University of Bergen

Master Thesis

Improved Combinatorial Bounds and Enumerators for the Connected Vertex Cover and Independent Feedback Vertex Set Problems

Author:
Thomas Wingsternes

Supervisor:
Daniel Lokshtanov

June 1, 2018

Abstract

In this thesis we shall study both the maximum number of minimal connected vertex covers and the maximum number of minimal independent feedback vertex sets in graphs. A subset S of the vertices of a graph G is a *vertex cover* if every edge in G is incident to some vertex in S. A vertex cover S in a graph G is a *connected vertex cover* if the subgraph of G induced by S is connected. A connected vertex cover S is a *minimal connected vertex cover* if no proper subset of S is also a connected vertex cover. A subset S of the vertices of a graph G is a *feedback vertex set* if the removal of S from G yields an acyclic graph. A feedback vertex set S in a graph G is an *independent feedback vertex set* if there are no edges in G which have both endpoints in S. An independent feedback vertex set S is a *minimal independent feedback vertex set* if no proper subset of S is also an independent feedback vertex set.

Golovach et al. [1] showed that there are at most 1.8668^n minimal connected vertex covers in a graph on n vertices, and that these can be enumerated in time $\mathcal{O}(1.8668^n)$. Furthermore, it was shown by Agrawal et al. [2] that there are at most 1.7485^n minimal independent feedback vertex sets in a graph on n vertices. The authors of [2] also showed that this bound is algorithmic, that the set of all minimal independent feedback vertex sets in an n-vertex graph can be enumerated in time $\mathcal{O}^*(1.7485^n)$, where the $\mathcal{O}^*$-notation supresses polynomial factors.

We present an upper bound $2 \cdot 1.7076^n$ on the number of minimal connected vertex covers in a graph on n vertices. We also present new bounds on the maximum number of minimal independent feedback vertex sets in a graph. In particular, we show that for a graph on n vertices, there are at most 1.7229^n minimal independent feedback vertex sets, and that there exists an n-vertex graph which contains 1.5067^n minimal independent feedback vertex sets. We show that the upper bounds we achieve are algorithmic by presenting an algorithm which enumerates all minimal connected vertex covers in time $\mathcal{O}^*(1.7076^n)$, and an algorithm which enumerates all minimal independent feedback vertex sets in time $\mathcal{O}^*(1.7229^n)$, where n is the number of vertices in the input graph.

To my loving parents, Torbjørn and May Britt.

Acknowledgements

First and foremost, I would like to thank my supervisor, Daniel. His guidance has been invaluable, and I must thank him for all the conversations we have had; they have been tremendously helpful while writing this thesis. I am also very grateful to my classmates, both for their support and for their good friendship. Finally, I would like to thank my father Torbjørn, and my partner Evy-Ann, for their love and support.

Contents

Chapter 1

Introduction

Graphs are interesting mathematical objects which are often very useful in modeling real-world problems. A graph consists of a set of vertices, and a set of edges between these vertices. In real-world applications, the vertices of a graph typically represent some object from the problem domain, and an edge between two vertices indicates that the objects corresponding to these vertices have some kind of relationship.

In the field of computer science and graph theory, much research has been dedicated to the problem of finding subsets of the vertices of a graph which have some specific property. For instance, in the Vertex Cover problem, one asks if there exists a subset of the vertices of size at most k such that every edge of the input graph is incident to some vertex in the subset. Another example is the Feedback Vertex Set problem. In this problem, we are given as input an integer k and a graph G, and the question is whether there exists a subset of the vertices of size at most k whose removal from G results in an acyclic graph.

Another well-studied problem is the problem of determining the maximum number of vertex subsets of a graph which can have some given property, as a function of the number of vertices in the graph. To make the problem more interesting, we typically require that the subsets are inclusion-wise minimal or inclusion-wise maximal with respect to the given property. To see why, consider the following. An *independent set* S in a graph G is a subset of the vertices of G such that there are no edges in G which have both endpoints in S. An independent set S is a *maximal independent set* if no proper superset of S is also an independent set. Now, consider an n-vertex graph G in which there are no edges. Since there are no edges in G, all subsets of the vertices of G are independent sets, and there are therefore 2^n independent sets in G. However, there is only *one maximal* independent set in G, the set which contains all vertices of G. Can there exist graphs on n vertices in which the number of maximal

independent sets is exponential in n? The answer is yes, and we shall see an example of such a graph shortly.

First, a *complete* graph is a graph in which there is an edge between every pair of vertices. We denote the complete graph on n vertices by K_n. Consider the n-vertex graph G composed of $n/3$ disjoint copies of K_3. We provide an illustration of G in Fig. 1.1. Any maximal independent set in G must contain exactly one vertex from each copy of K_3. As there are 3 vertices to choose from within each copy, and $n/3$ copies from which we must choose exactly one vertex, there are $3^{n/3}$ maximal independent sets in G. Interestingly, $3^{n/3}$ is the maximum number of maximal independent sets in a graph on n vertices, an acclaimed result of Moon and Moser [3].

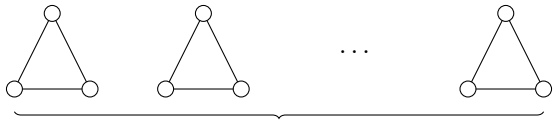


Figure 1.1: G is composed of $n/3$ disjoint copies of K_3.

Closely related to the problem of upper bounding the number of subsets having some given property is the problem of enumerating these subsets efficiently. If we can construct an algorithm that enumerates all subsets which have some given property, then an upper bound on the number of such subsets follows from the running time of the algorithm. For instance, Fomin et al. [4] constructed an algorithm for listing all minimal dominating sets of an n-vertex graph in time $\mathcal{O}(1.7159^n)$. As an immediate consequence of this, the authors proved that every graph on n vertices contains at most 1.7159^n minimal dominating sets.

Very often, enumeration algorithms are *branching algorithms*. The key to upper bounding the running time of a branching algorithm is usually to obtain a lower bound on the amount of *progress* made by the algorithm in each step of the execution. For graph problems, a simple and natural measure of progress is the number of vertices in the graph. However, a more involved choice of measure can have a significant impact on the worst-case running time one arrives at when analyzing the algorithm. In the preliminary version [5] of their paper [4], Fomin et al. had shown an upper bound of 1.7697^n on the number of minimal dominating sets in a graph on n vertices. Through a more careful choice of measure, the authors were able to reduce their previous result of 1.7697^n in [5] to the upper bound 1.7159^n achieved in [4]. The method of focusing on the choice of measure, rather than extending an algorithm with more and more rules, is known as *Measure & Conquer*.

Another interesting upper bound with respect to the dominating sets of a graph is the upper bound of Lokshtanov et al. [6] on the maximum number of minimal connected dominating sets in a graph. Lokshtanov et al. proved that there exists a constant $\epsilon > 10^{-50}$ such that every graph on n vertices has at most $\mathcal{O}(2^{(1-\epsilon)n})$ minimal connected dominating sets, thereby breaking the trivial barrier of 2^n. Furthermore, Couturier et al. [7] substantially improved the upper bound on the number of minimal dominating sets, for several classes of graphs. Other important results in the area of combinatorial bounds on the number of vertex subsets with given properties include [8, 9, 10].

In this thesis we shall study the maximum number of minimal connected vertex covers and the maximum number of minimal independent feedback vertex sets in graphs. A subset S of the vertices of a graph G is a *vertex cover* if every edge in G is incident to some vertex in S. A vertex cover S in a graph G is a *connected vertex cover* if the subgraph of G induced by S is connected. A connected vertex cover S is a *minimal connected vertex cover* if no proper subset of S is also a connected vertex cover. A subset S of the vertices of a graph G is a *feedback vertex set* if the removal of S from G yields an acyclic graph. A feedback vertex set S in a graph G is an *independent feedback vertex set* if there are no edges in G which have both endpoints in S. An independent feedback vertex set S is a *minimal independent feedback vertex set* if no proper subset of S is also an independent feedback vertex set.

Golovach et al. [1] described a branching algorithm for enumerating all minimal connected vertex covers of a graph, and showed that there are at most 1.8668^n minimal connected vertex covers in a graph on n vertices, and that these can be enumerated in time $\mathcal{O}(1.8668^n)$. The authors of [1] also presented a lower bound example, showing that there exists an n-vertex graph which contains $3^{\frac{n-1}{3}}$ minimal connected vertex covers. It was later shown by Ryland [11] that there exists a graph on n vertices which contains 1.51978^n minimal connected vertex covers.

Agrawal et al. [2] showed that there are at most 1.7485^n minimal independent feedback vertex sets in a graph on n vertices. Unfortunately, the proof was omitted from [2] due to space constraints. However, through personal communication, we have obtained the longer, unpublished version [12] of [2], which contains the proof. In [12], Agrawal et al. showed that the bound is algorithmic, that the set of all minimal independent feedback vertex sets in an n-vertex graph can be enumerated in time $\mathcal{O}^*(1.7485^n)$, where the $\mathcal{O}^*$-notation supresses polynomial factors.

We shall build upon the algorithm of Agrawal et al. [2, 12] and show that there are at most 1.7229^n minimal independent feedback vertex sets in a graph on n vertices, and further, that these can be enumerated in time $\mathcal{O}^*(1.7229^n)$. Using a similar approach, we shall show that there are at most $2 \cdot 1.7076^n$ minimal connected vertex covers in a graph on n vertices, and that we can enumerate

the set of all minimal connected vertex covers in time $\mathcal{O}^*(1.7076^n)$. We shall also show that there exists a graph which contains 1.5067^n minimal independent feedback vertex sets.

The remainder of this thesis is organized as follows. In Chapter 2 we will introduce the notation and terminology that we will be using throughout the thesis. We present our results in Chapter 3 and Chapter 4. In Chapter 3, we shall present our first result, the upper bound $2 \cdot 1.7076^n$ on the number of minimal connected vertex covers in a graph on n vertices. In Chapter 4 we present a lower bound example which shows that the maximum number of minimal independent feedback vertex sets in an n-vertex graph is at least 1.5067^n, and further, we present an improved upper bound 1.7229^n on the number of minimal independent feedback vertex sets in a graph on n vertices. We note that we arrived at the results presented in Chapter 4 before arriving at the results presented in Chapter 3. We have chosen to present the upper bound on the number of minimal connected vertex covers before preseting the upper bound on the number of minimal independent feedback vertex sets, because we feel that the proof of the former is not as involved as the proof of the latter. We conclude the thesis with a few open problems and some suggestions for future work in Chapter 5.

Chapter 2

Preliminaries

In this thesis we assume that the reader has knowledge of propositional logic. We refer the reader to [13] for an introduction to this topic.

2.1 Mathematical Notation

We denote the real numbers by $\mathbb{R}$, the positive real numbers by $\mathbb{R}^+$, and the natural numbers $\mathbb{N}$.

2.1.1 Asymptotic Notation

Let f and g be functions from the real numbers to the real numbers. If there exists a constant $c > 0$ and a real number n_0 such that for every $n \geq n_0$ we have $f(n) \leq cg(n)$, then we write $f(n) = \mathcal{O}(g(n))$, and we say that $g(n)$ is an asymptotic upper bound of $f(n)$. If $f(n) = \mathcal{O}(poly(n) \cdot g(n))$, for some polynomial $poly(n)$, then we write $f(n) = \mathcal{O}^*(g(n))$. In other words, O^*-notation supresses polynomial factors.

2.2 Set Theory

A *set* is an unordered collection of *distinct* elements. By distinct, we mean that every element in the set appears only once. A *multiset* S is an unordered collection of elements in which we allow the same element to occur more than

once. The number of times the element x occurs in S is called the *multiplicity* of x in S.

Let A and B be sets. By $x \in A$, we state that the element x is a *member* of the set A, and that A contains x. Conversely, by $x \notin A$ we mean that x is *not* a member of A. If the set A is entirely contained in the set B, we say that A is a *subset* of B, and we denote this by writing $A \subseteq B$. Formally, $A \subseteq B$ if and only if for every $x \in A$ we have $x \in B$. Two sets A and B are equal if and only if they are subsets of one another, that is, $A = B$ if and only if $A \subseteq B$ and $B \subseteq A$. If $A \subseteq B$ and $A \neq B$, then we say that A is a *proper* subset of B, written $A \subset B$.

The *cardinality* of a set A is the number of elements in A, and is denoted by $|A|$. If A has no elements, then we say that A is the *empty set*. We use the symbol $\emptyset$ to denote the empty set. The elements of a set may themselves be sets. The *power set* of a set S, denoted $\mathcal{P}(S)$, is the set of all subsets of S.

2.2.1 Set Operations

We often use a mathematical notation called *set-builder notation* when we describe sets of elements. A set S described in set-builder notation typically has the form $S = \{x \mid P(x)\}$, where $P(x)$ is a predicate, and S is then the set of all elements x for which $P(x)$ holds. We now define the *union*, *intersection* and *set difference* operations. The union of two sets A and B, $A \cup B$, is the set containing every element of A and every element of B. The intersection of A and B, $A \cap B$, is the set containing every element that is *both* in A *and* in B. Finally, the set difference of A and B, $A \setminus B$, is the set containing every element of A that is *not* in B. Formally,

$$A \cup B = \{x \mid x \in A \vee x \in B\}$$
$$A \cap B = \{x \mid x \in A \wedge x \in B\}$$
$$A \setminus B = \{x \mid x \in A \wedge x \notin B\}$$

We may express the union or intersection of k sets $S_1, S_2, \ldots, S_k$ in a compact manner by writing $\bigcup_{i=1}^{k} S_i$, or $\bigcap_{i=1}^{k} S_i$, respectively. That is,

$$\bigcup_{i=1}^{k} S_i = S_1 \cup S_2 \cup \ldots \cup S_k$$
$$\bigcap_{i=1}^{k} S_i = S_1 \cap S_2 \cap \ldots \cap S_k$$

If $A \cap B = \emptyset$, then A and B are said to be *disjoint*. A *partition* of a set S is a collection of non-empty pairwise disjoint subsets of S whose union is S. More

precisely, a partition of S is a collection of non-empty sets $S_1, S_2, \ldots, S_k$ such that for every $i \neq j$ we have $S_i \cap S_j = \emptyset$, and $\bigcup_{i=1}^{k} S_i = S$. The *cartesian product of* two sets A and B is the set

$$A \times B = \{(a, b) \mid a \in A, b \in B\}$$

A *binary relation* on a set A is a set $R \subseteq A \times A$. We say that a relation R on a set A is *symmetric* if for every $x, y \in A$ it holds that $(x, y) \in R$ if and only if $(y, x) \in R$.

2.3 Graph Theory

2.3.1 Graphs

Formally, a graph is a pair $G = (V, E)$, where V is the set of *vertices* in G, and $E \subseteq V \times V$ is the set of edges in G. For a graph G we often denote the set of vertices of G by $V(G)$, and the set of edges of G by $E(G)$. If $(u, v) \in E(G)$, then we say that u is *adjacent* to v. In this thesis we will concern ourselves only with *undirected* graphs. An undirected graph is a graph G for which $E(G)$ is symmetric.

2.3.2 The Neighborhood and Degree of a Vertex

If $e = (u, v)$ is an edge, then we say that e is *incident* to u and v, and that u and v are the *endpoints* of e. We may also say that e is the edge *between* u and v, or that e is the edge *from* u *to* v. Further, when $(u, v) \in E(G)$ we say that u and v are *neighbors*. We denote the set of neighbors of a vertex v by $N(v)$. That is, $N(v) = \{u \in V(G) \mid (u, v) \in E(G)\}$. We refer to $N(v)$ as the *neighborhood* of v. The *closed* neighborhood of v is the set $N[v] = N(v) \cup \{v\}$. If G is a graph, $v \in V(G)$ and $N[v] = V(G)$, then we say that v is a *universal* vertex. In other words, a vertex is universal if it is adjacent to every other vertex in the graph. If G is a graph on n vertices, and every vertex of G is universal, then G is the *complete* graph on n vertices. We denote the complete graph on n vertices by K_n. The *degree* of a vertex v, denoted $d(v)$, is the number of neighbors of v. That is, $d(v) = |N(v)|$. If $d(v) = 0$, then we say that v is an *isolated* vertex. By $\delta(G)$ and $\Delta(G)$ we denote the minimum degree and the maximum degree of the graph G, respectively. Formally,

$$\delta(G) = min\{d(v) \mid v \in V(G)\}$$
$$\Delta(G) = max\{d(v) \mid v \in V(G)\}$$

Often, when we are dealing with multiple graphs and it is not clear from context to which graph we are referring to, we may subscript $N(\cdot)$ and $d(\cdot)$ to remove

ambiguities. Thus, $N_G(v)$ and $d_G(v)$ will refer to the neighborhood of v in the graph G, and to the degree of v in the graph G, respectively.

2.3.3 Subgraphs and Graph Isomorphism

A subgraph of a graph $G = (V, E)$ is a graph $G' = (V', E')$, such that $V' \subseteq V$ and $E' \subseteq E$. Let G be a graph, and let $S \subseteq V(G)$. Let E_S be the set of edges that have both endpoints in S, i.e., let $E_S = \{(u, v) \in E(G) \mid u, v \in S\}$. We call $G[S] = (S, E_S)$ the subgraph of G *induced* by S. In other words, the subgraph of G induced by S is the subgraph obtained from G by removing all vertices of G that are not in S and all edges of G which have at least one endpoint in $V(G) \setminus S$. In many problems we look for a subset of the vertices of a graph whose removal yields a graph with some special properties. Again, let $S \subseteq V(G)$. By $G - S$ we denote the subgraph of G obtained by removing all vertices of S and all edges of G that have at least one endpoint in S. That is, $G - S = G[V(G) \setminus S]$. We shall overload this operation by allowing the removal of sets of edges as well. If $S \subseteq E(G)$, then by $G - S$ we mean the subgraph $(V, E \setminus S)$. If e is an edge in a graph $G = (V, E)$, then G/e denotes the graph obtained by *contracting* e. Formally, $G/(u, v) = (V', E')$, where

$$V' = (V \setminus \{u, v\}) \cup \{w\}$$
$$E' = E(G - \{u, v\}) \cup \{(w, x) \mid (u, x) \in E(G)\} \cup \{(w, x) \mid (v, x) \in E(G)\}$$

Another operation we shall need is *edge subdivision*. Let $(u, v) \in E(G)$. When we *subdivide* the edge (u, v), we obtain a new graph

$$G' = (V \cup \{w\}, (E \setminus \{(u, v)\}) \cup \{(u, w), (v, w)\})$$

Let G and H be graphs. If there exists a bijection $f : V(G) \to V(H)$ such that for every $u, v \in V(G)$ we have $(u, v) \in E(G)$ if and only if $(f(u), f(v)) \in E(H)$, then we say that G is *isomorphic* to H.

2.3.4 Multigraphs

Thus far we have been discussing *simple* graphs. We shall also be working with *multigraphs*. In a multigraph the set of edges is a multiset of multisets, thus allowing the presence of both multi-edges and self-loops, which we now define formally. Let G be a multigraph, then a *multi-edge* in G is an edge $e \in E(G)$ of multiplicity $m \geq 2$, and a *self-loop* on a vertex $v \in V(G)$ is an edge (v, v). In other words, a multi-edge is an edge that occurs more than once between the same pair of vertices, and a self-loop is an edge that begins and ends at the same vertex.

2.3.5 Paths, Cycles and Connectivity

A *path* in a graph G is a sequence of *distinct* vertices $P = (v_1, v_2, \ldots, v_k)$, such that $(v_i, v_{i+1}) \in E(G)$ for $i = 1, 2, \ldots, k-1$. We say that P is a path between v_1 and v_k. If $(v_1, v_2, \ldots, v_k)$ is a path in G, and $(v_k, v_1) \in E(G)$, then $(v_1, v_2, \ldots, v_k, v_1)$ is a *cycle* in G. Let G be a multigraph and let e be some edge in G. The endpoints of e form a cycle in G if e is a self-loop, or if e is a multi-edge. If a graph has no cycles, then it is *acyclic*. We say that a graph G is *connected* if for every pair of vertices $u, v \in V(G)$, there exsists a path between u and v. We note that a graph with no vertices is then connected. Let G be a graph, let $v \in V(G)$, and let C_v be the set of vertices u for which a path between u and v exists. That is, let

$$C_v = \{u \in V(G) \mid \text{there exists a path between } u \text{ and } v\}$$

We say that C_v is a *connected component* of G. If a graph has more than one connected component, then it is *disconnected*.

2.3.6 Trees and Spanning Trees

A *tree* is a connected acyclic graph. If G is a graph in which every connected component is a tree, then G is a *forest*. Let $G = (V, E)$ be a graph, then a *spanning tree* of G is a subgraph $T = (V, E')$ of G such that T is a tree. Note that the set of vertices of T is the same as the set of vertices of G. We say that E' *forms* a spanning tree in G. We often refer to the vertices of a tree as the *nodes* of the tree.

2.3.7 Independent sets and Bipartite Graphs

An *independent set* in a graph G is a set $S \subseteq V(G)$ such that for every $u, v \in S$ we have $(u, v) \notin E(G)$. If there exists a partition (A, B) of $V(G)$ such that both A and B are independent sets, then we say that G is *bipartite*, and that (A, B) is a *bipartition* of the graph.

