4-2022

Graph Arrowing: Constructions and Complexity

Zohair Raza Hassan
zh5337@rit.edu

Graph Arrowing: Constructions and Complexity

by

Zohair Raza Hassan

A thesis submitted in partial fulfillment of the
requirements for the degree of
Master of Science
in Computer Science

B. Thomas Golisano College of Computing and
Information Sciences

Rochester Institute of Technology
Rochester, New York
April 2022

Graph Arrowing: Constructions and Complexity

by

Zohair Raza Hassan

Edith Hemaspaandra — Date
Thesis Advisor

Stanisław Radziszowski — Date
Thesis Advisor

Ivona Bezáková — Date
Reader

Brendan Rooney — Date
Observer

Graph Arrowing: Constructions and Complexity

by

Zohair Raza Hassan

Submitted to the
B. Thomas Golisano College of Computing and Information Sciences
MS Program in Computer Science
in partial fulfillment of the requirements for the
Master of Science Degree
at the Rochester Institute of Technology

Abstract

Graph arrowing is concerned with determining which monochromatic subgraphs are unavoidable when coloring a given graph. There are two main avenues of research concerning arrowing: finding extremal Ramsey/Folkman graphs and categorizing the complexity of arrowing problems. Both avenues have been studied extensively for decades. In this thesis, we focus on graph arrowing problems where one of the monochromatic subgraphs being avoided is the path on three vertices, denoted as P_3. Our main contributions involve computing Folkman numbers by generating graphs up to 13 vertices and proving the coNP-completeness of some arrowing problems using a novel reduction framework geared towards avoiding P_3's.

The (P_3, H)-Arrowing Problem asks whether a given graph can be colored using two colors (red and blue) such that there are no red P_3's and no blue H's, where H is a fixed graph. The few previous hardness proofs for arrowing problems relied on ad-hoc, laborious constructions of "gadgets." We introduce a general framework that can be used to prove the coNP-completeness of (P_3, H)-arrowing problems. We search for gadgets computationally. These gadgets allow us to simulate variants of SAT, thus showing coNP-hardness.

Finally, we use our (P_3, H)-Arrowing hardness reductions to gain insight into variants of Monotone SAT. For fixed $k \in \{4, 5, 6\}$, we show that Monotone SAT remains NP-complete under the following constraints: 1) each clause consists of exactly two unnegated literals or exactly k negated literals, 2) the variables in each clause are distinct, and 3) the number of times a variable occurs in the formula is bounded by a constant.

For future work, we expect that the insight gained by our computationally assisted reductions will help us prove the complexity of other elusive arrowing problems.

Acknowledgments

I want to thank my advisors, Professors Edith and Staszek, for introducing me to a research topic I'm passionate about, allowing me to explore different ideas at my pace, and being a constant source of support throughout my graduate school experience. Thank you for believing in me. It has been a privilege to work under your supervision, and I look forward to holding on to this privilege for the next few years.

I'd also like to thank Professors Ivona and Brendan for reading my thesis, providing their feedback, and sharing my enthusiasm for the research presented in this work.

I am grateful to the United States Educational Foundation in Pakistan and the Institute of International Education for granting me the Fulbright scholarship and providing the opportunity to pursue higher education in the US.

I want to thank my friend Talha, who always believed in me and kept my head held high. I'd like to thank his parents, Drs. Khadija and Irfan, for regularly checking up on me, keeping in touch, and making sure I'm doing alright.

I want to thank my friends Bilal, Sayaan, and Sultan for their support, hospitality, and making my transition from Lahore to the US easier. I'd also like to thank my labmate, Maheen Contractor, for her friendship and for making my time in Rochester a fun and memorable experience.

Finally, I want to thank my family. After having lived in Lahore for all of my life, moving to the US and not having all of you by my side was a painful experience. However, every message, phone call, and Zoom session we shared made these circumstances a little bit easier. Thank you for providing the unconditional love and support I needed to make it this far and for continuing to support me, even while being 7000 miles away.

In loving memory of Talha.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Graph arrowing is concerned with coloring graphs such that the monochromatic subgraphs of each coloring adhere to a specified structural property. For example, take the complete graph on six vertices, commonly referred to as K_6 (see Figure 1.1).

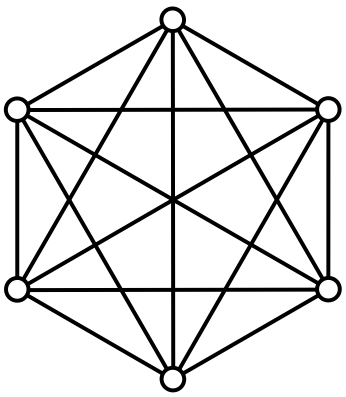


Figure 1.1: The complete graph on six vertices.

Assume that we are coloring the graph's edges with two colors: red and blue. It can be shown that in every possible coloring of K_6, there must exist either a red triangle or a blue triangle—where a triangle is the complete graph on three vertices. Some examples have been provided in Figure 1.2. This fact is also commonly expressed as "each 2-edge-coloring of K_6 must have a monochromatic triangle." The mathematical notation to put this concisely is $K_6 \to (K_3, K_3)^e$, pronounced, K_6 *arrows* K_3, K_3. The 'e' in the superscript denotes that edges are being colored. Note that this

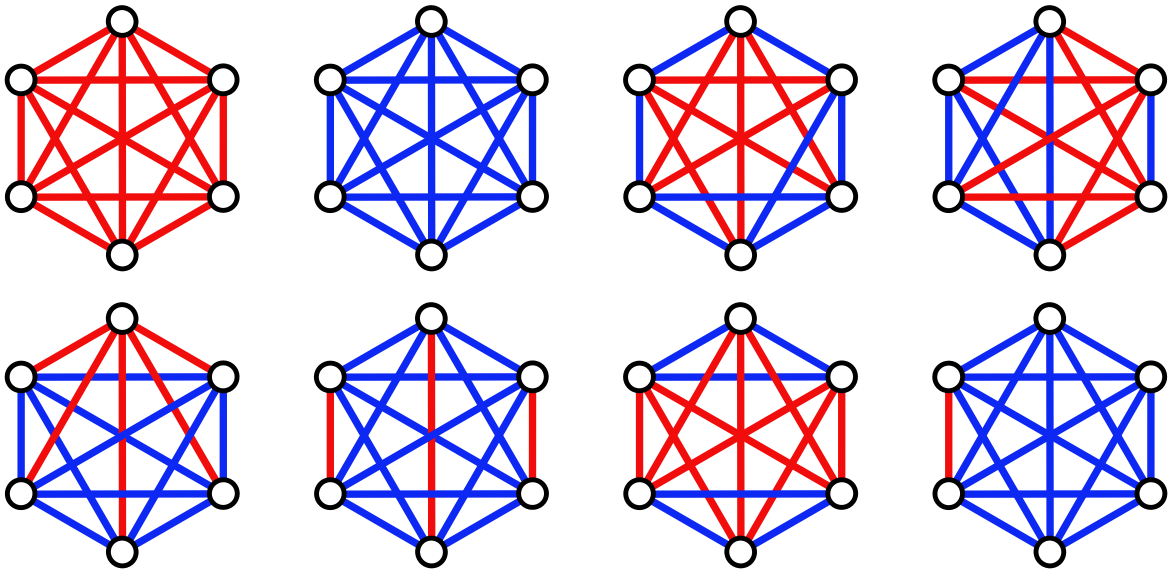


Figure 1.2: Eight out of 32,768 possible 2-(edge)-colorings of K_6.

concept can be generalized to k colors.

Graph arrowing can be traced back to 1930 when Ramsey formalized the concept of Ramsey numbers: the order of the smallest complete graph that arrows specified subgraphs [35]. For example, K_6 is the smallest complete graph that arrows $(K_3, K_3)^e$. In Ramsey notation, this is written as $R(K_3, K_3) = 6$. Since its inception, Ramsey numbers have been explored for many different families of graphs, such as cliques, paths, and cycles [5,6,15,34]. This concept was further generalized to "Folkman numbers," where the graph being colored is not necessarily a complete graph and must not contain a specified subgraph [2, 7, 13, 21, 27, 39]. Arrowing decision problems are concerned with determining whether a given graph arrows some fixed $(H_1, H_2, \ldots, H_k)^e$, where H_i is a graph and k is the number of colors being used. Many such problems have been proven to be solvable in polynomial time [3] or to be coNP-complete [4, 14, 36].

This thesis aims to solve problems concerning arrowing by exploring these problems through the lens of computational combinatorics and complexity theory. This includes finding extremal graphs that adhere to properties related to arrowing and classifying arrowing problems into various complexity classes. We list below a few reasons why we believe arrowing is interesting and why this work is worth exploring:

1. **Intellectual curiosity.** Ramsey theory is often described as "finding order within chaos."

In his seminal paper in 1930, Ramsey essentially proved that within large enough structures, specific patterns are unavoidable [22]. It is interesting to see how this observation unfolds for different structures, such as cliques, paths, cycles, etc. Even more so, it is interesting to see if these patterns remain unavoidable despite certain substructures being forbidden, as Folkman did.

2. **Delineating the line that divides P and NP-complete problems.** By exploring which subgraphs permit polynomial-time solutions and which don't, we may be able to understand better the line that divides P-time problems and NP-complete/coNP-complete problems—at least for arrowing problems. For instance, determining whether a graph arrows $(P_3, P_3)^e$ can be done in polynomial time, but doing so for $(P_4, P_4)^e$ is coNP-complete, where P_3 and P_4 are the path graphs on three and four vertices, respectively. We believe it would be interesting to obtain sets of problems in P and coNP-complete and compare them to see what inherent difference divides these problems.

3. **More problems for hardness proofs.** Many hardness proofs are based on reductions from variants of SAT. Different variants impose different constraints, allowing for more accessible hardness proofs. For instance, geometric NP-complete problems may have easier reductions from planar or rectilinear variants of SAT. Arrowing naturally has infinitely many variants, many of which are coNP-complete. Due to its diversity, we believe that arrowing problems could also serve as a useful source for reductions.

This work focuses on graph arrowing where P_3 is being avoided. We summarize our contributions below:

1. **Novel reduction framework for (P_3, H)-Arrowing.** For fixed H, the (P_3, H)-Arrowing problem decides whether the edges of a given graph can be colored red and blue such that there are no red P_3's and no blue H's. In Chapter 5, we describe a framework that allows us to prove hardness results. We first formally define graphs with special colorings, known as "gadgets." We then show how one can prove that (P_3, H)-Arrowing is coNP-complete using these gadgets. Finally, we describe a methodology that allows us to search for gadgets computationally.

2. **New complexity results.** In Chapter 5 we showcase the utility of the aforementioned reduction framework by showing that (P_3, H)-Arrowing is coNP-complete for seven distinct H. This result is formally stated in Theorem 5.1.

3. **New Folkman numbers.** In Chapter 6 we present new Folkman numbers of the form $F_e(P_3, H; I)$ where H is one of nine select graphs, and I is a cycle, complete graph, or complete graph missing an edge. We show the smallest graphs G such that G is I-free, and $G \to (P_3, H)^e$. The results are summarized in Theorem 6.2.

4. **Insight on SAT Variants.** In Chapter 7 we prove the NP-hardness of special variants of a variant of SAT called Monotone SAT. Monotone SAT and different variants of it have been explored in the past [1, 10, 11, 17]. By reducing from arrowing, we show that we are able to impose more constraints on the Monotone SAT variant than otherwise would be possible from a direct, trivial reduction from other SAT problems, while still maintaining the NP-hardness of the problem. Our main result is stated in Theorem 7.2.

We list all the major theorems proven in this work below. The graphs mentioned below have been defined in Table 2.1 of Chapter 2. Monotone $(2, k)$-SAT has been defined in Chapter 7.

Theorem 5.1. *(P_3, H)-Arrowing is* coNP*-complete for $H \in \{C_4, C_5, C_6, J_4, BT, P_5, P_6\}$.*

Theorem 6.2. *Let $\mathcal{Y} = \{K_3, P_4, P_5, P_6, C_4, C_5, C_6, J_4, BT\}$ and $\mathcal{I} = \{K_6, K_5, K_4, K_3, J_6, J_5, J_4, C_6, C_5, C_4\}$. For $H \in \mathcal{Y} \setminus \{BT\}$ and $I \in \mathcal{I}$, $F_e(P_3, H; I) \leq 13$, or does not exist. For $I \in \mathcal{I} \setminus \{J_4, C_4\}$, $F_e(P_3, BT; I) \leq 10$, or does not exist. The exact values for each number can be found in Table 6.2.*

Theorem 7.2. *For fixed $k \in \{4, 5, 6\}$, Monotone $(2, k)$-SAT is* NP*-complete even when the variables in each clause are distinct, and the number of times each variable occurs is bounded by a constant. In particular:*

1. *Monotone $(2, 4)$-SAT is* NP*-complete even when each variable appears at most 11 times, where a variable appears as an unnegated literal at most eight times and as a negated literal at most three times.*

2. *Monotone $(2, 5)$-SAT is* NP*-complete even when each variable appears at most 11 times, where a variable appears as an unnegated literal at most seven times and as a negated literal at most four times.*

3. *Monotone $(2, 6)$-SAT is* NP*-complete even when each variable appears at most 15 times, where a variable appears as an unnegated literal at most nine times and as a negated literal at most six times.*

The thesis is organized as follows. In Chapter 2 we describe the notation and terminology used throughout the thesis. In Chapter 3, we present a literature review of works related to graph

arrowing. In Chapter 4 we describe a basic methodology to generate the graphs that have "good" colorings. These graphs are used in Chapters 5 and 6 to obtain the results mentioned before. Chapter 7 uses the hardness results from Chapter 5 to explore variants of SAT. Finally, we conclude in Chapter 8, wherein we discuss the future directions of our work.

Chapter 2

Preliminaries

This chapter lists the notation and terminology used throughout the thesis.

2.1 Graphs

A graph $G = (V(G), E(G))$ is a two-tuple of vertices and edges. The vertex set is denoted as $V(G)$, and the edge set as $E(G)$. For a graph G, $|V(G)|$ is referred to as the order of said graph. Two graphs G and H are isomorphic if there exists a bijection $\pi : V(G) \leftrightarrow V(H)$ such that $\{\{\pi(u), \pi(v)\} \mid \{u, v\} \in E(G)\} = E(H)$. Some special graphs and their notation have been mentioned in Table 2.1.

For a graph G and a subset $V' \subseteq V(G)$, $G[V']$ is defined as the graph with vertex set $V(G[V']) = V'$ and edge set $E(G[V']) = \{\{u, v\} \in E(G) \mid u, v \in V'\}$. We define a similar concept for edges: given a subset of edges $E' \subseteq E(G)$, $G[E']$ is defined as the graph with $E(G[E']) = E'$ and $V(G'[E']) = \{u \in V(G) \mid \{u, v\} \in E'\}$.

We say that H is an induced subgraph of G if there exists a subset $V' \subseteq V(G)$ such that $G[V']$ is isomorphic to H. We say that H is a subgraph of G if there exists a subset $V' \subseteq V(G)$ such that the removal of some edges from $G[V']$ makes $G[V']$ isomorphic to H.

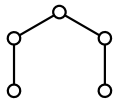
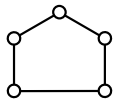
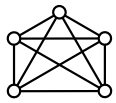
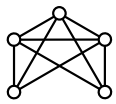
Notation	Description	Example on $k = 5$
P_k	The path graph on k vertices, where $k \geq 1$.	
C_k	The cycle graph on k vertices, where $k \geq 3$.	
K_k	The complete graph on k vertices, where $k \geq 1$.	
J_k	The complete graph minus one edge on k vertices, where $k \geq 2$.	

