vertex Optional. Let $C = (c_1, \dots, c_n, c_1)$ be a pendant cycle, and let c_i be a mandatory vertex with $\deg(c_i) = 2$. Let c_j be a vertex on C with an adjacent leaf l_j and $i < j < n$, such that every vertex c_x on C with $1 < x < j$ satisfies $\deg(c_x) = 2$. Assume that c_{j+1} is an optional vertex with $\deg(c_{j+1}) = 2$. Then remove c_{j+1}. See Figure 5.10.

Proof. We now show that there exists a minimum induced path partition of G in which c_{j+1} does not lie on any induced path, allowing us to remove c_{j+1}.

By Theorem 5.5, there exists a minimum induced path partition P of G such that c_j and c_{j+1} do not lie on the same induced path. If c_{j+1} does not lie on any induced path in P, the claim follows immediately.

Otherwise, suppose c_{j+1} lies on some induced path $p \in P$. Then c_{j+1} must be an endpoint of p. Since c_{j+1} is an optional vertex, it cannot be the only vertex on p, because in that case $P' = P \setminus \{p\}$ would still be an induced path partition of G, contradicting the assumption that P is a minimum induced path partition. Thus, c_{j+1} must be an endpoint of an induced path $p = (c_{j+1}, c_j, \dots) \in P$. We can remove c_{j+1} from p, obtaining the induced path $p' = (c_{j+2}, \dots)$. Then

$$P' = (P \setminus \{p\}) \cup \{p'\}$$

is also a minimum induced path partition of G in which c_{j+1} does not lie on any induced path. ∎

The next two reduction rules address the case where c_{j+1} has a neighbor that is an adjacent leaf. The following reduction rule assumes that there exists a vertex $c_\ell \in \{c_{j+1}, \dots, c_n\}$ that also has an adjacent leaf.

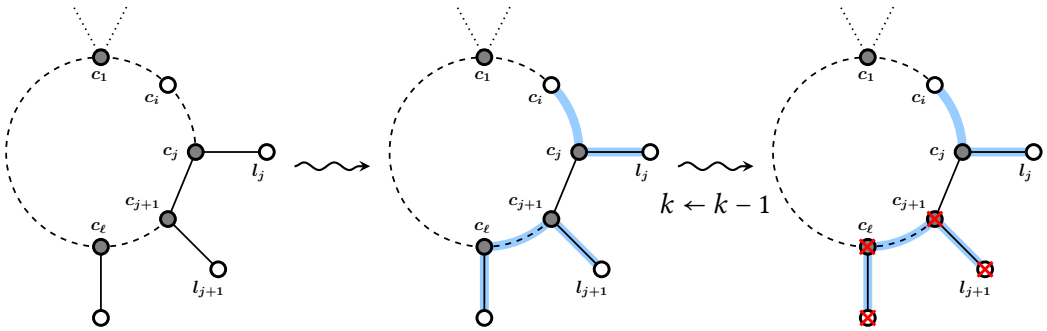


Figure 5.11: Reduction Next Vertex with Leaf.

Reduction Next Vertex with Leaf. Let $C = (c_1, \dots, c_n, c_1)$ be a pendant cycle, and let c_i be a mandatory vertex with $\deg(c_i) = 2$. Let c_j and c_{j+1} be vertices on C with adjacent leaves l_j and l_{j+1}, respectively, where $j > i$, such that every vertex c_x on C with $1 < x < j$ satisfies $\deg(c_x) = 2$. Let c_ℓ be another vertex on C with $\deg(c_\ell) = 3$ and $\ell > j+1$, such that all vertices c_y on C with $j+1 < y < \ell$ also satisfy $\deg(c_y) = 2$. Then remove $l_{j+1}, c_{j+1}, c_{j+2}, \dots, c_\ell, l_\ell$ and decrease k by one. See Figure 5.11.

Proof. We show that there exists a minimum induced path partition P of G that contains the induced path

$$p = (l_{j+1}, c_{j+1}, \dots, c_\ell, l_\ell).$$

This implies that we can remove the vertices of p and decrease k by one.

By Theorem 5.4, there exists a minimum induced path partition $P^{(1)}$ of G containing an induced path of the form

$$p_j = (l_j, c_j, c_{j-1}, \dots, c_i, \dots).$$

By Theorem 5.2, we can construct a minimum induced path partition $P^{(2)}$ from $P^{(1)}$ in which c_{j+1} and l_{j+1} lie on the same induced path. Applying Theorem 5.2 again, we obtain a minimum induced path partition $P^{(3)}$ from $P^{(2)}$ in which c_ℓ and l_ℓ also lie on the same induced path.

Thus, $P^{(3)}$ is a minimum induced path partition containing the induced path p_j, where c_{j+1} and l_{j+1} lie on the same induced path $p_{j+1} \in P^{(3)}$, and c_ℓ and l_ℓ lie on the same induced path $p_\ell \in P^{(3)}$.

If $\ell > j+2$, then all vertices $c_{j+2}, \dots, c_{\ell-1}$ must either lie on p_{j+1} or be optional vertices that do not lie on any induced path in $P^{(3)}$. Assume, for contradiction, that there exists a vertex $c_x \in \{c_{j+2}, \dots, c_{\ell-1}\}$ that lies on another induced path $p_x \in P^{(3)} \setminus \{p_{j+1}\}$, while its neighbor c_{x-1} lies on p_{j+1}. Then c_x must be an endpoint of p_x, and c_{x-1} must be an endpoint of p_{j+1}. Hence, we could merge p_x and p_{j+1} into a single induced path, contradicting the assumption that $P^{(3)}$ is a minimum induced path partition.

If there exists a vertex c_x on C with $j+1 < X < \ell$ that does not lie on any induced path in $P^{(3)}$, while c_{x-1} lies on p_{j+1}, then c_{x-1} must be an endpoint of p_{j+1}, and all vertices $c_{x+1}, \dots, c_{\ell-1}$ also do not lie on any induced path in $P^{(3)}$. We can therefore append the vertices $c_x, c_{x+1}, \dots, c_{\ell-1}$ to p_{j+1}, obtaining

$$p_{j+1} = (l_{j+1}, c_{j+1}, c_{j+2}, \dots, c_{\ell-1}, \dots).$$

If $p_{j+1} = p_\ell$, then p_ℓ already equals p. Otherwise, $c_{\ell-1}$ must be an endpoint of p_{j+1}. The vertex c_ℓ cannot be an endpoint of p_ℓ, since otherwise we could merge p_{j+1} and p_ℓ into a single induced path, contradicting the assumption that $P^{(3)}$ is a minimum induced path partition.

Hence, we can split p_ℓ into two induced paths,

$$p_\ell^{(1)} = (l_\ell, c_\ell) \quad \text{and} \quad p_\ell^{(2)} = (c_{\ell+1}, \dots).$$

Combining $p_\ell^{(1)}$ and p_{j+1} yields the induced path p, and

$$P = P^{(3)} \setminus \{p_\ell, p_{j+1}\} \cup \{p, p_\ell^{(2)}\}$$

is a minimum induced path partition of G that contains p. ∎

Next, we introduce a reduction rule for the case where no vertex c_ℓ on C with $\deg(c_\ell) = 3$ and $\ell > j+1$ exists.

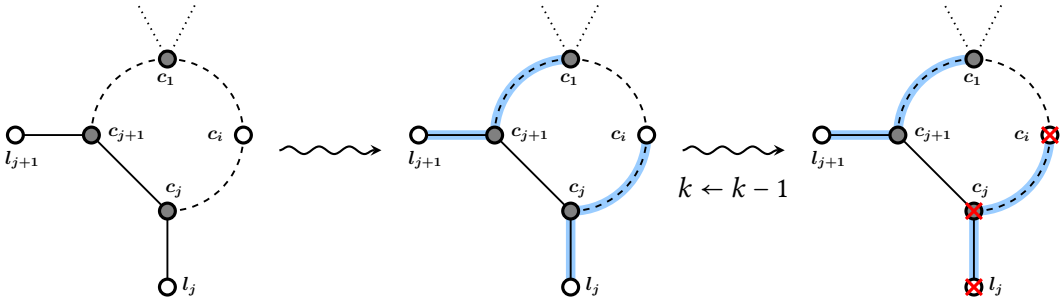


Figure 5.12: Reduction Next Vertex Only Leaf.

Reduction Next Vertex Only Leaf. Let $C = (c_1, \ldots, c_n, c_1)$ be a pendant cycle, and let c_i be a mandatory vertex on C with $\deg(c_i) = 2$ and $i > 1$. Let c_j and c_{j+1} be vertices on C with adjacent leaves l_j and l_{j+1}, respectively, where $j > i$. Every other vertex c_x on C with $1 < x < j$ or $x > j + 1$ satisfies $\deg(c_x) = 2$. Then remove the vertices $c_2, \ldots, c_i, \ldots, c_j, l_j$ and decrease k by one. See Figure 5.12.

Proof. We show that there exists a minimum induced path partition of G that contains the induced path

$$p = (c_2, \ldots, c_i, \ldots, c_j, l_j).$$

Hence, we may remove the vertices of p and decrease k by one.

By Theorem 5.4, there exists a minimum induced path partition $P^{(1)}$ of G containing an induced path of the form

$$p_j = (l_j, c_j, c_{j-1}, \ldots, c_i, \ldots).$$

By Theorem 5.2, we can construct a minimum induced path partition $P^{(2)}$ from $P^{(1)}$ such that c_{j+1} and its adjacent leaf l_{j+1} lie on the same induced path $p_{j+1} \in P^{(2)}$, while the vertices of p_j remain on the same induced path $p_j \in P^{(2)}$.