2.3.8 Feedback Vertex Sets

Let G be a graph, and let $S \subseteq V(G)$. If $G - S$ is a forest, then S is a *feedback vertex set* in G. If S is a feedback vertex set and S is independent, then S is an *independent feedback vertex set*. If S is an independent feedback vertex set and there exists no $X \subset S$ such that X is an independent feedback vertex set, then S is a *minimal independent feedback vertex set* in G.

2.3.9 Vertex Covers

Let G be a graph, and let $S \subseteq V(G)$. Let $(u, v) \in E(G)$. If $u \in S$ or $v \in S$, then we will say that S *covers* the edge (u, v). Let $Z \subseteq E(G)$. If S covers every edge $(u, v) \in Z$, then S covers Z. If S covers $E(G)$, then S is a *vertex cover* in G. That is, a vertex cover in a graph G is a set $S \subseteq V(G)$ such that for every $(u, v) \in E(G)$ we have $u \in S$ or $v \in S$. If S is a vertex cover in G and $G[S]$ is connected, then S is a *connected vertex cover* in G. If S is a connected vertex cover in G and there exists no $X \subset S$ such that X is a connected vertex cover in G, then S is a *minimal connected vertex cover* in G.

2.4 Computational Problems

In this section we introduce three types of computational problems: decision problems, search problems and enumeration problems. Before doing so, we provide some preliminary definitions. An *alphabet* Σ is a non-empty finite set of symbols, and a *string* over Σ is a sequence of symbols from Σ. If w is a string, then $|w|$ denotes the *length* of w, that is, the number of symbols in the sequence. By Σ^*, we denote the set of all possible strings over Σ. A *language* over Σ is a set of strings $L \subseteq \Sigma^*$.

A *decision problem* is a language L over some alphabet Σ. Let $L \subseteq \Sigma^*$ be a decision problem, and let $w \in \Sigma^*$ be some string. If $w \in L$, then we say that w is a *yes instance* of L, otherwise we say that w is a *no instance* of L. A *search problem* is a function $f : \Sigma^* \to \mathcal{P}(\Sigma^*)$. Likewise, we define an *enumeration problem* as a function $f : \Sigma^* \to \mathcal{P}(\Sigma^*)$. Note that although the definition of a search problem is the same as the definition of an enumeration problem, we will be using these terms in different contexts. In particular, we will see that *solving* a search problem is not the same as solving an enumeration problem. We will use the following terminology when discussing search problems or enumeration problems. Let f be a search problem or an enumeration problem, and let $w \in \Sigma^*$ such that $f(w)$ is non-empty. We call $f(w)$ the set of *feasible solutions* of w, and we call a member s of $f(w)$ a *solution* of w.

2.5 Algorithms

In this section we discuss what it means to solve the various computational problems introduced in Section 2.4, and we introduce algorithms, along with algorithmic complexity, and finally, polynomial delay in the context of enumeration algorithms.

The notion of *algorithms* can be formalized by a mathematical model of computation known as the *Turing machine*. However, for our purposes it will be sufficient to think of an algorithm as a procedure for solving some computational problem. We refer the reader to [14] for an introduction to Turing machines. Our algorithms accept as input some string w, called the *instance*, and yield some output upon halting. Note that we can encode any object as a string. Thus, by *instance*, we often mean the object represented by the input string. We will use the terms *instance*, *problem instance* and *input instance* interchangeably. Let $\mathcal{A}$ be an algorithm, and let w be a string, then $\mathcal{A}(w)$ denotes both the application of $\mathcal{A}$ to w, and the output of $\mathcal{A}$ when applied to w.

What it means to *solve* a problem naturally depends on what that problem is. We will consider what it means to solve a decision problem, a search problem, and an enumeration problem, respectively. An algorithm for a decision problem is typically called a *decider*. Given as input some instance w, a decider outputs either *yes* or *no*. Formally, a decider $\mathcal{A}$ of a decision problem $L \subseteq \Sigma^*$ solves L if for any instance $w \in \Sigma^*$, $\mathcal{A}$ outputs either *yes* or *no*, and it holds that w is a yes instance of L if and only if $\mathcal{A}$ outputs *yes* when applied to w.

An algorithm for a search problem $f : \Sigma^* \to \mathcal{P}(\Sigma^*)$, is an algorithm $\mathcal{A}$ that given an instance $w \in \Sigma^*$ outputs some $s \in \Sigma^*$. Formally, $\mathcal{A}$ solves f if for any instance $w \in \Sigma^*$ such that $f(w) \neq \emptyset$, $\mathcal{A}(w) \in f(w)$, and for any instance $w \in \Sigma^*$ such that $f(w) = \emptyset$, $\mathcal{A}$ concludes that w has no solution.

An algorithm for an enumeration problem is often called an *enumerator*. On encountering a feasible solution of the input instance, an enumerator typically outputs the solution immediately. However, for our purposes it will be more convenient to think of an enumerator as an algorithm that outputs the set of all feasible solutions of the input instance. That is, given as input some instance $w \in \Sigma^*$, an enumerator $\mathcal{A}$ for an enumeration problem $f : \Sigma^* \to \mathcal{P}(\Sigma^*)$ will output some $S \in \mathcal{P}(\Sigma^*)$. Whereas it is sufficient that an algorithm for a search problem finds *one* solution of the input instance, an algorithm for an enumeration problem must find *all* solutions of the input instance. Formally, $\mathcal{A}$ solves f if for any instance $w \in \Sigma^*$, $\mathcal{A}(w) = f(w)$. If $\mathcal{A}$ solves f, then we say that $\mathcal{A}$ *enumerates* f.

2.5.1 Algorithmic Complexity

The *complexity* of an algorithm $\mathcal{A}$ is the amount of computational resources $\mathcal{A}$ requires when applied to some instance, in the worst-case scenario. We distinguish between the *space complexity* and the *time complexity* of an algorithm. Let $\mathcal{A}$ be an algorithm. The space complexity of $\mathcal{A}$ is a function $f : \mathbb{N} \to \mathbb{N}$, where $f(n)$ is the maximum amount of computer memory used by $\mathcal{A}$ when applied to any instance of length n. Similarly, the time complexity of $\mathcal{A}$ is a function $g : \mathbb{N} \to \mathbb{N}$, where $g(n)$ is the maximum number of elementary computer

instructions executed by $\mathcal{A}$ when applied to any instance of length n.

Let $f, g : \mathbb{R} \to \mathbb{R}^+$ such that $f(n) = \mathcal{O}(g(n))$, and let $\mathcal{A}$ be an algorithm with time complexity $f(n)$, then we say that $\mathcal{A}$ runs in time bounded by $g(n)$, or that the running time of $\mathcal{A}$ is bounded by $g(n)$.

2.5.2 Polynomial Delay

Let $f : \Sigma^* \to \mathcal{P}(\Sigma^*)$ be an enumeration problem, and let $\mathcal{A}$ be an algorithm that enumerates f. In the following, we let s_k denote the k'th solution output by $\mathcal{A}$. $\mathcal{A}$ is said to have *polynomial delay* if for any instance w where $f(w) \neq \emptyset$, it holds that

i. $\mathcal{A}$ outputs the first feasible solution $s_1 \in f(w)$ within time bounded by a polynomial in $|w|$

ii. If $s_{i-1}, s_i \in f(w)$, then the time that elapses between when s_{i-1} and s_i is output is bounded by a polynomial in $|w|$

iii. $\mathcal{A}$ outputs no solution more than once

2.6 Branching Algorithms

In this section we give a brief introduction to the paradigm of *Branch & Reduce* in the context of enumeration problems. We discuss what comprises a branching algorithm for an enumeration problem, how to prove the correctness of such algorithms, and how a branching algorithm for a particular enumeration problem can be used to give an upper bound on the number of solutions of a problem instance. We refer the reader to the book of Fomin and Kratsch [15] for a more detailed introduction to branching algorithms.

Branching is an algorithmic technique in which an instance is solved by decomposing the instance into subproblems, recursively solving these subproblems, and then combining the solutions of these subproblems into a solution of the original instance. As branching is based on recursion, it is paramount that the subproblems we produce are *smaller* than the original instance, otherwise the algorithm could potentially never halt. The *measure* of an instance I is a function $\mu : \Sigma^* \to \mathbb{N}$ that is bounded by some function of natural parameters of I. It is, in essence, a measure of the size of an instance. We describe branching algorithms in terms of *reduction rules*, *halting rules* and *branching rules*. In the following, let $f : \Sigma^* \to \mathcal{P}(\Sigma^*)$ be an enumeration problem.

A *reduction rule* is a computable function which takes as input an instance I and outputs a *reduced* instance I'. By reduced, we mean that $\mu(I') < \mu(I)$. Recall that $f(I)$ and $f(I')$ is the set of feasible solutions of I and the set of feasible solutions of I', respectively. We say that $\mathcal{R}$ is *safe* if there exists a function $g : \Sigma^* \to \Sigma^*$ such that $f(I) \subseteq g(f(I'))$, where $g(f(I'))$ is the image of $f(I')$ under g. A *halting rule* $\mathcal{H}$ is a computable function which takes as input an instance I and outputs some $\mathcal{H}(I) \in \mathcal{P}(\Sigma^*)$. $\mathcal{H}$ is correct if $\mathcal{H}(I) = f(I)$.

A *branching rule* $\mathcal{B}$ is a computable function that given an instance I outputs $r \geq 2$ subproblems $I_1, I_2, \ldots, I_r$ such that r is bounded by $\mu(I)$ and for $i = 1, 2, \ldots, r$ we have $\mu(I_i) < \mu(I)$. We say that $\mathcal{B}$ is *safe* if there exists functions $g_i : \Sigma^* \to \Sigma^*$, such that $f(I) \subseteq \bigcup_{i=1}^{r} g_i(f(I_i))$. In this thesis, all branching rules will have $r \leq 4$.

Given as input some instance I, a branching algorithm either applies a reduction rule, a halting rule, or a branching rule. When the algorithm applies a halting rule, it solves the input instance outright and outputs $f(I)$. When the algorithm applies a reduction rule or a branching rule, the algorithm first generates r subproblems $I_1, I_2, \ldots, I_r$, where $r = 1$ when a reduction rule is applied, and $r \geq 2$ when a branching rule is applied. After generating the subproblem(s), the algorithm calls itself recursively on $I_1, I_2, \ldots, I_r$, obtaining $f(I_1), f(I_2), \ldots, f(I_r)$. It then computes $\zeta = \bigcup_{i=1}^{r} g_i(f(I_i))$, and thereafter outputs $f(I) = \{s \mid P(s), s \in \zeta\}$, where the predicate $P(s)$ holds if and only if $s \in f(I)$.

When we discuss a branching algorithm, we will refer to the reduction rules, halting rules and branching rules of the algorithm collectively as the *rules* of the algorithm. The correctness of a branching algorithm follows from the safeness of its reduction rules and branching rules, and the correctness of its halting rules, together with a proof that for any instance I, some rule of the the branching algorithm will apply to I.

2.6.1 Upper Bounding the Number of Solutions of an Instance

We shall now describe an approach to upper bounding the number of solutions $|f(I)|$ of any problem instance I, for any enumeration problem $f : \Sigma^* \to \Sigma^*$. We will need the following lemma.

Lemma 2.1. *Let $f : \Sigma^* \to \Sigma^*$ be an enumeration problem, and let $\mathcal{A}$ be a branching algorithm which enumerates f. We assume that all halting rules of $\mathcal{A}$ are correct, and that all reduction rules and branching rules of $\mathcal{A}$ are safe. Further, we assume that for any instance I, some rule of $\mathcal{A}$ is applicable to I. Suppose that there exists $x, a \in \mathbb{R}$ such that $x, a \geq 1$ and the following holds.*

a) For every branching rule $\mathcal{B}$ of $\mathcal{A}$, if $I_1, I_2, \ldots, I_r$ are the subproblems output by $\mathcal{B}$ when applied to an instance I, then there exists integer constants $d_i > 0$ such that $\mu(I_i) \leq \mu(I) - d_i$, for $i = 1, 2, \ldots, r$, and $\Sigma_{i=1}^{r} x^{-d_i} \leq 1$.

b) For every instance I to which some halting rule of $\mathcal{A}$ applies, it holds that $|f(I)| \leq ax^{\mu(I)}$.

Under this assumption, we have $|f(I)| \leq ax^{\mu(I)}$, for any instance I.

Proof. Let $\mathcal{I}$ be the set of all instances of f, and let

$$f(\mu) = max\{|f(I)| \mid \mu(I) \leq \mu, I \in \mathcal{I}\}$$

In order to prove our claim, we will prove by induction on μ that $f(\mu) \leq ax^{\mu}$. We note that for any instance $I \in \mathcal{I}$, we have $|f(I)| \leq f(\mu(I))$. Since we assume that $|f(I)| \leq ax^{\mu(I)}$ whenever I is an instance to which some halting rule applies, the base case holds. Suppose that for every $l < \mu$, we have $f(l) \leq ax^{l}$. Let I be an instance of measure $\mu(I) = \mu$.

Case 1: Some reduction rule $\mathcal{R}$ applies to I. Let $I' = \mathcal{R}(I)$ be the reduced instance, then $\mu(I') < \mu$. Since $\mathcal{R}$ is safe, we have $|f(I)| \leq |f(I')|$. Thus, by the induction hypothesis,

$$|f(I)| \leq |f(I')| \leq ax^{\mu(I')} \leq ax^{\mu}$$

Case 2: Some halting rule $\mathcal{H}$ applies to I. By assumption, $|f(I)| \leq ax^{\mu}$.

Case 3: Some branching rule $\mathcal{B}$ applies to I. Suppose $\mathcal{B}$ outputs the subproblems $I_1, I_2, \ldots, I_r$ when applied to I. Since $\mathcal{B}$ is safe, we have

$$\begin{aligned} |f(I)| &\leq |f(I_1)| + |f(I_2)| + \ldots + |f(I_r)| \\ &\leq \Sigma_{i=1}^{r} ax^{\mu(I_i)} \\ &\leq \Sigma_{i=1}^{r} ax^{\mu(I_i)} \cdot x^{\mu} \cdot x^{-\mu} \\ &\leq ax^{\mu} \cdot \Sigma_{i=1}^{r} x^{\mu(I_i)-\mu} \end{aligned}$$

Recall that we assume that there exists $d_i > 0$ such that $\mu - \mu(I_i) \geq d_i$, for $i = 1, 2, \ldots, r$, and that $\Sigma_{i=1}^{r} x^{-d_i} \leq 1$. Thus,

$$|f(I)| \leq ax^{\mu} \cdot \Sigma_{i=1}^{r} x^{\mu(I_i)-\mu} \leq ax^{\mu} \cdot \Sigma_{i=1}^{r} x^{-d_i} \leq ax^{\mu}$$

In every case we have $|f(I)| \leq ax^{\mu}$. Since I was an arbitrary instance of measure μ, it holds that $|f(I)| \leq ax^{\mu}$ whenever I is an instance that attains $|f(I)| = f(\mu)$. Thus, $f(\mu) \leq ax^{\mu}$, and the proof is complete. □

Branching vectors and their branching factors

Let $d_i > 0$ be constants, for $i = 1, 2, \ldots r$. Let $\mathcal{B}$ be a branching rule that generates $r \geq 2$ subproblems of measure at most $\mu - d_1, \mu - d_2, \ldots, \mu - d_r$ when applied to any instance of measure $\mu \geq max\{d_1, d_2, \ldots, d_r\}$. We call $(d_1, d_2, \ldots, d_r)$ the *branching vector* of $\mathcal{B}$. We define the *branching factor* of a branching vector $(d_1, d_2, \ldots, d_r)$ to be

$$\tau(d_1, d_2, \ldots, d_r) = min\{\alpha \mid \Sigma_{i=1}^{r} \alpha^{-d_i} \leq 1 \wedge \alpha \geq 1, \alpha \in \mathbb{R}\}$$

Observe that $\tau(d_1, d_2, \ldots, d_r)$ is the unique positive real root of

$$x^{-d_1} + x^{-d_2} + \ldots + x^{-d_r} = 1$$

When a branching algorithm applies a branching rule with branching vector (i, j), we say that the algorithm (i, j)-*branches*.

An approach to upper bounding the number of solutions of an enumeration problem

From Lemma 2.1, we obtain the following approach to upper bounding the number of solutions of a problem instance, for any enumeration problem f. First, we describe a branching algorithm $\mathcal{A}$ which enumerates f. We then show that all reduction rules and branching rules are safe, and that all halting rules are correct, and further, that for any instance I, some rule of $\mathcal{A}$ is applicable to I. Thereafter we analyze each branching rule separately and obtain a system B of branching vectors. Then we obtain an upper bound $\beta^{\mu(I)}$ on the number of solutions output by any halting rule when applied to an instance I. Finally, we choose

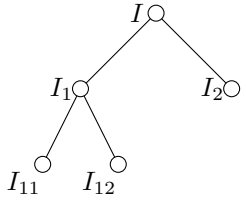
$$x = max\left\{max\{\tau(\vec{b}) \mid \vec{b} \in B\}, \beta\right\}$$

By Lemma 2.1, we then have that $|f(I)| \leq x^{\mu(I)}$, for any instance I of f. We will apply this approach in chapters 3 and 4, when we upper bound the number of minimal connected vertex covers in a graph, and the number of minimal independent feedback vertex sets in a graph, respectively.

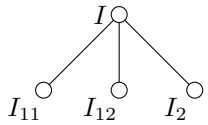
Addition of branching vectors

Let $\mathcal{A}$ be a branching algorithm, and let $\mathcal{B}_1$ and $\mathcal{B}_2$ be branching rules with corresponding branching vectors (i, j) and (k, l), respectively. Let I be an instance to which $\mathcal{B}_1$ applies, and let I_1 and I_2 be the subproblems output by $\mathcal{B}_1$ when applied to I, then $\mu(I_1) \leq \mu(I) - i$, and $\mu(I_2) \leq \mu(I) - j$. Consider an application of $\mathcal{B}_2$ to I_1, and let I_{11} and I_{12} be the subproblems output by $\mathcal{B}_2$, then $\mu(I_{11}) \leq \mu(I_1) - k \leq \mu(I) - (i + k)$, and $\mu(I_{12}) \leq \mu(I_1) - l \leq \mu(I) - (i + l)$.

Suppose $\mathcal{A}$ always applies $\mathcal{B}_2$ to I_1. We then obtain a branching rule $\mathcal{B}'$ which outputs the subproblems I_{11}, I_{12} and I_2 when applied to I. The branching vector corresponding to $\mathcal{B}'$ is $\vec{b'} = (i+k, i+l, j)$. We note that we could explicitly add the new rule $\mathcal{B}'$, and remove the rule $\mathcal{B}_1$, from the set of branching rules of $\mathcal{A}$. However, we shall not be adding $\mathcal{B}'$ to the set of branching rules of $\mathcal{A}$, rather, we shall only add the branching vector $\vec{b'}$ corresponding to $\mathcal{B}'$ to the system of branching vectors of $\mathcal{A}$. We call this *addition of branching vectors*. We illustrate addition of branching vectors in Fig. 2.1.



(a) $\mathcal{A}$ applies $\mathcal{B}_1$ to I, generating the subproblems I_1 and I_2. $\mathcal{A}$ then always applies $\mathcal{B}_2$ to I_1, generating the subproblems I_{11} and I_{12}.



(b) An application of $\mathcal{B}'$ to I generates the subproblems I_{11}, I_{12} and I_2.

Figure 2.1: Addition of branching vectors.

Chapter 3

Improved Combinatorial Bounds for the Connected Vertex Cover Problem

3.1 Introduction

In this chapter we present an improved upper bound on the maximum number of minimal connected vertex covers in a graph.

Definition 3.1. *Let G be a graph, and let $M \subseteq V(G)$. Let S be a connected vertex cover in G. If $M \subseteq S$, then S is a M-cvc in G.*

If S is a M-cvc in G and there is no $X \subset S$ such that X is a M-cvc in G, then S is a *minimal* M-cvc in G. Observe that a minimal M-cvc is not necessarily a minimal connected vertex cover in G. To see this, let $G = (\{u, v\}, \{(u, v)\})$ be a graph, and note that $\{u, v\}$ is a minimal $\{u, v\}$-cvc in G, but not a minimal connected vertex cover in G, since both $\{u\}$ and $\{v\}$ are connected vertex covers in G. We will let $C_G(M)$ denote the set of all *minimal* M-cvc in G. We note that $C_G(\emptyset)$ is then the set of *all* minimal connected vertex covers in G.

Let G be a graph. If $E(G) = \emptyset$, then $C_G(\emptyset) = \{\emptyset\}$. That is, G has exactly one minimal connected vertex cover, the empty set. Otherwise, let $(u, v) \in E(G)$ be any edge in G. If $S \in C_G(\emptyset)$, then we must have $u \in S$ or $v \in S$, thus

$$C_G(\emptyset) \subseteq C_G(\{u\}) \cup C_G(\{v\})$$

We make the following observation.

Observation 3.2. *Let a and c be constants such that $c \geq 1$ and $a > 0$, and let G be a graph on n vertices. If $|C_G(M)| \leq ac^n$ whenever $M \neq \emptyset$, then $|C_G(\emptyset)| \leq 2ac^n$.*

We claim that for any graph G on n vertices and $M \subseteq V(G)$ such that $M \neq \emptyset$, we have $|C_G(M)| \leq 1.7076^n$. By Observation 3.2, if our claim holds, then there are at most $2 \cdot 1.7076^n$ minimal connected vertex covers in any n-vertex graph.