Table 2.1: The special graphs used throughout this thesis.

2.2 Coloring & Arrowing

A k-coloring of the edges of a graph G is a partition of $E(G)$ into k sets. Similarly, a k-coloring of the vertices of a graph G is a partition of $V(G)$ into k sets. Examples of a 2-(edge)-coloring of K_6 are provided in Figure 1.2. The chromatic number of a graph G is denoted as $\chi(G)$ and is equal to the smallest number of colors required to color the graph's vertices, such that no adjacent vertices have the same color.

Definition 2.1 (Arrowing)**.** *Let $G, H_1, H_2, \ldots,$ and H_k be graphs, where $k \geq 1$. We say that $G \to (H_1, H_2, \ldots, H_k)^e$ if in every k-coloring of the edges of G, there exists at least one color i such that the subgraph corresponding to the i^{th} color has H_i as a subgraph. The notation $G \to (H_1, H_2, \ldots, H_k)^v$ is used similarly for vertex arrowing. The notation is summarized below:*

$$G \to (H_1, H_2, \ldots, H_k)^a$$

Graph being colored; Graphs to avoid in each color; Number of colors being used; Object being colored

As an example, consider $K_6 \to (K_3, K_3)^e$ again. Assume we are coloring the edges red and blue.

The notation summarizes that in every 2-coloring of K_6, the red subgraph of G contains K_3, or the blue subgraph of G contains K_3. In this work, we focus on edge arrowing with two colors. Thus, when we discuss colorings, we assume that we are coloring the edges red and blue. In this vein, we also define the following terminology:

Definition 2.2 ((H_1, H_2)-good coloring)**.** *An (H_1, H_2)-good coloring of a graph is one where there are no red H_1's and no blue H_2's when the edges are colored red or blue.*

Definition 2.3 ((H_1, H_2)-good graph)**.** *A graph G is (H_1, H_2)-good if there exists at least one (H_1, H_2)-good coloring of G.*

Note that we also use the phrase $G \not\to (H_1, H_2)^e$ to say that G is (H_1, H_2)-good. We now define the decision problem studied in this paper. Let H_1 and H_2 be fixed graphs. The basic edge arrowing problem on two colors is defined as follows:

Problem 2.1 ((H_1, H_2)-Arrowing)**.** *Given a graph G, does $G \to (H_1, H_2)^e$?*

Note that the problem is phrased in this manner due to convention [3]. However, for simplicity, we define the Non-Arrowing problem as it is easier to work with problems in NP rather than coNP:

Problem 2.2 ((H_1, H_2)-Non-Arrowing)**.** *Given a graph G, does $G \not\to (H_1, H_2)^e$?*

It is easy to see that (H_1, H_2)-Arrowing is in coNP since an (H_1, H_2)-good coloring is a certificate that can be verified in polynomial time for (H_1, H_2)-Non-Arrowing.

2.3 Ramsey & Folkman Numbers

Let I, $H_1, H_2, \ldots$, and H_k be graphs. We define Ramsey and Folkman numbers below:

Definition 2.4 (Ramsey numbers)**.** *The Ramsey number $R(H_1, H_2, \ldots, H_k)$ is defined as the smallest n such that $K_n \to (H_1, H_2, \ldots, H_k)^e$.*

Definition 2.5 (Folkman numbers)**.** *The Folkman number $F_e(H_1, H_2, \ldots, H_k; I)$ is the order of the smallest G such that G is I-free and $G \to (H_1, H_2, \ldots, H_k)^e$.*

Although in this work we focus on edge colorings, Folkman numbers are defined similarly for vertex colorings: $F_v(H_1, H_2, \ldots, H_k; I)$ is the order of the smallest G such that G is I-free and $G \to (H_1, H_2, \ldots, H_k)^v$.

Chapter 3

Related Work

This section provides a brief literature review of the works related to graph arrowing. We first discuss existing results on the computational complexity of graph arrowing in Section 3.1, followed by a brief history of Ramsey and Folkman numbers and the approaches taken to compute them in Section 3.2. In Sections 3.3 and 3.4, we discuss other problems with links to graph arrowing.

3.1 Complexity of Arrowing

In the late 1980s, Rutenburg [36] explored the Generalized Coloring Problem: $GCP_{\mathcal{F},k}$, where $k \geq 1$ is an integer and $\mathcal{F}$ is a finite family of graphs. The problem asks to determine whether a given graph's vertices can be colored using k colors such that there are no monochromatic subgraphs isomorphic to a graph in $\mathcal{F}$. Observe that when $k = 2$ and $\mathcal{F} = \{H\}$, the problem is equivalent to determining $G \not\to (H, H)^v$, and extends to more colors. Note that this covers all symmetric vertex arrowing problems, i.e., where the same subgraph is being avoided in each color.

The general edge arrowing problem is to determine whether $G \to (H_1, H_2)^e$, given graphs G, H_1, and H_2. This has been proven to be Π_2^p-complete by Schaefer [37]. However, one can construct several variants of the problem by considering cases where H_1 or H_2 are set to be specific graphs, as in Problem 2.1. Several of these problems have been shown to be in P or coNP-complete. For instance, one can devise a simple greedy algorithm to show that (P_3, P_3)-Arrowing is solvable in polynomial time. Burr et al. [3] showed that (H_1, H_2)-Arrowing is solvable in polynomial time when H_1 and H_2 are stars, or when H_1 is a fixed matching and H_2 is any fixed graph.

coNP-complete problems include (K_3, K_3)-Arrowing [14] and (P_4, P_4)-Arrowing [36]. The problem has also been shown to be coNP-complete for an infinite family of graphs: Let Γ_3 be the set of all 3-connected graphs[1] and K_3. It has been shown that (H_1, H_2)-Arrowing is coNP-complete when H_1 and H_2 are fixed graphs such that $H_1, H_2 \in \Gamma_3$ [4]. The proof behind this is based on the existence of graphs known as "senders." For (H_1, H_2)-coloring, positive senders are graphs with two edges e and f that have (H_1, H_2)-colorings such that e is red (resp., blue) iff f is red (resp., blue). Negative senders are defined similarly, but e is red (resp., blue) iff f is blue (resp., red). By proving the existence of these senders, one can construct gadgets that allow a reduction from a variant of SAT [4].

We note that there is little work on the case where only one graph is fixed, i.e., H_1 is fixed, and H_2 is part of the input. A result that is easy to observe is that when $H_1 = K_2$ and G and H_2 are provided as input, determining whether $G \to (H_1, H_2)^e$ is equivalent to the subgraph isomorphism problem, which is NP-complete [14].

3.2 Finding Ramsey and Folkman numbers

Ramsey Theory was introduced in the 1930s. Originally, it was concerned with finding the smallest complete graph K_n such that $K_n \to (K_j, K_\ell)^e$ for complete graphs K_j and K_ℓ. For given K_j and K_ℓ, this is known as the Ramsey number $R(j, \ell)$. In his seminal paper, Ramsey [35] proved the existence of these complete graphs for any given K_j and K_ℓ. Now, Ramsey theory has been extended to include k colors, and avoid any given subgraph: for graphs H_1, H_2, ..., H_k, $R(H_1, H_2, \ldots, H_k) = n$ is the order of the smallest K_n such that $K_n \to (H_1, H_2, \ldots, H_k)^e$. Small Ramsey numbers can be found via theoretical analysis, but significant strides were made in the field when we could harness computational power to aid in the search for Ramsey numbers. Radziszowski maintains a detailed survey [34] of discovered Ramsey numbers and their bounds.

Folkman numbers can be viewed as an extension of Ramsey numbers. Unlike Ramsey numbers, Folkman numbers look for any graph on n vertices instead of complete graphs and also require that the graph being colored must be I-free for some graph I. Folkman-esque problems have been around since before Folkman's theorem. Graham proved that $F_e(K_3, K_3; K_6) = 8$ in 1968, and Erdős and Hajnal first conjectured the existence of a graph G that is K_4-free and $G \to (K_3, K_3)^e$ in 1967. It wasn't until 1970 that Folkman proved the existence of such a graph and the concept of

[1] A 3-connected graph is a graph of order at least 4 that requires the removal of at least 3 vertices to become disconnected.

Folkman numbers was formalized. For in-depth surveys on Folkman numbers, we refer the reader to the works of Bikov [2] and Wood [39].

Finding Ramsey/Folkman numbers and their bounds is a computationally expensive task. For orders up to 12, one can enumerate over all graphs to find the desired graph. Going beyond 12 vertices is a daunting task, as there are 50 trillion connected graphs on 13 vertices, and exponentially more as we go forward. To explore graphs over more vertices requires a meticulous approach; by proving the necessary properties of the desired graphs, one can write programs to generate only these graphs and avoid enumeration over unwanted graphs.

For example, to show that the Folkman number $F_v(K_2, K_2, K_2, K_2; J_4) \leq 15$, Hassan et al. [21] first showed that if a corresponding graph G exists, then G can be split into subgraphs G_1 and G_2 such that G_1 is an independent set on four vertices, and G_2 is a J_4-free graph such that $\chi(G_2) \geq 4$.

We refer the reader to the following accessible works on vertex Folkman numbers to get an idea of this process: (1) Jensen and Royle's [25] and Goedgebeur's [16] work on K_3-free Folkman numbers, (2) Coles and Radziszowski's [7] and Lathrop and Radziszowski's [27] work on K_4-free Folkman numbers, and (3) Hassan et al.'s [21] work on J_4-free Folkman numbers.

3.3 Graph Coloring

In this section, we discuss other problems that involve coloring graphs. Coloring graphs has been a popular research avenue for decades. The most common practice in coloring is to find the minimum number of colors required to color the vertices of a given graph such that no vertices connected by an edge have the same color. This number is known as the chromatic number of the graph—a similar concept has been defined for edges, where no two adjacent edges may have the same color, and it is known as the chromatic index of a graph.

The k-colorability problem decides whether a graph's vertices can be colored with k colors, such that no adjacent vertices share the same color. This problem was one of the first problems proven to be NP-complete via a reduction from SAT by Karp in his seminal paper with 21 similar proofs via reductions [14, 26]. It is also known, for fixed k, that determining whether a graph's vertices can be colored using k colors is NP-complete when $k \geq 3$, but can be done in polynomial time when $k \leq 2$. Similarly, determining the k-colorability of edges is also NP-complete [23], for $k \geq 3$.

Observe that determining whether a graph G's vertices are 3-colorable is equivalent to deciding

whether $G \not\to (K_2, K_2, K_2)^v$. Similarly, determining whether a graph G's edges are 3-colorable is equivalent to determining whether $G \not\to (P_3, P_3, P_3)^e$. It is easy to see that these reductions generalize to larger k as well. Observe that graph arrowing explores coloring more generally: we may disallow any graph in each color instead of K_2's or P_3's.

We note that other coloring problems have also been explored. For example, the defective coloring problem studies vertex colorings of graphs where adjacent vertices are allowed to share colors under some constraints: in a (k, d)-coloring, vertices are colored with at most k colors, and a vertex is allowed to have at most d neighbors of the same color [5, 8, 9, 19]. Many instances of defective colorings are NP-complete. For instance, $(2, d)$-coloring is known to be NP-complete for $d \geq 1$.

3.4 Matching Removal

A matching M is defined as a set of disjoint edges of a graph, i.e., no two edges in M share a common vertex. Matching removal problems can generally be defined as follows: for a fixed graph property Π and some given graph G, does there exist a matching $M \subseteq E(G)$ such that the graph $G' = (V(G), E(G) \setminus M)$ has property Π? [29] We show below how certain arrowing problems can be rephrased as matching removal problems and discuss some matching removal problems that have been explored in the past.

Consider the following problem: Given G, does $G \not\to (P_3, H)^e$ for some fixed H? Note that in any $(P_3, H)^e$-good coloring of G, one may only have disjoint P_2's in the red subgraph, and that the blue subgraph must be H-free. This shows that determining whether $G \not\to (P_3, H)^e$ is equivalent to a matching removal problem where the property Π is that G' must be H-free.

Recently, work has been done on the acyclic property: determining if a matching exists that gives an acyclic graph on removal. This was proven to be NP-complete by Lima et al. [28], who also worked on a variant of this problem known as Odd Decycling Matching, where the graph must have no odd cycles after removal. This can also be viewed as "bipartization," i.e., making the resulting graph bipartite. They show that this problem is NP-complete [29].

An interesting point to note is the relationship between Odd Decycling Matching and defective coloring. It has been shown that a graph has a $(2, 1)$-coloring if and only if there exists an odd decycling matching [30].

Chapter 4

Generating (H_1, H_2)-good Graphs

Combinatorial computing methodologies begin with the generation of combinatorial objects. Once generated, the objects are analyzed to find ones with desired properties. In this section, we discuss a basic framework that iteratively generates (H_1, H_2)-good graphs, given graphs H_1 and H_2. In particular, we discuss how to generate all (H_1, H_2)-good graphs on $n+1$ vertices, given the set of all such graphs on n vertices. These graphs are used to find the gadgets in Chapter 5 and discover the Folkman numbers in Chapter 6. Section 4.1 introduces some tools required for our methodology described in Section 4.2.

4.1 Preliminary Tools

Our methodology requires three tools: an isomorphism checker, a graph extender, and finally, a graph colorer. This section elaborates on the utility of these three tools.

4.1.1 Isomorphism Checker

The Graph Isomorphism Problem is defined as follows:

Problem 4.1 (Graph Isomorphism)**.** *Given graphs G and F, is G isomorphic to F?*

As of yet, Graph Isomorphism has no known polynomial-time solution. However, in 1984, McKay developed `nauty`: a practical solution to the Graph Isomorphism problem [32]. `nauty` maps each

graph to a unique string. These strings are constructed based on the adjacency matrix of a graph. Since a single graph may be represented by many adjacency matrices, McKay proposed an algorithm that rearranges the order of a graph's vertices to a "canonical" ordering, allowing us to obtain unique string representations. In our methodology, we use `nauty` to discard duplicate graphs generated during our iterative process. Note that mapping graphs to strings allows us to quickly remove duplicates from lists of graphs by sorting the list of strings and removing adjacent matching strings.

4.1.2 Graph Extender

Let G be a graph on n vertices. We say that $\widehat{G}$ is an extension of G if $\widehat{G}$ is a graph on $n+1$ vertices and G is an induced subgraph of $\widehat{G}$. Given a graph G, a graph extender is a function that returns all possible extensions of G. A simple algorithm for such a function is given in Algorithm 1.

Algorithm 1: Graph Extender

Function `Extend(`G`)`

1 $X_G \leftarrow \emptyset$
2 Let $\mathcal{P}$ be the set of all subsets of $V(G)$
3 **for** $V' \in \mathcal{P}$ **do**
4 Construct $\widehat{G}$ by making a copy of G, adding a new vertex u, and adding an edge between u and each vertex in V'
5 Add $\widehat{G}$ to X_G
end
6 Remove duplicates from X_G using the isomorphism checker (Section 4.1.1)
7 **return** X_G

4.1.3 Graph Colorer

Given graphs G, H_1, and H_2, a graph colorer is a function that returns all possible (H_1, H_2)-good colorings of G. We note that it can be difficult to obtain a good coloring efficiently due to the hardness of graph arrowing. However, modern SAT solvers are practical enough to provide solutions efficiently for small instances. In this vein, we propose obtaining colorings by constructing a CNF-SAT formula $\phi_G^{H_1,H_2}$ such that there is a bijection between the satisfying assignments of $\phi_G^{H_1,H_2}$ and the (H_1, H_2)-good colorings of G.