We first extend p_j so that it contains all vertices $c_{i-1}, c_{i-2}, \ldots, c_2$. While there exists a vertex c_x on C with $1 < x < i$ that does not lie on p_j, but c_{x+1} does, we append c_x to p_j. If c_x lies on another induced path $p_x \in P^{(2)} \setminus \{p_j\}$, then p_x must have length at least two, since otherwise we could merge p_j and p_x into a single induced path, contradicting the assumption that $P^{(2)}$ is a minimum induced path partition. Therefore, we can remove c_x from p_x and append c_x to p_j. If instead c_x is optional and does not lie on any induced path in $P^{(2)}$, we can directly append c_x to p_j. Thus, the induced path p_j becomes of the form

$$p_j = (l_j, c_j, c_{j-1}, \ldots, c_i, \ldots, c_3, c_2, \ldots).$$

If c_1 does not lie on p_j, then the induced path p_j already equals p, and $P^{(2)}$ is a minimum induced path partition containing p.

If instead c_1 lies on p_j, then the other endpoint of p_{j+1} (besides l_{j+1}) must be a vertex c_x on C. While $x \neq n$, we can append c_{x+1} to p_{j+1}. If c_{x+1} already lies on another induced path p_{x+1}, then p_{x+1} must have length at least two, since otherwise we could merge p_{j+1} and p_{x+1} into a single induced path, contradicting the assumption that $P^{(2)}$ is a minimum induced path partition. Thus, we remove c_{x+1} from p_{x+1} and append it to p_{j+1}. If c_{x+1} is an optional vertex that does not lie on any induced path in $P^{(2)}$, we can directly append c_{x+1} to p_{j+1} without removing it from any induced path. Hence, the induced path p_{j+1} becomes of the form

$$p_{j+1} = (l_{j+1}, c_{j+1}, c_{j+2}, \ldots, c_n).$$

We now split p_j into two induced paths,

$$p_j^{(1)} = (l_j, c_j, \dots, c_1) = p \quad \text{and} \quad p_j^{(2)} = (c_1, \dots).$$

We then merge p_{j+1} and $p_j^{(2)}$ into a single induced path p'_{j+1}. Finally, we construct

$$P = P^{(2)} \setminus \{p_j, p_{j+1}\} \cup \{p, p'_{j+1}\},$$

which is again a minimum induced path partition containing p. ∎

If we cannot apply Reduction Next Vertex Optional, Reduction Next Vertex with Leaf, or Reduction Next Vertex Only Leaf, then either c_{j+1} is a mandatory vertex with $\deg(c_{j+1}) = 2$, or $c_j = c_n$. If c_{j+1} is a mandatory vertex with $\deg(c_{j+1}) = 2$, then all vertices c_x on C with $x > j$ must satisfy $\deg(c_x) = 2$, since otherwise we could apply Reduction Mandatory Vertex without Leaf between Vertices with Leaf. The following reduction rule covers the case where $c_j \neq c_n$.

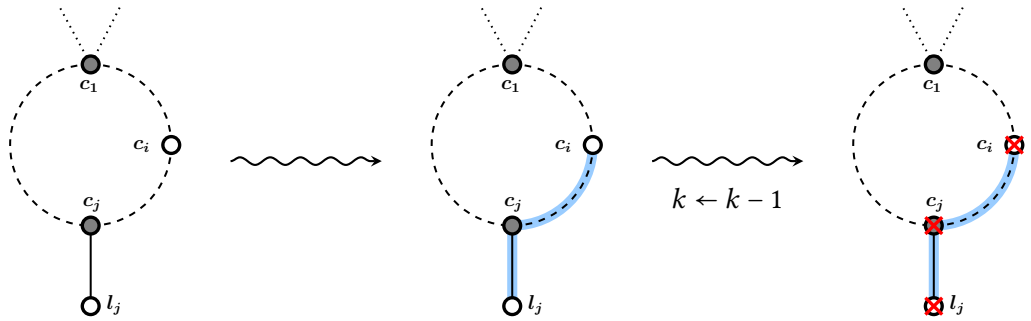


Figure 5.13: Reduction No Other Leaf.

Reduction No Other Leaf. Let $C = (c_1, \dots, c_n, c_1)$ be a pendant cycle, and let c_i be a mandatory vertex with $\deg(c_i) = 2$ and $i > 1$. Let c_j be a vertex on C with $i < j < n$ that has an adjacent leaf l_j. The vertex c_{j+1} is also mandatory, and all other vertices c_x on C with $x \neq j$ satisfy $\deg(c_x) = 2$. Then, remove the vertices $c_2, c_3, \dots, c_{j-1}, c_j, l_j$ and decrease k by one. See Figure 5.13.

Proof. We show that there exists a minimum induced path partition P of G containing the induced path

$$p = (c_2, \dots, c_i, \dots, c_j, l_j).$$

Hence, all vertices on p can be removed, and k can be decreased by one.

By Theorem 5.4, there exists a minimum induced path partition P of G that contains an induced path of the form

$$p_j = (l_j, c_j, c_{j-1}, \dots, c_i, \dots).$$

If $p_j = p$, then P is already a minimum induced path partition containing p. Otherwise, the other endpoint of p_j (besides l_j) must be either a vertex c_x on C with $1 < x \leq j$, or p_j contains the vertex c_1.

First, assume the other endpoint of p_j (besides l_j) is a vertex c_x on C with $1 < x \leq j$. In this case, we can remove the vertices $c_{x-1}, c_{x-2}, \dots, c_2$ from their respective induced paths (if they belong to any) and append them to p_j, thereby obtaining p.

Now assume that p_j contains the vertex c_1. Then c_{j+1} must be the endpoint of another induced path $p_{j+1} \neq p_j$ in P. The other endpoint of p_{j+1} (besides c_{j+1}) must be a vertex c_ℓ on C with $\ell \geq j$. If $\ell \neq n$, we remove all vertices c_x on C with $x > j$ that lie on any induced path $p_x \in P$, and append them to p_{j+1}, obtaining

$$p_{j+1} = (c_j, c_{j+1}, \ldots, c_n).$$

Next, we split p_j into two induced paths

$$p_j^{(1)} = (l_j, c_j, \ldots, c_2) = p \quad \text{and} \quad p_j^{(2)} = (c_1, \ldots).$$

We then merge p_{j+1} and $p_j^{(2)}$ into a new induced path p'_{j+1}. Consequently, we obtain a new minimum induced path partition

$$P' = P \setminus \{p_j, p_{j+1}\} \cup \{p, p'_{j+1}\},$$

which contains the induced path p. ∎

Next, we present reduction rules for the case where $c_j = c_n$. We must distinguish whether the vertex c_{n-1} is mandatory or optional, since Theorem 5.4 can only be applied if c_{n-1} is mandatory. Otherwise, a minimum induced path partition may contain the induced path

$$p = (l_n, c_n, c_1, c_2, \ldots, c_i, \ldots, c_{n-2}),$$

while c_{n-1} does not lie on any induced path. We first introduce a reduction rule for the case where c_{n-1} is optional.

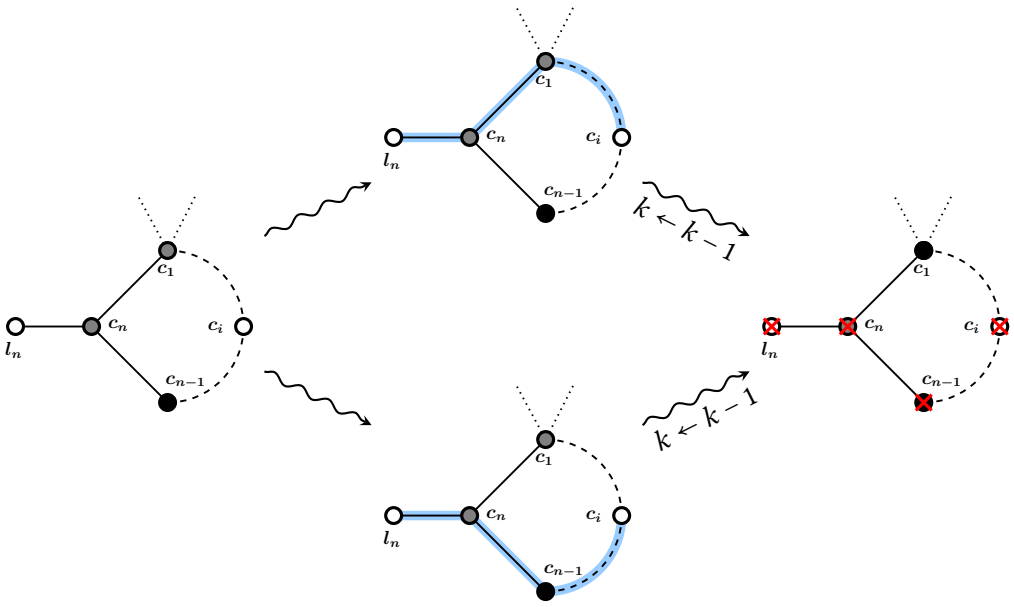


Figure 5.14: Reduction No Next Vertex and Optional Predecessor.

Reduction No Next Vertex and Optional Predecessor. Let $C = (c_1, \dots, c_n, c_1)$ be a pendant cycle, and let c_i be a mandatory vertex with $\deg(c_i) = 2$ and $i > 1$. Assume that c_n is a vertex with an adjacent leaf l_n, and that c_{n-1} is an optional vertex. Every other vertex c_x on C with $1 < x < n$ satisfies $\deg(c_x) = 2$. Then, remove the vertices $c_2, c_3, \dots, c_n, l_n$, mark c_1 as optional, and decrease k by one. See Figure 5.14.