3.2 Improving the Upper Bound

In order to prove our claim, we will apply the approach described in Section 2.6.1. We now establish some notation which we will be using throughout this chapter. We adopt the notation of Agrawal et al. [2]. Let G be a graph, and let $M \subseteq V(G)$. We let $U = V(G) \setminus M$. We shall refer to vertices in M as *marked* vertices, and to vertices in U as *unmarked* vertices, and we shall refer to edges for which both endpoints are marked as *marked* edges. We let u_v and m_v denote the number of unmarked neighbors of v and the number of marked neighbors of v, respectively. That is, $m_v = |N(v) \cap M|$, and $u_v = |N(v)| - m_v$. We define the measure of an instance $I = (G, M)$ to be

$$\mu(I) = \mu(G, M) = w_1 |U| + w_2 |M|$$

where $w_1, w_2 \in \mathbb{N}$ such that $w_1 > w_2 > 0$. Observe that

$$\begin{aligned} \mu(I) &= w_1 |U| + w_2 |M| \\ &\leq w_1(|U| + |M|) \\ &\leq w_1 |V(G)| \end{aligned}$$

In the following we will always assume that $M \neq \emptyset$. We will now construct a branching algorithm which enumerates all M-cvc of any graph G, for non-empty $M \subseteq V(G)$. We shall call this algorithm EnumMcvc. We apply the rules of EnumMcvc in ascending order, according to the numbering of the rules. That is, we always attempt to apply rule i before applying rule $j > i$. Therefore, if we apply rule j to some instance (G, M), then we can always assume that rule $i < j$ does not apply to (G, M).

3.2.1 Reduction Rules and Halting Rules

We will now describe the reduction rules and the halting rules of EnumMcvc.

Reduction Rule 3.1. *If there exists some vertex $v \in U$ such that $d(v) = 0$, then replace the instance (G, M) with the instance $(G - \{v\}, M)$.*

Reduction Rule 3.1 removes a single unmarked vertex from the graph, and thus reduces the measure by w_1.

Halting Rule 3.2. *If G is disconnected, then output $\emptyset$.*

Reduction Rule 3.3. *If there exists some edge $(u, v) \in E(G)$ such that $u, v \in M$, then let $G' = G/(u, v)$, and let $M' = (M \setminus \{u, v\}) \cup \{w\}$, where w is the vertex obtained by contracting (u, v), and replace the instance (G, M) with the instance (G', M').*

With respect to Reduction Rule 3.3, when we contract an edge e, we shall *not* keep any multi-edges or self-loops obtained by contracting e. Since Reduction Rule 3.3 contracts a single marked edge, an application of Reduction Rule 3.3 will reduce the measure by w_2.

Observation 3.3. *If (G, M) is an instance to which none of rules 3.1-3.3 apply, then M is an independent set.*

Halting Rule 3.4. *If $M = V(G)$, then output M.*

Reduction Rule 3.5. *If there exists some vertex $v \in U$ such that $G - \{v\}$ is disconnected, then replace the instance (G, M) with the instance $(G, M \cup \{v\})$.*

An application of Reduction Rule 3.5 will reduce the measure by $w_1 - w_2 > 0$, since Reduction Rule 3.5 marks a single unmarked vertex.

Observation 3.4. *If (G, M) is an instance to which none of rules 3.1-3.5 apply, then $G - \{v\}$ is connected, for every $v \in U$.*

Reduction Rule 3.6. *If there exists some vertex $v \in U$ such that $d(v) = 1$, then replace the instance (G, M) with the instance $(G - \{v\}, M)$.*

Reduction Rule 3.6 removes a single unmarked vertex from the graph. An application of Reduction Rule 3.6 therefore reduces the measure by w_1.

We shall now prove that reduction rules 3.1, 3.3, 3.5 and 3.6 are safe, and that halting rules 3.2 and 3.4 are correct. Let (G, M) be some instance, and let $\mathcal{R}$ be some reduction rule. Let $(G', M') = \mathcal{R}(G, M)$ be the instance obtained by applying $\mathcal{R}$ to (G, M). In order to show that $\mathcal{R}$ is safe, we must show that there exists a function g such that for any $S \in C_G(M)$, there exists a solution $S' \in C_{G'}(M')$ such that $g(S') = S$.

Lemma 3.5. *Reduction Rule 3.1 is safe.*

Proof. Let (G, M) be an instance to which Reduction Rule 3.1 applies, and let $v \in U$ be the vertex we remove from the graph, then $(G - \{v\}, M)$ is the instance output by Reduction Rule 3.1 when applied to (G, M). We let $g(S) = S$ be

the function which maps a solution of the reduced instance to a solution of the input instance. Let $S \in C_G(M)$, and let $G' = G - \{v\}$. In order to show that Reduction Rule 3.1 is safe, we must show that $S \in C_{G'}(M)$ as well. Observe that since $E(G') = E(G)$, S is a M-cvc in G'. Suppose $S \notin C_{G'}(M)$, and let $X \subset S$ be a M-cvc in G'. Again, since $E(G') = E(G)$, X is a M-cvc in G, contradicting our assumption that S is a minimal M-cvc in G. We see that if $S \in C_G(M)$, then $S \in C_{G'}(M)$ as well, and Reduction Rule 3.1 is therefore safe. □

Lemma 3.6. *Halting Rule 3.2 is correct.*

Proof. Let (G, M) be an instance to which Halting Rule 3.2 applies, then G is disconnected. We must show that $C_G(M) = \emptyset$. Suppose not, and let S be a minimal M-cvc in G. Let $C \subseteq V(G)$ be a connected component in G, and let $v \in C$. If $v \in U$, then $d(v) \geq 1$, as otherwise Reduction Rule 3.1 would have applied to (G, M). Thus, if $v \in U$, then there must be some edge $(u, v) \in E(G[C])$ which S covers. That is, if $v \in U$, we must have $S \cap C \neq \emptyset$. If $v \in M$, then $v \in S$, since $M \subseteq S$. Therefore, if $v \in M$, we must have $S \cap C$ as well. Let C_1 and C_2 be two connected components in G, then $S \cap C_1 \neq \emptyset$ and $S \cap C_2 \neq \emptyset$. However, this implies that $G[S]$ is disconnected, which contradicts our assumption that S is a M-cvc in G. We conclude that $C_G(M) = \emptyset$. □

Lemma 3.7. *Reduction Rule 3.3 is safe.*

Proof. Let (G, M) be an instance to which Reduction Rule 3.3 applies. Suppose $(u, v) \in E(G)$ is the edge we contract, then $u, v \in M$. Let w be the vertex obtained by contracting the edge (u, v), then

$$(G', M') = (G/(u, v), (M \setminus \{u, v\}) \cup \{w\})$$

is the instance obtained by contracting (u, v). Let $g(S') = (S' \setminus \{w\}) \cup \{u, v\}$. We will show that for every $S \in C_G(M)$ there exists some $S' \in C_{G'}(M')$ such that $g(S') = S$. Let $S \in C_G(M)$, and note that since $u, v \in M$ and $M \subseteq S$, we have $u, v \in S$. Let $S' = (S \setminus \{u, v\}) \cup \{w\}$, then $g(S') = S$. In order to show that Reduction Rule 3.3 is safe, we must show that $S' \in C_{G'}(M')$.

We first introduce some auxiliary notation. Let $E_X(H)$ denote the set of edges in the graph H which are incident to some vertex in $X \subseteq V(H)$. That is, we let

$$E_X(H) = \{(x, y) \mid \{x, y\} \cap X \neq \emptyset, (x, y) \in E(H)\}$$

We note that

$$E(G) \cap E(G') = E(G) \setminus E_{\{u,v\}}(G) = E(G') \setminus E_{\{w\}}(G')$$

We will now show that $S' \in C_{G'}(M')$. Note that since S covers $E(G)$, the set $(S \setminus \{u, v\}) \cup \{w\}$ covers

$$(E(G) \setminus E_{\{u,v\}}(G)) \cup E_{\{w\}}(G') = E(G')$$

That is, S' covers $E(G')$. Since $G'[S']$ is obtained by contracting an edge in the connected graph $G[S]$, we have that $G'[S']$ is connected as well. Finally, since $M \subseteq S$, we have $M' \subseteq S'$, and S' is thus a M'-cvc in G'. Suppose $S' \notin C_{G'}(M')$, then there must be some $X' \subset S'$ such that X' is a M'-cvc in G'. Let $X = g(X') = (X' \setminus \{w\}) \cup \{u, v\}$, and note that when $X' \subset S'$, we have $X \subset S$. We will show that X is a M-cvc in G, contradicting the minimality of S. Since X' covers $E(G')$, the set $(X' \setminus \{w\}) \cup \{u, v\}$ covers

$$(E(G') \setminus E_{\{w\}}(G')) \cup E_{\{u,v\}}(G) = E(G)$$

Thus, X is a vertex cover in G. Since $G'[X']$ is connected, then so is $G[X]$, and lastly, since $M' \subseteq X'$, we have $M \subseteq X$. We see that X is a M-cvc in G, which is not possible, since S is a minimal M-cvc in G. Our assumption that $S' \notin C_{G'}(M')$ leads to a contradiction, and we therefore conclude that $S' \in C_{G'}(M')$, and thus that Reduction Rule 3.3 is safe. □

Lemma 3.8. *Halting Rule 3.4 is correct.*

Proof. Let (G, M) be an instance to which Halting Rule 3.4 applies, we then have $M = V(G)$. We must show that $C_G(M) = \{V(G)\}$. Observe that since Halting Rule 3.2 does not apply to (G, M), we know that G is connected, and therefore also that M is a M-cvc in G. Clearly, there is no $S \subset M$ such that $M \subseteq S$, thus $M = V(G)$ is a minimal M-cvc in G. We have $C_G(M) = \{V(G)\}$, and Halting Rule 3.4 is therefore correct. □

Claim 3.9. *Let (G, M) be an instance to which none of rules 3.1-3.4 apply, then $|V(G)| \geq 2$ and $\delta(G) \geq 1$.*

Proof. Since Halting Rule 3.2 does not apply to (G, M), we know that G is a connected graph. Recall that we always assume that $M \neq \emptyset$. Since Halting Rule 3.4 does not apply to (G, M), we know that $M \subset V(G)$, and therefore that $U \neq \emptyset$. Since $U \neq \emptyset$ and $M \neq \emptyset$, we have that $|V(G)| \geq 2$. Let $v \in U$, and let $u \in M$. Since G is connected, there is a path P between u and v. The existence of P implies that $d(v) \geq 1$, and that $d(u) \geq 1$. Since we have $d(v) \geq 1$ for every $v \in U$, and $d(u) \geq 1$ for every $u \in M$, we have $\delta(G) \geq 1$. Thus, $|V(G)| \geq 2$, and $\delta(G) \geq 1$, and the claim holds. □

Lemma 3.10. *Let G be a connected graph, and let S be a connected vertex cover in G. If there is some $v \in V(G)$ such that $G - \{v\}$ is disconnected, then $v \in S$.*

Proof. Suppose not, and let C_1 and C_2 be two connected components in $G - \{v\}$. Let $u \in C_1$ such that $(u, v) \in E(G)$, and let $w \in C_2$ such that $(v, w) \in E(G)$. Since $v \notin S$, we must have $u, w \in S$, otherwise S would not cover the edges (u, v) and (v, w). However, there is no path between u and v in $G[S]$, thus $G[S]$ is not connected, a contradiction. □

Lemma 3.11. *Reduction Rule 3.5 is safe.*

Proof. Let (G, M) be an instance to which Reduction Rule 3.5 applies. Suppose $v \in U$ is the vertex we mark, then $(G, M \cup \{v\})$ is the instance output by Reduction Rule 3.5. We let $g(S) = S$ be the function which maps a solution of the reduced instance to a solution of the input instance. Let $S \in C_G(M)$, and let $M' = M \cup \{v\}$. In order to show that Reduction Rule 3.5 is safe, we must show that $S \in C_G(M')$ as well. We will first show that S is a M'-cvc in G. Since G is already a connected vertex cover in G by assumption, we need only show that $M' \subseteq S$. We know that $v \in S$, by Lemma 3.10. Since $M \subseteq S$ and $v \in S$, we have $M' \subseteq S$ as well, and S is thus a M'-cvc in G. Suppose that $S \notin C_G(M')$, and let $X \subset S$ be a M'-cvc in G. X must exist, since S is not minimal. However, when X is a M'-cvc in G, X is also a M-cvc in G, since $M \subseteq M'$. Thus, our assumption that $S \notin C_G(M')$ contradicts our assumption that $S \in C_G(M)$. We conclude that when $S \in C_G(M)$, we have $S \in C_G(M')$ as well, and Reduction Rule 3.5 is therefore safe. □

Lemma 3.12. *Let $I = (G, M)$ be an instance to which none of rules 3.1-3.5 apply. If there is some $v \in U$ such that $N(v) = \{u\}$, then $u \in M$.*

Proof. First, observe that since Halting Rule 3.2 does not apply to I, we know that G is connected. Now, suppose $u \notin M$. We then have $u, v \in U$. Since we always assume that $M \neq \emptyset$, and since $u, v \in U$, we know that there must be some vertex $w \in V(G) \setminus \{u, v\}$. Since G is connected, and since $N(v) = \{u\}$, we therefore know that $d(u) \geq 2$. However, if $d(u) \geq 2$, then $G - \{u\}$ is disconnected. Since Reduction Rule 3.5 does not apply to I, we know that there can be no $x \in U$ such that $G - \{x\}$ is disconnected. Hence, since $G - \{u\}$ is disconnected, we cannot have $u \in U$, which contradicts our assumption that $u \notin M$. Thus, $u \in M$, and the proof is complete. □

Lemma 3.13. *Reduction Rule 3.6 is safe.*

Proof. Let (G, M) be an instance to which Reduction Rule 3.6 applies. Suppose $v \in U$ is the vertex we remove from the graph, and let $u \in N(v)$. $(G - \{v\}, M)$ is the instance output by Reduction Rule 3.6. We let $g(S) = S$ be the function which maps a solution of the reduced instance to a solution of the input instance. Let $S \in C_G(M)$, and let $G' = G - \{v\}$. In order to show that Reduction Rule 3.6 is safe, we must show that $S \in C_{G'}(M)$ as well. We will first show that S is a M-cvc in G'. Since $E(G') \subset E(G)$, and since S covers $E(G)$, S covers $E(G')$ as well. Further, since $G[S]$ is connected, then so is $G'[S]$, and S is therefore a M-cvc in G'. Suppose $S \notin C_{G'}(M)$, then there must be some $X \subset S$ such that X is a M-cvc in G'. Observe that $E(G) = E(G') \cup \{(u, v)\}$. We know that X covers $E(G')$. In order to show that X covers $E(G)$, we must show that X covers (u, v) as well. Note that by Lemma 3.12, we have $u \in M$, and therefore $u \in X$. Thus, X covers $E(G)$. Since $G'[X]$ is connected, $G[X]$ is connected as

well, and X is therefore a M-cvc in G, which contradicts the minimality of S. We conclude that when $S \in C_G(M)$, we must have $S \in C_{G'}(M)$, and Reduction Rule 3.6 is therefore safe. □

Claim 3.14. *Let $I = (G, M)$ be an instance to which none of rules 3.1-3.6 apply, then $\delta(G) \geq 2$.*

Proof. Suppose not, then there must be some $v \in V(G)$ such that $d(v) = 1$, since $\delta(G) \geq 1$, by Claim 3.9. Observe that since Reduction Rule 3.6 does not apply to I, there exists no $x \in U$ such that $d(x) = 1$. Since $\delta(G) \geq 1$, we therefore know that $d(x) \geq 2$ whenever $x \in U$. Since $d(v) = 1$, we must have $v \in M$. Let $u \in N(v)$. By Observation 3.3, M is independent. Thus, since $v \in M$, we must have $u \in U$. We established that $d(x) \geq 2$, for any $x \in U$. We therefore know that $d(u) \geq 2$. However, observe that if $d(u) \geq 2$, then $G - \{u\}$ is disconnected. Since Reduction Rule 3.5 does not apply to I, we must have $u \in M$. We have $u \in U$ and $u \in M$, and $U \cap M = \emptyset$, a contradiction. □

3.2.2 Branching Rules

Let $I = (G, M)$ be an instance, and let $v \in U$. When we *branch* on v, we output two subproblems I_1 and I_2, where

$$I_1 = (G, M \cup \{v\})$$
$$I_2 = (G - \{v\}, M \cup N(v))$$

We shall refer to I_1 as the subproblem obtained by *selecting* v, and to I_2 as the subproblem obtained by *discarding* v. We will now describe the branching rules of EnumMcvc.

Branching Rule 3.7. *If there exists some vertex $v \in U$ such that $d(v) \geq 3$ and $m_v > 0$, then branch on v.*

Branching Rule 3.8. *If there exists some vertex $v \in U$ such that $d(v) \geq 3$ and there is some $u \in N(v)$ such that $d(u) = 2$, then branch on v.*

Branching Rule 3.9. *If there exists some $v \in U$ such that $d(v) = 2$ and $m_v = 1$, then branch on v.*

Lemma 3.15. *Branching rules 3.7, 3.8 and 3.9 are safe.*

Proof. Let $\mathcal{B}$ be one of branching rules 3.7, 3.8 or 3.9. Let $I = (G, M)$ be an instance to which $\mathcal{B}$ applies, and let $v \in U$ be the vertex we branch on, then $I_1 = (G, M \cup \{v\})$ and $I_2 = (G - \{v\}, M \cup N_G(v))$ are the subproblems output by $\mathcal{B}$, corresponding to selecting v and discarding v, respectively. We let $g_1(S) = S$ be the function which maps a solution of I_1 to a solution of I, and $g_2(S) = S$

be the function which maps a solution of I_2 to a solution of I. Let $S \in C_G(M)$. We must show that $S \in C_G(M \cup \{v\})$ or $S \in C_{G-\{v\}}(M \cup N_G(v))$. We consider two scenarios:

Case 1: $v \in S$**.** If $v \in S$, then S is a $(M \cup \{v\})$-cvc in G. We must show that S is a *minimal* $(M \cup \{v\})$-cvc in G. Suppose not, and let $X \subset S$ such that X is a $(M \cup \{v\})$-cvc in G. Since X is a $(M \cup \{v\})$-cvc in G, X certainly is a M-cvc in G, which contradicts the minimality of S. We must have $S \in C_G(M \cup \{v\})$.

Case 2: $v \notin S$**.** If $v \notin S$, then we must have $N_G(v) \subseteq S$, otherwise S would not be a vertex cover in G. Since $N_G(v) \subseteq S$, we have $M \cup N_G(v) \subseteq S$, and S is therefore a $(M \cup N_G(v))$-cvc in G. Further, S is a $(M \cup N_G(v))$-cvc in $G - \{v\}$, because if $v \notin S$, then $(G - \{v\})[S]$ must be connected, since $G[S]$ is connected. Suppose for the sake of obtaining a contradiction that $S \notin C_{G-\{v\}}(M \cup N_G(v))$, and let $X \subset S$ such that X is a $(M \cup N_G(v))$-cvc in $G - \{v\}$. Observe that since X is a $(M \cup N_G(v))$-cvc in $G - \{v\}$, we have $M \cup N_G(v) \subseteq X$, by definition of $(M \cup N_G(v))$-cvc. In particular, we have $N_G(v) \subseteq X$, which means that for any $(u, v) \in E(G)$ we have $u \in X$. Therefore, since X covers $E(G - \{v\})$, X covers $E(G)$ as well. Clearly, $M \subseteq X$, and X must be a M-cvc in G, contradicting our assumption that S was a minimal M-cvc in G. Thus, $S \in C_{G-\{v\}}(M \cup N_G(v))$.

We see that if $S \in C_G(M)$, then $S \in C_G(M \cup \{v\})$ or $S \in C_{G-\{v\}}(M \cup N_G(v))$, which is what we had to prove. We conclude that branching rules 3.7, 3.8 and 3.9 are safe. □

Branching Rule Analysis

We will now analyze branching rules 3.7, 3.8 and 3.9. Our goal is to obtain a system of branching vectors. Before we begin to analyze each branching rule separately, we will make some general observations. Let $I = (G, M)$ be an instance, and let $v \in U$ be some vertex. We are interested in a lower bound on the decrease in measure as we select or discard v. In order to quantify this lower bound, we must know how selecting or discarding v affects both the number of unmarked vertices, and the number of marked vertices.

Suppose we select v, and let $I' = (G', M')$ be the resulting subproblem. Clearly, the number of unmarked vertices decreases by 1, and the number of marked vertices increases by 1, as the unmarked vertex v is marked. However, for every $u \in N_G(v) \cap M$, we obtain a marked edge (u, v) in G'. Now, recall that Reduction Rule 3.3 contracts marked edges, and note that a single application of Reduction Rule 3.3 decreases the number of marked vertices by 1. Since G' contains m_v such edges, Reduction Rule 3.3 is applied repeatedly m_v times to

I', and the number of marked vertices therefore effectively decreases by $m_v - 1$. Thus, selecting v reduces the measure by $w_1 + (m_v - 1)w_2$. We note that $w_1 + (m_v - 1)w_2 > 0$ for $m_v \geq 0$, since $w_1 > w_2$.

