Markov-Chain-Based Heuristics for the Feedback Vertex Set Problem for Digraphs

I n a u g u r a l - D i s s e r t a t i o n

zur

Erlangung des Doktorgrades

der Mathematisch-Naturwissenschaftlichen Fakultät

der Universität zu Köln

vorgelegt von

Mile Lemaić
aus Wesseling

Köln 2008

Berichterstatter:	**Prof. Dr. Ewald Speckenmeyer**
	HD Dr. Bert Randerath

Tag der mündlichen Prüfung: 30. Juni 2008

Abstract

A feedback vertex set (FVS) of an undirected or directed graph $G = (V, A)$ is a set $F \subset V$ such that $G - F$ is acyclic. The minimum feedback vertex set problem asks for a FVS of G of minimum cardinality whereas the weighted minimum feedback vertex set problem consists of determining a FVS F of minimum weight $w(F)$ given a weight function $w : V \to \mathbb{R}_+$. Both problems are NP-hard [28]. Nethertheless, they have been found to have applications in many fields. So one is naturally interested in approximation algorithms.

While most of the existing approximation algorithms for feedback vertex set problems rely on local properties of G only, this thesis explores strategies that use global information about G in order to determine good solutions. The pioneering work in this direction has been initiated by Speckenmeyer [56]. He demonstrated the use of Markov chains for determining low cardinality FVSs. Based on his ideas, new approximation algorithms are developed for both the unweighted and the weighted minimum feedback vertex set problem for digraphs. According to the experimental results presented in this thesis, these new algorithms outperform all other existing approximation algorithms.

An additional contribution, not related to Markov chains, is the identification of a new class of digraphs $G = (V, A)$ which permit the determination of an optimum FVS in time $O(|V|^4)$. This class strictly encompasses the completely contractible graphs [36].

Zusammenfassung

Ein Feedback-Vertex-Set (FVS) eines ungerichteten oder gerichteten Graphen $G = (V, A)$ ist eine Menge $F \subset V$ derart, dass $G - F$ azyklisch ist. Das Minimum-Feedback-Vertex-Set Problem verlangt nach einem FVS minimaler Kardinalität, wohingegen das gewichtete Minimum-Feedback-Vertex-Set Problem aus dem Bestimmen eines FVSs F minimalen Gewichtes $w(F)$ besteht, ausgehend von einer Gewichtsfunktion $w : V \to \mathbb{R}_+$. Beide Probleme sind NP-schwer [28]. Dennoch haben sie in vielen Bereichen Anwendung gefunden. Daher ist man naturgemäß an Approximationsalgorithmen interessiert.

Während die meisten der vorhandenen Approximationsalgorithmen für Feedback-Vertex-Set Probleme lediglich auf lokalen Eigenschaften von G beruhen, untersucht die vorliegende Dissertation Strategien, die zur Bestimmung guter Lösungen globale Informationen über G benutzen. Die Pionierarbeit in dieser Richtung wurde von Speckenmeyer [56] geleistet. Er demonstrierte den Einsatz von Markovketten zur Bestimmung von FVSs kleiner Kardinalität. Auf seinen Ideen basierend werden sowohl für das ungewichtete als auch für das gewichtete Minimum-Feedback-Vertex-Set Problem für gerichtete Graphen neue Algorithmen entwickelt. Gemäß den experimentellen Ergebnissen dieser Dissertation übertreffen die neuen Algorithmen leistungsmäßig alle bisher vorhandenen Approximationsalgorithmen.

Ein weiterer Beitrag, der keinen Bezug zu Markovketten hat, ist die Benennung einer neuen Klasse von gerichteten Graphen $G = (V, A)$, die die Bestimmung optimaler FVSs in Zeit $O(|V|^4)$ erlauben. Diese Klasse enthält die 'completely contractible graphs' [36] als echte Teilklasse.