For graphs G and H, let S_G^H be the set of all subsets of $E(G)$ such that $G[E']$ is isomorphic to H. Let e_i be the i^{th} edge in $E(G)$. Let x_i be a variable in ϕ_G corresponding to e_i. Recall that we are working with two colors: red and blue. Let the value of a variable correspond to a color: without loss of generality, assume that x_i is true iff e_i is blue. Let $S \in S_G^{H_1}$. Clearly, we must have at least one red edge in S in any (H_1, H_2)-good coloring of G. This rule can be represented by a CNF clause like so:

$$\bigvee_{e_i \in S} x_i$$

Similarly, for any $S \in S_G^{H_2}$ to have at least one blue edge, we can use the clause:

$$\bigvee_{e_i \in S} \overline{x_i}$$

Thus, we can construct $\phi_G^{H_1,H_2}$ like so:

$$\phi_G^{H_1,H_2} = \bigwedge_{S \in S_G^{H_1}} \left(\bigvee_{e_i \in S} x_i \right) \wedge \bigwedge_{S \in S_G^{H_2}} \left(\bigvee_{e_i \in S} \overline{x_i} \right)$$

Note that we can generate $\phi_G^{H_1,H_2}$ in polynomial time when H_1 and H_2 are fixed graphs. Also note that when generating graphs, only one good coloring is required to verify that the graph is (H_1, H_2)-good. Thus, in our implementation, we only checked the existence of a single satisfying assignment for the corresponding formula.

Algorithm 2: Constructing (H_1, H_2)-good graphs

Input : $\mathcal{G}_n$ = All (H_1, H_2)-good graphs on n vertices

Output: $\mathcal{G}_{n+1}$ = All (H_1, H_2)-good graphs on $n+1$ vertices

1 $\mathcal{G}_{n+1} \leftarrow \emptyset$
2 **for** $G \in \mathcal{G}_n$ **do**
3 $\quad X_G \leftarrow$ `Extend`(G)
4 $\quad$ Add all graphs from X_G to $\mathcal{G}_{n+1}$, removing duplicates if necessary
end
5 **for** $G \in \mathcal{G}_{n+1}$ **do**
6 $\quad$ Let Y_G be the set of all connected subgraphs of G of order n
7 $\quad$ **if** any $G' \in Y_G$ is not a member of $\mathcal{G}_n$ **then**
8 $\quad\quad$ Remove G from $\mathcal{G}_{n+1}$
$\quad$ **else**
9 $\quad\quad$ Check if G has a good coloring using the graph colorer
10 $\quad\quad$ **if** G does not have a good coloring **then**
11 $\quad\quad\quad$ Remove G from $\mathcal{G}_{n+1}$
$\quad\quad$ **end**
$\quad$ **end**
end
12 **return** $\mathcal{G}_{n+1}$

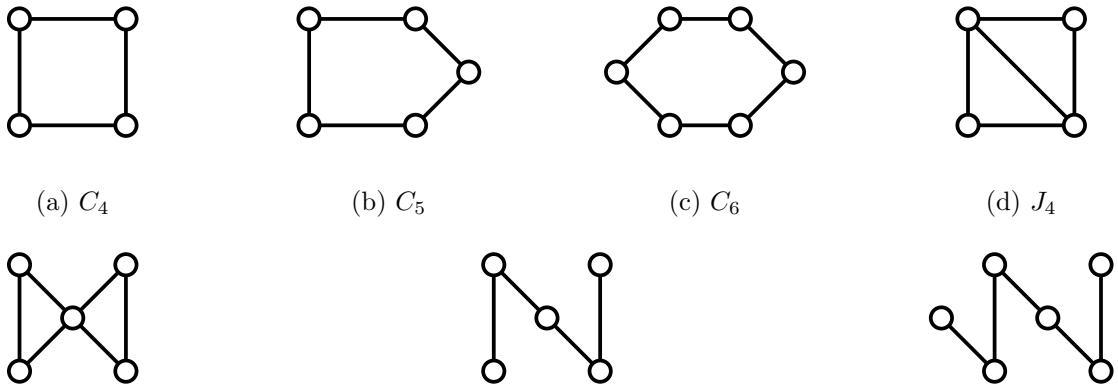
4.2 Methodology

Using the tools described above, we implemented the methodology described in Algorithm 2 to incrementally generate our (H_1, H_2)-good graphs. Observe that in lines 2–4, we obtain all possibly good graphs from the previous set, and in lines 5–11, we filter out graphs which arrow $(H_1, H_2)^e$. Note that in lines 6–8, we check subgraphs of each graph to see if each subgraph is (H_1, H_2)-good. This subgraph check allows us to skip calling the SAT solver, allowing for more efficient computation.

In our implementation, we started with all connected graphs on three vertices (since there are only two: P_3 and K_3) and iteratively generated the next sets. The methodology was implemented using C and Python. We want to acknowledge the use of NetworkX [18], pynauty [33], pySAT [24], and GrandIso [31] in our implementation.

Chapter 5

(P_3, H)-Arrowing for some fixed H



(a) C_4 (b) C_5 (c) C_6 (d) J_4

(e) BT (f) P_5 (g) P_6

Figure 5.1: Graphs in $\mathcal{Z}$.

In this chapter, we propose a simple framework for coNP-hardness proofs of (P_3, H)-Arrowing problems, where H is a fixed graph. We show that (P_3, H)-Arrowing is coNP-complete for $H \in \mathcal{Z} = \{C_4, C_5, C_6, J_4, BT, P_5, P_6\}$, where BT—referred to as the butterfly graph—is the graph where two triangles share a vertex. Each graph is shown in Figure 5.1.

For each $H \in \mathcal{Z}$, we build "gadgets" that allow us to simulate CNF-SAT formulas as arrowing problems. The term gadget refers to a small graph that simulates a component of a boolean formula. In our reduction, there are two types of gadgets: variable gadgets and clause gadgets. Variable and clause gadgets will be connected according to their arrangement within the given

formula. Thus, for any given formula ϕ, we will construct a graph G_ϕ that is (P_3, H)-good if and only if ϕ is satisfiable.

We use two variants of SAT to reduce from, which are discussed in Section 5.1. The reduction is explained in Section 5.2. Finally, the gadgets for each H are discussed in Section 5.4.

5.1 Variants of SAT

We use the following variants of SAT for our reductions:

Problem 5.1 ($(3,1)$-3SAT [10])**.** *Let ϕ be a 3CNF formula where 1) each clause has exactly three distinct variables, and 2) each variable appears as an unnegated literal three times and as a negated literal once. Does there exist a satisfying assignment for ϕ?*

Problem 5.2 ($(2,2)$-3SAT [1])**.** *Let ϕ be a 3CNF formula where 1) each clause has exactly three distinct variables, and 2) each variable appears as an unnegated literal twice and as a negated literal twice. Does there exist a satisfying assignment for ϕ?*

Note that the important properties of these problems are the limit on the number of occurrences of each variable and that clauses have exactly three distinct variables. The benefit of using these two problems is that:

1. Each variable appears exactly four times. This allows us to construct variable gadgets with only four output vertices instead of constructing one that must be modified according to the number of occurrences.
2. By knowing exactly how many times a variable is negated, we avoid having to create negation gadgets and instead have negated output vertices within the variable gadgets themselves.
3. Each clause having exactly three literals allows us to construct only one kind of clause gadget.
4. Having distinct variables in each clause eliminates the creation of unwanted subgraphs when joining gadgets. The utility of this fact will be clearer when we explore the cases where H is a cycle.

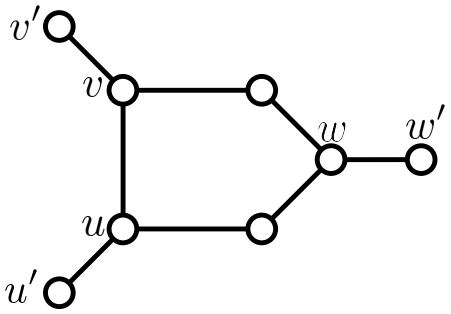


Figure 5.2: Clause gadget example.

5.2 Reduction Idea

We make use of the following simple observation, which shows that red edges are especially restrictive in (P_3, H)-good colorings:

Observation 5.1. *In any (P_3, H)-good coloring of a graph G, if an edge e is colored red, then all edges adjacent to e must be blue.*

This is clearly true because two adjacent red edges would form a red P_3. Based on this observation, we construct gadgets for variables and clauses to simulate a SAT formula. We explain the properties of these gadgets below, after which we discuss how these gadgets are connected to simulate a given boolean formula.

5.2.1 Gadgets

Clause Gadget. To better explain the functionality of the clause gadget, we focus our attention on a specific H. Consider the (P_3, C_5)-Non-Arrowing problem. Let u, v, and w be three vertices of a C_5 such that u and w are adjacent, and v is not adjacent to either u or w. Let u', v', and w' be three vertices not in the C_5, such that u', v', and w' are connected only to u, v, and w respectively. Note that if the three edges (u', u), (v', v), and (w', w) are colored red, then, by Observation 5.1, the C_5 must be colored blue, resulting in an invalid coloring. It follows that at least one edge in $\{(u', u), (v', v), (w', w)\}$ must be blue in any (P_3, C_5)-good coloring of the constructed graph. Further analysis shows that all seven combinations of colorings with at least one blue edge of

the set $\{(u', u), (v', v), (w', w)\}$ can be extended to a (P_3, C_5)-good coloring. This is illustrated in Figure 5.2. Note that this is similar to how a CNF-SAT clause works: At least one true literal (blue edge) is required to satisfy the clause (allow a good coloring).

Thus, for (P_3, C_5)-Non-Arrowing, one can use a C_5 as a clause gadget, where the vertices u, v, and w are "input vertices" from variable gadgets. For general H, we define the clause gadget as a graph with three vertices, u, v, and w, referred to as "input vertices." Let u', v', and w' be vertices not in the gadget such that u', v', and w' are connected to u, v, and w, respectively. Observe that there are eight possible combinations for the colorings of $\{(u, u'), (v, v'), (w, w')\}$. For each combination, except where all three edges are colored red, there must exist a (P_3, H)-good coloring of the clause gadget, and no good coloring should exist when all edges are colored red.

Variable Gadget. We first define red and blue vertices. Let G be a (P_3, H)-good graph. In a (P_3, H)-good coloring of G, a red vertex is a vertex that is adjacent to a red edge, and a blue vertex is a vertex that is adjacent to no red edges. Note that according to our clause gadget, a red vertex acts as a false input. Consider the (P_3, C_5)-Non-Arrowing example again. If the three vertices u, v, and w were red vertices, where the red edge lies outside the C_5, the gadget does not have a good coloring, i.e., the clause is not satisfied.

Recall that the SAT variants we are reducing from constrain the number of occurrences of each variable. Thus, a variable gadget will have four "output vertices," one for each occurrence of the corresponding variable. Each gadget will have unnegated output vertices and negated output vertices. These vertices will have the following properties:

1. In each coloring, if a single unnegated (resp., negated) output vertex is a red vertex, then all unnegated (resp., negated) output vertices must be red vertices.
2. In each coloring, if a single unnegated (resp., negated) output vertex is a blue vertex, then all unnegated (resp., negated) output vertices must be blue vertices.
3. In each coloring, if the unnegated output vertices are red (resp., blue), then the negated output vertices must be blue (resp., red).
4. For each output vertex, there must exist at least one coloring where it is a red vertex and at least one coloring where it is a blue vertex.

The number of these vertices will depend on the SAT variant we are reducing from. If we are reducing from $(3, 1)$-3SAT, then we will have three unnegated output vertices and one negated

Algorithm 3: (P_3, H) Reduction

Input : A formula ϕ with variables $X = \{x_1, x_2, \ldots, x_n\}$ and clauses $Y = \{Y_1, Y_2, \ldots, Y_m\}$

Output: A graph G s.t. G is (P_3, H)-good iff ϕ is satisfiable

```
Let G be the empty graph
for Y_i ∈ Y do
    Add a copy of the clause gadget to G
end
for x_i ∈ X do
    Add a copy of the variable gadget to G
    for each clause Y_j that x_i belongs to do
        if x_i appears as a negated literal in Y_j then
            Join a free negated output vertex of x_i's variable gadget to a free input vertex of
             Y_j's clause gadget
        else
            Join a free unnegated output vertex of x_i's variable gadget to a free input vertex of
             Y_j's clause gadget
        end
    end
end
return G
```

output vertex. If we are reducing from $(2, 2)$-3SAT, then we will have two unnegated output vertices and two negated output vertices.

5.2.2 Joining Gadgets

For each H, Algorithm 3 is used to reduce an instance of one of the SAT variants to (P_3, H)-Non-Arrowing. To explain the algorithm, we first define a "join" operation on the vertices of a graph. When two vertices $u, w \in V(G)$ in a graph are joined, a new vertex v is added such that v is connected to all neighbors of u and w, and the vertices u and w are removed. Note that some texts refer to this operation as a vertex contraction or vertex identification operation.[1] Let a "free vertex" be a vertex that has not been created via the join operation.

[1] `https://mathworld.wolfram.com/VertexContraction.html`

Note that it is not enough for each clause gadget to have at least one blue vertex as input to obtain a good coloring; we must also show that no blue H is formed when the coloring of the variables corresponds to a satisfying assignment. In our reductions below, we show that this is the case for each $H \in \mathcal{Z}$.

5.3 Finding Gadgets

For most $H \in \mathcal{Z} = \{P_5, P_6, C_4, C_5, C_6, J_4, BT\}$, the graphs themselves act as clause gadgets: Each $H \in \{P_5, P_6, C_5, C_6, J_4, BT\}$ has three vertices that can act as input vertices. This is illustrated in each gadget's respective section. For $H = C_4$, it is easy to see that the combination of two C_4's is enough to make a valid clause gadget.

For a fixed H, the variable gadgets are found by enumerating over (P_3, H)-good graphs and finding graphs that meet the criteria described for said gadgets. Assume we are given a graph G and four vertices v_1, v_2, v_3, and v_4. For each (P_3, H)-good coloring of G, we create a 4-tuple: (k_1, k_2, k_3, k_4), where $k_i \in \{R, B\}$ corresponds to vertex v_i, such that R and B are labels corresponding to whether vertex v_i is a red vertex or a blue vertex, respectively. All colorings of G are enumerated, and each 4-tuple is stored in a set, L_G. It is clear that if G and the vertices $v_1, \ldots, v_4$ form a valid variable gadget for $(3, 1)$-3SAT (resp., $(2, 2)$-3SAT), L_G must lie in the set $A_{3,1}$ (resp., $A_{2,2}$), defined below:

$$A_{3,1} = \begin{array}{l} \big\{\{(R,R,R,B),(B,B,B,R)\}, \\ \{(R,R,B,R),(B,B,R,B)\}, \\ \{(R,B,R,R),(B,R,B,B)\}, \\ \{(B,R,R,R),(R,B,B,B)\}\big\} \end{array}$$

$$A_{2,2} = \begin{array}{l} \big\{\{(R,R,B,B),(B,B,R,R)\}, \\ \{(R,B,R,B),(B,R,B,R)\}, \\ \{(R,B,B,R),(B,R,R,B)\}\big\} \end{array}$$

The (P_3, H)-good graphs are generated as described in Chapter 4, where the number of vertices is in the range $[3, 10]$. The procedure described above is performed for all combinations of four vertices in all generated graphs. If a graph matching the criteria for either variable gadget is found, we store it. In the end, we are left with a number of graphs that may or may not be suitable gadgets. Since there are very few graphs that meet our criteria for each H, each is analyzed, and the best one is selected. If a viable gadget is not found, we use these generated graphs to construct

a new one.