Suppose now that we discard v. Let $I' = (G', M')$ be the resulting subproblem. As v is removed from the graph, the number of unmarked vertices decreases by 1. Moreover, discarding v will mark every unmarked neighbor of v. There are u_v such neighbors, thus the number of unmarked vertices further decreases by u_v, while the number of marked vertices increases by u_v. Consequently, discarding v reduces the measure by $(u_v + 1)w_1 - u_v w_2$. Again, note that $w_1 > w_2$ implies that $(u_v + 1)w_1 - u_v w_2 > 0$ for any $u_v \geq 0$. That is, the decrease in measure is always positive.

Branching Rule 3.7
Consider an application of Branching Rule 3.7 to some instance $I = (G, M)$, and let $v \in U$ be the vertex we branch on. Since selecting v reduces the measure by $w_1 + (m_v - 1)w_2$ and discarding v reduces the measure by $(u_v + 1)w_1 - u_v w_2$, the branching vectors corresponding to Branching Rule 3.7 are

$$\begin{aligned} (w_1, 3w_1 - 2w_2) \quad & \text{for } u_v \geq 2, m_v = 1 \\ (w_1 + w_2, 2w_1 - w_2) \quad & \text{for } u_v \geq 1, m_v = 2 \\ (w_1 + 2w_2, w_1) \quad & \text{for } u_v \geq 0, m_v \geq 3 \end{aligned}$$

Branching Rule 3.8
We now analyze Branching Rule 3.8. Let $I = (G, M)$ be an instance to which Branching Rule 3.8 applies, and let $v \in U$ be the vertex we branch on. Recall that selecting and discarding v reduces the measure by $w_1 + (m_v - 1)w_2$ and $(u_v + 1)w_1 - u_v w_2$, respectively. We know that $u_v = d(v)$ and $m_v = 0$. Therefore, selecting v reduces the measure by at least $w_1 - w_2$, and discarding v reduces the measure by at least $(d(v) + 1)w_1 - d(v)w_2$.

We let X be the set of all neighbors of v which have degree exactly 2, that is,

$$X = \{x \in N(v) \mid d(x) = 2\}$$

Claim. *X is an independent set.*

Proof. Suppose not, then there is some $(u, w) \in E(G)$ such that $u, w \in X$. Since $N[u] = N[w] = \{u, v, w\}$, and since $d(v) \geq 3$, we know that $G - \{v\}$ is disconnected. However, we know that since Branching Rule 3.8 is applicable to (G, M), Reduction Rule 3.5 is not, which implies that $v \in M$, a contradiction. □

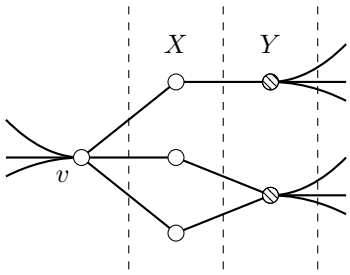


Figure 3.1: Branching Rule 3.8. Every vertex in Y is either already marked, or will be marked by Reduction Rule 3.5 after we discard v.

We let $n_2 = |X|$. Let $Y = \bigcup_{x \in X} N(x) \setminus \{v\}$, and observe that for every $y \in Y$, we have that $G - \{v, y\}$ is disconnected, thus we know that either $y \in M$ or that Reduction Rule 3.5 will apply to y if we discard v. See Fig. 3.1. Observe that since X is independent, we know that $Y \neq \emptyset$. We note that since $v \in U$, $G - \{v\}$ must be connected, by Observation 3.4. We can therefore be certain that Halting Rule 3.2 will not apply to the instance obtained by discarding v. Since every $x \in X$ is incident to some $y \in Y$, we know that for every $x \in X$, after we discard v, we will obtain a marked edge (x, y), for some $y \in Y$. Further, we know that every such edge will be contracted by Reduction Rule 3.3. In other words, after we discard v, we know that X will be removed from the graph. Therefore, discarding v reduces the measure by at least $(d(v)+1)w_1 - d(v)w_2 + n_2 w_2$. We consider two cases: $n_2 = d(v)$ and $n_2 < d(v)$.

Case 1: $n_2 = d(v)$. Since $n_2 = d(v) \geq 3$, discarding v reduces the measure by at least

$$\begin{aligned}(d(v)+1)w_1 - d(v)w_2 + n_2 w_2 &= (d(v)+1)w_1 - d(v)w_2 + d(v)w_2 \\ &= (d(v)+1)w_1 \\ &\geq 4w_1\end{aligned}$$

The branching vector corresponding to the case of $n_2 = d(v)$ is therefore

$$(w_1 - w_2, 4w_1)$$

Case 2: $n_2 < d(v)$. We know that $n_2 \geq 1$, by definition of Branching Rule 3.8. Since $n_2 \geq 1$ and $d(v) \geq 3$, discarding v will reduce the measure by at least

$$\begin{aligned}(d(v)+1)w_1 - d(v)w_2 + n_2 w_2 &\geq (d(v)+1)w_1 - d(v)w_2 + w_2 \\ &\geq 4w_1 - 2w_2\end{aligned}$$

Let $I' = (G', M')$ be the instance obtained by selecting v. Since $n_2 < d(v)$, we know that there is some $u \in N_{G'}(v)$ that satisfies $d(u) \geq 3$ and $m_u = 1$

in I'. Observe that none of rules 3.1-3.6 are applicable to I'. Consequently, the algorithm will apply Branching Rule 3.7 to u following the selection of v. When we select u, the measure is reduced by $w_1+(m_u-1)w_2 = w_1$, and when we discard u, the measure is reduced by $(u_u+1)w_1-u_uw_2$, which is at least $3w_1 - 2w_2$. That is, we can $(w_1, 3w_1 - 2w_2)$-branch immediately after selecting v. By addition of branching vectors, we obtain

$$\Big(\underbrace{(w_1 - w_2) + w_1}_{\text{select } v\text{, then select } u}, \underbrace{(w_1 - w_2) + 3w_1 - 2w_2}_{\text{select } v\text{, then discard } u}, \underbrace{4w_1 - 2w_2}_{\text{discard } v}\Big)$$

thus, the branching vector corresponding to the case of $n_2 < d(v)$ is

$$(2w_1 - w_2, 4w_1 - 3w_2, 4w_1 - 2w_2)$$

Branching Rule 3.9

Finally, consider an application of Branching Rule 3.9 to some instance $I = (G, M)$. Let $v \in U$ be the vertex we branch on, and let $N(v) = \{x, y\}$, where $x \in M$ and $y \in U$. Note that by Claim 3.14, we have $\delta(G) \geq 2$. Since Branching Rule 3.8 does not apply to I, and since $\delta(G) \geq 2$, we know that $d(y) = 2$. Consider the neighborhood of y. Since $d(y) = 2$, there are two cases: $N(y) = \{v, x\}$ (Fig. 3.2), or $N(y) = \{v, z\}$ for $z \neq x$ (Fig. 3.3).

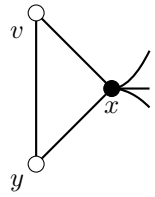


Figure 3.2: Branching Rule 3.9, case of $N(y) = \{v, x\}$.

Let us first consider the case of $N(y) = \{v, x\}$. When we select v the measure first decreases by $w_1 - w_2$ as v is marked. Since v has a marked neighbor x, we know that Reduction Rule 3.3 will contract (v, x) after v is marked, further reducing the measure by w_2. After we contract (v, x), the degree of y will be 1, and Reduction Rule 3.6 will thus remove y from the graph, decreasing the measure by w_1 for a total of $2w_1$. When we discard v, the measure decreases by $2w_1 - w_2$ as v is removed from the graph and y is marked. Reduction Rule 3.5 then contracts (x, y), reducing the measure by another w_2. In total, discarding v reduces the measure by $2w_1$, and the branching vector corresponding to the case of $N(y) = \{v, x\}$ is thus

$$(2w_1, 2w_1)$$

We now consider the case of $N(y) = \{v, z\}$, where $z \neq x$. When we select v, the measure is reduced by w_1 as v is marked and (v, x) is subsequently contracted

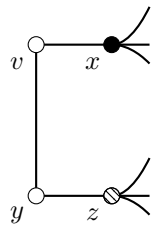


Figure 3.3: Branching Rule 3.9, case of $N(y) = \{v, z\}$.

by Reduction Rule 3.3. When we discard v, the measure first decreases by $2w_1 - w_2$ as v is removed from the graph and y is marked. Since $G - \{v, z\}$ is disconnected, we know that z is either already marked, or that Reduction Rule 3.5 will mark z after we remove v from the graph. In any case, after we remove v from the graph, we will contract the edge (x, y) and reduce the measure by w_2. Thus, discarding v reduces the measure by at least $2w_1$, and the branching vector corresponding to the case of $N(y) = \{v, z\}$ is therefore

$$(w_1, 2w_1)$$

Branching Vectors		
Branching Rule	Case	Branching Vector
3.7	$u_v \geq 2,\ m_v = 1$	$(w_1, 3w_1 - 2w_2)$
	$u_v \geq 1,\ m_v = 2$	$(w_1 + w_2, 2w_1 - w_2)$
	$u_v \geq 0,\ m_v \geq 3$	$(w_1 + 2w_2, w_1)$
3.8	$n_2 = d(v)$	$(w_1 - w_2, 4w_1)$
	$n_2 < d(v)$	$(2w_1 - w_2, 4w_1 - 3w_2, 4w_1 - 2w_2)$
3.9	$N(y) = \{v, x\}$	$(2w_1, 2w_1)$
	$N(y) = \{v, z\}$	$(w_1, 2w_1)$

Table 3.1: Branching rules and their corresponding branching vectors, parameterized by w_1 and w_2.

See Table 3.1 for a summary of branching rules and their corresponding branching vectors, parameterized by w_1 and w_2.

3.2.3 Special Instances

When proving the upper bound of 1.7485^n on the number of minimal independent feedback vertex sets in a graph on n vertices, Agrawal et al. [2, 12] introduced *special instances* and showed that there is a one-to-one correspondence between the solutions of a special instance and the spanning trees of an auxiliary graph corresponding to the special instance. It turns out that special

instances are very useful in proving an upper bound on the number of minimal connected vertex covers of a graph as well.

In this section we shall provide the definition of a special instance, and we shall show that if (G, M) is an instance to which none of rules 3.1-3.9 apply, then (G, M) is a special instance. We shall also introduce the final halting rule of EnumMcvc, which the algorithm will apply to special instances, and we shall obtain an upper bound on the number of solutions of a special instance. We note that some of the following lemmata and their proofs have been copied from Agrawal et al. [12], with some modularization, minor alterations and expanded calculations added for clarity. More precisely, we have copied lemmata 3.17, 3.27 and 3.28 from [12].

Definition 3.16. *Let G be a graph, and let $M \subseteq V(G)$. If G is connected, and $\delta(G) \geq 2$, and for every $v \in U$ we have $d(v) = 2$, and M and U are both independent, then we say that (G, M) is a special instance.*

Lemma 3.17. *[2, 12] Let $I = (G, M)$ be a special instance, then $|U| \geq |M|$.*

Proof. Since I is a special instance, we know that $d(v) = 2$ for every $v \in U$, and that $d(v) \geq 2$ for every $v \in M$. Further, we know that G is bipartite with bipartition (U, M), which implies that $\Sigma_{v \in U} d(v) = \Sigma_{v \in M} d(v)$. Now, since $d(v) = 2$ for every $v \in U$, we have $\Sigma_{v \in U} d(v) = 2\,|U|$, and since $d(v) \geq 2$ for every $v \in M$, we have $\Sigma_{v \in M} d(v) \geq 2\,|M|$. Thus,

$$2\,|U| = \Sigma_{v \in U} d(v) = \Sigma_{v \in M} d(v) \geq 2\,|M|$$

That is, $|U| \geq |M|$, and the proof is complete. □

Corollary 3.18. *Let $I = (G, M)$ be a special instance, and let n be the number of vertices in G, then $|M| \leq \frac{1}{2}n$.*

To see that Corollary 3.18 holds, first note that $n = |U| + |M|$. Since $|U| \geq |M|$, we have $|U| + |M| \geq 2\,|M|$, or equivalently, $n \geq 2\,|M|$, thus $|M| \leq \frac{1}{2}n$.

Definition 3.19. *Let $I = (G, M)$ be a special instance, and let $S \subseteq U$. In the context of special instances, we say that*

$$E_S = \{(x, y) \mid x, y \in N(v), v \in S\}$$

is the set of edges corresponding to S.

With respect to Definition 3.19, we keep multi-edges in E_S. We note that none of the edges corresponding to $S \subseteq U$ are present in the special instance itself. They are, however, present in the *special graph* corresponding to that special instance, which we now define.

Definition 3.20. *Let $I = (G, M)$ be a special instance. We call $G^{sp} = (M, E_U)$ the special graph corresponding to I.*

Lemma 3.21. *Let $I = (G, M)$ be a special instance, and let $S \subseteq U$, then $G[S \cup M]$ is connected if and only if the graph (M, E_S) is connected.*

Proof. We will first prove that the forward direction holds. Suppose $G[S \cup M]$ is connected, and let $m_1, m_l \in M$. Let P be a path between m_1 and m_l in $G[S \cup M]$. Since M and U are both independent, we have $P = (m_1, u_1, m_2, \ldots, u_{l-1}, m_l)$, where $u_i \in U$ and $m_i \in M$. We note that since $N(u_i) = \{m_i, m_{i+1}\}$, we have $(m_i, m_{i+1}) \in E_S$, for $i = 1, 2, \ldots, l-1$. Therefore, by removing all unmarked vertices from P, we obtain a path $P' = (m_1, m_2, \ldots, m_l)$ in (M, E_S). Thus, for every $u, v \in M$ there is a path between u and v in (M, E_S), and (M, E_S) is therefore connected.

We will now prove that the reverse direction holds as well. Suppose (M, E_S) is connected. Let G' be the graph obtained by subdividing every edge in (M, E_S), then G' is connected. Observe that G' is isomorphic to $G[S \cup M]$. Therefore, since G' is connected, $G[S \cup M]$ is connected as well. □

Corollary 3.22. *Let (G, M) be a special instance, and let G^{sp} be the special graph corresponding to (G, M), then G^{sp} is a connected graph.*

Corollary 3.22 follows from Lemma 3.21, since $G = G[U \cup M]$, and since G is connected whenever (G, M) is special.

Lemma 3.23. *Let $I = (G, M)$ be a special instance, then M is a vertex cover in G.*

Proof. Since I is special a special instance, we have that both U and M are independent sets. Let $e = (u, v)$ be some edge in G, and observe that since U and M are independent, e must have one endpoint in U, and the other endpoint in M. Consequently, every edge in G is incident to M, and M therefore covers $E(G)$. □

Lemma 3.24. *Let $I = (G, M)$ be a special instance, and let $G^{sp} = (M, E_U)$ be the special graph corresponding to I. Let $S \subseteq U$, then $S \cup M \in C_G(M)$ if and only if E_S forms a spanning tree in G^{sp}.*

Proof. For the forward direction, suppose $S \cup M \in C_G(M)$, and that E_S does not form a spanning tree in G^{sp}. That is, we assume that (M, E_S) is not a tree. Since (M, E_S) is not a tree, it must hold that there is some cycle in (M, E_S), since (M, E_S) is connected by Lemma 3.21. Let $e = (m_1, m_2) \in E_S$ such that $(M, E_S) - e$ is connected, i.e., let e be an edge in G^{sp} whose removal does not disconnect G^{sp}. We know that e exists, since (M, E_S) has a cycle. Let $u \in S$

such that $N(u) = \{m_1, m_2\}$, that is, let u be the vertex in S corresponding to the edge e. We note that $(M, E_S) - e = (M, E_{S\setminus\{u\}})$. Since $(M, E_{S\setminus\{u\}})$ is connected, then so is $G[(S \setminus \{u\}) \cup M]$, by Lemma 3.21. Now, observe that by Lemma 3.23, M covers $E(G)$, thus $(S \setminus \{u\}) \cup M$ is a M-cvc in G, which contradicts the minimality of $S \cup M$ in G. The forward direction holds.

For the reverse direction, suppose E_S forms a spanning tree in G^{sp}. We will show that $S \cup M$ is then a minimal M-cvc in G. By Lemma 3.23, $S \cup M$ is a vertex cover in G, and by Lemma 3.21, $G[S \cup M]$ is connected as well. It is therefore sufficient to show that $S \cup M$ is minimal. Suppose not, i.e., suppose there is some $v \in S \cup M$ such that $(S \cup M) \setminus \{v\}$ is a M-cvc in G. Clearly, $v \in S$ and $(S \cup M) \setminus \{v\} = (S \setminus \{v\}) \cup M$, since otherwise $M \not\subseteq (S \cup M) \setminus \{v\}$. Since $G[(S \setminus \{v\}) \cup M]$ is connected, $(M, E_{S\setminus\{v\}})$ is also connected. However, this is impossible, since (M, E_S) was a tree by assumption. This concludes the proof. □

Corollary 3.25. *Let $I = (G, M)$ be a special instance, and let G^{sp} be the special graph corresponding to I, then there is a one-to-one correspondence between $C_G(M)$ and the set of all spanning trees of G^{sp}.*

Corollary 3.25 follows from Lemma 3.24 and the fact that there is a one-to-one correspondence between U and $E(G^{sp})$.

In the following, let EnumST be an algorithm which takes as input a graph H, and then enumerates all spanning trees of H.

Halting Rule 3.10. *If (G, M) is a special instance, then output the set*

$$\{S \cup M \mid E_S \in \textsc{EnumST}(G^{sp}), S \subseteq U\}$$

where G^{sp} is the special graph corresponding to (G, M), and $\textsc{EnumST}(G^{sp})$ is the set of all spanning trees in G^{sp}.

Lemma 3.26. *Halting Rule 3.10 is correct.*

Lemma 3.26 follows from Lemma 3.24.

Lemma 3.27. *[2, 12] Let H be a connected graph on $n \geq 2$ vertices and m edges, then there are at most $\left(\frac{2m}{n}\right)^n$ spanning trees in H.*

Proof. Grimmett [16] showed that there are at most $\frac{1}{n}\left(\frac{2m}{n-1}\right)^{n-1}$ spanning trees in any connected graph on n vertices and m edges. Thus, in order to prove the claim, we must show that

$$\frac{1}{n}\left(\frac{2m}{n-1}\right)^{n-1} \leq \left(\frac{2m}{n}\right)^n$$

We note that

$$\frac{1}{n}\left(\frac{2m}{n-1}\right)^{n-1} = (2m)^n \cdot \frac{1}{2mn} \cdot \left(\frac{1}{n-1}\right)^{n-1}$$

and further, that

$$\begin{aligned} (2m)^n \cdot \frac{1}{2mn} \cdot \left(\frac{1}{n-1}\right)^{n-1} &\leq (2m)^n \cdot \left(\frac{1}{n}\right)^n \qquad \Longleftrightarrow \\ \frac{1}{2mn} \cdot \left(\frac{1}{n-1}\right)^{n-1} &\leq \left(\frac{1}{n}\right)^n \end{aligned}$$

Since H is a connected graph, we must have $m \geq n-1$, thus

$$\frac{1}{2mn} \cdot \left(\frac{1}{n-1}\right)^{n-1} \leq \frac{1}{2n(n-1)} \cdot \left(\frac{1}{n-1}\right)^{n-1} = \frac{1}{2n} \cdot \left(\frac{1}{n-1}\right)^n$$

Observe that

$$\begin{aligned} \frac{1}{2n} \cdot \left(\frac{1}{n-1}\right)^n &\leq \left(\frac{1}{n}\right)^n \qquad \Longleftrightarrow \\ \left(\frac{n}{n-1}\right)^n &\leq 2n \qquad \Longleftrightarrow \\ \left(\frac{n}{n-1}\right)^{n-1} &\leq 2n\left(\frac{n-1}{n}\right) \qquad \Longleftrightarrow \\ \left(1+\frac{1}{n-1}\right)^{n-1} &\leq 2(n-1) \end{aligned} \tag{3.1}$$

and that

$$\lim_{n\to\infty}\left(1+\frac{1}{n-1}\right)^{n-1} = e$$

For $n = 2$, (3.1) clearly holds. For $n \geq 3$, we have $2(n-1) \geq 4 > e$. Therefore, for $n \geq 2$, it holds that

$$\frac{1}{n}\left(\frac{2m}{n-1}\right)^{n-1} \leq \left(\frac{2m}{n}\right)^n$$

and the proof is complete. □

Lemma 3.28. *[2, 12] Let $I = (G, M)$ be a special instance, and let $n = |V(G)|$, then*

$$|C_G(M)| \leq \left(\frac{2\alpha}{1-\alpha}\right)^{(1-\alpha)n}$$

where $|M| = (1-\alpha)n$, and $\frac{1}{2} \leq \alpha \leq 1$.