Contents

Mathematical symbols

$\mathbb{N}$	Set of natural numbers, i.e. $\mathbb{N} = \{1, 2, 3, \ldots\}$
$\mathbb{Q}$	Set of rational numbers
$\mathbb{R}$	Set of real numbers
$\mathbb{Q}_+$	Set of positive rational numbers
$\mathbb{Q}_{\geq 0}$	Set of non-negative rational numbers
$\mathbb{R}_+$	Set of positive real numbers
$\mathbb{R}_{\geq 0}$	Set of non-negative real numbers
$\mathcal{P}(X)$	Power set of set X
$\lvert X\rvert$	Cardinality of set X or the absolute value of X if X is a real number
$\lVert x\rVert_1$	Taxicab norm of vector $x = (x_i)_{i\in I}$, i.e. $\lVert x\rVert_1 = \sum_{i\in I} \lvert x_i\rvert$
$\lVert x\rVert_\infty$	Maximum norm of vector $x = (x_i)_{i\in I}$, i.e. $\lVert x\rVert_\infty = \max\{\lvert x_i\rvert : i \in I\}$
$A \dot\cup B$	Disjoint union of sets A and B
$f_{\mid S}$	Restriction of the map $f : X \to Y$ to the set $S \subset X$

Algorithms

CHAPTER 1

Introduction

"B-but, my dear f-fellows," said Feodor Simeonovich, having diligently deciphered the handwriting. "This is B-Ben B-Beczalel's problem! Didn't C-Cagliostro prove th-that it had no s-solution?"
"We know that it has no solution, too," said Junta, bristling immediately. "But we wish to learn how to solve it"
"H-how strangely you r-reason, C-Cristo. ... H-how can you look for a solution, where it d-does not exist? It's s-some sort of n-nonsense."
"Excuse me, Feodor, but it's you who are reasoning strangely. It's nonsense to look for a solution if it already exists. We are talking about how to deal with a problem that has no solution. [...] "

Monday Begins on Saturday,
Arkadi and Boris Strugatski

Think of the task of testing a VLSI circuit for defects, (re)evaluating the probability that a hypothesis may be true based on evidence using Bayesian inference, or preventing deadlocks in computer systems, i.e. situations in which two or more processes mutually wait for each other to release particular resources. What do these seemingly unrelated tasks have in common? Their core problems and many others can be formulated as feedback vertex set problems [35, 38, 47].

The feedback vertex set of an undirected or directed graph is a subset of vertices whose removal destroys (or breaks) all cycles of the graph. Feedback arc sets are defined analogously. For practically all applications one is interested in determining feedback vertex sets of minimum cardinality or minimum weight in the case of vertex-weighted graphs.

Given the variety of applications, one would expect feedback vertex set problems to be well analysed. Yet, the contrary is rather the case. Despite the well known fact that the problem of determining a minimum cardinality (resp. weight) feedback vertex set is NP-hard for both undirected and directed graphs [28], in [60] it is called the 'probably [...] least understood of the classic problems' — at least for digraphs. Of course, in the more than 20 years since that quote we have seen some progress. Approximation algorithms with provable performance-ratio have been developed and new classes of polynomially solvable graphs have been identified. However, particularly for the directed case, feedback set problems remain poorly understood compared to the SAT problem or the Travelling Salesman problem. This is especially true in view of the small number of existing heuristics for feedback set problems on digraphs. This thesis is intended to narrow the gap.

Almost all existing heuristic algorithms for the minimum feedback vertex set problem on digraphs base their decision which vertex v to take into a feedback vertex set of the digraph G on local properties of v, i.e. on the degree of v or on its neighbourhood. The problem with these approaches is that cycles — which shall be broken by a feedback vertex set — are global objects of G, so intuitively, it is obvious that local information about v is no good indicator of whether v belongs to a low cardinality feedback vertex set of G. While this problem is less acute on dense digraphs because the cycles tend to be small, it becomes dominant on sparse digraphs where cycle lengths are typically larger. This is the starting point of this thesis.