5.4 Gadgets for each $H \in \mathcal{Z}$

In this section, we prove the following theorem:

Theorem 5.1. *(P_3, H)-Arrowing is* coNP*-complete for $H \in \{C_4, C_5, C_6, J_4, BT, P_5, P_6\}$.*

For each H, we found gadgets as discussed in the previous section. These gadgets are shown in this section. We also show the following:

1. Each coloring of incoming edges for a clause gadget corresponding to a satisfying assignment is valid, i.e., we show that good colorings exist for each incoming color combination to the input vertices.
2. For variable gadgets, each possible coloring is shown. The unnegated and negated output vertices are labeled.
3. No blue H can be formed while joining the gadgets when: (1) the variable gadgets adhere to their good colorings, and (2) the variable gadgets are colored corresponding to a satisfying assignment of the formula ϕ. Essentially, we show that no "unwanted" blue H's are formed while joining the gadgets. This requires special attention for each H, and is discussed separately for all cases. Cases for which these proofs are similar are discussed together.

5.4.1 Cycles: C_4, C_5, and C_6

Gadgets for arrowing (P_3, C_4), (P_3, C_5), and (P_3, C_6), are presented in Figures 5.4, 5.5, and 5.6, respectively. The reductions for (P_3, C_4) and (P_3, C_5)-Non-Arrowing use $(3, 1)$-3SAT, while the reduction for (P_3, C_6)-Non-Arrowing uses $(2, 2)$-3SAT. For each $k \in \{4, 5, 6\}$, to show that no unwanted blue C_k's are formed, we show that joining the gadgets does not created any new C_k's. The same argument works for all three cases. We first discuss the proof for C_4, then show the same applies to C_5 and C_6.

Recall that each clause in the formula ϕ must have three distinct variables. Thus, a new cycle is created only when two variables appear together in two clauses.

Note that in the variable gadget for (P_3, C_4) (Figure 5.4), the shortest path between any two output vertices has two edges. Similarly, in the clause gadget, the shortest path between any two input vertices has at least one edge. Thus, when two variables appear in two clauses, as discussed previously, the cycle created must have at least $2 + 1 + 2 + 1 = 6$ edges, making a C_6, i.e., no new C_4 is formed when joining the gadgets.

A similar argument can be made for C_5 and C_6. The (P_3, C_5) variable and clause gadgets also have two and one edges at minimum, respectively, separating their output/input vertices. Thus, the shortest new cycle formed is a C_6. For both of (P_3, C_6)'s gadgets, we note that there are at minimum two edges between the output/input vertices too. Thus, the shortest possible cycle made is of order $2 + 2 + 2 + 2 = 8$.

5.4.2 J_4 and BT

$(3, 1)$-3SAT was reduced to (P_3, J_4)-Non-Arrowing via the gadgets shown in Figure 5.7. $(2, 2)$-3SAT was reduced to (P_3, BT)-Non-Arrowing via the gadgets shown in Figure 5.8.

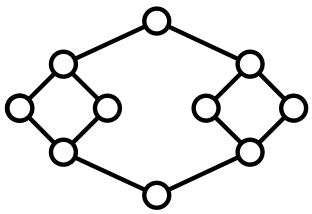
When constructing the graph for (P_3, J_4)-Non-Arrowing's reduction, it is clear to see that joining vertices can not form new J_4's. Thus, unwanted blue J_4's are not formed. For (P_3, BT)-Non-Arrowing, note that BT's are formed when joining the gadgets. However, the output vertices of the variable gadgets never belong to blue K_3's. Thus, unwanted BT's can not be formed.

5.4.3 Paths: P_5 and P_6

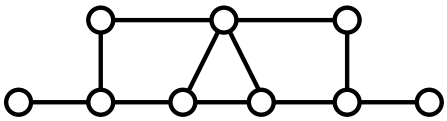
$(2, 2)$-3SAT was reduced to (P_3, P_5) and (P_3, P_6)-Non-Arrowing. The gadgets for these are shown in Figures 5.9 and 5.10, respectively.

Note that to ensure no unwanted P_5's or P_6's are formed when joining the gadgets, it is necessary to ensure that the input edges to the clause gadget are not parts of larger monochromatic paths. For instance, consider (P_3, P_5)-Arrowing and refer to the clause gadget in Figure 5.9. If the blue input edges were already a part of blue P_3's, then we could not have a valid coloring in the case where three blue edges are input to the clause gadget. Thus, we had to ensure that the output vertices of our variable gadget end at blue/red P_2's.

Since no such graphs were found up to 10 vertices, the graphs found using our gadget search were modified to obtain the graphs presented. The original graphs found are shown in Figure 5.3.



(a) The original gadget for (P_3, P_5).



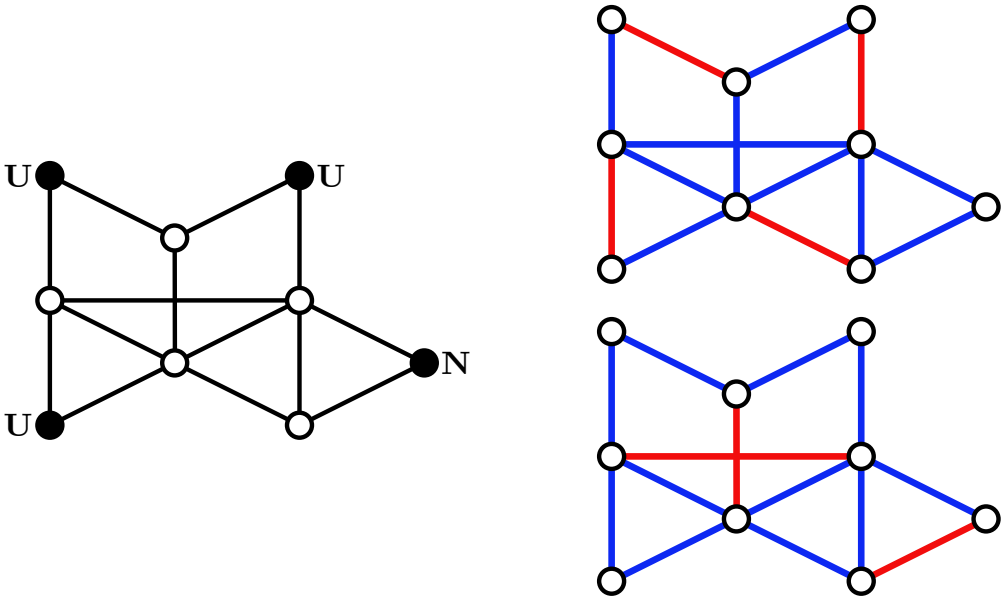
(b) The original gadget for (P_3, P_6).

Figure 5.3: The original gadgets found via enumeration for (P_3, P_5) and (P_3, P_6)-Non-Arrowing, when reducing from $(2, 2)$-3SAT. These were modified by hand to obtain the gadgets necessary for the final reductions.

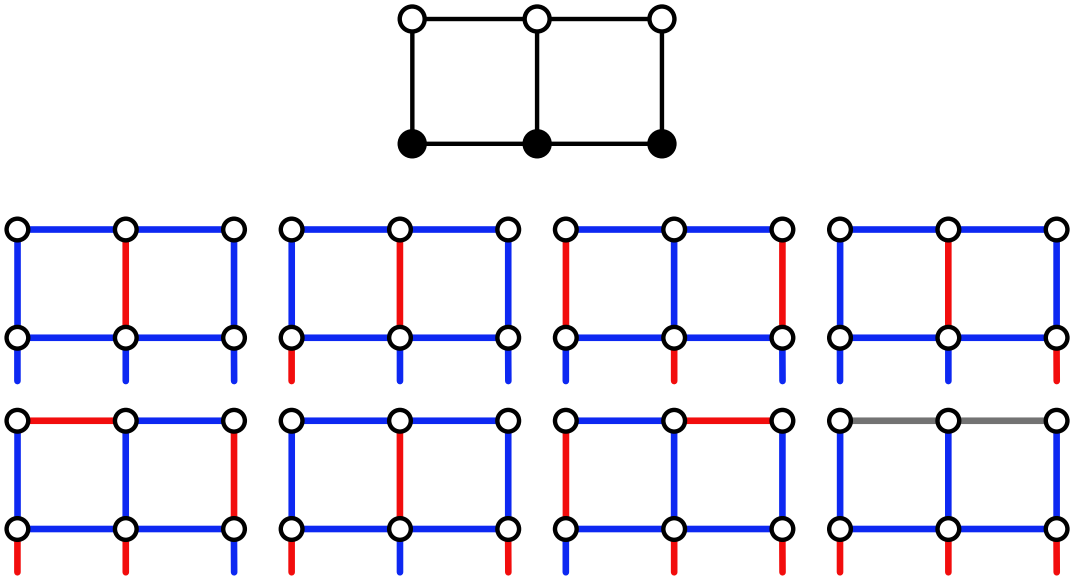
5.5 Remarks

While the problems discussed in this chapter are coNP-complete, we note that there do exist P_3 arrowing problems that are in P. (P_3, P_3)-Arrowing is in P, as there is a simple greedy algorithm one can employ to check if a good coloring exists. (P_3, P_4)-Arrowing and (P_3, K_3)-Arrowing are also solvable in polynomial time [20]. (P_3, P_4)-Arrowing was proven to be in P via a reduction to 2SAT, while (P_3, K_3)-Arrowing was shown to be in P via a reduction to maximum weight matching.

We also want to point out that the framework discussed in the chapter is not the only way one can construct gadgets for arrowing problems. Our initial reductions for (P_3, P_5) and (P_3, P_6)-Arrowing were from a variant of 3SAT that restricted each variable to appear only four times in the formula, but did not restrict the number of times it is negated [38]. Keeping this in mind, we also constructed NOT gadgets, which have an input vertex and an output vertex. The property of the NOT gadget was that the output vertex is a red vertex (resp., blue vertex) if the input vertex is a blue vertex (resp., red vertex). Moreover, the variable gadgets had only unnegated output vertices, as the negation was handled via NOT gadgets. While this allowed us to have easier gadgets for (P_3, P_5) and (P_3, P_6)-Arrowing, this methodology was not fruitful for (P_3, H)-Arrowing where H is not a path graph.

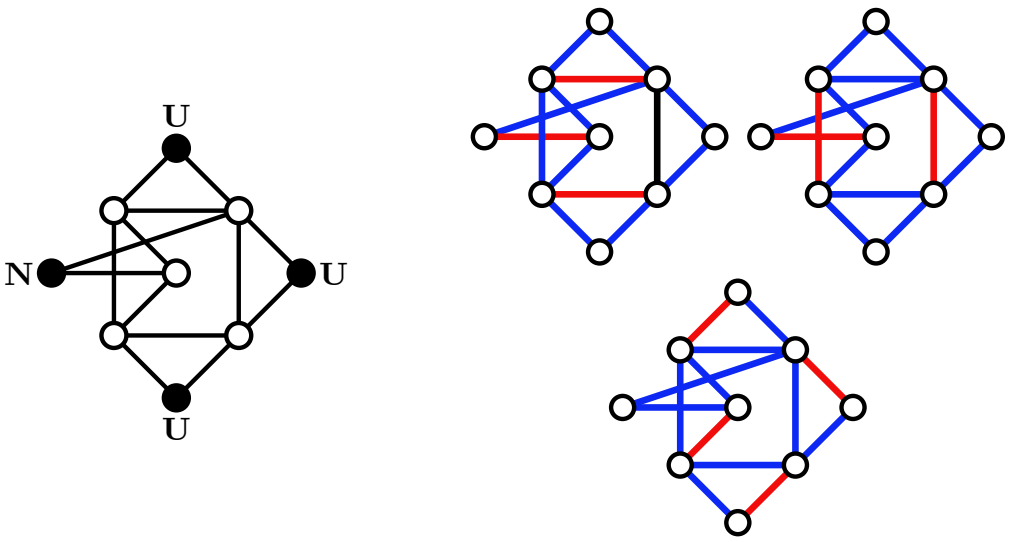


(a) The variable gadget is shown on the left. Output vertices are filled in. Unnegated output vertices are marked **U**, and negated output vertices are marked **N**. All good colorings of the gadget are shown on the right.

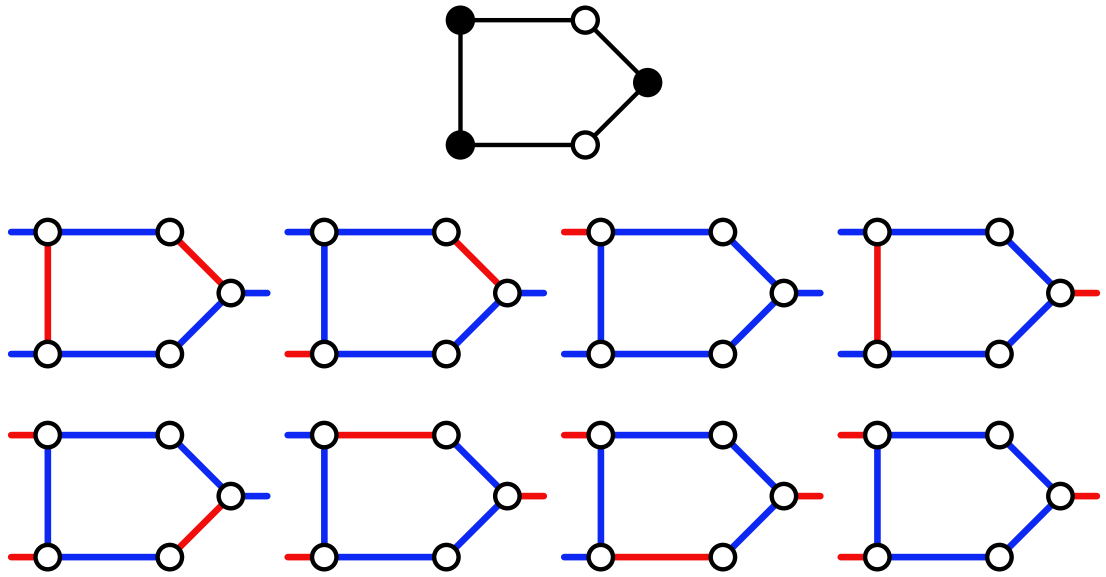


(b) The clause gadget is shown on the top. Input vertices are filled in. One coloring for each possible combination of input edge colorings is shown; note that there exists a good coloring for all combinations except the one where there are three red input edges. Note that any coloring of the gray edges with at least one blue edge will lead to a blue C_4.

Figure 5.4: Gadgets to reduce $(3, 1)$-3SAT to (P_3, C_4)-Non-Arrowing.

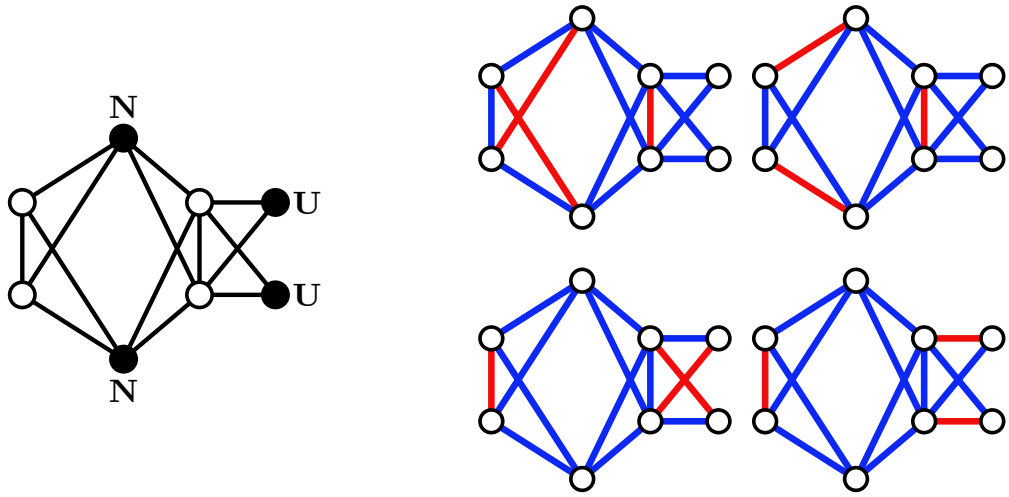


(a) The variable gadget is shown on the left. Output vertices are filled in. Unnegated output vertices are marked **U**, and negated output vertices are marked **N**. All good colorings of the gadget are shown on the right.