Proof. We first note that by Corollary 3.18, we have $(1-\alpha)n \leq \frac{1}{2}n$, which implies $\alpha \geq \frac{1}{2}$. Let $\zeta(H)$ denote the number of spanning trees in the graph H. By assumption $|M| = (1-\alpha)n$, therefore $|U| = \alpha n$. Since G^{sp} is a connected graph on $|M| = (1-\alpha)n$ vertices and $|U| = \alpha n$ edges, we have

$$\zeta(G^{sp}) \leq \left(\frac{2\alpha n}{(1-\alpha)n}\right)^{(1-\alpha)n} = \left(\frac{2\alpha}{1-\alpha}\right)^{(1-\alpha)n}$$

by Lemma 3.27. By Corollary 3.25 there is a one-to-one correspondence between minimal M-cvc in G and spanning trees in G^{sp}, thus

$$|C_G(M)| = \zeta(G^{sp}) \leq \left(\frac{2\alpha}{1-\alpha}\right)^{(1-\alpha)n}$$

□

Claim 3.29. *Let $I = (G, M)$ be an instance to which none of rules 3.1-3.9 apply, then I is a special instance.*

Proof. We know that G is connected, since Halting Rule 3.2 does not apply to I. Since Reduction Rule 3.3 does not apply to I, M must be an independent set. Further, since none of rules 3.1-3.6 are applicable, we know that $\delta(G) \geq 2$, by Claim 3.14. Thus, in order to show that I is a special instance, it is sufficient to show that U is independent, and that $d(v) = 2$, for every $v \in U$. Recall that we always assume that $M \neq \emptyset$.

Let $u \in M$ and let $v \in N(u)$, then $v \in U$, since M is independent. We know that $d(v) = 2$, as otherwise Branching Rule 3.7 would have applied to I. Moreover, since $d(v) = 2$ and v has a marked neighbor u, it must hold that $m_v = 2$, since $m_v = 1$ implies that Branching Rule 3.9 would have applied to I. In other words, whenever $v \in U$ is adjacent to some $u \in M$, we know that $d(v) = 2$ and that $m_v = 2$. Consequently, since G is connected and $M \neq \emptyset$, we know that $d(v) = m_v = 2$, for every $v \in U$. That is, U is an independent set in which every vertex has degree *exactly* 2, which is what we had to show. □

3.2.4 An Improved Upper Bound

Lemma 3.30. *Let I be an instance, then some rule of* EnumMcvc *is applicable to I.*

Proof. If one of rules 3.1-3.9 apply to I, then the claim holds, and there is nothing to prove. Suppose none of rules 3.1-3.9 apply to I, then I is a special instance, by Claim 3.29. Since I is a special instance, we know that Halting Rule 3.10 will apply to I. The proof is complete. □

Lemma 3.31. *Let G be a graph on n vertices, and let $M \subseteq V(G)$ such that $M \neq \emptyset$, then $|C_G(M)| \leq 1.7076^n$.*

Proof. We have presented a branching algorithm EnumMcvc which enumerates all minimal M-cvc of a graph G, for non-empty $M \subseteq V(G)$. By lemmata 3.5, 3.7, 3.11 and 3.13, all reduction rules are safe, and by Lemma 3.15, all branching rules are safe. Further, by lemmata 3.6, 3.8 and 3.26, all halting rules are correct. We know that for any instance I, some rule of EnumMcvc will apply to I, by Lemma 3.30.

We let the measure of an instance (G, M) be $\mu(G, M) = 42\,|U| + 25\,|M|$. We present the branching vectors of EnumMcvc and their corresponding branching factors in Table 3.2, for $w_1 = 42$ and $w_2 = 25$. Let $x = 1.01282$, and note that for each branching vector $\vec{b}$, we have $\tau(\vec{b}) \leq x$.

Let (G, M) be an instance to which some halting rule applies. We will show that $|C_G(M)| \leq x^{\mu(G,M)}$. If Halting Rule 3.2 or Halting Rule 3.4 applies to (G, M), then

$$|C_G(M)| \leq 1 \leq x^{\mu(G,M)}$$

Suppose that Halting Rule 3.10 applies to (G, M), and let n be the number of vertices in G. Let $\frac{1}{2} \leq \alpha \leq 1$, and let $|M| = (1-\alpha)n$. We have

$$\begin{aligned}\mu(G, M) &= 42\,|U| + 25\,|M| \\ &= 42\alpha n + 25(1-\alpha)n \\ &= n(17\alpha + 25)\end{aligned}$$

By Lemma 3.28,

$$|C_G(M)| \leq \max_{\frac{1}{2} \leq \alpha \leq 1} \left\{ \left(\frac{2\alpha}{1-\alpha} \right)^{(1-\alpha)n} \right\} \leq \max_{\frac{1}{2} \leq \alpha \leq 1} \left\{ \left(\frac{2\alpha}{1-\alpha} \right)^{\frac{(1-\alpha)\mu(G,M)}{17\alpha+25}} \right\}$$

We have $\max\limits_{\frac{1}{2} \leq \alpha \leq 1} \left\{ \left(\frac{2\alpha}{1-\alpha} \right)^{\frac{1-\alpha}{17\alpha+25}} \right\} = 1.01282 \leq x$, thus $|C_G(M)| \leq x^{\mu(G,M)}$.

We see that whenever (G, M) is an instance to which some halting rule applies, it holds that $|C_G(M)| \leq x^{\mu(G,M)}$. By Lemma 2.1, we have

$$|C_G(M)| \leq 1.01282^{\mu(G,M)}$$

Thus, for any graph G on n vertices and $M \subseteq V(G)$ such that $M \neq \emptyset$,

$$|C_G(M)| \leq 1.01282^{\mu(G,M)} \leq 1.01282^{42n} \leq 1.7076^n$$

This concludes the proof. □

Branching Vectors				
Branching Rule	Case	Branching Vector	c^{μ}	c^{42n}
3.7	$u_v \geq 2,\ m_v = 1$	$(42, 76)$	1.012178^{μ}	1.6627^{n}
	$u_v \geq 1,\ m_v = 2$	$(67, 59)$	1.011079^{μ}	1.5885^{n}
	$u_v \geq 0,\ m_v \geq 3$	$(92, 42)$	1.010952^{μ}	1.5801^{n}
3.8	$n_2 = d(v)$	$(17, 168)$	1.010743^{μ}	1.5665^{n}
	$n_2 < d(v)$	$(59, 93, 118)$	1.012819^{μ}	1.7074^{n}
3.9	$N(y) = \{v, x\}$	$(84, 84)$	1.008286^{μ}	1.4143^{n}
	$N(y) = \{v, z\}$	$(42, 84)$	1.011523^{μ}	1.6181^{n}

Table 3.2: Branching vectors and their corresponding branching factors, for measure $\mu(G, M) = 42\,|U| + 25\,|M|$.

With respect to the proof of Lemma 3.31, we note that we arrived at the weights $w_1 = 42$ and $w_2 = 25$ of the measure programmatically. That is, for every possible combination of values of w_1 and w_2 in the range $[1, 100)$ for which $w_1 > w_2$, we computed the branching factor of every parameterized branching vector of EnumMcvc, along with the upper bound β on $|C_G(M)|$ for a special instance (G, M). The values of w_1 and w_2 which attained the smallest maximum among the branching factors and β were $w_1 = 42$ and $w_2 = 25$.

Theorem 3.32. *There are at most $2 \cdot 1.7076^n$ minimal connected vertex covers in any graph on n vertices.*

Theorem 3.32 follows from Lemma 3.31 and Observation 3.2. We note that since all reduction rules and branching rules of EnumMcvc can be implemented to run in time polynomial in the measure of the input instance, and since the spanning trees of a graph can be enumerated with polynomial delay, we can enumerate all minimal connected vertex covers of an n-vertex graph in time $\mathcal{O}^*(1.7076^n)$. We end this chapter with an important observation.

Remark 3.33. *Consider Table 3.2, and in particular the branching vector corresponding to Branching Rule 3.8, case of $n_2 < d(v)$. The branching factor of this branching vector is substantially worse than the branching factor of any other branching vector in Table 3.2. Therefore, if we are to attempt further improvements on the upper bound, then we must somehow eliminate this branching vector from the system.*

Chapter 4

Improved Combinatorial Bounds for the Independent Feedback Vertex Set Problem

4.1 Introduction

In this chapter we present improved combinatorial bounds on the maximum number of minimal independent feedback vertex sets in a graph. We improve the lower bound and the upper bound on the maximum number of minimal independent feedback vertex sets in a graph in Section 4.2 and Section 4.3, respectively.

4.2 Improving the Lower Bound

By virtue of the following example from [2], Agrawal et al. showed that the maximum number of minimal independent feedback vertex sets in any graph on n vertices is at least $3^{n/3}$. Let n be a positive multiple of 3, and let G be a graph on n vertices such that G has $n/3$ connected components, each of which induces a K_3. In order to construct a minimal independent feedback vertex set in G we must select exactly one vertex from each component of G, and within each component there are 3 choices for which vertex to select. Thus, as there

are $n/3$ components in G, there are in total $3^{n/3}$ minimal independent feedback vertex sets in G. We provide an illustration of the graph G in Fig. 4.1.

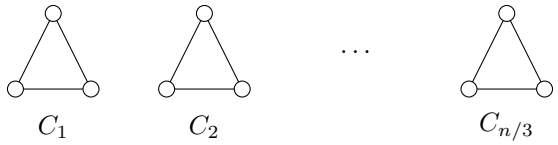


Figure 4.1: The depicted graph G has $3^{n/3}$ minimal independent feedback vertex sets, where $n = |V(G)|$. Here, C_i denotes the i'th component of G.

Let H be a graph on a vertices, and let x be the number of minimal independent feedback vertex sets in H. Further, let G be a graph on $n = ta$ vertices, such that G has t connected components $C_1, C_2, \ldots, C_t$, where $G[C_i]$ is isomorphic to H, for $i = 1, 2, \ldots, t$. Any minimal independent feedback vertex set in G has the form $S = \bigcup_{i=1}^{t} S_i$, where $S_i \subseteq C_i$ is one of the x minimal independent feedback vertex sets in $G[C_i]$. In other words, in order to obtain a minimal independent feedback vertex set S in G, one must select a minimal independent feedback vertex set S_i from each individual component C_i of G. Since each component C_i presents x choices for S_i, and since there are t components, there are in total $x^t = x^{n/a} = (x^{1/a})^n$ minimal independent feedback vertex sets in G. For a graph G on $a > 0$ vertices, we define

$$\rho(G) = x^{1/a}$$

where x is the number of minimal independent feedback vertex sets in G. We make the following observation.

Observation 4.1. *If there exists some graph G on $a > 0$ vertices such that $\rho(G) \geq c$, for some constant c, then the maximum number of minimal independent feedback vertex sets in any graph on n vertices is at least c^n.*

nauty and **Traces** are programs for computing automorphism groups of both undirected an directed graphs. Using the **geng** program contained in the **gtools** program suite included in the nauty and Traces package, we generated the set of all non-isomorphic graphs on $n \leq 11$ vertices, and then maximized $\rho(\cdot)$ over this set, using brute force to count the number of minimal independent feedback vertex sets in each graph. We present the graph which attained the maximum in Lemma 4.2. We refer the reader to [17] for a description of the algorithms behind nauty and Traces.

Lemma 4.2. *There exists a graph G which attains $\rho(G) \geq 1.4752$.*

Proof. Let $w \in \mathbb{N}$ such that $w \geq 2$, and let $G = (V, E)$ be a graph, where

$$V = \{L, R\} \cup \bigcup_{i=1}^{w} \{l_i, r_i\}$$

$$E = \bigcup_{i=1}^{w} \{(L, l_i), (l_i, r_i), (r_i, R)\}$$

We depict G in Fig. 4.2. Let S be a minimal independent feedback vertex set in G which is disjoint from $\{L, R\}$, then there is some j such that $\{l_j, r_j\} \cap S = \emptyset$, and for every $i \neq j$ we have $|S \cap \{l_i, r_i\}| = 1$. There are w choices for j, and for each choice of j, there are 2^{w-1} ways in which S may intersect with the remaining $\{l_i, r_i\}$. Thus, there are in total $w \cdot 2^{w-1}$ minimal independent feedback vertex sets in G which are disjoint from $\{L, R\}$. We note that the number of vertices in G is $2 + 2w$. If we let $w = 12$, then $\rho(G) \geq 1.4752$. This concludes the proof. □

Regarding the proof of Lemma 4.2, a natural way of arriving at $w = 12$ is to maximize the function $(w \cdot 2^{w-1})^{\frac{1}{2+2w}}$. We have

$$\max_{w \geq 2} \left\{ (w \cdot 2^{w-1})^{\frac{1}{2+2w}} \right\} \approx 1.47526$$

at $w \approx 11.832$.

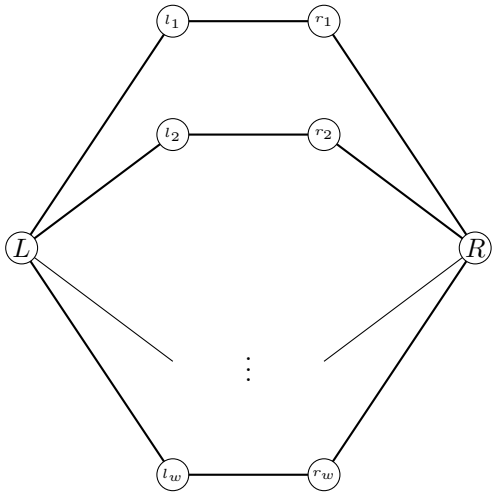


Figure 4.2: $\rho(G) \geq 1.4752$ for $w = 12$.

Ryland [11] discovered a graph which can be used to show that 1.51978^n is a lower bound on the maximum number of minimal connected vertex covers in

any graph on n vertices. Through personal communication, we learned that Ryland had arrived at the graph presented in Lemma 4.2 independently, and further discussion revealed that this graph had equally many minimal connected vertex covers and minimal independent feedback vertex sets, which we found interesting. We learned that using the graph from Lemma 4.2, one can construct a larger graph G which can be used to show an improved lower bound on the maximum number of minimal connected vertex covers in a graph. Out of curiosity, we decided to evaluate $\rho(G)$, and we were enthusiastic to see that $\rho(G) \geq 1.5067$. We present the graph G in Lemma 4.3.

Lemma 4.3. *There exists a graph G which attains $\rho(G) \geq 1.5067$.*

Proof. Let $w \in \mathbb{N}$ such that $w \geq 2$, and let $G = (V, E)$ be a graph, where

$$\begin{aligned}
V &= \{U, D, L, R\} \cup \bigcup_{i=1}^{4} \bigcup_{j=1}^{w} \{l_{ij}, r_{ij}\} \\
E &= E_L \cup E_R \\
E_L &= \bigcup_{j=1}^{w} \{(U, l_{1j}), (l_{1j}, l_{2j}), (l_{2j}, L), (L, l_{3j}), (l_{3j}, l_{4j}), (l_{4j}, D)\} \\
E_R &= \bigcup_{j=1}^{w} \{(U, r_{1j}), (r_{1j}, r_{2j}), (r_{2j}, R), (R, r_{3j}), (r_{3j}, r_{4j}), (r_{4j}, D)\}
\end{aligned}$$

We depict G in Fig. 4.3. We let

$$\begin{aligned}
V_1 &= \{U, R\} \cup \bigcup_{i=1}^{w} \{r_{1i}, r_{2i}\} \\
V_2 &= \{U, L\} \cup \bigcup_{i=1}^{w} \{l_{1i}, l_{2i}\} \\
V_3 &= \{D, L\} \cup \bigcup_{i=1}^{w} \{l_{3i}, l_{4i}\} \\
V_4 &= \{D, R\} \cup \bigcup_{i=1}^{w} \{r_{3i}, r_{4i}\}
\end{aligned}$$

and we let $G_i = G[V_i]$, for $i = 1, 2, 3, 4$. Let H be the graph presented in the proof of Lemma 4.2 such that H contains $2 + 2w$ vertices, and observe that G_i is isomorphic to H, for $i = 1, 2, 3, 4$. We shall not count every minimal independent feedback vertex set in G, but rather we shall provide a sufficient lower bound on the number of minimal independent feedback vertex sets in G. Let $M = \{U, D, L, R\}$, and let S be a minimal independent feedback vertex set in G which is disjoint from M, then there must be some $1 \leq i \leq 4$ such that the following holds.

a) There exists $X \subseteq V_i \setminus M$ such that $G_i - X$ is disconnected, X is independent, and $X \subseteq S$.

b) For every $j \neq i$, there exists a minimal independent feedback vertex set S_j in G_j such that $S_j \cap M = \emptyset$, and $S_j \subseteq S$.

c) $S = X \cup \bigcup_{j \neq i} S_j$.

Observe that if (a) does not hold, then there exists $v_i \in V(G) \setminus S$ such that

$$(R, v_1, v_2, U, v_3, v_4, L, v_5, v_6, D, v_7, v_8, R)$$

is a cycle in $G - S$. Further, note that if (b) does not hold, then there exists some $j \neq i$ for which $G_j - S$ contains a cycle, which implies that $G - S$ contains a cycle as well. Thus, in order to obtain a minimal independent feedback vertex set in G which is disjoint from M, we must choose some $1 \leq i \leq 4$, and then some independent set $X \subseteq V_i \setminus M$ such that $G_i - X$ is disconnected. There are 2^w ways to choose such a set X from $V_i \setminus M$. Thereafter, for all $j \neq i$, we must choose a minimal independent feedback vertex set S_j in G_j such that $S_j \cap M = \emptyset$. Since G_j is isomorphic to H, there are $w \cdot 2^{w-1}$ minimal independent feedback vertex sets of G_j which are disjoint from M. We see that there are in total

$$4 \cdot 2^w \cdot (w \cdot 2^{w-1})^3 = 2^{4w-1} \cdot w^3$$

minimal independent feedback vertex sets in G which are disjoint from M. We note that the number of vertices in G is $4 + 8w$. If we let $w = 6$, then $\rho(G) \geq 1.5067$. This concludes the proof. □

With respect to the proof of Lemma 4.3, we note that we obtained $w = 6$ by maximizing the function $(2^{4w-1} \cdot w^3)^{\frac{1}{4+8w}}$.

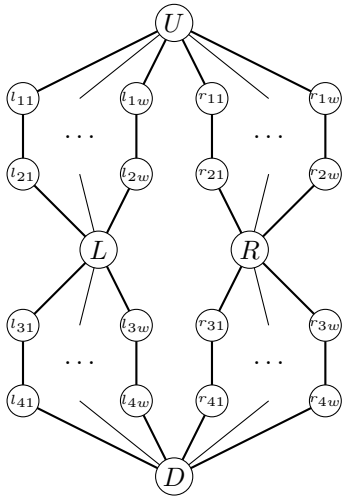


Figure 4.3: $\rho(G) \geq 1.5067$ for $w = 6$.

Theorem 4.4. *The maximum number of minimal independent feedback vertex sets in any graph on n vertices is at least 1.5067^n.*

Theorem 4.4 follows from Lemma 4.3 and Observation 4.1.

4.3 Improving the Upper Bound

In this section we present an improved upper bound on the maximum number of minimal independent feedback vertex sets in a graph.

Definition 4.5. *Let G be a graph, and let $M \subseteq V(G)$. Let S be an independent feedback vertex set in G. If $S \cap M = \emptyset$, then S is an M-ifvs in G.*

If S is an M-ifvs in G and there is no $X \subset S$ such that X is an M-ifvs in G, then S is a *minimal* M-ifvs in G. We denote the set of all *minimal* M-ifvs of a graph G by $I_G(M)$. We note that $I_G(\emptyset)$ is then the set of *all* minimal independent feedback vertex sets in the graph G.

Definition 4.6. *Let G be a multigraph, and let $M \subseteq V(G)$. If it holds that for every multi-edge $(u, v) \in E(G)$ we have $u \in M$ or $v \in M$, and that for every self-loop $(v, v) \in E(G)$ we have $v \in M$, then we will say that (G, M) is a nice instance.*

We note that for any simple graph G and $M \subseteq V(G)$, (G, M) is a nice instance.

Observation 4.7. *If there is some constant $c \geq 1$ such that $|I_G(M)| \leq c^n$ for any nice instance (G, M), where $n = |V(G)|$, then there are at most c^n minimal independent feedback vertex sets in any simple graph on n vertices.*

In order to upper bound the maximum number of minimal independent feedback vertex sets in a graph, we will obtain an upper bound on $|I_G(M)|$, for any nice instance (G, M). We will obtain our upper bound by applying the approach described in Section 2.6.1. Thus, we dedicate the remainder of this chapter to constructing a branching algorithm EnumMifvs which takes as input a nice instance (G, M) and enumerates all minimal M-ifvs of G. The algorithm EnumMifvs builds upon the algorithm of Agrawal et al. [2, 12].

We adopt the notation of Agrawal et al. [2]. Let $I = (G, M)$ be an instance. We let $U = V(G) \setminus M$. We shall refer to vertices in M as *marked* vertices, and to vertices in U as *unmarked* vertices, and we shall refer to edges for which both endpoints are marked as *marked* edges. We let $m_v = |N(v) \cap M|$, and $u_v = |N(v)| - m_v$, for any $v \in V(G)$. We define the measure of I to be

$$\mu(I) = \mu(G, M) = w_1 |U| + w_2 |M|$$

where $w_1, w_2 \in \mathbb{N}$ such that $w_1 > w_2 > 0$. Note that we are using the same notation and measure as in Section 3.2. In *this* section however, if we have $v \in M$, then v *cannot* belong to the solution, whereas in Section 3.2, if $v \in M$ then v *must* belong to the solution. We will be choosing values for w_1 and w_2 at the end of this section which will differ from the values chosen at the end of Section 3.2.

We apply the rules of EnumMifvs in ascending order, according to the numbering of each rule. In the following, unless we state otherwise, we shall assume that any instance (G, M) under consideration is a nice instance. Since we are working with multigraphs, we let G/e denote the graph obtained from G by contracting the edge e and *keeping* all multi-edges and self-loops. Observe that this is in contrast to Section 3.2, where any multi-edge or self-loop obtained by contracting an edge was discarded.