Proof. We show that there exists a minimum induced path partition of G containing either of the induced paths

$$p_1 = (l_n, c_n, c_1, c_2, \dots, c_i, \dots, c_{n-2}) \quad \text{or} \quad p_2 = (l_n, c_n, c_{n-1}, \dots, c_i, \dots, c_2).$$

Since both induced paths cover the vertices $c_2, c_3, \dots, c_{n-2}, c_n, l_n$, and since c_{n-1} is optional while c_1 lies only on p_1, we remove the vertices $c_2, c_3, \dots, c_n, l_n$, mark c_1 as optional, and decrease k by one.

By Theorem 5.2, there exists a minimum induced path partition P of G containing an induced path $p_n \in P$ on which both c_n and l_n lie. If $p_n = p_1$ or $p_n = p_2$, then P already contains one of the desired induced paths.

If c_1 lies on p_n, we modify p_n so that it becomes the induced path p_1. If the other endpoint of p_n (besides l_n) is a vertex c_ℓ on C, then all vertices of C lying on any induced path in P must also lie on p_n. Assume, for contradiction, that there exists another induced path $p_x = (c_x, c_{x+1}, \dots, c_y) \in P$. The induced path p_x must cover at least one mandatory vertex. If $c_y = c_{n-1}$, we remove c_y from p_x, since c_{n-1} is optional. Next, we append the vertices $c_2, c_3, \dots, c_{x-1}$ to p_x, removing them from their respective induced paths in P if necessary. Thus, $p_n = (c_2, c_3, \dots, c_{n-2})$. However, this implies that we could merge p_n and p_x into a single induced path, contradicting the assumption that P is a minimum induced path partition. Hence, the vertex c_i must lie on p_n, and we can append the vertices $c_{i+1}, c_{i+2}, \dots, c_{n-2}$ to p_n, obtaining the induced path p_1.

If instead the other endpoint of p_n (besides l_n) is a vertex outside C, then p_n must be of the form

$$p_n = (l_n, c_n, c_1, v, \dots)$$

for some vertex v outside C. Since c_i is a mandatory vertex on C that does not lie on p_n, there must exist another induced path $p_i \in P$ covering all remaining mandatory vertices c_x on C with $1 < x < n$. We then split p_n into two induced paths

$$p_n^{(1)} = (l_n, c_n, c_1), \qquad p_n^{(2)} = (v, \dots).$$

Finally, we merge p_i and $p_n^{(1)}$ into the single induced path p_1, and define

$$P' = P \setminus \{p_i, p_n\} \cup \{p_1, p_n^{(2)}\},$$

obtaining a minimum induced path partition containing the induced path p_1.

If, on the other hand, c_1 does not lie on p_n, we can append the vertices $c_{n-1}, c_{n-2}, \dots, c_2$ to p_n, removing them from their respective induced paths in P if necessary, thus obtaining the induced path p_2. ∎

We now present the reduction rule for the case where c_{n-1} is mandatory.

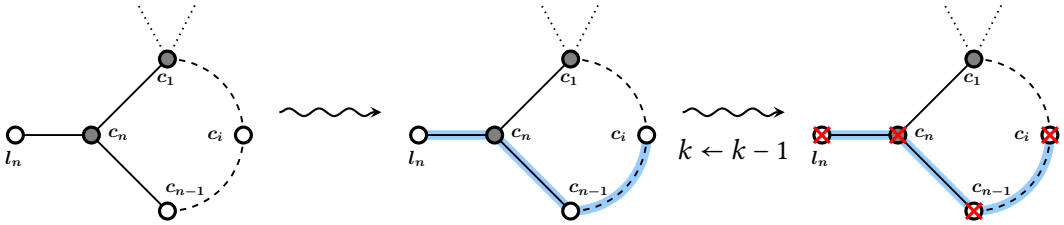


Figure 5.15: Reduction No Next Vertex and Mandatory Predecessor.

Reduction No Next Vertex and Mandatory Predecessor. Let $C = (c_1, \ldots, c_n, c_1)$ be a pendant cycle. Assume that c_n is a vertex with an adjacent leaf l_n, and that c_{n-1} is a mandatory vertex. Every other vertex c_x on C with $1 < x < n$ satisfies $\deg(c_x) = 2$. Then remove the vertices $c_2, c_3, \ldots, c_n, l_n$ and decrease k by one. See Figure 5.15.

Proof. We show that there exists a minimum induced path partition of G containing the induced path

$$p = (c_2, c_3, \ldots, c_n, l_n).$$

Hence, all vertices on p can be removed, and k can be decreased by one.

By Theorem 5.4, there exists a minimum induced path partition P of G containing the induced path

$$p_n = (l_n, c_n, c_{n-1}, \ldots, c_i, \ldots).$$

If $p_n = p$, then P already contains the desired induced path. Otherwise, the other endpoint of p_n (besides l_n) must be a vertex c_x on C with $1 < x \leq i$. In that case, remove the vertices $c_{x-1}, c_{x-2}, \ldots, c_2$ from their respective induced paths, if they lie on any, and append them to p_n. Thus, we obtain $p_n = p$, and therefore P becomes a minimum induced path partition containing the induced path p. ∎

Finally, we show that for any pendant cycle containing both a vertex with an adjacent leaf and a mandatory vertex without an adjacent leaf, one of the reduction rules presented in this section can be applied.

Theorem 5.6: *Let $C = (c_1, \ldots, c_n, c_1)$ be a pendant cycle such that at least one of the vertices $c_2, \ldots, c_n$ has an adjacent leaf, and at least one vertex is mandatory and has no neighbor outside C. Then, one of the following reduction rules can be applied: Reduction Mandatory Vertex without Leaf between Vertices with Leaf, Reduction Next Vertex Optional, Reduction Next Vertex with Leaf, Reduction Next Vertex Only Leaf, Reduction No Other Leaf, Reduction No Next Vertex and Optional Predecessor, or Reduction No Next Vertex and Mandatory Predecessor.*

Proof. First, assume there exist two vertices c_ℓ and c_j on C that each have an adjacent leaf, and a vertex c_i on C with $\deg(c_i) = 2$ such that $1 < \ell < i < j \leq n$. In this case, we can apply Reduction Mandatory Vertex without Leaf between Vertices with Leaf.

Otherwise, there must exist a mandatory vertex c_i on C with $\deg(c_i) = 2$ and a vertex c_j on C with an adjacent leaf such that:

- if $i < j$, then all vertices c_x on C with $1 < x < j$ have no adjacent leaf, and
- if $j < i$, then all vertices c_x on C with $x > j$ have no adjacent leaf.

Without loss of generality, we may assume $i < j$, because otherwise we can reverse the labeling of the vertices of C.

Next, assume $c_j \neq c_n$. If c_{j+1} is an optional vertex without an adjacent leaf, we can apply Reduction Next Vertex Optional. If c_{j+1} has an adjacent leaf and there exists another vertex c_ℓ on C with $\ell > j+1$ that also has an adjacent leaf, then we can apply Reduction Next Vertex with Leaf. If instead c_{j+1} has an adjacent leaf while all vertices c_x on C with $x > j+1$ have no neighbor outside C, then we can apply Reduction Next Vertex Only Leaf. If all other vertices c_x on C with $x \neq j$ satisfy $\deg(c_x) = 2$ and c_{j+1} is mandatory, we can apply Reduction No Other Leaf.

Otherwise, $c_j = c_n$ and all other vertices c_x on C wit $1 < x < n$ have no neighbor outside C. In this case, we can apply Reduction No Next Vertex and Optional Predecessor if c_{n-1} is optional, or Reduction No Next Vertex and Mandatory Predecessor if c_{n-1} is mandatory. ■

5.2.4 Reduction Rules for Cycles with Leaves and without Mandatory Vertices

In this subsection, we present reduction rules for a pendant cycle $C = (c_1, \ldots, c_n, c_1)$ in which every vertex c_x on C with $1 < x \leq n$ either has an adjacent leaf l_x or is an optional vertex. Let n_L denote the number of vertices c_x on C that have an adjacent leaf l_x. Throughout this subsection, we refer to the leaf l_j for which exactly $i-1$ vertices c_x with $1 < x < j$ have adjacent leaves as the *i-th leaf* of C.

We distinguish between reduction rules depending on whether n_L is even or odd. We begin with the case where n_L is even.

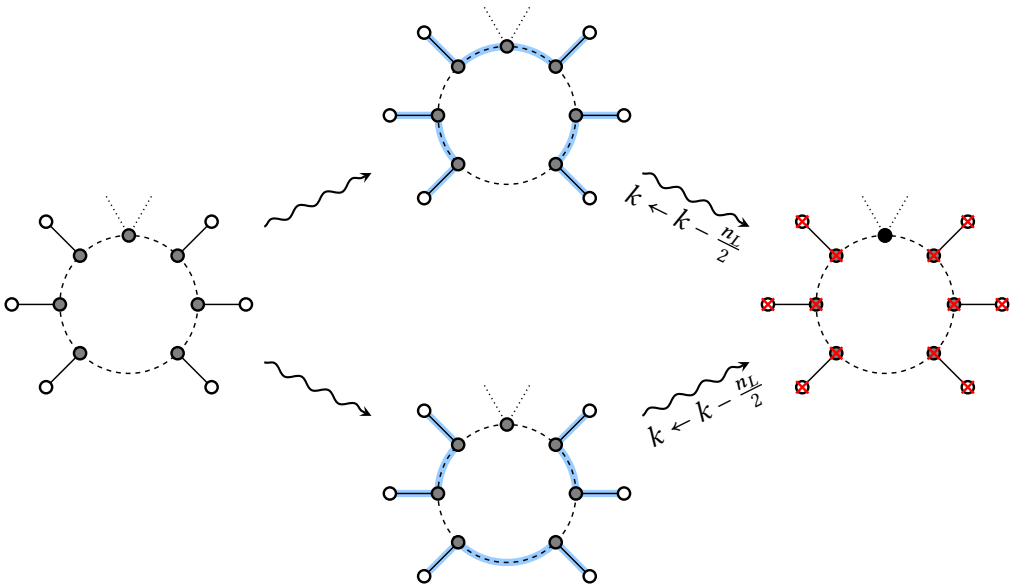


Figure 5.16: Reduction Pendant Cycle with n_L Even.