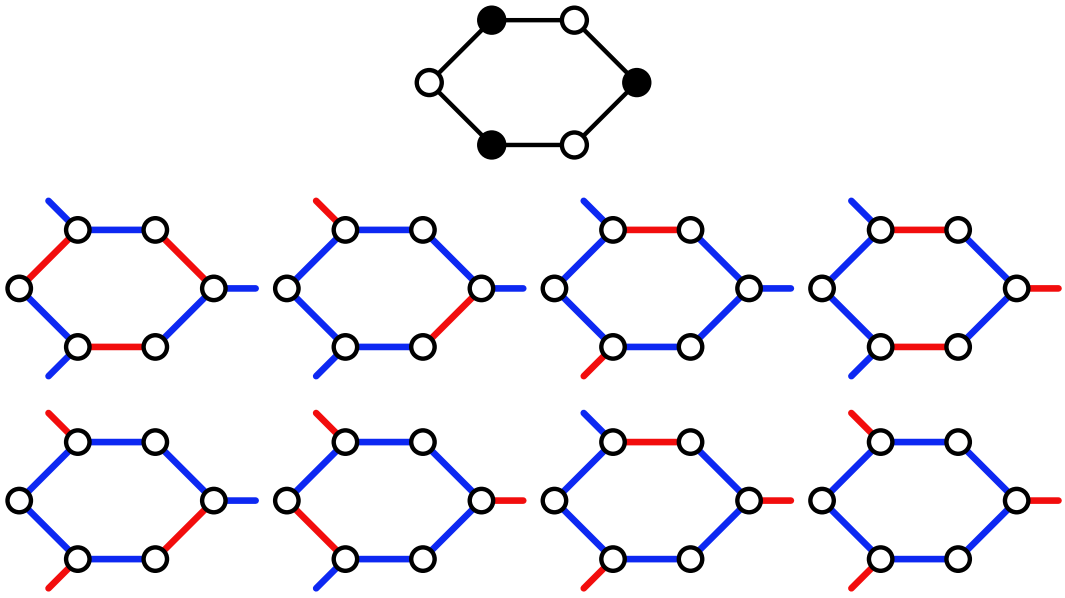


(b) The clause gadget is shown on the top. Input vertices are filled in. One coloring for each possible combination of input edge colorings is shown; note that there exists a good coloring for all combinations except the one where there are three red input edges.

Figure 5.5: Gadgets to reduce $(3, 1)$-3SAT to (P_3, C_5)-Non-Arrowing.

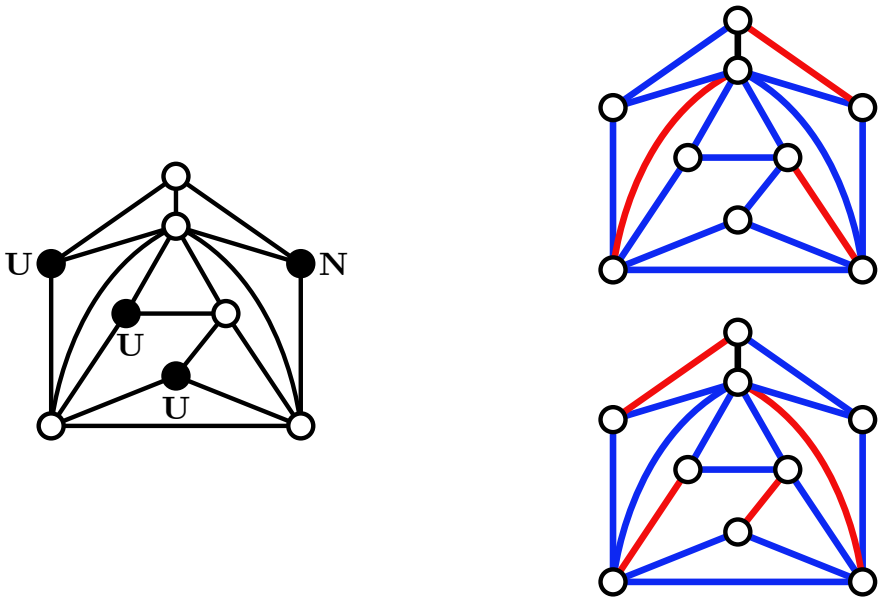


(a) The variable gadget is shown on the left. Output vertices are filled in. Unnegated output vertices are marked **U**, and negated output vertices are marked **N**. All good colorings of the gadget are shown on the right.

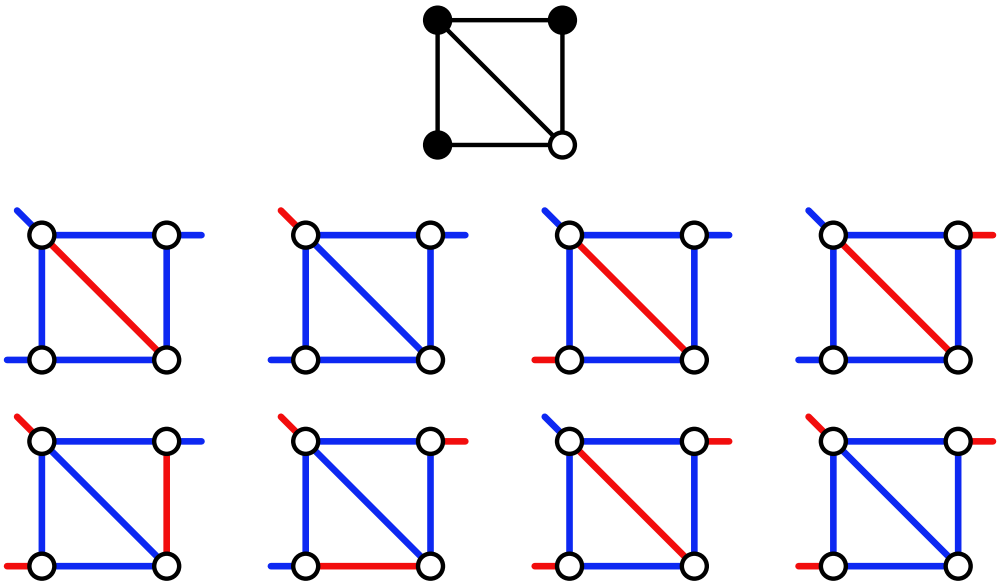


(b) The clause gadget is shown on the top. Input vertices are filled in. One coloring for each possible combination of input edge colorings is shown; note that there exists a good coloring for all combinations except the one where there are three red input edges.

Figure 5.6: Gadgets to reduce $(2, 2)$-3SAT to (P_3, C_6)-Non-Arrowing.

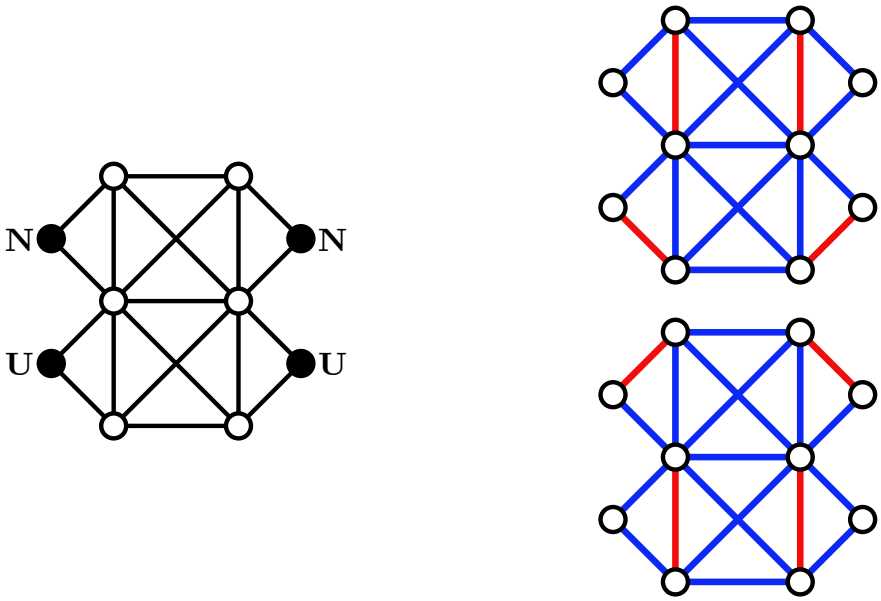


(a) The variable gadget is shown on the left. Output vertices are filled in. Unnegated output vertices are marked **U**, and negated output vertices are marked **N**. All good colorings of the gadget are shown on the right.

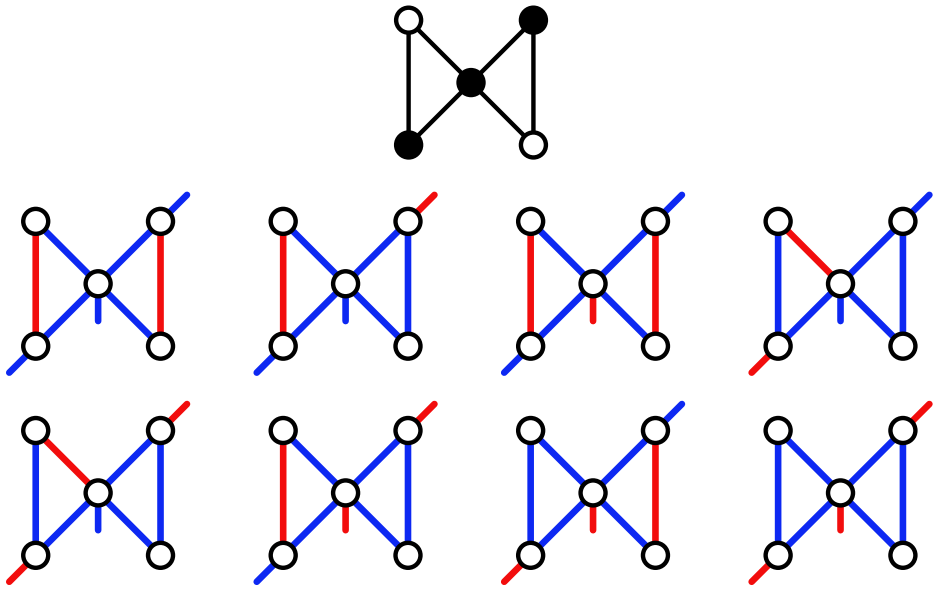


(b) The clause gadget is shown on the top. Input vertices are filled in. One coloring for each possible combination of input edge colorings is shown; note that there exists a good coloring for all combinations except the one where there are three red input edges.

Figure 5.7: Gadgets to reduce $(3, 1)$-3SAT to (P_3, J_4)-Non-Arrowing.

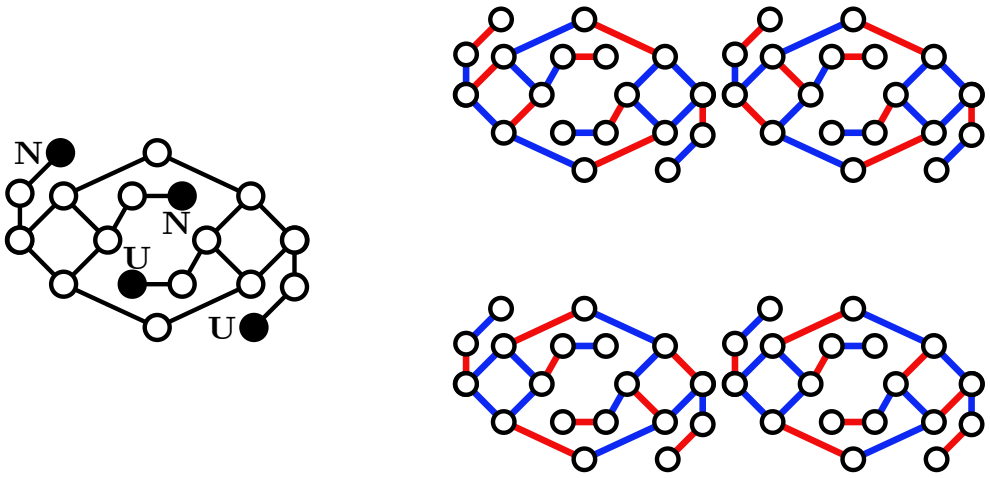


(a) The variable gadget is shown on the left. Output vertices are filled in. Unnegated output vertices are marked **U**, and negated output vertices are marked **N**. All good colorings of the gadget are shown on the right.

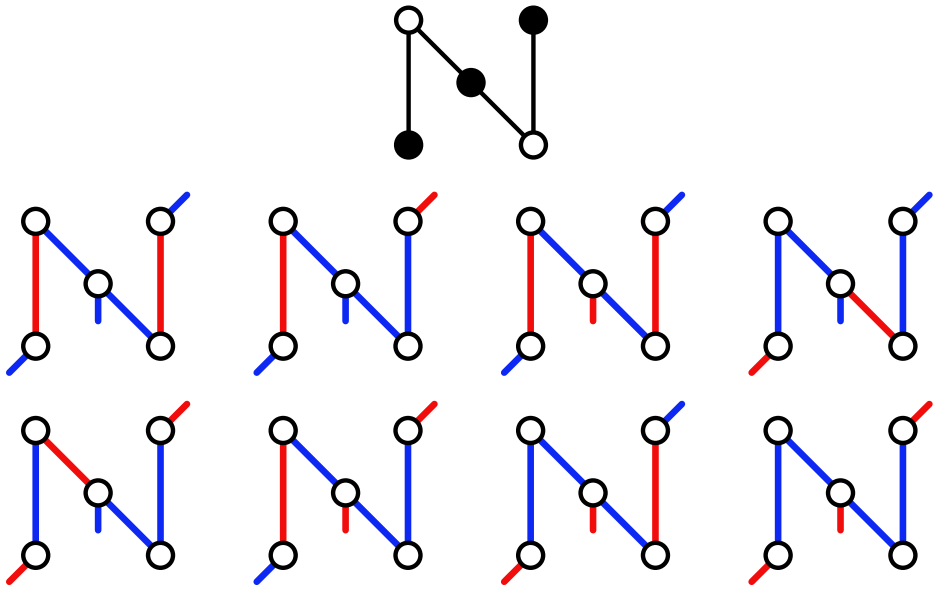


(b) The clause gadget is shown on the top. Input vertices are filled in. One coloring for each possible combination of input edge colorings is shown; note that there exists a good coloring for all combinations except the one where there are three red input edges.

Figure 5.8: Gadgets to reduce $(2, 2)$-3SAT to (P_3, BT)-Non-Arrowing.