The algorithms developed in the present thesis attempt to overcome the mentioned deficiencies. Based on the ideas of Speckenmeyer [56] Markov chains can be associated to instances of the (weighted) minimum feedback vertex set problem in a natural way. Markov chains in turn take the global cycle structure of the problem instance into account, so that algorithms can be devised that do not solely rely on local properties of the digraph. These deterministic algorithms can be randomised to achieve even better results, which is also demonstrated. The distinction between deterministic and randomised algorithms is made as follows: In the course of execution, deterministic algorithms may have a choice and the chosen alternative may depend on the encoding of the input instance. But given the same encoding a deterministic algorithm will always produce the same output. For randomised algorithms this is not the case. They will typically produce different outputs although the inputs have been the same.

The minimum feedback vertex set problem on digraphs comprises an interesting feature. Like other classic NP-hard optimisation problems it concedes rules by which the size of the input instance can be reduced. These rules will be called digraph reductions. The most prominent and effective ones among them are those of Levy and Low [36]. The digraphs that can be fully reduced by these reductions are called completely contractable graphs. In addition, there are the digraph reductions of Smith and Walford [55] which are only theoretically relevant due to their large runtime. Besides these classic reductions, several new digraph reductions have been developed over the past decade [37, 44]. Although they are effective in practice, they do not lead to new polynomially solvable digraph classes because it has not been shown that they possess the Church-Rosser property. Another minor topic of the thesis is the formulation of four new digraph reductions. For a subset of these reductions we prove the Church-Rosser property, for the others we show that the property does not hold. The resulting class of digraphs, which strictly encompasses the completely contractable graphs, is denoted as DICLIQUE-1 reducible graphs.

The thesis is organised as follows: The rest of the chapter is devoted to fixing the notation and introducing the basic concepts with respect to feedback set problems. Chapter 2 gives an overview of the current state of the art. Some algorithms mentioned therein will motivate the design of our own ones. As these new algorithms make heavy use of Markov chains, Chapter 3 presents a brief introduction. Chapters 4 and 5 introduce the developed approximation algorithms which are all based on Markov chains. While Chapter 4 describes the deterministic algorithms, Chapter 5 deals with the randomised algorithms. Chapter 6 contains the experimental results for the algorithms of Chapters 4 and 5 and some concluding remarks. Appendix A introduces the new digraph reductions which generalise the Levy-Low reductions.

While Chapters 1–3 are reproductive, Chapters 4–6 as well as Appendix A are original. Exceptions to this framework are cited.

1.1 Notation

The notation used here is mainly adopted from [17]. As the title of the thesis indicates, the emphasis lies on digraphs, and so the terminology for undirected graphs will be omitted in this section. It will be presented when needed.

A digraph $G = (V, A)$ consists of a finite vertex set V and an arc set $A \subset V \times V$. Occasionally, for V and A the notation $V =: V(G)$ and $A =: A(G)$ will be used. Given an arc $a = (u, v) \in A$ we will say that a is directed from u to v whereby u being the **source** and v the **target** of a. Arc a is said to be **incident** to both u and v. The definition of a digraph implies that it cannot have **parallel arcs** – different arcs with the same sources and targets. However, a digraph can have **anti-parallel arcs**, i.e. arcs of the form (u, v) and (v, u). If for any arc $(u, v) \in A$ the digraph G also contains the anti-parallel arc $(v, u) \in A$, then G is called **symmetric**. An arc (v, v) is called a **loop**. A vertex $v \in V$ is said to be **loop-free** if $(v, v) \notin A$. If this holds for every vertex $v \in V$, G is called loop-free.