Reduction Pendant Cycle with n_L Even. Let $C = (c_1, \dots, c_n, c_1)$ be a pendant cycle in which every vertex c_x with $1 < x \leq n$ either has an adjacent leaf l_x or is optional, and let $n_L \geq 2$ be even. If $n_L = 2$, assume that the two vertices with adjacent leaves are not adjacent to each other. Then remove $c_2, \dots, c_n$ and their adjacent leaves, mark c_1 as optional, and decrease k by $\frac{n_L}{2}$. See Figure 5.16.

Proof. We show that there exists a minimum induced path partition of G that covers the vertices $c_2, \dots, c_n$ and their adjacent leaves using $\frac{n_L}{2}$ induced paths. These vertices are covered either by the set of induced paths $P_1 = \{p_{n_L,1}, p_{2,3}, \dots, p_{n_L-2,n_L-1}\}$ or by $P_2 = \{p_{1,2}, p_{3,4}, \dots, p_{n_L-1,n_L}\}$, where each induced path $p_{x,x+1}$ is defined as

$$p_{x,x+1} = (l_i, c_i, c_{i+1}, \dots, c_{j-1}, c_j, l_j),$$

with l_i and l_j denoting the x-th and $(x+1)$-th leaves of C, respectively. The path $p_{n_L,1}$ is defined as

$$p_{n_L,1} = (l_i, c_i, c_{i+1}, \dots, c_n, c_1, \dots, c_{j-1}, c_j, l_j),$$

where l_i and l_j correspond to the last and the first leaves of C, respectively. These paths are induced because, even when $n_L = 2$, the vertices c_i and c_j corresponding to the first and second leaves are not adjacent. Hence, all vertices $c_2, \dots, c_n$ and their leaves can be covered by the $\frac{n_L}{2}$ induced paths in either P_1 or P_2, while c_1 lies only on an induced path in P_1. Thus, we remove all vertices $c_2, \dots, c_n$ and their leaves, mark c_1 as optional, and decrease k by $\frac{n_L}{2}$.

Now, let P be a minimum induced path partition of G. Since every leaf must be an endpoint of an induced path, the vertices of C must lie on at least $\frac{n_L}{2}$ induced paths.

If the leaves of C are covered by exactly $\frac{n_L}{2}$ induced paths in P, then the endpoints of each induced path $p \in P$ must be two distinct leaves l_i and l_j. Any vertex on p other than c_i, c_j, and c_1 cannot have an adjacent leaf, as such a leaf would have to lie on an induced path of length one, contradicting that both endpoints of p are distinct leaves. Therefore, the induced path $p \in P$ containing the first leaf of C must have its other endpoint at either the second leaf ($p = p_{1,2}$) or the last leaf ($p = p_{n_L,1}$). If $p = p_{1,2}$, then the induced path in P containing the third leaf must also contain the fourth leaf, and $p_{3,4} \in P$. By induction, all vertices of P_2 must also be contained in P. If instead $p = p_{n_L,1}$, then the second and third leaf must lie on the same induced path $p_{2,3}$, and again, by induction, all vertices of P_1 are contained in P.

Now suppose that the leaves of C are covered by more than $\frac{n_L}{2}$ induced paths. Assume, for contradiction, that c_1 does not lie on the same induced path as either c_2 or c_n. Since no vertex of C other than c_1 has a neighbor outside C, we could cover all remaining vertices of C by the $\frac{n_L}{2}$ induced paths in P_2, contradicting the assumption that P is a minimum induced path partition. Similarly, if c_1 lies on the same induced path as both c_2 and c_n, we could again replace the induced paths covering C by those in P_1, contradicting the assumption that P is a minimum induced path partition. Hence, c_1 must lie on an induced path of the form $p_1 = (\dots, c_i, c_1, v, \dots)$, where $i \in \{2, n\}$ and v is a vertex outside C. We can then split p_1 into two induced paths, $p_1^{(1)} = (\dots, c_i)$ and $p_1^{(2)} = (c_1, v, \dots)$. Since we can again cover all vertices on C except c_1 using the $\frac{n_L}{2}$ induced paths in P_2, we can replace all induced paths covering vertices of C by those in P_2 together with $p_1^{(2)}$. ∎

If $n_L = 2$ and two adjacent vertices $c_i, c_{i+1} \in \{c_2, \dots, c_{n-1}\}$ each have an adjacent leaf, then Reduction Pendant Cycle with n_L Even cannot be applied, since the path $(l_{i+1}, c_{i+1}, \dots, c_n, c_1, c_i, l_i)$ is not induced, as c_i and c_{i+1} are adjacent. For such graphs, we require the following reduction rule.

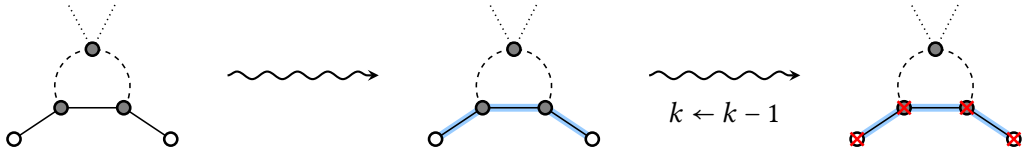


Figure 5.17: Reduction Cycle with Two Adjacent Leaves.

Reduction Cycle with Two Adjacent Leaves. Let $C = (c_1, \dots, c_n, c_1)$ be a pendant cycle. Suppose c_i and c_{i+1} are two adjacent vertices on C with adjacent leaves l_i and l_{i+1}, respectively, where $1 < i < n$. All other vertices c_j on C with $1 < j < i$ or $i + 1 < j < n$ are optional and have degree two. Then remove c_i, l_i, c_{i+1}, and l_{i+1}, and decrease k by one. See Figure 5.17.

Proof. We show that there exists a minimum induced path partition of G containing the induced path $p = (l_i, c_i, c_{i+1}, l_{i+1})$, which allows us to remove c_i, l_i, c_{i+1}, and l_{i+1} and to decrease k by one.

Let P be a minimum induced path partition of G. If c_i and l_i do not lie on the same induced path in P, we can apply Theorem 5.2 to obtain a new minimum induced path partition $P^{(1)}$ in which c_i and l_i lie on the same induced path. Applying the same theorem again yields another minimum induced path partition $P^{(2)}$ derived from $P^{(1)}$, where c_{i+1} and l_{i+1} also lie on the same induced path $p_{i+1} \in P^{(2)}$, while c_i and l_i remain on the same induced path $p_i \in P^{(2)}$.

If $p_i = p_{i+1}$, then $p_i = (l_i, c_i, c_{i+1}, l_{i+1})$, since c_i and c_{i+1} are adjacent. Otherwise, the vertex c_1 must lie on either p_i or p_{i+1}. Assume, for contradiction, that c_1 lies on neither p_i nor p_{i+1}. Then all vertices of p_i and p_{i+1} except c_i, l_i, c_{i+1}, and l_{i+1} are optional, and

$$P^{(2)\prime} = P^{(2)} \cup \{p\} \setminus \{p_i, p_{i+1}\}$$

would be an induced path partition of smaller size, contradicting the assumption that $P^{(2)}$ is a minimum induced path partition. Therefore, either p_i or p_{i+1} must be of the form $p_j = (l_j, c_j, \dots, c_1, \dots)$. We can then split p_j into two induced paths, $p_j^{(1)} = (l_j, c_j)$ and $p_j^{(2)} = (c_1, \dots)$. Finally,

$$P^{(3)} = P^{(2)} \setminus \{p_i, p_j\} \cup \{p, p_j^{(2)}\}$$

is a minimum induced path partition that contains p. ∎

Next, we consider the case where n_L is odd.

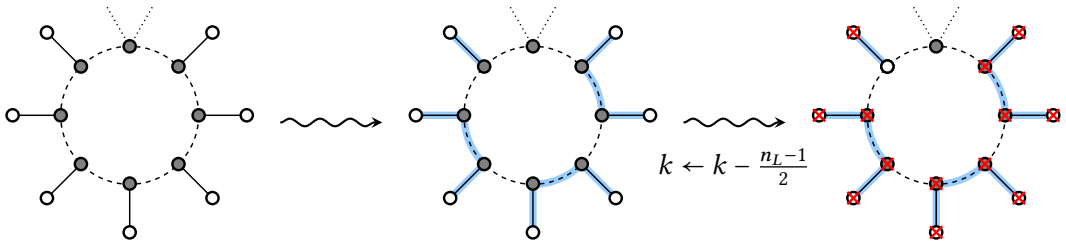


Figure 5.18: Reduction Cycle with an Odd Number of Leaves.

Reduction Cycle with an Odd Number of Leaves. Let $C = (c_1, \dots, c_n, c_1)$ be a pendant cycle in which all vertices of degree two are optional and n_L is odd. Then remove $c_2, \dots, c_{n-1}$ and all leaves of C, mark c_n as mandatory, and decrease k by $\frac{n_L - 1}{2}$. See Figure 5.18.

Proof. We distinguish two cases.