4.3.1 Reduction Rules and Halting Rules

We will now describe the reduction rules and the halting rules of EnumMifvs.

Reduction Rule 4.1. *If there exists some $v \in V(G)$ such that $d(v) \leq 1$, and there is no edge e incident to v such that e is a self-loop or a multi-edge, then replace the instance (G, M) with the instance $(G - \{v\}, M)$.*

An application of Reduction Rule 4.1 removes a single vertex v from the graph. If $v \in U$, then the measure is reduced by w_1, and if $v \in M$, then the measure is

reduced by w_2. Thus, an application of Reduction Rule 4.1 reduces the measure by at least w_2.

Reduction Rule 4.2. *If there is some edge $(u, v) \in E(G)$ such that $u, v \in M$, then let $G' = G/(u, v)$, and let $M' = (M \setminus \{u, v\}) \cup \{w\}$, where w is the vertex obtained by contracting (u, v), and replace the instance (G, M) with the instance (G', M').*

Since Reduction Rule 4.2 contracts a single marked edge, an application of Reduction Rule 4.2 will reduce the measure by w_2.

Observation 4.8. *If (G, M) is an instance to which rules 4.1-4.2 do not apply, then M is an independent set.*

Reduction Rule 4.3. *If there is some $v \in U$ for which there is some $u \in M$ such that (u, v) is a multi-edge, then replace the instance (G, M) with the instance $(G - \{v\}, M \cup N(v))$.*

An application of Reduction Rule 4.3 will reduce the measure by at least w_1.

Halting Rule 4.4. *If $G[M]$ contains a cycle, output $\emptyset$.*

Halting Rule 4.5. *If G is acyclic, then output $\{\emptyset\}$.*

Halting Rule 4.6. *If G contains K_4 as an induced subgraph, then output $\emptyset$.*

Claim 4.9. *If (G, M) is an instance to which rules 4.1-4.4 do not apply, then G contains no self-loops and no multi-edges, and $\delta(G) \geq 2$.*

Proof. We shall first prove that (G, M) contains no self-loops and no multi-edges. Suppose not, and let $e \in E(G)$ such that e is a self-loop or a multi-edge. Recall that we always assume that (G, M) is a nice instance. Let us first assume that e is a self-loop. We let $e = (v, v)$, for some $v \in V(G)$. Since (G, M) is a nice instance, we know that $v \in M$. However, if $v \in M$, then $G[M]$ contains a cycle (v, v). We know that this is not possible, because $G[M]$ must be acyclic, since Halting Rule 4.4 is not applicable to (G, M). Therefore, we must have $v \in U$, which contradicts our assumption that (G, M) is a nice instance. Since e cannot be a self-loop, e must be a multi-edge. We let $e = (u, v)$ for some $u, v \in V(G)$. Since (G, M) is a nice instance, we know that $u \in M$ or $v \in M$. If $u, v \in M$, then $G[M]$ contains a cycle (u, v, u), another contradiction. Thus, without loss of generality we must have $u \in M$ and $v \in U$. However, if $u \in M$ and $v \in U$, and (u, v) is a multi-edge, then Reduction Rule 4.3 would be applicable to (G, M), which it is not. We must therefore have $u, v \in U$, which contradicts our assumption that (G, M) was a nice instance. We see that when (G, M) is an instance to which rules 4.1-4.4 do not apply, then G contains no self-loops and no multi-edges. Now, observe that since there are no self-loops and no multi-edges in G, and since Reduction Rule 4.1 is not applicable to (G, M), we know

that $\delta(G) \geq 2$, because if there was some $v \in V(G)$ such that $d(v) \leq 1$, then there would be no self-loop or multi-edge incident to v, and hence Reduction Rule 4.1 would be applicable to (G, M), contradicting our assumption that none of rules 4.1-4.4 apply to (G, M). □

We shall now prove that reduction rules 4.1, 4.2 and 4.3 are safe, and that halting rules 4.4, 4.5 and 4.6 are correct.

Lemma 4.10. *Reduction Rule 4.1 is safe.*

Proof. Let (G, M) be an instance to which Reduction Rule 4.1 applies, and let $v \in V(G)$ be the vertex we remove from the graph, then $(G - \{v\}, M)$ is the instance output by Reduction Rule 4.1 when applied to (G, M). We let $g(S) = S$ be the function which maps a solution of the reduced instance to a solution of the input instance. Let $S \in I_G(M)$. In order to show that Reduction Rule 4.1 is safe, we must show that $S \in I_{G-\{v\}}(M)$. We know that S is an M-ifvs in $G-\{v\}$, since $(G-\{v\})-S$ must be acyclic whenever $G-S$ is acyclic. We must show that S is a minimal M-ifvs in $G - \{v\}$. Suppose not, and let $X \subset S$ such that X is an M-ifvs in $G - \{v\}$. Observe that since $d_G(v) \leq 1$ and there is no edge e which is incident to v such that e is a self-loop or a multi-edge, v cannot participate in any cycle in G. Since X is an M-ifvs in $G - \{v\}$, $(G - \{v\}) - X$ is acyclic. However, so is $G - X$, because v does not participate in any cycle in G. Consequently, X is an M-ifvs in G as well, which contradicts the minimality of S in G. Thus, we must have $S \in I_{G-\{v\}}(M)$. We conclude that Reduction Rule 4.1 is safe. □

Lemma 4.11. *Let G be a graph, and let $(u, v) \in E(G)$, then G is acyclic if and only if $G/(u, v)$ is acyclic.*

Proof. We let w be the vertex obtained by contracting (u, v). We shall first show that the forward direction holds. Suppose that G is acyclic, but that $G/(u, v)$ contains some cycle C. If C does not contain w, then C is a cycle in G as well, contradicting our assumption that G is acyclic. We therefore assume that C *does* contain w. If C is a self-loop (w, w), then either (u, u) or (v, v) is a self-loop in G, or (u, v) is a multi-edge in G. In either case, G contains a cycle, a contradiction. If C is a multi-edge between w and some $x \in N_{G/(u,v)}(w)$, then one of (u, x, u), (v, x, v) and (u, v, x, u) must be a cycle in G. Thus, C is neither a self-loop nor a multi-edge. Let x and y be the neighbors of w in C, and note that $x, y \in N_G(u) \cup N_G(v)$. There are two cases which we must consider:

Case 1: $x, y \in N_G(u)$, or $x, y \in N_G(v)$. If $x, y \in N_G(u)$, then we obtain a cycle in G by replacing w with u in C. The same argument holds for the case of $x, y \in N_G(v)$; simply replace w with v instead of u.

Case 2: Without loss of generality $x \in N_G(u)$ and $y \in N_G(v)$. We obtain a cycle in G by replacing w with the edge (u, v) in C.

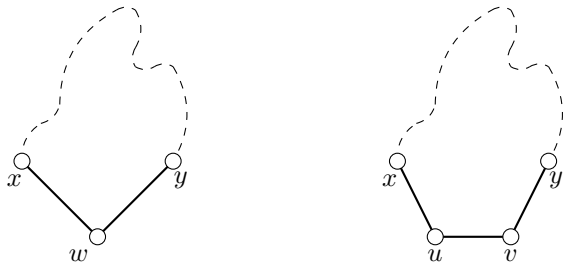
We provide an illustration of the latter case in Fig. 4.4. Our assumption that $G/(u, v)$ contains a cycle contradicts our assumption that G is acyclic, and $G/(u, v)$ must therefore be acyclic.

We shall now show that the reverse direction holds as well. Suppose that $G/(u, v)$ is acyclic, but that G contains some cycle C. The cycle C either contains only u or only v, or it contains both u and v, because if neither u nor v is in C, then C is a cycle in $G/(u, v)$ as well. We must consider two cases:

Case 1: C contains exactly one of u and v. Suppose C contains u, but not v. We obtain a cycle in $G/(u, v)$ by replacing u with w in C. Correspondingly, if C contains v, but not u, replacing v with w in C yields a cycle in $G/(u, v)$.

Case 2: C contains both u and v. Observe that it is safe to assume that (u, v) is not a multi-edge, because if it was, then we would have a self-loop (w, w) in $G/(u, v)$, which cannot be the case, since $G/(u, v)$ is acyclic. Thus, there must be some path between u and v that does not use the edge (u, v). We know that there is no $z \in V(G)$ for which (u, z, v) is a path, because this would imply that (w, z) is a multi-edge in $G/(u, v)$. Thus, let $x \in N_G(u)$ and $y \in N_G(v)$ such that $(u, x, \ldots, y, v)$ is a path between u and v, and note that $(x, \ldots, y)$ is a path between x and y in $G/(u, v)$. Since $(w, x), (w, y) \in E(G/(u, v))$, and since there is a path $(x, \ldots, y)$ between x and y in $G/(u, v)$, $(w, x, \ldots, y, w)$ is a cycle in $G/(u, v)$.

Our assumption that G contains a cycle contradicts our assumption that $G/(u, v)$ is acyclic, and G must therefore be acyclic. Both the forward direction and the reverse direction of the proof hold, and the proof is complete. □



(a) The cycle C in $G/(u,v)$.

(b) The cycle obtained in G by replacing w with the edge (u,v) in C.

Figure 4.4: If $x \in N_G(u)$ and $y \in N_G(v)$, then we obtain a cycle in G by replacing w with the edge (u,v) in C.

Lemma 4.12. *Reduction Rule 4.2 is safe.*

Proof. Let (G, M) be an instance to which Reduction Rule 4.2 applies. Suppose $(u,v) \in E(G)$ is the edge we contract, then $u, v \in M$. Let w be the vertex obtained by contracting the edge (u,v), then

$$(G', M') = (G/(u,v), (M \setminus \{u,v\}) \cup \{w\})$$

is the instance obtained by contracting (u,v). We let $g(S) = S$ be the function which maps a solution of the reduced instance to a solution of the input instance. Let $S \in I_G(M)$. In order to show that Reduction Rule 4.2 is safe, we must show that $S \in I_{G'}(M')$ as well.

We will first show that S is an M'-ifvs in G'. We note that since S is disjoint from M, and since $w \notin S$, S is disjoint from M' as well. Since $u, v, w \notin S$, we have $G[S] = G'[S]$. Therefore, since S is an independent set in G, S must also be an independent set in G'. Now, observe that since $u, v \notin S$, we have that

$$(G/(u,v)) - S = (G - S)/(u,v)$$

That is, $G' - S = (G - S)/(u,v)$. Since $G - S$ is acyclic, $(G - S)/(u,v)$ must be acyclic as well, by Lemma 4.11. Thus, $G' - S$ is acyclic, and S is an M'-ifvs in G'.

We must show that S is a *minimal* M'-ifvs in G'. Suppose not, and let $X \subset S$ such that X is an M'-ifvs in G'. We will show that X must be an M-ifvs in G as well, which will contradict the minimality of S. We know that S is independent in G, and that $S \cap M = \emptyset$. Since $X \subset S$, it then follows that X is independent in G, and that $X \cap M = \emptyset$. Thus, in order to show that X is an M-ifvs in G, we need only show that $G - X$ is acyclic. Since $u, v \notin S$ and $X \subset S$, we must have $u, v \notin X$, and thus

$$(G/(u,v)) - X = (G - X)/(u,v)$$

I.e., $G' - X = (G - X)/(u, v)$. Since $G' - X$ is acyclic, $(G - X)/(u, v)$ is also acyclic. Further, since $(G - X)/(u, v)$ is acyclic, it must hold that $G - X$ is acyclic, by Lemma 4.11. We see that X is an M-ifvs in G, which contradicts our assumption that S is a minimal M-ifvs in G. Thus, we must have $S \in I_{G'}(M')$. We conclude that Reduction Rule 4.2 is safe. □

Lemma 4.13. *Reduction Rule 4.3 is safe.*

Proof. Let (G, M) be an instance to which Reduction Rule 4.3 applies, and let $v \in U$ be the vertex we remove from the graph, then $(G - \{v\}, M \cup N(v))$ is the instance output by Reduction Rule 4.3 when applied to (G, M). We let $g(S') = S' \cup \{v\}$ be the function which maps a solution of the reduced instance to a solution of the input instance. Let $S \in I_G(M)$, let $G' = G - \{v\}$, and let $M' = M \cup N(v)$. In order to show that Reduction Rule 4.3 is safe, we must show that there exists some $S' \in I_{G'}(M')$ such that $g(S') = S$. Let $u \in N(v) \cap M$ such that (u, v) is a multi-edge. Since $S \cap M = \emptyset$, we must have $v \in S$, otherwise (u, v, u) would be a cycle in $G - S$, contradicting our assumption that S is a feedback vertex set in G. Now, let $S' = S \setminus \{v\}$, and note that $g(S') = S$. We will show that $S' \in I_{G'}(M')$. We will first show that S' is an M'-ifvs in G'. Since $v \in S$ and S is independent, we have $S \cap N(v) = \emptyset$. Thus, since $S \cap M = \emptyset$, we have $S \cap (M \cup N(v)) = S \cap M' = \emptyset$. Finally, since $S' \subset S$, we have $S' \cap M' = \emptyset$ as well. Clearly, since S is independent, then so is S'. In order to show that S' is an M'-ifvs in G', what remains is to show that S' is a feedback vertex set in G'. Observe that

$$G - S = (G - \{v\}) - (S \setminus \{v\}) = G' - S' \tag{4.1}$$

Thus, $G - S = G' - S'$, which implies that $G' - S'$ is acyclic. We see that S' is an M'-ifvs in G'. Suppose $S' \notin I_{G'}(M')$, then there must be some $X' \subset S'$ such that X' is an M'-ifvs in G'. Let $X = g(X') = X' \cup \{v\}$. Since $X' \subset S'$, we have $X \subset S$. Observe that (4.1) holds when we replace S and S' with X and X', respectively. Thus, since $G' - X'$ is acyclic, $G - X$ is acyclic as well, which contradicts our assumption that S is a minimal M-ifvs in G. We conclude that $S' \in I_{G'}(M')$, and therefore also that Reduction Rule 4.3 is safe. □

Lemma 4.14. *Halting Rule 4.4 is correct.*

Proof. Let (G, M) be an instance to which Halting Rule 4.4 applies, then $G[M]$ contains a cycle. In order to show that Halting Rule 4.4 is correct, we must show that $I_G(M) = \emptyset$. Suppose not, and let $S \in I_G(M)$. Consider $G - S$. Since $S \cap M = \emptyset$, $G[M]$ is a subgraph of $G - S$, and any cycle in $G[M]$ is therefore also a cycle in $G - S$. Thus, since $G[M]$ contains a cycle, so does $G - S$, a contradiction. S cannot exist, and we must have $I_G(M) = \emptyset$. □

Lemma 4.15. *Halting Rule 4.5 is correct.*

Proof. Let (G, M) be an instance to which Halting Rule 4.5 applies, then G is acyclic. In order to show that Halting Rule 4.5 is correct, we must show that $I_G(M) = \{\emptyset\}$. Clearly, if G is acyclic, then $G - \emptyset$ is acyclic, and thus $\emptyset \in I_G(M)$. Since $\emptyset \in I_G(M)$, there cannot exist any $S \neq \emptyset$ such that $S \in I_G(M)$, since $\emptyset \subset S$. Thus, $I_G(M) = \{\emptyset\}$, and Halting Rule 4.5 is correct. □

Lemma 4.16. *Halting Rule 4.6 is correct.*

Proof. Let (G, M) be an instance to which Halting Rule 4.6 applies, then G contains K_4 as an induced subgraph. In order to show that Halting Rule 4.6 is correct, we must show that $I_G(M) = \emptyset$. Suppose not, and let $S \in I_G(M)$. Let $C \subseteq V(G)$ such that $G[C]$ is isomorphic to K_4. Observe that since S is independent, $|S \cap C| \leq 1$. Suppose $S \cap C = \emptyset$, then $G[C]$ is a subgraph of $G - S$. However, $G[C]$ is isomorphic to K_4, and $G - S$ therefore contains a cycle, a contradiction. Suppose instead that $S \cap C = \{x\}$, and note that $G[C \setminus \{x\}]$ is isomorphic to K_3. Since $G[C \setminus \{x\}]$ is a subgraph of $G - S$, $G - S$ contains a cycle. We have $S \in I_G(M)$, yet in every scenario $G - S$ contains a cycle. Thus, S cannot exist, and $I_G(M)$ must therefore be empty. □

4.3.2 Branching Rules

Let $I = (G, M)$ be an instance, and let $v \in U$. When we *branch* on v, we output two subproblems I_1 and I_2, where

$$I_1 = (G - \{v\}, M \cup N(v))$$
$$I_2 = (G, M \cup \{v\})$$

We shall refer to I_1 as the subproblem obtained by *selecting* v, and to I_2 as the subproblem obtained by *discarding* v. We will now describe the branching rules of EnumMifvs.

Branching Rule 4.7. *If there exists some vertex $v \in U$ such that $d(v) \geq 3$ and $m_v > 0$, then branch on v.*

Branching Rule 4.8. *If there exists some vertex $v \in U$ such that $d(v) \geq 3$ and there is some $u \in N(v)$ such that $d(u) = 2$, then branch on v.*

Branching Rule 4.9. *If there exists some $v \in U$ such that $d(v) \geq 3$ and $N[v] \neq C_v$, where C_v is the connected component of G containing v, then branch on v.*

Branching Rule 4.10. *If there exists some $v \in U$ such that $d(v) = 2$ and $u_v > 0$, then branch on v.*

Lemma 4.17. *Branching rules 4.7, 4.8, 4.9, and 4.10 are safe.*

Proof. Let $\mathcal{B}$ be one of branching rules 4.7, 4.8, 4.9 or 4.10. Let $I = (G, M)$ be an instance to which $\mathcal{B}$ applies, and let $v \in U$ be the vertex we branch on, then $\mathcal{B}$ outputs $I_1 = (G - \{v\}, M \cup N_G(v))$ when we select v, and $I_2 = (G, M \cup \{v\})$ when we discard v. We let $g_1(S') = S' \cup \{v\}$ be the function which maps a solution of I_1 to a solution of I, and $g_2(S) = S$ be the function which maps a solution of I_2 to a solution of I. Let $S \in I_G(M)$. We must show that there either exists some $S' \in I_{G-\{v\}}(M \cup N_G(v))$ such that $g_1(S') = S$, or that we have $S \in I_G(M \cup \{v\})$. There are two cases which we must consider:

Case 1: $v \in S$**.** Let $S' = S \setminus \{v\}$, and note that $g_1(S') = S$. We will show that $S' \in I_{G-\{v\}}(M \cup N_G(v))$. We will first show that S' is an $(M \cup N_G(v))$-ifvs in $G - \{v\}$. Now, if $G - S$ is acyclic, then so is $(G - \{v\}) - S'$, and S' is therefore a feedback vertex set in $G - \{v\}$. Since S is independent, S' is independent as well. Furthermore, since $v \in S$, we know that S must be disjoint from $N_G(v)$, and thus that S' must be disjoint from $N_G(v)$ as well. Thus, since $S \cap M = \emptyset$, we have $S' \cap M \cup N_G(v) = \emptyset$. We see that S is an $(M \cup N_G(v))$-ifvs in $G - \{v\}$. We must now show that S' is a *minimal* $(M \cup N_G(v))$-ifvs in $G - \{v\}$. Suppose not, and let $X' \subset S'$ such that X' is a $(M \cup N_G(v))$-ifvs in $G - \{v\}$. Let $X = g_1(X') = X' \cup \{v\}$, and note that since $X' \subset S'$, we have that $X \subset S$. We shall show that X is an M-ifvs in G, which will contradict our assumption that S is a minimal M-ifvs in G. Observe that since

$$(G - \{v\}) - X' = G - X' \cup \{v\} = G - X$$

and since $(G - \{v\}) - X'$ is acyclic, $G - X$ is acyclic as well. Since $v \in S$, we know that $v \notin M$. Therefore, since $X' \cap (M \cup N_G(v)) = \emptyset$, we have $X \cap (M \cup N_G(v)) = \emptyset$ as well. Since $X \cap N_G(v) = \emptyset$, and since X' is independent, X must also be independent. We see that X is an M-ifvs in G, a contradiction. We must have $S \in I_{G-\{v\}}(M \cup N_G(v))$.

Case 2: $v \notin S$**.** Since S is disjoint from M, and since $v \notin S$, we know that S is disjoint from $M \cup \{v\}$ as well. Thus, S is an $(M \cup \{v\})$-ifvs in G. Suppose $S \notin I_G(M \cup \{v\})$, and let $X \subset S$ such that X is an $(M \cup \{v\})$-ifvs in G. Clearly, if X is an $(M \cup \{v\})$-ifvs in G, then X is an M-ifvs in G as well, contradicting our assumption that S is a minimal M-ifvs in G. We must have $S \in I_G(M \cup \{v\})$.