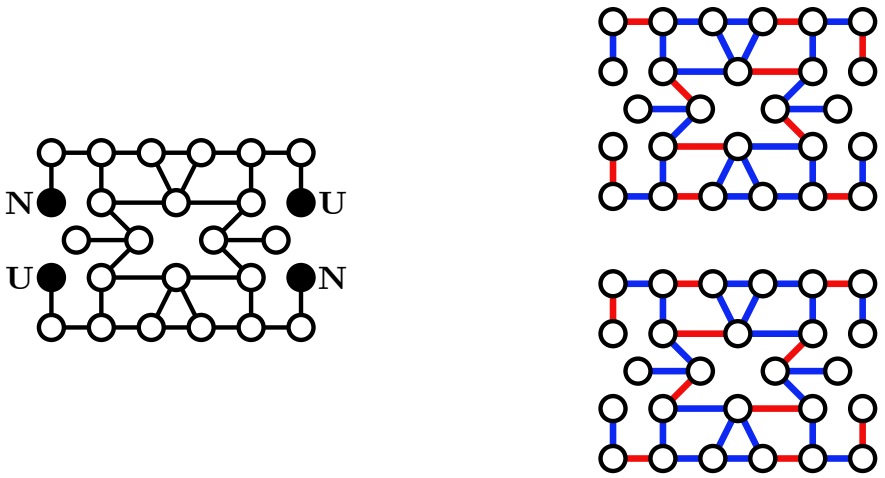


(a) The variable gadget is shown on the left. Output vertices are filled in. Unnegated output vertices are marked **U**, and negated output vertices are marked **N**. All good colorings of the gadget are shown on the right.

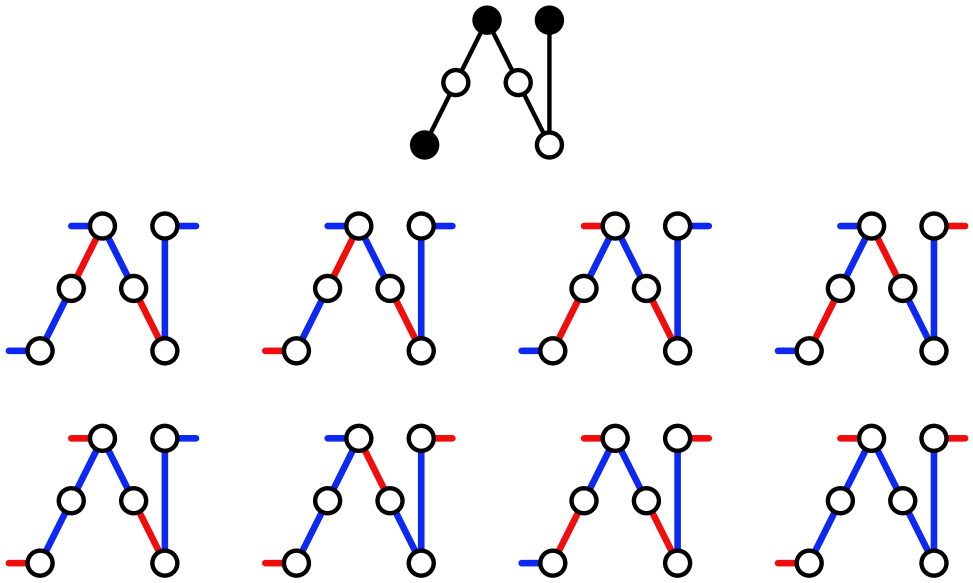


(b) The clause gadget is shown on the top. Input vertices are filled in. One coloring for each possible combination of input edge colorings is shown; note that there exists a good coloring for all combinations except the one where there are three red input edges.

Figure 5.9: Gadgets to reduce $(2, 2)$-3SAT to (P_3, P_5)-Non-Arrowing.



(a) The variable gadget is shown on the left. Output vertices are filled in. Unnegated output vertices are marked **U**, and negated output vertices are marked **N**. All good colorings of the gadget are shown on the right.



(b) The clause gadget is shown on the top. Input vertices are filled in. One coloring for each possible combination of input edge colorings is shown; note that there exists a good coloring for all combinations except the one where there are three red input edges.

Figure 5.10: Gadgets to reduce $(2, 2)$-3SAT to (P_3, P_6)-Non-Arrowing.

Chapter 6

Some Small Ramsey & Folkman Numbers

Let H_1, H_2, and I be graphs. The Ramsey number $R(H_1, H_2)$ is defined as the smallest n such that $K_n \to (H_1, H_2)^e$. The Folkman number $F_e(H_1, H_2; I)$ is defined as the order of the smallest I-free graph G such that $G \to (H_1, H_2)^e$.

In this chapter, we compute Ramsey numbers of the form $R(P_3, H)$, where $H \in \mathcal{Y}$ and $\mathcal{Y} = \{K_3, P_4, P_5, P_6, C_4, C_5, C_6, J_4, BT\}$. Moreover, we present some Folkman numbers of the form $F_e(P_3, H; I)$, where $H \in \mathcal{Y}$, and I is a cycle, a complete graph, or a complete graph missing an edge. Mainly, we present all such numbers that lie within the range $[3, 10]$.

Note that all Ramsey numbers presented in this chapter, except $R(P_3, BT)$, have been discovered before this work and were recomputed in our work for completeness. For instance, in 1972, Chvátal and Harary determined the numbers $R(P_3, H)$, where H is an "isolate-free" graph on at most four vertices; this includes the graphs P_4, C_4, and J_4 [6]. Ramsey numbers of the form $R(P_n, P_m)$ were classified in 1967 by Gerencsér and Gyárfás [15]. Numbers of the form $R(P_n, C_m)$ were classified in 1974 by Faudree et al. [12].

6.1 Ramsey Numbers

To find Ramsey numbers $R(P_3, H)$, each K_n, where $n \in [3, 10]$, was checked using a graph colorer (Section 4.1.3) iteratively. We stopped the process once the smallest n was found. The numbers are shown in Table 6.1.

H	K_3	P_4	P_5	P_6	C_4	C_5	C_6	J_4	BT
$R(P_3, H)$	5	4	5	6	4	5	6	5	5

Table 6.1: Ramsey numbers $R(P_3, H)$ for $H \in \mathcal{Y}$.

We summarize our results in the following theorem:

Theorem 6.1. *$R(P_3, P_4) = R(P_3, C_4) = 4$, $R(P_3, K_3) = R(P_3, P_5) = R(P_3, C_5) = R(P_3, J_4) = R(P_3, BT) = 5$, and $R(P_3, P_6) = R(P_3, C_6) = 6$.*

6.2 Folkman Numbers

For each $H \in \mathcal{Y}$, we look for $F_e(P_3, H; I)$ where I is a C_k, K_k, or J_k. Note that certain I's are ignored for some H. To explain why, we first note the following observation:

Observation 6.1. *If $|V(I)| > R(H_1, H_2)$, then $F_e(H_1, H_2; I) = R(H_1, H_2)$.*

This is true because $n = R(H_1, H_2)$ is the order of the smallest graph G that arrows $(H_1, H_2)^e$, and K_n must be I-free since there are not enough vertices in K_n to include an I. Thus, we ignore I's in the following two cases:

1. The number is clearly equal to the Ramsey number by Observation 6.1, e.g., $F_e(P_3, J_4; C_6)$ is ignored because $R(P_3, J_4) = 5$. Thus, $F_e(P_3, J_4; C_6) = 5$ since K_5 is clearly C_6-free.
2. The number clearly cannot exist, e.g., $F_e(P_3, J_4; C_4)$ cannot exist because any C_4-free graph must be J_4-free and thereby has the trivial coloring where all edges are colored blue.

To show that a Folkman number is equal to n, it is sufficient to show that there exists no graph with the desired properties on $n - 1$ vertices and show at least one graph with the selected properties

on n vertices. Said graph on n vertices is typically referred to as a "witness graph." For example, Figure 6.1 is a witness graph for $F_e(P_3, P_4; K_4)$ since it contains no K_4's and arrows $(P_3, P_4)^e$.

We take two approaches to obtain the Folkman numbers presented in this work. Most of the numbers were found using the approach in Section 6.2.1. Both approaches are discussed below. For each H, the Folkman numbers found are shown in Table 6.2. For all Folkman numbers, witness graphs have been illustrated in Figures 6.1–6.21. Table 6.3 shows the label of the corresponding witness graph for each Folkman number discovered. Our results are summarized in the following theorem:

Theorem 6.2. *Let* $\mathcal{Y} = \{K_3, P_4, P_5, P_6, C_4, C_5, C_6, J_4, BT\}$ *and* $\mathcal{I} = \{K_6, K_5, K_4, K_3, J_6, J_5, J_4, C_6, C_5, C_4\}$. *For* $H \in \mathcal{Y} \setminus \{BT\}$ *and* $I \in \mathcal{I}$, $F_e(P_3, H; I) \leq 13$, *or does not exist. For* $I \in \mathcal{I} \setminus \{J_4, C_4\}$, $F_e(P_3, BT; I) \leq 10$, *or does not exist. The exact values for each number can be found in Table 6.2.*

6.2.1 Approach 1: Targeting H

For our first approach, we used the graphs generated using the methodology discussed in Chapter 4. Let $\mathfrak{G}$ be the set of all graphs that have between three and ten vertices. Note that this set can be obtained using `nauty`'s `geng` function. For a fixed H, let $\mathcal{G}_H$ be the set of all (P_3, H)-good graphs in $\mathfrak{G}$. Observe that we can find the set of graphs in $\mathfrak{G}$ that arrow $(P_3, H)^e$, denoted as $\overline{\mathcal{G}_H}$, by computing $\mathfrak{G} \setminus \mathcal{G}_H$. To find the Folkman number $F_e(P_3, H; I)$, we selected all I-free graphs from $\overline{\mathcal{G}_H}$ and obtained the smallest such graph.

6.2.2 Approach 2: Targeting I

In our second approach, we iteratively generated all I-free graphs. We can do this similarly to the process in Chapter 4: Given the set of I-free graphs on n vertices, use the graph extender to obtain graphs on $n+1$ vertices and remove any graphs that are not I-free. This approach works since for any I-free G, the subgraph must be I-free as well.

To find the Folkman number $F_e(P_3, H; I)$, we ran the graph colorer on all I-free graphs generated using the approach above. Since we found all numbers bounded by ten using the first approach, this approach provided all Folkman numbers reported beyond ten.

Note that the only I for which this approach was necessary were C_4 and J_4. I-free graphs up to 13 and 11 vertices, respectively, were generated in our attempt to find our desired Folkman numbers.

6.2.3 Open Cases

Note that the only missing numbers are $F_e(P_3, BT; J_4)$ and $F_e(P_3, BT; C_4)$. Thus, we pose the following question:

Open Problem 6.1. *Do the Folkman numbers $F_e(P_3, BT; J_4)$ and $F_e(P_3, BT; C_4)$ exist? If yes, what are their values?*

Observe that we pose the question of "existence" in our open problem. This is because Folkman numbers, unlike Ramsey numbers, have not been proven to exist for all possible cases. An example we saw earlier was $F_e(P_3, J_4; C_4)$, whose nonexistence is trivial to prove. However, some cases require more careful reasoning. For example, $F_e(K_3, K_3; B_3)$, whose nonexistence was proven by Xu et al. [40].[1]

Note that adding an edge to the BT graph forms a J_4/C_4, significantly limiting the number of ways two BT's can interact within a graph. This suggests that perhaps all J_4-free graphs have a (P_3, BT)-good coloring. If there do exist J_4/C_4-free graphs that arrow (P_3, BT), then we know via our computation that $F_e(P_3, BT; J_4) > 11$, and $F_e(P_3, BT; C_4) > 13$.

[1] B_3 is the graph on five vertices and seven edges, where one edge belongs to exactly three triangles.

	I									
H	K_6	K_5	K_4	K_3	J_6	J_5	J_4	C_6	C_5	C_4
K_3	R	5	5	NE	R	5	9	R	7	13
P_4	R	R	4	5	R	R	5	R	R	6
P_5	R	5	5	5	R	5	5	R	5	7
P_6	6	6	6	6	6	6	6	7	6	8
C_4	R	R	6	6	R	R	6	R	R	NE
C_5	R	5	6	9	R	6	9	R	NE	13
C_6	6	6	6	6	6	6	6	NE	6	12
J_4	R	5	7	NE	R	7	NE	R	7	NE
BT	R	6	7	NE	R	7	?	R	10	?

Table 6.2: Folkman numbers $F_e(P_3, H; I)$ for $H \in \mathcal{Y}$. Cells marked are R if the number is clearly equal to the corresponding Ramsey number. Cells marked are NE if the number clearly cannot exist. Cells marked ? were not found during our enumeration.

	I									
H	K_6	K_5	K_4	K_3	J_6	J_5	J_4	C_6	C_5	C_4
K_3		6.7	6.6			6.6	6.14		6.9	6.16
P_4			6.1	6.2			6.2			6.3
P_5		6.2	6.2	6.2		6.2	6.2		6.2	6.4
P_6	6.10	6.10	6.10	6.10	6.10	6.10	6.10	6.11	6.10	6.13
C_4			6.5	6.5			6.5			
C_5		6.7	6.12	6.15		6.12	6.15			6.19
C_6	6.10	6.10	6.10	6.10	6.10	6.10	6.10		6.10	6.21
J_4		6.7	6.8			6.8			6.9	
BT		6.20	6.17			6.17			6.18	

Table 6.3: Witness graphs for $F_e(P_3, H; I)$ for $H \in \mathcal{Y}$. Each cell refers to the label of a figure.

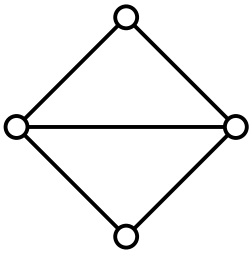


Figure 6.1: Witness graph for $F_e(P_3, P_4; K_4)$.

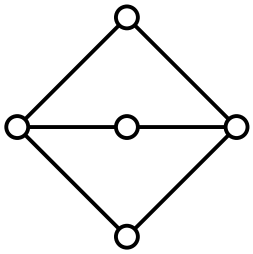


Figure 6.2: Witness graph for $F_e(P_3, P_5; I)$ for $I \in \{K_5, K_4, K_3, J_5, J_4, C_5\}$ and $F_e(P_3, P_4; I)$ for $I \in \{K_3, J_4\}$.

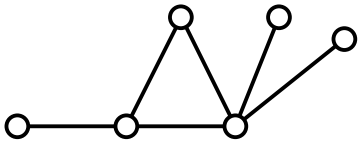


Figure 6.3: Witness graph for $F_e(P_3, P_4; C_4)$.

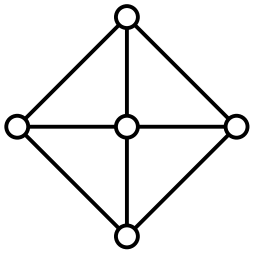


Figure 6.4: Witness graph for $F_e(P_3, K_3; I)$ for $I \in \{K_4, J_5\}$.

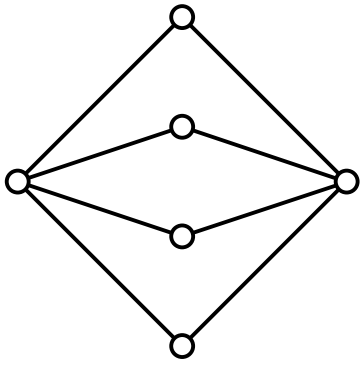


Figure 6.5: Witness graph for $F_e(P_3, C_4; I)$ for $I \in \{K_4, K_3, J_4\}$.

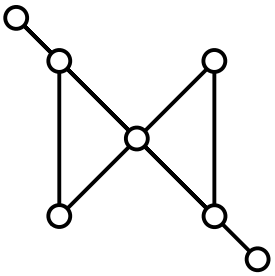


Figure 6.6: Witness graph for $F_e(P_3, P_5; C_4)$.