Case 1: $n_L = 1$. Let p be the induced path containing the only leaf l_i of C. We show that there exists a minimum induced path partition of G containing an induced path p of the form $p = (l_i, c_1, \dots, c_j, \dots)$, where $c_j = c_2$ if $c_i \neq c_n$, and $c_j = c_n$ otherwise. This allows us to remove $c_3, \dots, c_n$ and l_i, and mark c_2 as mandatory. This is valid because c_2 must then be an endpoint of an induced path in a minimum induced path partition p' of $G' = G - \{c_3, \dots, c_n, l_i\}$. We obtain a minimum induced path partition of G by replacing c_2 with c_j in p' and extending p' to include c_i and l_i. Then p' is of the required form p.

Let P be a minimum induced path partition of G. By Theorem 5.2.1, we can construct a minimum induced path partition $P^{(1)}$ from P in which c_i and l_i lie on the same induced path $p_i \in P^{(1)}$. If c_1 lies on p_i, then p_i already has the desired form p. Otherwise, let $p_1 \in P^{(1)}$ be the induced path containing c_1. The vertex c_1 must be the only vertex of C on p_1, and c_1 cannot be an endpoint of p_1, as otherwise p_i and p_1 could be merged into a single induced path, contradicting the assumption that $P^{(1)}$ is a minimum induced path partition. Thus, p_1 must be of the form $p_1 = (\dots, v, c_1, w, \dots)$ for some vertices v and w not lying on C. We then split p_1 into two induced paths: $p_1^{(1)} = (\dots, v)$ and $p_1^{(2)} = (c_1, w, \dots)$. If $c_j = c_2$, we append the vertices c_2 and l_2 to $p_1^{(2)}$. Otherwise, we append the vertices $c_n, c_{n-1}, \dots, c_j, l_j$ to $p_1^{(2)}$. In both cases, $p_1^{(2)}$ becomes an induced path of the required form p. Then

$$P^{(2)} = P^{(1)} \setminus \{p_1, p_i\} \cup \{p_1^{(1)}, p_1^{(2)}\}$$

is a minimum induced path partition containing an induced path of the form p.

Case 2: $n_L \geq 3$. We show that there exists a minimum induced path partition of G that contains the induced paths in

$$P_C = \{p_{1,2}, p_{3,4}, \dots, p_{n_L-1,n_L}\},$$

where each induced path $p_{x,x+1}$ is defined as

$$p_{x,x+1} = (l_i, c_i, c_{i+1}, \dots, c_{j-1}, c_j, l_j),$$

with l_i and l_j being the x-th and $(x+1)$-th leaves of C, respectively.

Let P be a minimum induced path partition of G. Since every leaf of C must be an endpoint of an induced path, they collectively lie on at least $\frac{n_L+1}{2}$ induced paths.

If the vertex c_1 does not lie on the same induced path as any other mandatory vertex of C, then all induced paths in P containing vertices of C other than c_1 consist solely of vertices of C, since c_1 is the only vertex of C adjacent to a vertex outside the pendant cycle. As the remaining mandatory vertices of C can be covered by the induced paths $p_{1,2}$, $p_{3,4}, \dots$, and by $p_i = (l_i, c_i)$, where l_i is the last leaf of C, we can replace the induced paths covering the vertices of C (except c_1) with the induced paths in P_C and p_i, obtaining a minimum induced path partition of G containing the induced paths in P_C.

Otherwise, c_1 lies on an induced path p_1 of the form $p_1 = (\dots, c_j, c_1, x)$, where $c_j = c_2$ or $c_j = c_n$, and x is some vertex. If x is not a vertex of C, we split p_1 into two induced paths: $p_1^{(1)} = (\dots, c_j)$ and $p_1^{(2)} = (c_1, x, \dots)$. We then extend $p_1^{(2)}$ to include the last leaf l_i of C and the vertices $c_i, c_{i+1}, \dots, c_n$. By replacing all induced paths in P with the induced paths

in P_C and the extended induced path $p_1^{(2)}$, we obtain a minimum induced path partition of G containing the induced paths in P_C. If x is a vertex of C, then p_1 must be of the form $p_1 = (\dots, c_n, c_1, c_2, \dots)$. In this case, all vertices of C are covered by $\frac{n_L+1}{2}$ induced paths, none of which contain vertices outside C. Hence, we can replace these induced paths with those in P_C and $p_i = (l_i, c_i)$, where l_i is the last leaf of C, obtaining a minimum induced path partition of G that contains the induced paths in P_C.

Therefore, there exists a minimum induced path partition containing the induced paths in P_C. Consequently, we can remove all vertices lying on these induced paths. The remaining vertices of C are $c_j, c_{j+1}, \dots, c_i, \dots, c_1$, and l_i, where l_i is the last leaf of C. Next, we can iteratively apply Reduction Optional Leaf to remove the vertices $c_j, c_{j+1}, \dots, c_{i-1}$, and finally apply Reduction Path to remove $l_i, c_i, c_{i+1}, \dots, c_{n-1}$. Thus, all vertices of C except c_1 and c_n are removed, and c_n is marked as optional by Reduction Path. ∎

The following theorem shows that we can apply one of the reduction rules presented in this subsection to every pendant cycle that contains at least one vertex with an adjacent leaf and whose mandatory vertices all have adjacent leaves.

Theorem 5.7: *Let $C = (c_1, \dots, c_n, c_1)$ be a pendant cycle such that at least one of the vertices $c_2, \dots, c_n$ has an adjacent leaf and every mandatory vertex among $\{c_2, \dots, c_n\}$ has an adjacent leaf. Then we can apply Reduction Pendant Cycle with n_L Even, Reduction Cycle with Two Adjacent Leaves, or Reduction Cycle with an Odd Number of Leaves.*

Proof. If n_L is even and $n_L > 2$, or if $n_L = 2$, and the two vertices with adjacent leaves are not adjacent, then we can apply Reduction Pendant Cycle with n_L Even. If $n_L = 2$ and the two vertices with adjacent leaves are adjacent, we can apply Reduction Cycle with Two Adjacent Leaves. Otherwise, n_L must be odd, and we can apply Reduction Cycle with an Odd Number of Leaves. ∎

Finally, we show that the reduction rules presented in Subsection 5.2.2, Subsection 5.2.3, and this subsection are sufficient to ensure that we can apply a reduction rule to any pendant cycle.

Theorem 5.8: *Let $C = (c_1, \dots, c_n, c_1)$ be a pendant cycle. Then we can remove at least one of the vertices $c_2, \dots, c_n$ by applying a reduction rule.*

Proof. If none of the vertices $c_2, \dots, c_n$ has an adjacent leaf, then by Theorem 5.3, we can apply one of the reduction rules from Subsection 5.2.2. Otherwise, there exists at least one vertex with an adjacent leaf. If one of the vertices $c_2, \dots, c_n$ is mandatory and does not have a neighbor outside C, then by Theorem 5.6, we can apply a reduction rule from Subsection 5.2.3. Otherwise, we can apply one of the reduction rules from this subsection by Theorem 5.7. Thus, a reduction rule can always be applied, and since each reduction rule removes at least one vertex, we can remove at least one vertex from C in every case. ∎

5.3 Linear-Time Algorithm for Induced Path Partition

In this section, we present a linear-time algorithm that decides whether a cactus $G = (V, E)$ has a zero forcing set of size at most k. The algorithm repeatedly applies the reduction rules from Section 5.2 until all vertices are removed. If, upon termination, $k \geq 0$, then G has a zero forcing set of size at most k.

5.3.1 Overview

We begin by selecting an arbitrary vertex $v \in V$. Then we add a new optional vertex r and the edge $e_r = \{r, v\}$ to G. Note that r can be removed by applying Reduction Optional Leaf, restoring the original graph and preserving the zero forcing number. Hence, the original graph has a zero forcing set of size at most k if and only if the modified graph does. The addition of r ensures that the block-cutpoint tree T of G contains at least one bridge whose endpoint is a leaf.

We orient each bridge as a directed edge $b = (u, v)$, where either $u = r$ or u lies on the path from v to r. Similarly, for each cycle $C = (c_1, \ldots, c_n, c_1)$, we label the vertices so that c_1 connects the remaining vertices $c_2, \ldots, c_n$ to r.

Next, we perform a DFS traversal on the block-cutpoint tree $T = (V', E')$ of G, starting from the bridge e_r. We apply reduction rules during the backtracking phase of the DFS. After each backtracking step, the following invariants must hold:

Invariant 1: *The graph G remains connected.*

Invariant 2: *Each finished block is a bridge (u, v) where v is a mandatory leaf.*

Invariant 3: *Each finished cutpoint lies on an unfinished cycle and exactly one finished block.*

Initially, all invariants hold, since the original graph G is connected and contains neither finished blocks nor finished cutpoints.

5.3.2 Backtracking a Cutpoint

Assume we backtrack a cutpoint $v \in V$. Let $b \in V'$ be the unfinished block containing v, and let $b_1, \ldots, b_n \in V' \setminus \{b\}$ be the other blocks containing v. These blocks $b_1, \ldots, b_n$ are finished, and therefore each b_i is a bridge $b_i = (v, x_i)$ where x_i is a mandatory leaf by Invariant 2.

If $n \geq 2$, we apply Reduction Pendant Star to remove x_1, x_2, and v. The remaining vertices $x_3, \ldots, x_n$ become isolated and are removed by applying Reduction Isolated Vertex, ensuring that G remains connected. If the unfinished block containing v is a cycle $C = (c_1, \ldots, c_n, c_1)$, that block in the block-cutpoint tree is replaced by bridges and cutpoints between the former vertices of the cycle. We mark the newly created bridges and cutpoints as unvisited and continue the DFS at c_1. We maintain Invariants 2 and 3 because no additional block or cutpoint becomes finished.