We see that if $S \in I_G(M)$, then there either exists some $S' \in I_{G-\{v\}}(M \cup N_G(v))$ such that $g_1(S') = S$, or we have $S \in I_G(M \cup \{v\})$, which is what we had to prove. We conclude that branching rules 4.7, 4.8, 4.9 and 4.10 are safe. □

Claim 4.18. *Let $I = (G, M)$ be an instance to which none of rules 4.1-4.7 apply, and let $v \in U$ such that $d(v) \geq 3$, then none of rules 4.1-4.6 will apply to the instance $(G, M \cup \{v\})$.*

Proof. We let $I' = (G, M \cup \{v\})$. We will give a proof by contradiction. Suppose some rule $\mathcal{R}$ among rules 4.1-4.6, *does* apply to I'. Observe that since we were unable to apply Branching Rule 4.7 to I, we know that $m_v = 0$. Clearly, $\mathcal{R}$ cannot be Reduction Rule 4.1 or Reduction Rule 4.3, because by Claim 4.9, G contains no self-loops and no multi-edges, and $\delta(G) \geq 2$. Furthermore, rule $\mathcal{R}$ cannot be Reduction Rule 4.2, because $m_v = 0$. Since $G[M]$ is acyclic, and since G contains no self-loops and no multi-edges, and further, since $m_v = 0$, we know that $G[M \cup \{v\}]$ must be acyclic as well. In other words, $\mathcal{R}$ cannot be Halting Rule 4.4. Lastly, we know that G is not acyclic, and that G does not contain K_4 as an induced subgraph, because neither Halting Rule 4.5 nor Halting Rule 4.6 were applicable to I. Therefore, halting rules 4.5 and 4.6 cannot be applied to I' either. We have exhausted all cases, and see that none of rules 4.1-4.6 can be applied to I'. □

Claim 4.19. *Let $I = (G, M)$ be an instance to which none of rules 4.1-4.8 apply, and let $v \in U$. If $d(v) \geq 3$, then $\delta(G[C_v]) \geq 3$, where C_v is the connected component of G containing v.*

Proof. For the sake of obtaining a contradiction, let us assume that there is some vertex $u \in C_v$ such that $d(u) < 3$. We know that $\delta(G) \geq 2$, from Claim 4.9, and therefore also that $d(u) = 2$. Let P be a path between u and v, and let x be the first *unmarked* vertex along P which has degree at least 3. Note that x must exist, since the vertex v is in P. Consider the vertex z occuring immediately before x in P. Since x was the first unmarked vertex of degree at least 3 to occur along P, we know that either $z \in M$, or $d(z) = 2$. However, note that if $z \in M$, then Branching Rule 4.7 is applicable to x, and if $d(z) = 2$, then Branching Rule 4.8 is applicable to x. By assumption, none of rules 4.1-4.8 are applicable to I, and we therefore know that we must have $z \in U$, and that $d(z) \geq 3$. However, this contradicts our assumption that x was the first unmarked vertex of degree at least 3 to occur along P. We conclude that $\delta(G[C_v]) \geq 3$, and the proof is complete. □

Claim 4.20. *Let $I = (G, M)$ be an instance to which none of rules 4.1-4.9 apply, then we have $d(v) = 2$, for every vertex $v \in U$.*

Proof. By Claim 4.9, we have $\delta(G) \geq 2$. Suppose there is some vertex $v \in U$ such that $d(v) \geq 3$, and let C_v be the connected component G containing v. By Claim 4.19, we must have $\delta(G[C_v]) \geq 3$. Since Branching Rule 4.9 does not apply to I, there can be no vertex $u \in C_v$ such that u is not universal in C_v. That is, for every $u \in C_v$ we have $N[u] = C_v$. Since $\delta(G[C_v]) \geq 3$, $G[C_v]$ must therefore contain K_4 as an induced subgraph. This is not possible, because Halting Rule 4.6 would then have applied to I. Our assumption that there is some $v \in U$ such that $d(v) \geq 3$ leads to a contradiction. Thus, as $\delta(G) \geq 2$, we conclude that for every $v \in U$ we have $d(u) = 2$. □

Branching Rule Analysis

We will now analyze branching rules 4.7, 4.8, 4.9 and 4.10. We note that if one of branching rules 4.7, 4.8, 4.9 or 4.10 can be applied to (G, M), then rules 4.1-4.6 cannot. We therefore know that if we are able to apply some branching rule to (G, M), then by Observation 4.8, M is an independent set, and by Claim 4.9, G contains no self-loops and no multi-edges, and $\delta(G) \geq 2$.

Let $I = (G, M)$ be an instance, and let $v \in U$ be some vertex. When we select v, v is removed from the graph, and the neighborhood of v is marked. Thus, when we select v, the measure is reduced by $(u_v + 1)w_1 - u_v w_2$. When we discard v, v is marked, and we obtain a marked edge (u, v) for every $u \in N_G(v) \cap M$. Since v has m_v marked neighbors, Reduction Rule 4.2 will be applied successively m_v times after we discard v, and each application decreases the measure by w_2. Thus, when we discard v, the measure is reduced by $w_1 + (m_v - 1)w_2$.

Branching Rule 4.7
Suppose that Branching Rule 4.7 is applied to some instance $I = (G, M)$, and let v be the vertex we branch on, then $d(v) \geq 3$ and $m_v > 0$. Since selecting and discarding v reduces the measure by $(u_v + 1)w_1 - u_v w_2$ and $w_1 + (m_v - 1)w_2$, respectively, the branching vectors corresponding to Branching Rule 4.7 are

$$\begin{aligned} (3w_1 - 2w_2, w_1) \quad & \text{for } u_v \geq 2, m_v = 1 \\ (2w_1 - w_2, w_1 + w_2) \quad & \text{for } u_v \geq 1, m_v = 2 \\ (w_1, w_1 + 2w_2) \quad & \text{for } u_v \geq 0, m_v \geq 3 \end{aligned}$$

Branching Rule 4.8
Consider now an application of Branching Rule 4.8 to an instance $I = (G, M)$, and let $v \in U$ be the vertex we branch on, then $d(v) \geq 3$ and there is at least one vertex $u \in N_G(v)$ such that $d(u) = 2$. Observe that since Branching Rule 4.8 applies to I, we know that Branching Rule 4.7 does not, and therefore that $m_v = 0$. Correspondingly, we have $u_v = d(v)$. We let n_2 denote the number of neighbors of v which have degree 2.

Let (G', M') be the instance obtained by selecting v, and observe that for every vertex $u \in N_G(v)$ we have $d_{G'}(u) = 1$. Thus, after we select v, Reduction Rule 4.1 will apply n_2 times, and each application will remove some marked vertex from the graph. Therefore, when we select v we will reduce the measure by

$$(u_v + 1)w_1 - u_v w_2 + n_2 w_2 = (d(v) + 1)w_1 - d(v)w_2 + n_2 w_2$$

When we discard v, the measure is reduced by $w_1 - w_2$, since $m_v = 0$. We consider two cases with respect to n_2:

Case 1: $n_2 = d(v)$**.** Since $n_2 = d(v) \geq 3$, selecting v reduces the measure by at least

$$\begin{aligned}(d(v)+1)w_1 - d(v)w_2 + n_2 w_2 &= (d(v)+1)w_1 - d(v)w_2 + d(v)w_2 \\ &= (d(v)+1)w_1 \\ &\geq 4w_1\end{aligned}$$

The branching vector corresponding to the case of $n_2 = d(v)$ is

$$(4w_1, w_1 - w_2)$$

Case 2: $n_2 < d(v)$**.** By definition of Branching Rule 4.8, we have $n_2 \geq 1$. Now, since $n_2 \geq 1$ and $d(v) \geq 3$, selecting v will reduce the measure by at least

$$(d(v)+1)w_1 - d(v)w_2 + n_2 w_2 \geq 4w_1 - 2w_2$$

Let $I' = (G, M \cup \{v\})$ be the instance obtained by discarding v. When we discard v, v is marked and the measure decreases by $w_1 - w_2$. Observe that by Claim 4.18, none of rules 4.1-4.6 will apply to I'. Since $n_2 < d(v)$, there must be some $u \in N(v)$ such that $d(u) \geq 3$. Thus, after discarding v, we know that the algorithm will branch on some vertex $u \in N(v)$ which satisfies $d(u) \geq 3$ and $m_u = 1$. That is, after discarding v, we know that the algorithm will $(3w_1 - 2w_2, w_1)$-branch on some vertex $u \in N(v)$. By addition of branching vectors, the vector

$$(4w_1 - 2w_2, 4w_1 - 3w_2, 2w_1 - w_2)$$

is the branching vector corresponding to the case of $n_2 < d(v)$.

Branching Rule 4.9

We will now analyze Branching Rule 4.9. Let $I = (G, M)$ be an instance to which Branching Rule 4.9 applies, let $v \in U$ be the vertex we branch on, and let C_v be the connected component of G containing v. We have that $d(v) \geq 3$ and that $N[v] \neq C_v$. We note that since we were unable to apply Branching Rule 4.7 to I, we know that $m_v = 0$, and therefore also that $u_v = d(v)$. Observe that by Claim 4.19, we have $\delta(G[C_v]) \geq 3$.

Let $I' = (G, M \cup \{v\})$ be the instance obtained by discarding v. When we discard v, v is marked and the measure decreases by $w_1 - w_2$. After discarding v, there will be some $u \in N(v)$ for which $d(u) \geq 3$ and $m_u = 1$. We note that by Claim 4.18, none of rules 4.1-4.6 will apply to I'. Therefore, after we discard v, the algorithm will $(3w_1 - 2w_2, w_1)$-branch on some $u \in N(v)$ which satisfies $d(u) \geq 3$ and $m_u = 1$. The branching vector corresponding to discarding v and then branching on u is the vector

$$(4w_1 - 3w_2, 2w_1 - w_2)$$

Let $N^2(v)$ denote the set of vertices at distance 2 from v in G, i.e.,

$$N^2(v) = \left(\bigcup_{u \in N(v)} N(u) \right) \setminus N[v]$$

Since $N[v] \neq C_v$ and $\delta(G[C_v]) \geq 3$, we know that $N^2(v) \neq \emptyset$, and that for every $w \in N^2(v)$ we have $d(w) \geq 3$. When we select v, the measure decreases by

$$(d(v)+1)w_1 - d(v)w_2 \geq 4w_1 - 3w_2$$

We shall consider the subproblem $I' = (G', M')$ obtained by selecting v.

Claim. *If some rule $\mathcal{R}$ among rules 4.1-4.6 applies to I', then $\mathcal{R}$ must be Reduction Rule 4.2.*

Proof. Since we have $\delta(G[C_v]) \geq 3$, we know that $d_{G'}(u) \geq 2$ for all $u \in N_G(v)$, which implies that we cannot apply Reduction Rule 4.1 to I'. It is further implied that $\delta(G') \geq 2$, and consequently that G' must contain some cycle. Therefore, Halting Rule 4.5 cannot be applied to I' either. We note that by Claim 4.9, G does not contain any self-loops or multi-edges, and for that reason, neither does G', which implies that Reduction Rule 4.3 does not apply to I'. Clearly, G' cannot contain K_4 as an induced subgraph when G does not contain K_4 as an induced subgraph, and Halting Rule 4.6 is therefore not applicable to I'. Thus far, we have established that none of rules 4.1, 4.3, 4.5 or 4.6 can be applied to I'. Suppose for the sake of obtaining a contradiction that Halting Rule 4.4 is applicable to I', but Reduction Rule 4.2 is not, and let C be some cycle in $G'[M']$. We know that C cannot be a self-loop, because G' does not contain any self-loops. Consequently, there must exist $x, y \in M'$ such that x and y participate in C and $(x, y) \in E(G')$. However, this implies that Reduction Rule 4.2 is applicable to I', which we have assumed not to be the case. We have obtained our contradiction, and conclude that if some rule $\mathcal{R}$ among rules 4.1-4.6 applies to I', then $\mathcal{R}$ must be Reduction Rule 4.2. □

We will now consider two cases with respect to I':

Case 1: Reduction Rule 4.2 does *not* apply to I' When Reduction Rule 4.2 does *not* apply to I', we know that the algorithm will apply Branching Rule 4.7 to I' and branch on some $w \in N^2(v)$ which satisfies $d_{G'}(w) \geq 3$ and $m_w > 0$. We provide an illustration of this scenario in Fig. 4.5. Recall from our analysis of Branching Rule 4.7 that the branching vectors corresponding to branching on w are

$$\begin{aligned} (3w_1 - 2w_2, w_1) &\quad \text{for } u_w \geq 2, m_w = 1 \\ (2w_1 - w_2, w_1 + w_2) &\quad \text{for } u_w \geq 1, m_w = 2 \\ (w_1, w_1 + 2w_2) &\quad \text{for } u_w \geq 0, m_w \geq 3 \end{aligned}$$

The branching vectors corresponding to selecting v and then branching on w are therefore

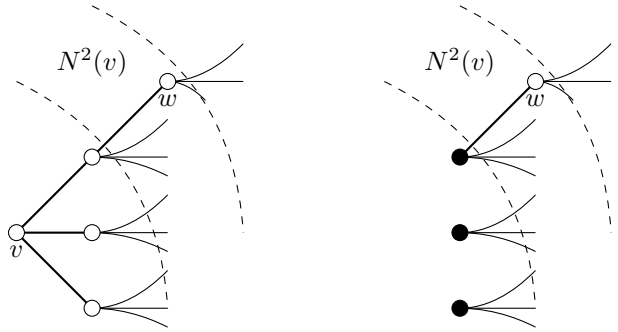
$$\begin{aligned}(7w_1 - 5w_2, 5w_1 - 3w_2)\\ (6w_1 - 4w_2, 5w_1 - 2w_2)\\ (5w_1 - 3w_2, 5w_1 - w_2)\end{aligned}$$

Hence, the branching vectors of Branching Rule 4.9 corresponding to the case in which Reduction Rule 4.2 does *not* apply to I' are

$$\begin{aligned}(7w_1 - 5w_2, 5w_1 - 3w_2, 4w_1 - 3w_2, 2w_1 - w_2) &\quad \text{for } u_w \geq 2, m_w = 1\\ (6w_1 - 4w_2, 5w_1 - 2w_2, 4w_1 - 3w_2, 2w_1 - w_2) &\quad \text{for } u_w \geq 1, m_w = 2\\ (5w_1 - 3w_2, 5w_1 - w_2, 4w_1 - 3w_2, 2w_1 - w_2) &\quad \text{for } u_w \geq 0, m_w \geq 3\end{aligned}$$

Case 2: Reduction Rule 4.2 *does* apply to I'. Since an application of Reduction Rule 4.2 decreases the measure by w_2, we know that selecting v will reduce the measure by $4w_1 - 3w_2 + w_2 = 4w_1 - 2w_2$. Thus, the branching vector of Branching Rule 4.9 corresponding to the case in which Reduction Rule 4.2 *does* apply to I' is

$$(4w_1 - 2w_2, 4w_1 - 3w_2, 2w_1 - w_2)$$



(a) The instance $I = (G, M)$.

(b) The subproblem $I' = (G', M')$ obtained by selecting v

Figure 4.5: When Reduction Rule 4.2 does *not* apply to I', we know that the algorithm will apply Branching Rule 4.7 to I' and branch on some $w \in N^2(v)$ which satisfies $d_{G'}(w) \geq 3$ and $m_w > 0$.

Branching Rule 4.10

Finally, let $I = (G, M)$ be an instance, and consider an application of Branching Rule 4.10 to I. Let $v \in U$ be the vertex we branch on, then $d(v) = 2$ and $u_v \in \{1, 2\}$.

Case 1: $u_v = 1$. Let $x, y \in N(v)$ such that $x \in M$ and $y \in U$. By Claim 4.20 we have $d(y) = 2$. Since $d(y) = 2$, selecting v will reduce the measure by $2w_1$, as the removal of v from the graph will cause Reduction Rule 4.1 to remove y in the next step as well. When we discard v the measure is reduced by $w_1 + (m_v - 1)w_2 = w_1$. Thus, the branching vector corresponding to the case in which $u_v = 1$ is

$$(2w_1, w_1)$$

Case 2: $u_v = 2$. Let $x, y \in N(v)$, then $x, y \in U$. By Claim 4.20 we have $d(x) = d(y) = 2$. Therefore, by selecting v we will reduce the measure by $3w_1$, as x and y will be removed from the graph by Reduction Rule 4.1 following the removal of v. When we discard v, the measure is reduced by $w_1 - w_2$. Thus, the branching vector corresponding to the case in which $u_v = 2$ is

$$(3w_1, w_1 - w_2)$$

Branching Vectors		
Branching Rule	Case	Branching Vector
4.7	$u_v \geq 2,\ m_v = 1$ $u_v \geq 1,\ m_v = 2$ $u_v \geq 0,\ m_v \geq 3$	$(3w_1 - 2w_2, w_1)$ $(2w_1 - w_2, w_1 + w_2)$ $(w_1, w_1 + 2w_2)$
4.8	$n_2 = d(v)$ $n_2 < d(v)$	$(4w_1, w_1 - w_2)$ $(4w_1 - 2w_2, 4w_1 - 3w_2, 2w_1 - w_2)$
4.9	$u_w \geq 2, m_w = 1$ $u_w \geq 1, m_w = 2$ $u_w \geq 0, m_w \geq 3$ Case 2	$(7w_1 - 5w_2, 5w_1 - 3w_2, 4w_1 - 3w_2, 2w_1 - w_2)$ $(6w_1 - 4w_2, 5w_1 - 2w_2, 4w_1 - 3w_2, 2w_1 - w_2)$ $(5w_1 - 3w_2, 5w_1 - w_2, 4w_1 - 3w_2, 2w_1 - w_2)$ $(4w_1 - 2w_2, 4w_1 - 3w_2, 2w_1 - w_2)$
4.10	$u_v = 1$ $u_v = 2$	$(2w_1, w_1)$ $(3w_1, w_1 - w_2)$

Table 4.1: Branching rules and their possible branching vectors, parameterized by w_1 and w_2.

See Table 4.1 for a summary of branching rules and their possible branching vectors, parameterized by w_1 and w_2.

4.3.3 Special Instances

In Section 3.2.3 we introduced special instances, as defined by Agrawal et al. in [2, 12]. Let (G, M) be an instance, and recall that if G is connected, and $\delta(G) \geq 2$, and for every $v \in U$ we have $d(v) = 2$, and M and U are both independent, then we say that (G, M) is a *special instance*. Furthermore, recall

that if (G, M) is a special instance, then for $S \subseteq U$,

$$E_S = \{(x, y) \mid x, y \in N(v), v \in S\}$$

is the set of edges corresponding to S, and $G^{sp} = (M, E_U)$ is the special graph corresponding to (G, M). We keep multi-edges in E_S.

We will now show that if (G, M) is an instance to which none of rules 4.1-4.10 apply, then for every connected component C of G, the instance $(G[C], M \cap C)$ is a special instance. We will also introduce the final halting rule of EnumMifvs, which the algorithm will apply to instances in which every connected component corresponds to a special instance, and further, we shall obtain an upper bound the number the number of solutions of such instances. We note that lemmata 4.21, 4.22 and 4.27 and their proofs have been copied from Agrawal et al. [12].

Lemma 4.21. *[2, 12] Let $I = (G, M)$ be a special instance, and let $S \subseteq U$, then S is an M-ifvs in G if and only if $G^{sp} - E_S$ is a forest.*

Proof. For the forward direction, suppose that S is an M-ifvs in G, but $G^{sp} - E_S$ has a cycle $C = (m_0, m_1, \ldots, m_l, m_0)$. For every edge $(m_i, m_{(i+1) \bmod (l+1)})$, for $i = 0, 1, \ldots, l$, there is a vertex $u_i \in U \setminus S$ such that $N(u_i) = \{m_i, m_{(i+1) \bmod (l+1)}\}$. Therefore, whenever $(m_0, m_1, \ldots, m_l, m_0)$ is a cycle in $G^{sp} - E_S$, we have that $(m_0, u_0, m_1, \ldots, m_l, u_0)$ is a cycle in $G - S$. Thus, the existence of C implies that $G - S$ has a cycle, which contradicts our assumption that S is an M-ifvs in G, and the forward direction therefore holds.

For the reverse direction, suppose $G^{sp} - E_S$ is a forest, and that S is *not* an M-ifvs in G. Since S is not an M-ifvs in G, $G - S$ must contain some cycle C. Since U and M are both independent, the cycle C must alternate between marked and unmarked vertices. Let $u \in U \setminus S$ be some unmarked vertex participating in C, and let $m_1, m_2 \in M$ be the neighbors of u in C, then $N(u) = \{m_1, m_2\}$, and thus $(m_1, m_2) \in E_{U \setminus S}$, where $E_{U \setminus S} = E(G^{sp} - E_S)$. Therefore, by removing all unmarked vertices from C, we obtain a cycle C' in $G^{sp} - E_S$, which is impossible, since $G^{sp} - E_S$ is a forest. We conclude that if $G^{sp} - E_S$ is a forest, then S is an M-ifvs in G. Both directions of the proof hold, and the proof is complete. □

Lemma 4.22. *[2, 12] Let $I = (G, M)$ be a special instance, and let $S \in I_G(M)$, then $E_{U \setminus S}$ forms a spanning tree in G^{sp}.*

Proof. First, recall that G^{sp} is connected, by Corollary 3.22. Since S is an M-ifvs in G, we have that $G^{sp} - E_S$ is a forest, by Lemma 4.21. If $G^{sp} - E_S$ is connected as well, then $G^{sp} - E_S$ is a tree, which means that $E_{U \setminus S}$ forms a spanning tree in G^{sp}. We will show that $G^{sp} - E_S$ is connected. Suppose not, and let $C \subseteq M$ be one of at least two components in $G^{sp} - E_S$. Consider the partition $(C, M \setminus C)$ of $V(G^{sp})$. There must be some edge $(m_1, m_2) \in E_U$ such that $m_1 \in C$ and $m_2 \in M \setminus C$, otherwise the graph G^{sp} would not be connected.