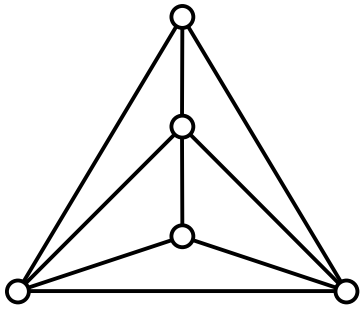


Figure 6.7: Witness graph for $F_e(P_3, H; K_5)$ for $H \in \{K_3, J_4, C_5\}$.

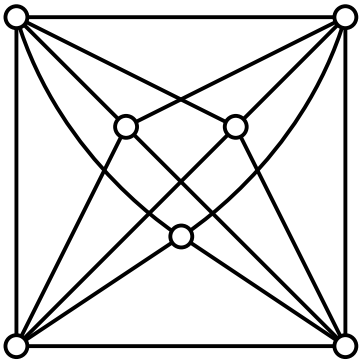


Figure 6.8: Witness graph for $F_e(P_3, J_4; I)$ for $I \in \{K_4, J_5\}$.

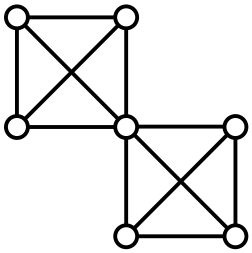


Figure 6.9: Witness graph for $F_e(P_3, K_3; C_5)$ and $F_e(P_3, J_4; I)$ for $I \in \{J_5, C_5\}$.

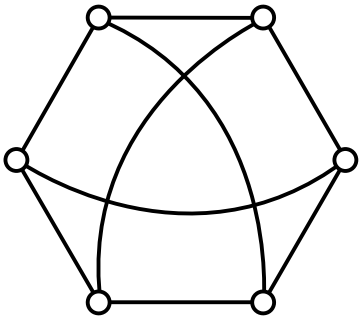


Figure 6.10: Witness graph for $F_e(P_3, H; I)$ for $I \in \{K_6, K_5, K_4, K_3, J_6, J_5, J_4, C_5\}$ and $H \in \{P_6, C_6\}$.

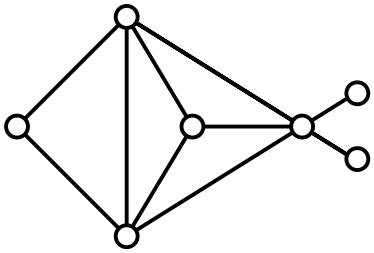


Figure 6.11: Witness graph for $F_e(P_3, P_6; C_6)$.

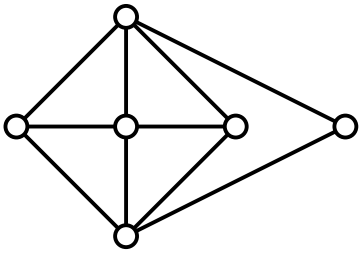


Figure 6.12: Witness graph for $F_e(P_3, C_5; I)$ for $I \in \{K_4, J_5\}$.

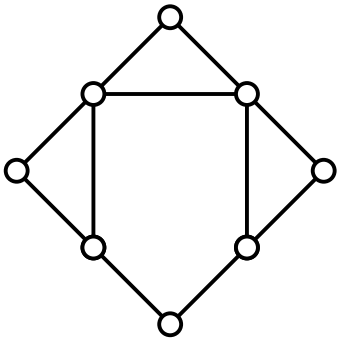


Figure 6.13: Witness graph for $F_e(P_3, P_6; C_4)$.

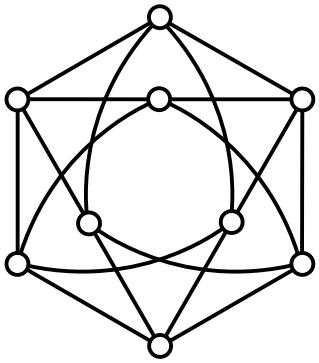


Figure 6.14: Witness graph for $F_e(P_3, K_3; J_4)$.

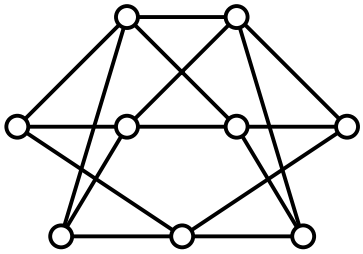


Figure 6.15: Witness graph for $F_e(P_3, C_5; I)$ for $I \in \{K_3, J_4\}$.

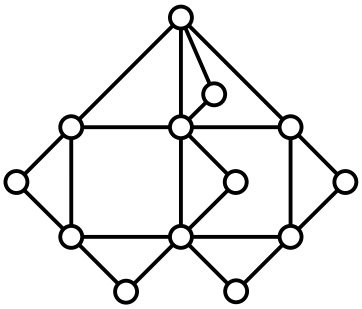


Figure 6.16: Witness graph for $F_e(P_3, K_3; C_4)$.

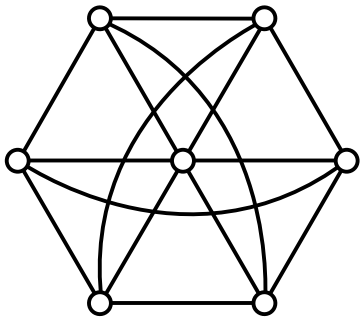


Figure 6.17: Witness graph for $F_e(P_3, BT; K_4)$ and $F_e(P_3, BT; J_5)$.

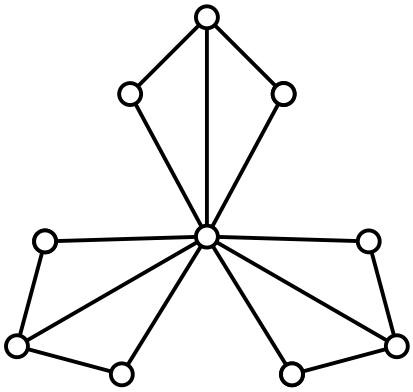


Figure 6.18: Witness graph for $F_e(P_3, BT; C_5)$.

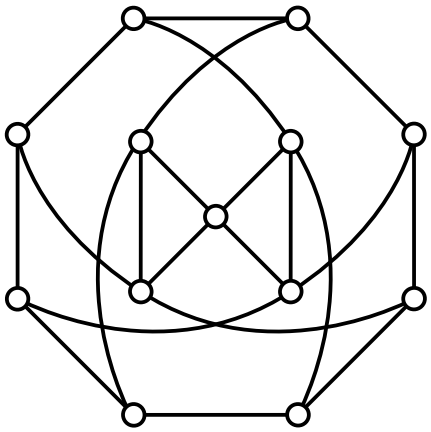


Figure 6.19: Witness graph for $F_e(P_3, C_5; C_4)$.

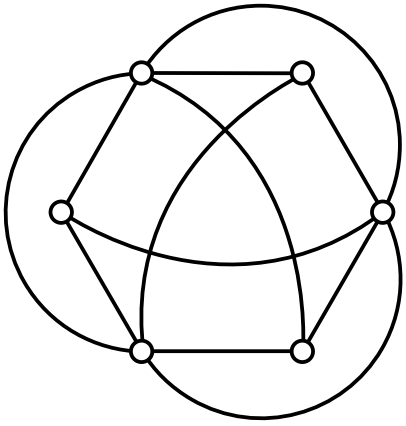


Figure 6.20: Witness graph for $F_e(P_3, BT; K_5)$.

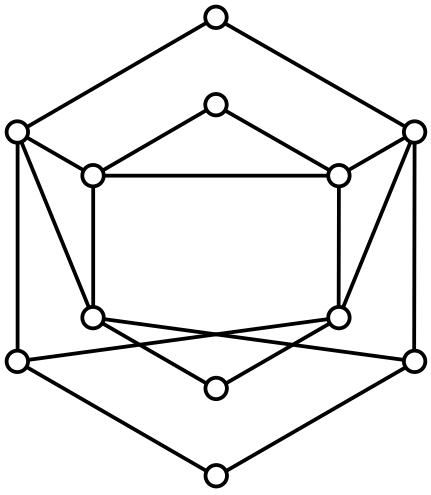


Figure 6.21: Witness graph for $F_e(P_3, C_6; C_4)$.

Chapter 7

Some Insight on Variants of Monotone SAT

Monotone SAT is a variant of the satisfiability problem where the input formula is in CNF, and each clause has only unnegated literals or only negated literals. Monotone 3SAT, where each clause has three literals, was shown to be NP-complete in 1978 [17]. More recently, work has been done to show that the problem remains NP-complete for other clause sizes and when the number of times a variable occurs in the formula is bounded [10, 11]. In this chapter, we provide some insight on a variant of Monotone SAT that allows clauses of two different sizes. We first define said variant of the Monotone SAT problem below:

Problem 7.1 (Monotone (p, q)-SAT)**.** *Let ϕ be a CNF-SAT formula such that there are two types of clauses: clauses with p unnegated literals and clauses with q negated literals. Does there exist a satisfying assignment for ϕ?*

The problem is clearly in NP for all p, q, since any satisfying assignment acts as a certificate verifiable in polynomial time. We provide new results on the hardness of three variants of this problem. Namely, Monotone $(2, 4)$-SAT, Monotone $(2, 5)$-SAT, and Monotone $(2, 6)$-SAT. The hardness results themselves are easy corollaries of the (P_3, H)-Non-Arrowing hardness results discussed in Chapter 5. However, we analyze the reductions of some select problems to bound the number of variable occurrences as well.

Before we discuss the results obtained via a reduction from (P_3, H)-Arrowing, we note that one can easily show Monotone $(2, k)$-SAT is NP-complete for all fixed $k \geq 4$, using the fact that Monotone

$(2,3)$-SAT is NP-complete, which was proven in [11]:

Let ϕ be a Monotone $(2,3)$-SAT formula, and ϕ' be the Monotone $(2,k)$-SAT formula we construct. Each three-literal clause in ϕ can be replaced with a k-literal clause including the three original literals and $k-3$ duplicates of one of the original literals. Note that the variables in each clause are not distinct, and if a variable appeared ℓ times in the original formula, it might appear at most $(k-2)\times\ell$ times in the new formula. We make note of this to juxtapose that reducing via arrowing allows us to have constant bounds for variable occurrences while ensuring that there are no duplicate variables in each clause.

7.1 The Hardness of Monotone $(2,k)$-SAT

We first make note of the following lemma, which links (H_1,H_2)-Non-Arrowing and Monotone (p,q)-SAT:

Lemma 7.1. *(H_1,H_2)-Non-Arrowing $\leq_m^p$ Monotone $(|E(H_1)|,|E(H_2)|)$-SAT.*

Proof. The reduction was stated in Section 4.1.3. The reduction takes polynomial time because H_1 and H_2 are fixed graphs. □

Note that this reduction also implies that the resulting SAT problem remains NP-complete even when the variables in each clause are distinct. Recall that the set of graphs we explored the hardness of is $\mathcal{Z}=\{C_4,C_5,C_6,J_4,BT,P_5,P_6\}$, and it is easy to see that $|E(H)|\in\{4,5,6\}$ for all $H\in\mathcal{Z}$. Thus, Lemma 7.1 and Theorem 5.1 together imply the following:

Theorem 7.1. *Monotone $(2,k)$-SAT is* NP*-complete for fixed $k\in\{4,5,6\}$, even when the variables in each clause are distinct.*

7.2 Bounding Variable Occurrences

In this section we explore the gadgets for the proofs of (P_3,C_k)-Non-Arrowing for $k\in\{4,5,6\}$ to show that Monotone $(2,k)$-SAT remains NP-complete even though each variable can occur at most y_k times, where y_k is some constant. Moreover, we bound how many times a variable appears as a negated and unnegated literal in the formula.

Recall from the reduction to Monotone (p, q)-SAT that each edge of the input graph corresponds to a variable. Moreover, there is a clause for each H_1 and H_2 present in the graph. Thus, to show our bounds, we analyze the number of times an edge appears in a P_3 and C_k in the graph G constructed via the (P_3, C_k)-Non-Arrowing reduction. This is especially easy for (P_3, C_k) since we showed earlier that no new C_k's are formed when joining the gadgets. Thus, we need only be concerned with new P_3's that are created when joining the gadgets.

Checking how many C_k's each edge belongs to can be done by enumerating over each edge in the variable and clause gadgets. Checking how many P_3's requires some modifications of the gadgets before we can check computationally. Note that new P_3's are formed only at the input/output vertices. Also, note that an edge belongs to a P_3 with only those edges that it is adjacent to. Thus, we can construct new graphs which include the maximum amount of these edges at each input/output vertex to obtain our bounds for P_3 occurrences for each edge. We elaborate on this with an example.

Consider the (P_3, C_4)-Non-Arrowing gadgets (Figure 5.4). Note that the maximum degree of an output vertex of the variable gadget is two, while the maximum degree of an input vertex of a clause gadget is three. To bound the number of times an edge appears in a P_3 in the graph G constructed via the reduction, we can analyze the graphs in Figure 7.1. Note that in Figure 7.1, the input and output vertices from the gadgets in Figure 5.4 have new leaf vertices to account for new P_3's formed while connecting gadgets. After our analysis, we find that the maximum number of times an edge appears in a P_3 is eight and a C_4 is three. Moreover, the maximum number of times an edge appears in either a P_3 or C_4 is 11.

Thus, the formula constructed using G via the reduction to Monotone $(2, 4)$-SAT has at most 11 occurrences of each variable. Furthermore, each variable may appear unnegated eight times and negated three times, at most.

The same methodology can be applied, mutatis mutandis, on the graphs in Figures 7.2 and 7.3 and to obtain similar bounds for for Monotone $(2, 5)$ and $(2, 6)$-SAT. With these bounds, we can assert the following theorem:

Theorem 7.2. *For fixed $k \in \{4, 5, 6\}$, Monotone $(2, k)$-SAT is* NP*-complete even when the variables in each clause are distinct, and the number of times each variable occurs is bounded by a constant. In particular:*

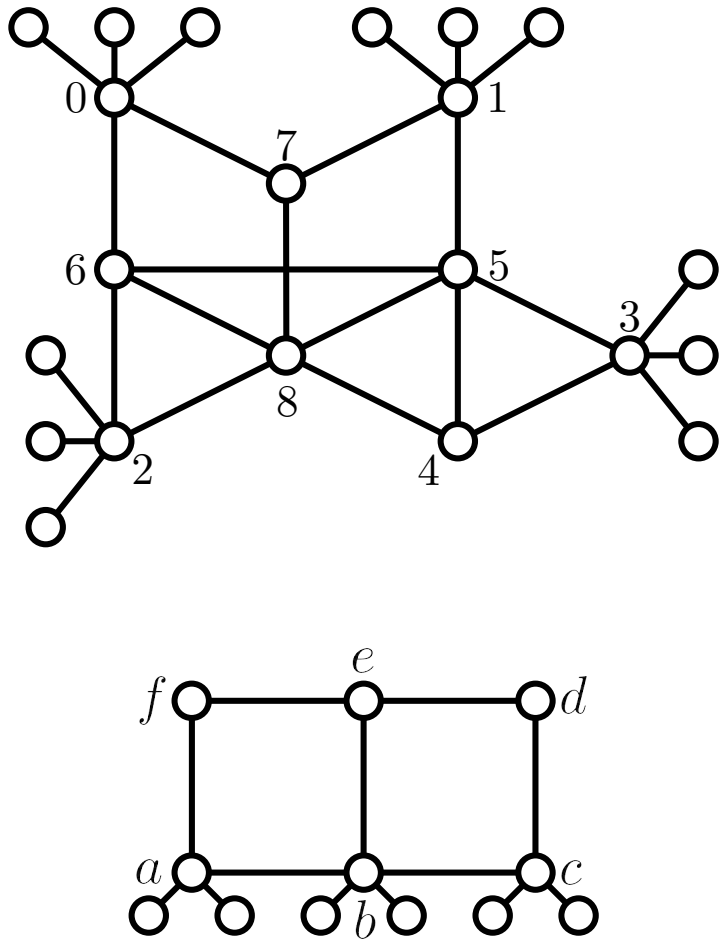
1. *Monotone $(2, 4)$-SAT is* NP*-complete even when each variable appears at most 11 times, where a variable appears as an unnegated literal at most eight times and as a negated literal at most*

three times.