If instead the unfinished block containing v is a bridge $b = (u, v)$, we continue the DFS at u. The invariants are preserved because G remains connected and no block or cutpoint becomes finished.

If $n = 1$, then $\deg(v) = 2$, and we may apply Reduction Path to remove the leaf x_1. We preserve Invariant 3 because v is no longer a cutpoint, and Invariant 2 because no block becomes finished. Invariant 1 is preserved because G remains connected.

5.3.3 Backtracking a Bridge

Suppose we backtrack a bridge $e = (u, v)$. The vertex v must be a leaf because otherwise v would be a finished cutpoint lying on a finished block, contradicting Invariant 3, since v does not lie on a cycle.

If v is a mandatory leaf, we do not apply any reduction rule, since continuing does not violate any invariant. If v is an optional leaf, we remove it by applying Reduction Optional Leaf. This removes the bridge e, thereby preserving Invariant 2.

5.3.4 Backtracking a Cycle

Suppose we now backtrack a cycle $C = (c_1, \dots, c_n, c_1)$. Every vertex $c_i \in \{c_2, \dots, c_n\}$ that is a cutpoint is already finished. By Invariant 3, such a vertex c_i lies on at most one finished block, which must be a bridge $e_i = (c_i, l_i)$ where l_i is a mandatory leaf by Invariant 2. Thus, C is a pendant cycle.

By Theorem 5.8, there exists a reduction rule that we can apply. Each such reduction rule removes at least one of the vertices $c_2, \dots, c_n$ while keeping G connected. After applying the reduction rule, the cycle C is replaced in the block-cutpoint tree by bridges and cutpoints connecting adjacent vertices from C. We then mark all new bridges and all vertices of C that are cutpoints as unvisited and continue the DFS from the cutpoint c_1. We preserve Invariants 2 and 3 because all children of c_1 have become unfinished.

5.3.5 Final Steps

The vertex r is not a cutpoint and initially lies only on the bridge $e_r = (r, v)$ chosen as the root. Thus, r has not been removed. The reduction rules applied while backtracking a bridge do not remove the first vertex of that bridge. The vertex r remains a leaf because no reduction rule adds edges to G. Therefore, v is the only possible neighbor of r.

If G contained more than two vertices after the DFS, there would exist a vertex $x \in V$ with $x \neq r$ that is adjacent to v, since G is connected by Invariant 1. In that case, v would be a finished cutpoint, contradicting Invariant 3. Hence, after the DFS we have $V = \{r\}$ or $V = \{r, v\}$.

If $V = \{r, v\}$, then either v or r must be optional by Invariant 3. We remove that vertex by applying Reduction Optional Leaf. Now assume that G consists of only one vertex. If the final vertex is optional, we remove it by applying Reduction Optional Leaf. Otherwise, it is mandatory, and we remove it by applying Reduction Isolated Vertex. Thus, we have removed all vertices by applying reduction rules.

5.3.6 Correctness and Runtime

We proved the correctness of each individual reduction rule when introducing it. Therefore the original graph has a zero forcing set of size at most k if and only if the graph obtained after applying the reduction rules has a zero forcing set of size at most the updated value of k. Since the algorithm removes all vertices of G, the original graph must have a zero forcing set of size at most k if and only if $k \geq 0$ after the algorithm terminates

The algorithm runs in $\mathcal{O}(|V|)$ time. A DFS on a block-cutpoint tree takes $\mathcal{O}(|V|)$ time. When backtracking from a cutpoint or a bridge, we can determine in $\mathcal{O}(1)$ time whether, and which, reduction rule to apply. The only exception occurs when backtracking a cutpoint that lies on at most two finished blocks, as we may have to apply Reduction Isolated Vertex multiple times and possibly remove a cycle from G.

The additional removal of optional vertices does not affect the overall runtime, since Reduction Isolated Vertex is applied at most $|V|$ times in total. If we remove a vertex from a cycle with n vertices, we mark $\mathcal{O}(n)$ cutpoints and bridges as unvisited. These can be revisited in $\mathcal{O}(n)$ time, since the cutpoints that become unvisited do not lie on a cycle. As each cycle vertex can be removed at most once using Reduction Isolated Vertex, and every edge in a cactus lies on at most one cycle, the additional time is bounded by $\mathcal{O}(|E|) = \mathcal{O}(|V|)$.

When backtracking a cycle with n vertices, we can identify the applicable reduction rule by iterating over the vertices of the cycle in $\mathcal{O}(n)$ time. The corresponding reduction rule can also be applied in $\mathcal{O}(n)$ time. After applying the rule, we mark $\mathcal{O}(n)$ bridges and cutpoints as unvisited, which can again be revisited in $\mathcal{O}(n)$ total time, since each bridge or cutpoint is backtracked in $\mathcal{O}(1)$ time, and none of the revisited cutpoints lies on a cycle. Because each vertex lies on at most one cycle, and each cycle is backtracked at most once, the total time spent backtracking all cycles is $\mathcal{O}(|E|) = \mathcal{O}(|V|)$.

Hence, the overall runtime of the algorithm is linear in the size of the input graph.

5.4 From Induced Path Partition to Zero Forcing

In this chapter, we have presented an algorithm that decides whether there exists a zero forcing set of a cactus G of size at most k by deciding whether there exists an induced path partition of G of size at most k. We can extend our algorithm to compute a minimum induced path partition of G by tracking, for each reduction rule, which edges are part of an induced path. For certain reduction rules, such as Reduction Cycle with Optional Vertex and c_2, c_n Mandatory, there may be multiple sets of induced paths covering the removed vertices. In such a case, the decision regarding which induced paths belong to the minimum induced path partition is deferred. This decision depends on whether an optional vertex will later be used in another induced path. Consequently, we can determine at a later stage which paths from the reduction rule contribute to the minimum induced path partition.

In this section, we show that every induced path partition P of a cactus is a yes-instance of Induced Path Orientation. More precisely, there exists an orientation $\overrightarrow{P}$ of the undirected paths in P such that the set containing the first vertex of each directed induced path in $\overrightarrow{P}$ is a zero forcing set of G. Moreover, $\overrightarrow{P}$ can be computed in linear time. Hence, we can compute a minimum induced path partition P of a cactus in linear time using the algorithm from Section 5.3, and then derive a minimum zero forcing set S from P. This implies that a minimum zero forcing set of a cactus can be obtained in linear time.

Let $G = (V, E)$ be a cactus with a minimum induced path partition P. The set containing the first vertex of each directed induced path in some orientation $\overrightarrow{P}$ of P is a zero forcing set if and only if G, with respect to $\overrightarrow{P}$, does not contain a chain twist. To ensure the absence of a chain twist, we perform a DFS on the block-cutpoint tree of G. Whenever we visit a cycle C, that is a chain twist, we reverse one directed induced path $\overrightarrow{p} \in \overrightarrow{P}$ on C so that C is no longer a chain twist. Since every cycle must be covered by at least two induced paths, we can choose $\overrightarrow{p}$ such that, if C is not the root of the DFS in the block-cutpoint tree of G, the preceding cutpoint in the DFS does not lie on $\overrightarrow{p}$. This ensures that no previously visited cycle becomes a chain twist, as $\overrightarrow{p}$ contains no vertex from any already visited cycle.

After completing the DFS on the block-cutpoint tree of G, the cactus contains no chain twist with respect to $\overrightarrow{P}$. Therefore, the set $S \subseteq V$, consisting of the first vertex of each directed induced path in $\overrightarrow{P}$, is a zero forcing set of G with $|S| = |P|$. This also proves that if P is a minimum induced path partition of G, then there exists a zero forcing set of G of equal size, thereby proving Theorem 5.1.

We can compute $\overrightarrow{P}$ in $\mathcal{O}(|V|)$ time because we can perform a DFS on the block-cutpoint tree in $\mathcal{O}(|V|)$ time, and decide for a cycle C consisting of i vertices in $\mathcal{O}(i)$ time whether the cycle is a chain twist. If C is a chain twist, we can also reverse a directed induced path on

C in $\mathcal{O}(i)$ time. Since G contains $O(|V|)$ edges and every edge lies on at most one cycle, the total time complexity for checking whether a cycle is a chain twist and reversing a directed induced path in case of a chain twist sums up to $\mathcal{O}(|V|)$.

6 FPT Algorithm

One objective of this thesis was to investigate whether an FPT algorithm exists for Zero Forcing on an undirected graph $G = (V, E)$ with $n = |V|$ vertices, when parameterized by the zero forcing number. Since the pathwidth of a graph is known to be a lower bound on its zero forcing number [Aaz08], and since computing a tree decomposition of minimum width is FPT when parameterized by the treewidth [Bod93], the existence of an FPT algorithm for Zero Forcing parameterized by treewidth would imply an FPT algorithm for Zero Forcing parameterized by the zero forcing number.

As shown by Guo et al., an FPT algorithm already exists for the Power Dominating Set problem when parameterized by the treewidth of a graph [GNR08]. They define an edge-orientation problem that is equivalent to Power Dominating Set and design a bottom-up dynamic programming algorithm to solve it. For each subgraph induced by a bag in the tree decomposition, they compute the optimal solution size under all possible valid orientations, which they refer to as *bag states*. Given a tree decomposition of width k, their dynamic program runs in time $\mathcal{O}(c^{k^2} \cdot n)$ for some constant c.

Our initial approach was to either reduce Zero Forcing to Power Dominating Set or to adapt the dynamic programming algorithm of Guo et al. in order to obtain an FPT algorithm for Zero Forcing parameterized by treewidth.