Let u be the vertex corresponding to the edge $e_u = (m_1, m_2)$, that is, let $u \in U$ such that $N(u) = \{m_1, m_2\}$. Since $G^{sp} - E_S$ is disconnected, we must have $e_u \in E_S$, and therefore also $u \in S$. Note however that by adding the edge e_u to $G^{sp} - E_S$, we connect two acyclic components, which *cannot* introduce a cycle. That is, $G^{sp} - (E_S \setminus \{e_u\}) = G^{sp} - E_{S \setminus \{u\}}$ is a forest, and by Lemma 4.21, $S \setminus \{u\}$ is therefore an M-ifvs in G, which contradicts the minimality of S. We conclude that $G^{sp} - E_S = (M, E_{U \setminus S})$ is both acyclic and connected, and thus that $E_{U \setminus S}$ forms a spanning tree in G^{sp}. □

Lemma 4.23. *Let $I = (G, M)$ be a special instance, and let $S \subseteq U$. If $E_{U \setminus S}$ forms a spanning tree in G^{sp}, then $S \in I_G(M)$.*

Proof. Suppose $(M, E_{U \setminus S})$ is a tree, but $S \notin I_G(M)$. By Lemma 4.21, since $(M, E_{U \setminus S}) = G^{sp} - E_S$ is a forest, we have that S is an M-ifvs in G. Therefore, since $S \notin I_G(M)$, S must not be minimal. Let $X \subset S$ such that X *is* a minimal M-ifvs in G. Observe that when $X \in I_G(M)$, we have that $E_{U \setminus X}$ forms a spanning tree in G^{sp}, by Lemma 4.22. That is, $(M, E_{U \setminus X})$ is a tree. However, since $X \subset S$, we know that $|X| < |S|$, and in turn that

$$|M| - 1 = \left|E_{U \setminus S}\right| < \left|E_{U \setminus X}\right| = |M| - 1$$

which is not possible. Since our assumption that $S \notin I_G(M)$ leads to a contradiction, we must have $S \in I_G(M)$ whenever $E_{U \setminus S}$ forms a spanning tree in G^{sp}. The proof is complete. □

Corollary 4.24. *Let $I = (G, M)$ be a special instance, then there is a one-to-one correspondence between $I_G(M)$ and spanning trees in G^{sp}.*

Corollary 4.24 follows from lemmata 4.22 and 4.23.

Lemma 4.25. *Let (G, M) be a special instance, and let G^{sp} be the special graph corresponding to (G, M). Let $\zeta(G^{sp})$ denote the set of all spanning trees in G^{sp}, then*

$$I_G(M) = \{S \mid E_{U \setminus S} \in \zeta(G^{sp}), S \subseteq U\}$$

Lemma 4.25 follows from lemmata 4.22 and 4.23.

In the following, let EnumST be an algorithm which takes as input a graph H, and then enumerates all spanning trees of H. Further, let C_i denote the i'th component of G, and let $G_i = G[C_i]$, $M_i = M \cap C_i$, and $U_i = C_i \setminus M_i$.

Halting Rule 4.11. *If (G_i, M_i) is a special instance, for all G_i, then compute*

$$\zeta_i = \{S_i \mid E_{U_i \setminus S_i} \in \textsc{EnumST}((M_i, E_{U_i})), S_i \subseteq U_i\}$$

where (M_i, E_{U_i}) is the special graph corresponding to (G_i, M_i), for $i = 1, 2, \ldots, k$, k being the number of connected components in G, and output the set

$$\{S \mid S = \bigcup_{i=1}^{k} S_i, S_i \in \zeta_i\}$$

Lemma 4.26. *Halting Rule 4.11 is correct.*

Proof. Let (G, M) be an instance to which Halting Rule 4.11 applies, and suppose that G has $k \geq 1$ connected components, then the instance (G_i, M_i) is special, for $i = 1, 2, \ldots, k$. If $S \in I_G(M)$, then $S = \bigcup_{i=1}^{k} S_i$, where $S_i \subseteq U_i$ is a minimal M_i-ifvs in G_i. Thus,

$$I_G(M) = \{S \mid S = \bigcup_{i=1}^{k} S_i, S_i \in I_{G_i}(M_i)\}$$

When applied to (G, M), Halting Rule 4.11 first computes

$$\zeta_i = \{S_i \mid E_{U_i \setminus S_i} \in \textsc{EnumST}((M_i, E_{U_i})), S_i \subseteq U_i\}$$

for $1 \leq i \leq k$, and then outputs the set

$$\{S \mid S = \bigcup_{i=1}^{k} S_i, S_i \in \zeta_i\}$$

Since (G_i, M_i) is a special instance with corresponding special graph (M_i, E_{U_i}), we know that $\zeta_i = I_{G_i}(M_i)$, by Lemma 4.25. Halting Rule 4.11 is therefore correct. □

Lemma 4.27. *[2, 12] Let $I = (G, M)$ be a special instance, and let $n = |V(G)|$, then*

$$|I_G(M)| \leq \left(\frac{2\alpha}{1-\alpha}\right)^{(1-\alpha)n}$$

where $|M| = (1-\alpha)n$, and $\frac{1}{2} \leq \alpha \leq 1$.

Lemma 4.27 follows from the proof of Lemma 3.28 and Corollary 4.24. That is, one can show that Lemma 4.27 holds by giving a proof near identical to the proof of Lemma 3.28.

Lemma 4.28. *Let $x \in \mathbb{R}$ such that $x \geq 1$, and suppose that $|I_G(M)| \leq x^{\mu(G,M)}$ whenever (G, M) is a special instance, then it holds that for any instance (G, M) to which Halting Rule 4.11 applies, we have $|I_G(M)| \leq x^{\mu(G,M)}$.*

Proof. We will prove by induction on $\mu(G, M)$ that the claim holds. Our base case occurs when (G, M) is a special instance, in which case the claim holds by assumption. Suppose that for any $l < \mu$, we have that $|I_G(M)| \leq x^l$, whenever (G, M) is an instance of measure $\mu(G, M) \leq l$ and Halting Rule 4.11 applies to (G, M). Let (G, M) be an instance of measure $\mu(G, M) = \mu$, and suppose that Halting Rule 4.11 is applicable to (G, M). We must consider two cases:

Case 1: G is connected. Since G is connected, the instance (G, M) is special, and we know by assumption that $|I_G(M)| \leq x^\mu$.

Case 2: G is disconnected. Suppose that G has $k \geq 2$ connected components, and note that $\mu(G_i, M_i) < \mu$, for $i = 1, 2, \ldots, k$. Let $S \in I_G(M)$, then $S = \bigcup_{i=1}^{k} S_i$, where $S_i \in I_{G_i}(M_i)$. We therefore have

$$|I_G(M)| \leq \prod_{i=1}^{k} |I_{G_i}(M_i)| \leq \prod_{i=1}^{k} x^{\mu(G_i, M_i)} \leq x^{\Sigma_{i=1}^{k} \mu(G_i, M_i)}$$

where

$$\begin{aligned} x^{\Sigma_{i=1}^{k} \mu(G_i, M_i)} &\leq x^{\Sigma_{i=1}^{k} w_1 |U_i| + w_2 |M_i|} \\ &\leq x^{w_1 \Sigma_{i=1}^{k} |U_i| + w_2 \Sigma_{i=1}^{k} |M_i|} \\ &\leq x^{w_1 |U| + w_2 |M|} \\ &\leq x^{\mu} \end{aligned}$$

That is, we have $|I_G(M)| \leq x^\mu$, and the claim holds.

We have $|I_G(M)| \leq x^\mu$ in both cases, and therefore conclude that the lemma holds. □

Claim 4.29. *Let $I = (G, M)$ be an instance to which none of rules 4.1-4.10 apply, then $(G[C], M \cap C)$ is a special instance, for every connected component C of G.*

Proof. By Claim 4.9, we know that $\delta(G) \geq 2$. Further, since none of rules 4.1-4.9 are applicable to I, we know by Claim 4.20 that for every $v \in U$, we have $d(v) = 2$. Observe that since Branching Rule 4.10 does not apply to I, we know that for every $v \in U$ we have $d(v) = m_v = 2$, which implies that U is an independent set. By Observation 4.8, M is an independent set as well. We have established that U and M are both independent, that $\delta(G) \geq 2$, and that for any $v \in U$, we have $d(v) = 2$. Therefore, for every connected component C of G, the instance $(G[C], M \cap C)$ is special. This concludes the proof. □

4.3.4 An Improved Upper Bound

Lemma 4.30. *Let I be an instance, then some rule of* EnumMifvs *is applicable to I.*

Proof. If one of rules 4.1-4.10 apply to I, then the claim holds, and there is nothing to prove. Suppose none of rules 4.1-4.10 apply to I. We then know

that for every connected component C of G, $(G[C], M \cap C)$ is a special instance, by Claim 4.29. Consequently, Halting Rule 4.11 will apply to I. The proof is complete. □

Lemma 4.31. *Let (G, M) be a nice instance, and let (G', M') be an instance output by some reduction rule or branching rule of* EnumMifvs *when applied to (G, M), then (G', M') is a nice instance.*

Proof. Observe that the only rule of EnumMifvs which introduces new edges is Reduction Rule 4.2. However, Reduction Rule 4.2 will only contract an edge (u, v) if we have $u, v \in M$. We therefore know that whenever an application of Reduction Rule 4.2 introduces a new edge e, at least one endpoint of e is marked. Thus, no rule of EnumMifvs can introduce an edge for which both endpoints are unmarked. Therefore, since (G, M) is a nice instance, (G', M') must be a nice instance as well. This concludes the proof. □

Lemma 4.32. *Let G be a graph on n vertices, and let $M \subseteq V(G)$ such that (G, M) is a nice instance, then $|I_G(M)| \leq 1.7229^n$.*

Proof. We have presented a branching algorithm EnumMifvs which enumerates all minimal M-ifvs of a nice instance (G, M). By lemmata 4.10, 4.12 and 4.13, all reduction rules are safe, and by Lemma 4.17, all branching rules are safe. Further, by lemmata 4.14, 4.15, 4.16 and 4.26, all halting rules are correct. We know that for any instance I, some rule of EnumMifvs will apply to I, by Lemma 4.30.

We let the measure of an instance (G, M) be $\mu(G, M) = 272\,|U| + 151\,|M|$. We present the branching vectors of EnumMifvs and their corresponding branching factors in Table 4.2, for $w_1 = 272$ and $w_2 = 151$. Let $x = 1.002002$, and note that for each branching vector $\vec{b}$, we have $\tau(\vec{b}) \leq x$.

Let (G, M) be an instance to which some halting rule applies. We will show that $|I_G(M)| \leq x^{\mu(G,M)}$. If Halting Rule 4.4, Halting Rule 4.5 or Halting Rule 4.6 applies to (G, M), then

$$|I_G(M)| \leq 1 \leq x^{\mu(G,M)}$$

Suppose Halting Rule 4.11 applies to (G, M). We will make use of Lemma 4.28 in order to show that we have $|I_G(M)| \leq x^{\mu(G,M)}$. However, we must first show that for any special instance $I' = (G', M')$ we have $|I_{G'}(M')| \leq x^{\mu(G',M')}$. We let n be the number of vertices in G'. Let $\frac{1}{2} \leq \alpha \leq 1$, and let $|M'| = (1 - \alpha)n$, then $|U'| = \alpha n$, where $U' = V(G') \setminus M'$. We have

$$\begin{aligned}
\mu(G', M') &= 272\,|U'| + 151\,|M'| \\
&= 272\alpha n + 151(1 - \alpha)n \\
&= n(121\alpha + 151)
\end{aligned}$$

By Lemma 4.27,

$$|I_{G'}(M')| \leq \max_{\frac{1}{2}\leq\alpha\leq 1}\left\{\left(\frac{2\alpha}{1-\alpha}\right)^{(1-\alpha)n}\right\} \leq \max_{\frac{1}{2}\leq\alpha\leq 1}\left\{\left(\frac{2\alpha}{1-\alpha}\right)^{\frac{(1-\alpha)\mu(G',M')}{121\alpha+151}}\right\}$$

We have $\max_{\frac{1}{2}\leq\alpha\leq 1}\left\{\left(\frac{2\alpha}{1-\alpha}\right)^{\frac{1-\alpha}{121\alpha+151}}\right\} = 1.002002 \leq x$, thus $|I_{G'}(M')| \leq x^{\mu(G',M')}$.

Since $|I_{G'}(M')| \leq x^{\mu(G',M')}$, for any special instance (G', M'), we know that if Halting Rule 4.11 applies to (G, M), then we have $|I_G(M)| \leq x^{\mu(G,M)}$, by Lemma 4.28. We see that if (G, M) is an instance to which some halting rule applies, then $|I_G(M)| \leq x^{\mu(G,M)}$.

By Lemma 2.1, we have

$$|I_G(M)| \leq 1.002002^{\mu(G,M)}$$

Thus, for any nice instance (G, M),

$$|I_G(M)| \leq 1.002002^{\mu(G,M)} \leq 1.002002^{272n} \leq 1.7229^n$$

where $n = |V(G)|$. This concludes the proof. □

Branching Vectors				
Branching Rule	Case	Branching Vector	c^{μ}	c^{272n}
4.7	$u_v \geq 2,\ m_v = 1$	$(514, 272)$	1.001827^{μ}	1.6430^n
	$u_v \geq 1,\ m_v = 2$	$(393, 423)$	1.001701^{μ}	1.5878^n
	$u_v \geq 0,\ m_v \geq 3$	$(272, 574)$	1.001719^{μ}	1.5954^n
4.8	$n_2 = d(v)$	$(1088, 121)$	1.001599^{μ}	1.5444^n
	$n_2 < d(v)$	$(786, 635, 393)$	1.001898^{μ}	1.6749^n
4.9	$u_w \geq 2, m_w = 1$	$(1149, 907, 635, 393)$	1.002002^{μ}	1.7229^n
	$u_w \geq 1, m_w = 2$	$(1028, 1058, 635, 393)$	1.001978^{μ}	1.7117^n
	$u_w \geq 0, m_w \geq 3$	$(907, 1209, 635, 393)$	1.001983^{μ}	1.7143^n
	Case 2	$(786, 635, 393)$	1.001898^{μ}	1.6749^n
4.10	$u_v = 1$	$(544, 272)$	1.001771^{μ}	1.6181^n
	$u_v = 2$	$(816, 121)$	1.001927^{μ}	1.6883^n

Table 4.2: Branching vectors and their corresponding branching factors, for measure $\mu(I) = 272\,|U| + 151\,|M|$.

Similarly to the proof of Lemma 3.31, we found the values of w_1 and w_2 in the proof of Lemma 4.32 through a programmatic approach.

Theorem 4.33. *There are at most* 1.7229^n *minimal independent feedback vertex sets in any graph on* n *vertices.*

Theorem 4.33 follows from Observation 4.7 and Lemma 4.32. We note that since all reduction rules and branching rules of EnumMifvs can be implemented to run in time polynomial in the measure of the input instance, and since the spanning trees of a graph can be enumerated with polynomial delay, we can enumerate all minimal independent feedback vertex sets of an n-vertex graph in time $\mathcal{O}^*(1.7229^n)$.

Remark 4.34. *Observe that the only bad branching vectors of* EnumMifvs *are the branching vectors corresponding to Branching Rule 4.9, more specifically, the case of Branching Rule 4.9 in which Reduction Rule 4.2 does not apply to the instance obtained by selecting v. If we wish to extend* EnumMifvs *in an attempt to achieve a tigher upper bound, then we must remove these branching vectors from the system.*

Chapter 5

Conclusion

Using the technique of branching in conjunction with a non-trivial measure, we have given improved bounds on both the maximum number of minimal connected vertex covers and the maximum number of minimal independent feedback vertex sets in a graph. We have shown that there are at most $2 \cdot 1.7076^n$ minimal connected vertex covers in a graph on n vertices, and that these can be enumerated in time $\mathcal{O}^*(1.7076^n)$. Moreover, we have shown that the maximum number of minimal independent feedback vertex sets in a graph on n vertices is at least 1.5067^n, but no more than 1.7229^n, and further, that the set of all minimal independent feedback vertex sets of an n-vertex graph can be enumerated in time $\mathcal{O}^*(1.7229^n)$. We conclude this thesis with a few open problems and some suggestions for future work.

5.1 Open Problems

The lower bound on the maximum number of minimal connected vertex covers in an n-vertex graph is 1.51978^n, by a result of Ryland [11]. Can we tighten the gap between the lower bound 1.51978^n and the upper bound $2 \cdot 1.7076^n$ on the maximum number of minimal connected vertex covers in a graph on n vertices? Likewise, can we tighten the gap between the lower bound 1.5067^n and the upper bound 1.7229^n on the maximum number of minimal independent feedback vertex sets in an n-vertex graph?

5.2 Future Work

It is possible that for some other choice of measure function, we can obtain better upper bounds without having to extensively modify the EnumMcvc or EnumMifvs algorithm. Alternatively, with respect to obtaining a better upper bound on the number of minimal connected vertex covers in a graph, we can try to find some way to avoid Branching Rule 3.8, case of $n_2 < d(v)$. We are quite confident that if we can eliminate the corresponding branching vector from the system of branching vectors of EnumMcvc, then we can adjust the weights of the measure and obtain a better upper bound. Similarly, consider the branching vectors of EnumMifvs corresponding to the case of Branching Rule 4.9 in which Reduction Rule 4.2 does *not* apply to the instance obtained by selecting v. These branching vectors attain substantially greater branching factors than the other branching vectors of EnumMifvs. If we can eliminate these branching vectors from the system, then we should be able to further improve the upper bound on the number of minimal independent feedback vertex sets by adjusting the weights of the measure.

Bibliography

[1] P. A. Golovach, P. Heggernes, and D. Kratsch, "Enumeration and maximum number of minimal connected vertex covers in graphs," *CoRR*, vol. abs/1602.07504, 2016.

[2] A. Agrawal, S. Gupta, S. Saurabh, and R. Sharma, "Improved Algorithms and Combinatorial Bounds for Independent Feedback Vertex Set," in *11th International Symposium on Parameterized and Exact Computation (IPEC 2016)* (J. Guo and D. Hermelin, eds.), vol. 63 of *Leibniz International Proceedings in Informatics (LIPIcs)*, (Dagstuhl, Germany), pp. 2:1–2:14, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.

[3] J. W. Moon and L. Moser, "On cliques in graphs," *Israel Journal of Mathematics*, vol. 3, pp. 23–28, Mar 1965.

[4] F. V. Fomin, F. Grandoni, A. V. Pyatkin, and A. A. Stepanov, "Combinatorial bounds via measure and conquer: Bounding minimal dominating sets and applications," *ACM Trans. Algorithms*, vol. 5, pp. 9:1–9:17, Dec. 2008.

[5] F. V. Fomin, F. Grandoni, A. V. Pyatkin, and A. A. Stepanov, "Bounding the number of minimal dominating sets: A measure and conquer approach," in *Algorithms and Computation* (X. Deng and D.-Z. Du, eds.), (Berlin, Heidelberg), pp. 573–582, Springer Berlin Heidelberg, 2005.

[6] D. Lokshtanov, M. Pilipczuk, and S. Saurabh, "Below all subsets for minimal connected dominating set," *CoRR*, vol. abs/1611.00840, 2016.

[7] J.-F. Couturier, P. Heggernes, P. van 't Hof, and D. Kratsch, "Minimal dominating sets in graph classes: Combinatorial bounds and enumeration," *Theoretical Computer Science*, vol. 487, pp. 82 – 94, 2013.

[8] F. V. Fomin, S. Gaspers, A. V. Pyatkin, and I. Razgon, "On the minimum feedback vertex set problem: Exact and enumeration algorithms," *Algorithmica*, vol. 52, pp. 293–307, Oct 2008.

[9] J.-F. Couturier, P. Heggernes, P. van 't Hof, and Y. Villanger, "Maximum number of minimal feedback vertex sets in chordal graphs and cographs," in *Computing and Combinatorics* (J. Gudmundsson, J. Mestre, and T. Viglas, eds.), (Berlin, Heidelberg), pp. 133–144, Springer Berlin Heidelberg, 2012.

[10] S. Gaspers and M. Mnich, "Feedback vertex sets in tournaments," in *Algorithms – ESA 2010* (M. de Berg and U. Meyer, eds.), (Berlin, Heidelberg), pp. 267–277, Springer Berlin Heidelberg, 2010.

[11] I. Ryland, "New lower bounds on the maximum number of minimal connected vertex covers," 2017.

[12] A. Agrawal, S. Gupta, S. Saurabh, and R. Sharma, "Improved Algorithms and Combinatorial Bounds for Independent Feedback Vertex Set." Personal communication, 2017.

[13] K. H. Rosen and K. Krithivasan, *Discrete mathematics and its applications*. McGraw-Hill Education, 2015.

[14] M. Sipser, *Introduction to the Theory of Computation*. International Thomson Publishing, 1st ed., 1996.

[15] F. V. Fomin and D. Kratsch, *Exact exponential algorithms*. Springer, 2010.

[16] G. Grimmett, "An upper bound for the number of spanning trees of a graph," *Discrete Mathematics*, vol. 16(4), pp. 323 – 324, 1976.

[17] B. D. McKay and A. Piperno, "Practical graph isomorphism, {II}," *Journal of Symbolic Computation*, vol. 60, no. 0, pp. 94 – 112, 2014.