2. *Monotone* $(2,5)$*-SAT is* NP*-complete even when each variable appears at most 11 times, where a variable appears as an unnegated literal at most seven times and as a negated literal at most four times.*

3. *Monotone* $(2,6)$*-SAT is* NP*-complete even when each variable appears at most 15 times, where a variable appears as an unnegated literal at most nine times and as a negated literal at most six times.*

Our main takeaway from these theorems is that the number of occurrences of the variables can be bounded by a constant while having distinct variables in each clause. The actual bounds are not very impressive; there is much room for improvement, considering that many variants of SAT bound the number of variable occurrences by four. While it may be possible to improve these bounds by finding better gadgets, we suspect that the improvement to the bounds will be minuscule using our technique. For completeness, we note that Darmann et al. [11] showed that Monotone $(2,3)$-SAT is NP-complete when each variable occurs exactly four times, and the variables in each clause are distinct.

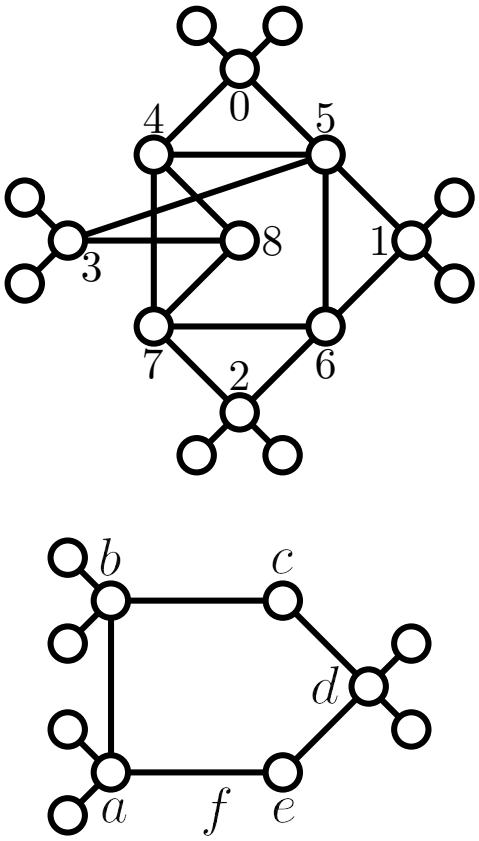


(a) Modified gadget graphs of (P_3, C_4).

Edge	P_3	C_4	**Total**
$(0,6)$	7	1	8
$(0,7)$	6	1	7
$(1,5)$	**8**	1	9
$(1,7)$	6	1	7
$(2,6)$	7	1	8
$(2,8)$	**8**	1	9
$(3,4)$	6	1	7
$(3,5)$	**8**	1	9
$(4,5)$	6	1	7
$(4,8)$	6	2	8
$(5,6)$	7	2	9
$(5,8)$	**8**	**3**	**11**
$(6,8)$	7	2	9
$(7,8)$	6	2	8
(a,b)	7	1	8
(a,f)	4	1	5
(b,c)	7	1	8
(b,e)	6	2	8
(c,d)	4	1	5
(f,e)	3	1	4
(e,d)	3	1	4

(b) Table of occurrences.

Figure 7.1: The graphs analyzed to bound the variable occurrences in Monotone $(2,4)$-SAT. In the table on the right, we show for each edge how many times it appeared in a P_3 and a C_4.

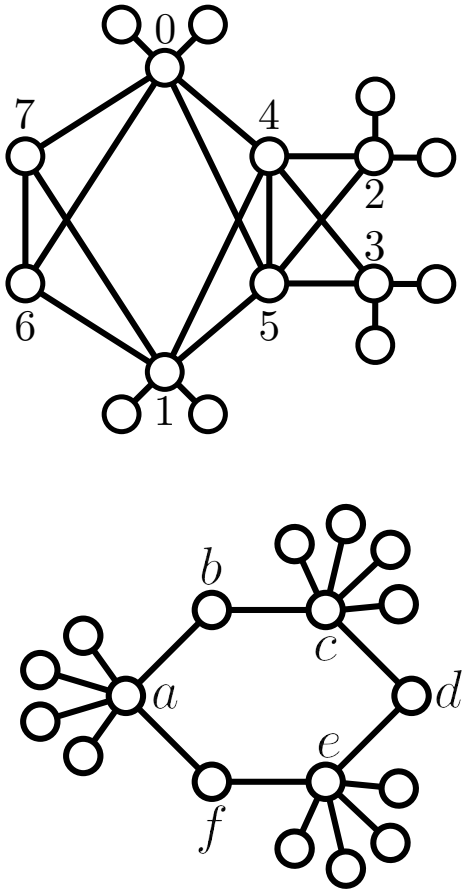


(a) Modified gadget graphs of (P_3, C_5).

Edge	P_3	C_5	**Total**
$(0,4)$	6	2	8
$(0,5)$	**7**	2	9
$(1,5)$	**7**	1	8
$(1,6)$	6	1	7
$(2,6)$	6	1	7
$(2,7)$	6	1	7
$(3,5)$	**7**	3	10
$(3,8)$	5	3	8
$(4,5)$	**7**	**4**	**11**
$(4,7)$	6	**4**	10
$(4,8)$	5	2	7
$(5,6)$	**7**	**4**	**11**
$(6,7)$	6	**4**	10
$(7,8)$	5	3	8
(a,b)	6	1	7
(a,e)	4	1	5
(b,c)	4	1	5
(c,d)	4	1	5
(d,e)	4	1	5

(b) Table of occurrences.

Figure 7.2: The graphs analyzed to bound the variable occurrences in Monotone $(2,5)$-SAT. In the table on the right, we show for each edge how many times it appeared in a P_3 and a C_5.



(a) Modified gadget graphs of (P_3, C_6).

Edge	P_3	C_6	**Total**
$(0,4)$	**9**	**6**	**15**
$(0,5)$	**9**	**6**	**15**
$(0,6)$	7	**6**	13
$(0,7)$	7	**6**	13
$(1,4)$	**9**	**6**	**15**
$(1,5)$	**9**	**6**	**15**
$(1,6)$	7	**6**	13
$(1,7)$	7	**6**	13
$(2,4)$	7	4	11
$(2,5)$	7	4	11
$(3,4)$	7	4	11
$(3,5)$	7	4	11
$(4,5)$	8	4	12
$(6,7)$	4	4	8
(a,b)	6	1	7
(a,f)	6	1	7
(b,c)	6	1	7
(c,d)	6	1	7
(d,e)	6	1	7
(e,f)	6	1	7

(b) Table of occurrences.

Figure 7.3: The graphs analyzed to bound the variable occurrences in Monotone $(2,6)$-SAT. In the table on the right, we show for each edge how many times it appeared in a P_3 and a C_6.

Chapter 8

Conclusion and Future Work

In this work, we tackled a number of graph arrowing problems via a combinatorial computing approach. We focused on the case where one must avoid red P_3's. We discussed a general methodology to generate (H_1, H_2)-good graphs, and employed this to generate (P_3, H)-good graphs up to ten vertices. For seven distinct H, we showed that (P_3, H)-Arrowing is coNP-complete via a framework based on finding variable and clause gadgets to reduce from variants of SAT. The gadgets used for these reductions were found computationally, by enumerating over the generated (P_3, H)-good graphs. For nine distinct H, we found almost all Folkman numbers of interest. All of these were found computationally. The only open cases are $F_e(P_3, BT; J_4)$ and $F_e(P_3, BT; C_4)$, which we plan on exploring in the future.

Furthermore, we hope to use the insight gained by our computationally assisted reductions to find patterns that will allow us to show that (P_3, P_k)-Arrowing or (P_3, C_k)-Arrowing are coNP-complete for general $k \geq 5$. We believe that this is a feasible task given the insight gained from our computational approach. A more ambitious goal is to prove a dichotomy theorem for (P_3, H)-Arrowing: categorizing problems to be in P or coNP-complete for all possible H.

We plan on exploring other arrowing problems as well. In particular, we will explore (P_k, P_ℓ)-Arrowing. We conjecture that this is coNP-complete for all fixed $k, \ell \geq 4$. So far, this has only be shown for $k = \ell = 4$.

Bibliography

[1] Piotr Berman, Marek Karpinski, and Alex Scott. Approximation hardness of short symmetric instances of max-3sat. *Electronic Colloquium on Computational Complexity*, 01 2003.

[2] Aleksandar Bikov. *Computation and Bounding of Folkman Numbers.* PhD thesis, Sofia University "St. Kliment Ohridski", 06 2018.

[3] Stefan A. Burr. On the computational complexity of Ramsey-type problems. In *Mathematics of Ramsey theory*, pages 46–52. Springer, 1990.

[4] Stefan A. Burr, Jaroslav Nešetřil, and Vojtech Rödl. On the use of senders in generalized Ramsey theory for graphs. *Discrete mathematics*, 54(1):1–13, 1985.

[5] G. Chartrand, D.P. Geller, and S. Hedetniemi. A generalization of the chromatic number. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 64, pages 265–271. Cambridge University Press, 1968.

[6] Václav Chvátal and Frank Harary. Generalized Ramsey theory for graphs. iii. small off-diagonal numbers. *Pacific Journal of mathematics*, 41(2):335–345, 1972.

[7] Jonathan Coles and Stanisław Radziszowski. Computing the Folkman number $F_v(2, 2, 3; 4)$. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 58:13–22, 2006.

[8] Lenore Cowen, Wayne Goddard, and C. Esther Jesurum. Defective coloring revisited. *Journal of Graph Theory*, 24(3):205–219, 1997.

[9] Lenore J. Cowen, Robert H. Cowen, and Douglas R. Woodall. Defective colorings of graphs in surfaces: partitions into subgraphs of bounded valency. *Journal of Graph Theory*, 10(2):187–195, 1986.

[10] Andreas Darmann and Janosch Döcker. On simplified NP-complete variants of monotone3-sat. *Discret. Appl. Math.*, 292:45–58, 2021.

[11] Andreas Darmann, Janosch Döcker, and Britta Dorn. The monotone satisfiability problem with bounded variable appearances. *Int. J. Found. Comput. Sci.*, 29(6):979–993, 2018.

[12] Ralph J. Faudree, S.L. Lawrence, T.D. Parsons, and Richard H. Schelp. Path-cycle Ramsey numbers. *Discrete Mathematics*, 10(2):269–277, 1974.

[13] Jon Folkman. Graphs with monochromatic complete subgraphs in every edge coloring. *SIAM Journal on Applied Mathematics*, 18(1):19–24, 1970.

[14] Michael R. Garey and David S. Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

[15] László Gerencsér and András Gyárfás. On Ramsey-type problems. *Ann. Univ. Sci. Budapest. Eötvös Sect. Math*, 10:167–170, 1967.

[16] Jan Goedgebeur. On minimal triangle-free 6-chromatic graphs. *Journal of Graph Theory*, 93(1):34–48, 2020.

[17] E. Mark Gold. Complexity of automaton identification from given data. *Inf. Control.*, 37(3):302–320, 1978.

[18] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using NetworkX. Technical report, Los Alamos National Lab, Los Alamos, NM (United States), 2008.

[19] F. Harary and K. Jones. Conditional colorability ii: Bipartite variations. *Congressus Numerantium*, 50:205–218, 1985.

[20] Zohair Raza Hassan, Edith Hemaspaandra, and Stanisław Radziszowski. Ramsey theory: Constructions & complexity. Manuscript on Arrowing Problems, 2022.

[21] Zohair Raza Hassan, Yu Jiang, David E. Narváez, Stanisław Radziszowski, and Xiaodong Xu. On some generalized vertex Folkman numbers. *arXiv preprint arXiv:2110.03121*, 2021.

[22] Justen Holl, Elizabeth Tso, and Julia Balla. Ramsey theory: Order from chaos. `https://math.mit.edu/~ngadish/notes/18204/ramsey.pdf`, September 2020.

[23] Ian Holyer. The NP-completeness of edge-coloring. *SIAM Journal on computing*, 10(4):718–720, 1981.

[24] Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. PySAT: A Python toolkit for prototyping with SAT oracles. In *SAT*, pages 428–437, 2018.

[25] Tommy R. Jensen and Gordon F. Royle. Small graphs with chromatic number 5: A computer search. *Journal of Graph Theory*, 19(1):107–116, 1995.

[26] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Symposium on the Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.

[27] Joel Lathrop and Stanisław Radziszowski. Computing the Folkman number $F_v(2,2,2,2,2;4)$. *JCMCC. The Journal of Combinatorial Mathematics and Combinatorial Computing*, 78, 08 2011.

[28] Carlos V.G.C. Lima, Dieter Rautenbach, Uéverton S. Souza, and Jayme L. Szwarcfiter. Decycling with a matching. *Information Processing Letters*, 124:26–29, 2017.

[29] Carlos V.G.C. Lima, Dieter Rautenbach, Uéverton S. Souza, and Jayme L. Szwarcfiter. Eliminating odd cycles by removing a matching. *arXiv preprint arXiv:1710.07741*, 2017.

[30] Carlos V.G.C. Lima, Dieter Rautenbach, Uéverton S. Souza, and Jayme L. Szwarcfiter. Bipartizing with a matching. In *International Conference on Combinatorial Optimization and Applications*, pages 198–213. Springer, 2018.

[31] Jordan K. Matelsky, Elizabeth P. Reilly, Erik C. Johnson, Jennifer Stiso, Danielle S. Bassett, Brock A. Wester, and William Gray-Roncal. DotMotif: an open-source tool for connectome subgraph isomorphism search and graph queries. *Scientific Reports*, 11(1), Jun 2021.

[32] Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60:94–112, 2014.

[33] Pdobsan. Pdobsan/pynauty: Isomorphism testing and automorphisms of graphs. `https://github.com/pdobsan/pynauty`.

[34] Stanisław Radziszowski. Small Ramsey numbers. *The Electronic Journal of Combinatorics*, 1000, 2011.

[35] F. Ramsey. On a problem in formal logic. *Proceedings of the London Mathematical Society*, 30:264–286, 1930.

[36] Vladislav Rutenburg. Complexity of generalized graph coloring. In *Mathematical Foundations of Computer Science 1986, Bratislava, Czechoslovakia, August 25-29, 1996, Proceedings*, volume 233 of *Lecture Notes in Computer Science*, pages 573–581. Springer, 1986.

[37] M. Schaefer. Graph Ramsey theory and the polynomial hierarchy. *Journal of Computer and System Sciences*, 62(2):290–322, 2001.

[38] Craig A. Tovey. A simplified NP-complete satisfiability problem. *Discrete applied mathematics*, 8(1):85–89, 1984.

[39] Christopher A. Wood. Small Folkman numbers. In *https://www.cs.rit.edu/~spr/COURSES/CCOMP/cawfolk.pdf*, 2014.

[40] Xiaodong Xu, Meilian Liang, and Stanislaw P. Radziszowski. On the nonexistence of some generalized Folkman numbers. *Graphs Comb.*, 34(5):1101–1110, 2018.