However, during the course of this thesis, Bhyravarapu et al. modified the dynamic programming algorithm of Guo et al. to obtain an FPT algorithm for Zero Forcing parameterized by treewidth [Bhy+25]. Consequently, this result also yields an FPT algorithm for Zero Forcing parameterized by the zero forcing number. Shortly thereafter, Robert Scheffler introduced an algorithm for Zero Forcing with runtime $2^{\mathcal{O}(k^2)} \cdot n$, where $k = Z(G)$ denotes the zero forcing number of G [Sch25].

We were not able to design an FPT algorithm for Zero Forcing that improves upon Scheffler's runtime, nor to establish a reduction from Zero Forcing to Power Dominating Set without significantly increasing the treewidth of the graph.

7 Zero Forcing on Directed Graphs

Up to this point, we have only studied Zero Forcing on undirected graphs. In this chapter, we extend the study of Zero Forcing to directed graphs. We first generalize the definition of Zero Forcing to directed graphs and then prove that Zero Forcing is NP-hard on DAGs.

7.1 Defining Zero Forcing on Directed Graphs

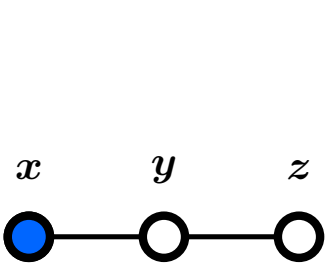
We define Zero Forcing on directed graphs analogously to the undirected case. Let $S \subseteq V$ be a set of initially blue vertices, while all remaining vertices $V \setminus S$ are colored white. We then iteratively apply the *color-change rule*, which for directed graphs is defined as follows:

If there exists a blue vertex $u \in V$ and a white vertex $v \in V$ with $(u, v) \in E$ such that for all other outgoing edges $(u, w) \in E \setminus \{(u, v)\}$ the vertex w is colored blue, then v becomes blue. We denote this application of the color-change rule by $u \to v$.

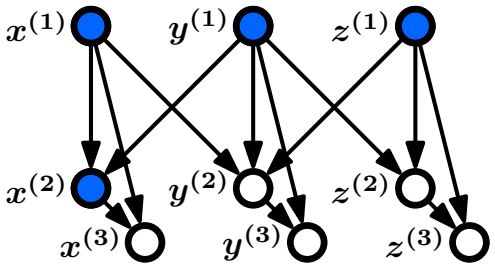
A set $S \subseteq V$ is a *zero forcing set* if, starting with S colored blue and $V \setminus S$ colored white, we can iteratively apply the color-change rule until all vertices are blue. A zero forcing set S is a *minimum zero forcing set* of a directed graph G if there does not exist another zero forcing set S' of G with $|S'| < |S|$. The *zero forcing number* $Z(G)$ of a directed graph G is the minimum integer k such that there exists a zero forcing set of G of size k.

7.2 Zero Forcing is NP-hard on DAGs

In this section, we prove that Zero Forcing is NP-hard on DAGs by a reduction from Zero Forcing on undirected graphs.



(a) Undirected graph with zero forcing set $S = \{x\}$.



(b) DAG with zero forcing set $S' = \{x^{(1)}, y^{(1)}, z^{(1)}, x^{(2)}\}$.

Figure 7.1: Illustration of the reduction from an undirected graph to a DAG.

Theorem 7.1: *The problem Zero Forcing is NP-hard on DAGs.*

Proof. By Theorem 3.17, the problem Zero Forcing is NP-hard on undirected graphs. Let $G = (V, E)$ be an undirected graph. We construct a DAG $G' = (V', E')$ from G as follows:

For each vertex $v \in V$, the directed graph G' contains three vertices $v^{(1)}$, $v^{(2)}$, and $v^{(3)}$. For each undirected edge $\{u, v\} \in E$, we add the directed edges $(u^{(1)}, v^{(2)})$ and $(v^{(1)}, u^{(2)})$ to E'. Additionally, for every $v \in V$, we include the edges $(v^{(1)}, v^{(2)})$, $(v^{(1)}, v^{(3)})$, and $(v^{(2)}, v^{(3)})$.

The construction of G' is illustrated in Figure 7.1. Since every edge in E' is directed from a vertex $u^{(i)}$ to a vertex $v^{(j)}$ with $i < j$, the graph G' is acyclic and therefore a DAG.

It remains to show that there exists a zero forcing set for G of size at most k if and only if there exists a zero forcing set for G' of size at most $k + |V|$.

Assume there exists a zero forcing set $S \subseteq V$ of G of size at most k. Then all vertices in V can be colored blue when the vertices in S are initially blue and those in $V \setminus S$ are initially white, by iteratively applying the color-change rules $u_1 \to v_1, \ldots, u_n \to v_n$.

Define

$$S' = \bigcup_{v \in V} \{v^{(1)}\} \cup \bigcup_{v \in S} \{v^{(2)}\}.$$

Thus, S' contains $|V| + |S| \leq |V| + k$ vertices.

We now show that S' is a zero forcing set of G'. Initially, all vertices in S' are colored blue and all others white. For each $v \in S$, we can apply the color-change rule $v^{(2)} \to v^{(3)}$, since $v^{(2)} \in S'$ and $(v^{(2)}, v^{(3)})$ is the only outgoing edge of $v^{(2)}$.

We now establish the invariant that for each $v \in V$, the vertices $v^{(2)}$ and $v^{(3)}$ in V' are blue if and only if v is blue in G. This invariant holds initially because for each vertex $v \in V$, the vertex $v^{(2)} \in V'$ is in S' if and only if $v \in S$, and if $v^{(2)} \in S'$, then $v^{(3)}$ is also colored blue by $v^{(2)} \to v^{(3)}$.

Now consider a color-change rule $u_i \to v_i$ in G. Since u_i and all its neighbors $w \in N(u_i) \setminus \{v_i\}$ are already colored blue in G, the vertices $u_i^{(2)}$, $u_i^{(3)}$, and $w^{(1)}$ are colored blue in G' by the invariant, while $v_i^{(2)}$ is still white. Thus, $v_i^{(2)}$ is the only white neighbor of $u_i^{(1)}$, and since $u_i^{(1)} \in S'$, we can apply the color-change rule $u_i^{(1)} \to v_i^{(2)}$ in G'. Afterwards, we apply $v_i^{(2)} \to v_i^{(3)}$, because $v_i^{(2)}$ has only one outgoing edge to $v_i^{(3)}$. Having colored the vertices $v_i^{(2)}$ and $v_i^{(3)}$ blue, the invariant is preserved.

After applying all color-change rules $u_1 \to v_1, \ldots, u_n \to v_n$ in G, all vertices in V are colored blue. Consequently, for each $v \in V$, the vertex $v^{(1)}$ is colored blue because $v^{(1)} \in S'$, and the vertices $v^{(2)}$ and $v^{(3)}$ are colored blue by the invariant. Hence, S' is a zero forcing set of G' of size at most $|V| + k$.

Now, let $G' = (V', E')$ be the DAG obtained from an undirected graph $G = (V, E)$. Suppose there exists a zero forcing set $S' \subseteq V'$ of G' of size at most $|V| + k$. Since for each vertex $v \in V$, the vertex $v^{(1)} \in V'$ has no incoming edges, it cannot be colored blue by any color-change rule. Thus, we must have $v^{(1)} \in S'$ for all $v \in V$.

Define

$$S = \{v \in V : v^{(2)} \in S' \text{ or } v^{(3)} \in S'\}.$$

Then $|S| \leq |S'| - |V| \leq k$. We now show that S is a zero forcing set of G.

Since S' is a zero forcing set of G', there exists a sequence of color-change rules $u_1 \to v_1, \ldots, u_n \to v_n$ that colors all vertices of G' blue when initially only the vertices in S' are blue. We simulate this sequence on G, maintaining the invariant that for each $v \in V$, the vertex v is colored blue in G if and only if $v^{(2)}$ or $v^{(3)}$ is colored blue in G'. Initially, this invariant holds by the definition of S.

Now consider a color-change rule in G' of the form $v^{(1)} \to v^{(2)}$, $v^{(1)} \to v^{(3)}$, or $v^{(2)} \to v^{(3)}$ for some vertex $v \in V$. In all these cases, the vertex v must already be colored blue by the invariant. If the color-change rule is of the form $v^{(1)} \to v^{(2)}$, then $v^{(3)}$ must already be colored

blue in G', since $(v^{(1)}, v^{(3)}) \in E'$. Similarly, if the color-change rule is of the form $v^{(1)} \to v^{(3)}$, then $v^{(2)}$ must already be colored blue in G', since $(v^{(1)}, v^{(2)}) \in E'$. If the color-change rule is of the form $v^{(2)} \to v^{(3)}$, then $v^{(2)}$ must be blue in G' to apply $v^{(2)} \to v^{(3)}$.

If a color-change rule in G' is not of these forms, then it must be of the form $u^{(1)} \to v^{(2)}$ for some distinct vertices $u, v \in V$. To apply this color-change rule, the vertices $u^{(2)}$ and $u^{(3)}$ must already be colored blue in G', implying that u is blue in G by the invariant. For every other neighbor $x \in N(u) \setminus \{v\}$, the vertex $x^{(2)}$ must also already be colored blue in G', because $(u^{(1)}, x^{(2)}) \in E'$, and therefore the vertex x must also be colored blue. Thus, in G, the vertex u and all neighbors of u except v are colored blue, which allows us to apply the color-change rule $u \to v$ in G if v is white. This ensures that the invariant continues to hold, since when $v^{(2)}$ becomes blue in G', we also color v blue in G.

After simulating all color-change rules $u_1 \to v_1, \ldots, u_n \to v_n$, all vertices are colored blue in G, because for each vertex $v \in V$ the vertices $v^{(2)}$ and $v^{(3)}$ are colored blue in G', and by the invariant, this implies v is colored blue in G. Hence, S is a zero forcing set of G with size at most k. ∎

8 Conclusion

In this thesis, we presented several complexity results for Zero Forcing. We began by introducing the concept of forcing chains and the related problem Induced Path Partition for undirected graphs. We compared the zero forcing number $Z(G)$ of an undirected graph G with its induced path partition number $P(G)$, showing that $P(G)$ serves as an upper bound for $Z(G)$. Furthermore, we demonstrated that there exist graphs for which the zero forcing number exceeds the induced path partition number by an arbitrarily large factor.

We revisited a known characterization of forcing chains and introduced a novel characterization that enables the verification of whether the directed induced paths of an orientation of an induced path partition are forcing chains of a zero forcing set in an undirected graph. Using this new characterization, we proved that for every non-isolated vertex, there exists a minimum zero forcing set that does not contain that vertex.

Furthermore, we showed that it is NP-hard to determine, for a given induced path partition, whether there exists an orientation of the undirected induced paths such that the set containing the first vertex of each directed path is a zero forcing set. We also provided a new proof showing that Zero Forcing remains NP-hard on undirected bipartite graphs with maximum degree three.

Leveraging the concept of forcing chains, we proved that for any undirected graph with n vertices and m edges, if there exists a zero forcing set S of size k, then the inequality $m \leq k \cdot \left(n - \frac{k+1}{2}\right)$ must hold. Moreover, we showed that it is possible to add edges to any graph until this inequality becomes tight, while ensuring that S remains a zero forcing set. If the inequality is tight and $S \neq V$, then S must be a minimum zero forcing set.

We also showed that for every induced path partition of a cactus, it is possible to orient the undirected induced paths such that they are forcing chains of a zero forcing set. This result enabled us to design a linear-time algorithm for solving Zero Forcing on cacti. Although we discussed recent FPT algorithms for Zero Forcing, we did not present new results. Finally, we extended the definition of Zero Forcing to directed graphs and proved that the problem remains NP-hard on DAGs.

Bibliography

[Aaz08] Ashkan Aazami. “Hardness results and approximation algorithms for some problems on graphs”. University of Waterloo Waterloo, Ontario, Canada, 2008.

[ACDP15] David Amos, Yair Caro, Randy Davila, and Ryan Pepper. “Upper bounds on the k-forcing number of a graph”. In: *Discrete Applied Mathematics* Volume 181 (2015), pp. 1–10.

[AIM08] AIM Minimum Rank – Special Graphs Work Group. “Zero forcing sets and the minimum rank of graphs”. In: *Linear Algebra and its Applications* Volume 428 (2008), pp. 1628–1648. ISSN: 0024-3795. DOI: *https://doi.org/10.1016/j.laa.2007.10.009*.

[ARS20] Meysam Alishahi, Elahe Rezaei-Sani, and Elahe Sharifi. “Maximum nullity and zero forcing number on graphs with maximum degree at most three”. In: *Discrete Applied Mathematics* Volume 284 (2020), pp. 179–194.

[Bar+10] Francesco Barioli, Wayne Barrett, Shaun M Fallat, H Tracy Hall, Leslie Hogben, Bryan Shader, Pauline Van Den Driessche, and Hein Van Der Holst. “Zero forcing parameters and minimum rank problems”. In: *Linear Algebra and its Applications* Volume 433 (2010), pp. 401–411.

[BCDH24] Boštjan Brešar, María Gracia Cornet, Tanja Dravec, and Michael Henning. “Bounds on zero forcing using (upper) total domination and minimum degree”. In: *Bulletin of the Malaysian Mathematical Sciences Society* Volume 47 (2024), p. 143.

[BFH19] Boris Brimkov, Caleb C Fast, and Illya V Hicks. “Computational approaches for zero forcing and related problems”. In: *European Journal of Operational Research* Volume 273 (2019), pp. 889–903.

[BG07] Daniel Burgarth and Vittorio Giovannetti. “Full control by locally induced relaxation”. In: *Physical review letters* Volume 99 (2007), p. 100501.

[BG24] Thomas Bläsius and Max Göttlicher. “An efficient algorithm for power dominating set”. In: *Algorithmica* (2024), pp. 1–33.

[BH17] Boris Brimkov and Illya V Hicks. “Complexity and computation of connected zero forcing”. In: *Discrete Applied Mathematics* Volume 229 (2017), pp. 31–45.

[Bhy+25] Sriram Bhyravarapu, Lawqueen Kanesh, Madhumita Kundu, Daniel Lokshtanov, and Saket Saurabh. “Parameterized Algorithms for Power Edge Set and Zero Forcing Set”. In: *International Workshop on Combinatorial Algorithms*. Springer. 2025, pp. 349–361.

[Bod93] Hans L Bodlaender. “A linear time algorithm for finding tree-decompositions of small treewidth”. In: *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*. 1993, pp. 226–234.

[Bre24] Zachary Brennan. “Probabilistic zero forcing with vertex reversion”. In: *arXiv preprint arXiv:2404.15049* (2024).

[Bri+25] Christopher Brice, Erin Meger, Nhat-Dinh Nguyen, Allen Rakhamimov, and Abigail Raz. "Zero Forcing on Iterated Graph Models". In: *arXiv preprint arXiv:2507.12579* (2025).

[Bur+13] Daniel Burgarth, Domenico D'Alessandro, Leslie Hogben, Simone Severini, and Michael Young. "Zero forcing, linear and quantum controllability for systems evolving on networks". In: *IEEE Transactions on Automatic Control* Volume 58 (2013), pp. 2349–2354.

[CJMZ24] Ben Cameron, Jeannette Janssen, Rogers Matthew, and Zhiyuan Zhang. "An approximation algorithm for zero forcing". In: *arXiv preprint arXiv:2402.08866* (2024).

[DD19] Andreas Darmann and Janosch Döcker. "On simplified NP-complete variants of Not-All-Equal 3-SAT and 3-SAT". In: *arXiv preprint arXiv:1908.04198* (2019).

[DeA14] Luz M DeAlba. "Some results on minimum skew zero forcing sets, and skew zero forcing number". In: *arXiv preprint arXiv:1404.1618* (2014).

[Die25] Reinhard Diestel. *Graph theory*. Vol. 173. Springer Nature, 2025.

[DK19] Shannon Dillman and Franklin Kenter. "Leaky forcing: a new variation of zero forcing". In: *arXiv preprint arXiv:1910.00168* (2019).

[DKS18] Randy Davila, Thomas Kalinowski, and Sudeep Stephen. "A lower bound on the zero forcing number". In: *Discrete Applied Mathematics* Volume 250 (2018), pp. 363–367.

[FJS14] Katherine Fetcie, Bonnie Jacob, and Daniel Saavedra. "The failed zero forcing number of a graph". In: *Involve, a Journal of Mathematics* Volume 8 (2014), pp. 99–117.

[FMY16] Shaun Fallat, Karen Meagher, and Boting Yang. "On the complexity of the positive semidefinite zero forcing number". In: *Linear Algebra and its Applications* Volume 491 (2016), pp. 101–122.

[GNR08] Jiong Guo, Rolf Niedermeier, and Daniel Raible. "Improved algorithms and complexity results for power domination in graphs". In: *Algorithmica* Volume 52 (2008), pp. 177–202.

[GPRS16] Michael Gentner, Lucia D Penso, Dieter Rautenbach, and Uéverton S Souza. "Extremal values and bounds for the zero forcing number". In: *Discrete applied mathematics* Volume 214 (2016), pp. 196–200.

[Gro+08] AIM Minimum Rank–Special Graphs Work Group et al. "Zero forcing sets and the minimum rank of graphs". In: *Linear algebra and its applications* Volume 428 (2008), pp. 1628–1648.

[HHHH02] Teresa W Haynes, Sandra M Hedetniemi, Stephen T Hedetniemi, and Michael A Henning. "Domination in graphs applied to electric power networks". In: *SIAM journal on discrete mathematics* Volume 15 (2002), pp. 519–529.

[Hog10] Leslie Hogben. "Minimum rank problems". In: *Linear Algebra and its Applications* Volume 432 (2010), pp. 1961–1974.

[JZJ23] Yu Jing, Wenqian Zhang, and Shengjin Ji. "The zero forcing number of graphs with the matching number and the cyclomatic number". In: *Graphs and Combinatorics* Volume 39 (2023), p. 72.

[LLX22] Yihui Liang, Jianxi Li, and Shou-Jun Xu. “On Extremal Graphs for Zero Forcing Number”. In: *Graphs and Combinatorics* Volume 38 (2022). DOI: *10.1007/s00373-022-02591-y*.

[LM+03] Hoàng-Oanh Le, Haiko Müller, et al. “Splitting a graph into disjoint induced paths or cycles”. In: *Discrete applied mathematics* Volume 131 (2003), pp. 199–212.

[Row12a] Darren Row. “Zero forcing number, path cover number, and maximum nullity of cacti”. In: *Involve, a Journal of Mathematics* Volume 4 (2012), pp. 277–291.

[Row12b] Darren D. Row. “A technique for computing the zero forcing number of a graph with a cut-vertex”. In: *Linear Algebra and its Applications* Volume 436 (2012), pp. 4423–4432. ISSN: 0024-3795. DOI: *https://doi.org/10.1016/j.laa.2011.05.012*.

[Sch25] Robert Scheffler. “On the Parameterized Complexity of Grundy Domination and Zero Forcing Problems”. In: *arXiv preprint arXiv:2508.18104* (2025).

[TD15] Maguy Trefois and Jean-Charles Delvenne. “Zero forcing number, constrained matchings and strong structural controllability”. In: *Linear Algebra and its Applications* Volume 484 (2015), pp. 199–218.