The sets

$$\mathrm{N}^-_G(v) := \{u \in V : (u, v) \in A\} \quad \text{and} \quad \mathrm{N}^+_G(v) := \{u \in V : (v, u) \in A\}$$

are said to be the set of all **predecessors** and the set of all **successors** of v, respectively. The **neighbours** $\mathrm{N}_G(v)$ of v are the union of predecessors and successors of v, i.e. $\mathrm{N}_G(v) := \mathrm{N}^-_G(v) \cup \mathrm{N}^+_G(v)$ while the **mates** $\mathrm{N}^=_G(v)$ of v are their intersection, i.e. $\mathrm{N}^=_G(v) := \mathrm{N}^-_G(v) \cap \mathrm{N}^+_G(v)$. The cardinalities

$$\mathrm{d}^-_G(v) := |\,\mathrm{N}^-_G(v)|, \quad \mathrm{d}^+_G(v) := |\,\mathrm{N}^+_G(v)| \quad \text{and} \quad \mathrm{d}^=_G(v) := |\,\mathrm{N}^=_G(v)|$$

are referred to as **in-**, **out-** and **matedegree** of v. The **degree** of v is defined as $\mathrm{d}_G(v) := \mathrm{d}^-_G(v) + \mathrm{d}^+_G(v)$. The index G will be omitted if the meaning is clear from the context. Furthermore, another digraph $G' = (V', A')$ is a **subgraph** of G (written as $G' \subseteq G$) if $V' \subseteq V$ and $A' \subseteq A$. The digraph **induced** by a vertex set $U \subseteq V$ is a special subgraph. It is defined as $G[U] := (U, A \cap U \times U)$.

A digraph $D = (V_D, A_D)$ is called a **directed clique (diclique)** of size $|V_D|$ if it is complete, which is to say $A_D = \{(u, v) \in V_D \times V_D : u \neq v\}$. For notational convenience the empty digraph and the digraph consisting of one vertex only are considered to be dicliques of size 0 and 1, respectively.

Finally, we introduce the **inverse digraph** G^{-1} of G defined by

$$G^{-1} := (V, A') \quad \text{with} \quad A' := \{(v,u) : (u,v) \in A\}.$$

It is obtained from G by altering the directions of the arcs of G.

A **path** $P = (v_0, \dots, v_k)$ is a sequence of vertices $v_i \in V$ $(i = 0, \dots, k)$ such that $(v_i, v_{i+1}) \in A$ for $i = 0, \dots, k-1$. It it said to be **simple** if all the vertices are distinct. In addition, the following notation is introduced:

$$\text{head}(P) := v_k, \quad \text{tail}(P) := v_0, \quad \text{length}(P) := k.$$

If $\text{head}(P) = \text{tail}(P)$, then P is called a **cycle**. Note that a loop is a cycle of length 1. A digraph which does not contain cycles is called **acyclic**. The cycle is simple if $v_i \neq v_j$ for $0 \leq i < j < k$. Like for digraphs the notation

$$V(P) := \{v_0, \dots, v_k\} \quad \text{and} \quad A(P) := \{(v_i, v_{i+1}) : 0 \leq i < k\}$$

will also be used for paths. Furthermore, the **acyclic vertices** and **acyclic arcs** of G are defined as

$$\begin{aligned} & V_{acyc}(G) := \{v \in V : v \notin V(C) : C \text{ any cycle in } G\} \\ \text{and} \quad & A_{acyc}(G) := \{a \in A : a \notin A(C) : C \text{ any cycle in } G\}, \end{aligned}$$

respectively.

To cope with feedback problems on digraphs one has to modify them. Thus, let us define some operations on digraphs beginning with the most basic ones, i.e. the **inclusion** and **deletion** of arcs and vertices. For a vertex v and an arc $a \in V \times V$ the following notation will be used:

$$\begin{aligned} G + a &:= (V, A \cup \{a\}) \\ G + v &:= (V \cup \{v\}, A) \\ G - a &:= (V, A \setminus \{a\}) \\ G - v &:= (V \setminus \{v\}, A \cap (V \setminus \{v\}) \times (V \setminus \{v\})) \end{aligned}$$

Figure 1.1 illustrates the deletion of a vertex v. Because the order in which vertices (resp. arcs) are included in G obviously does not matter, we shall write $G + S$ for a vertex (resp. arc) set $S = \{s_1, \dots, s_k\}$ instead of $G + s_1 + \dots + s_k$. By applying the same reasoning a similar notation will be used for the deletion.

Next, we introduce an operation $\circ$ defined for pairs consisting of a digraph G and a vertex $v \in V(G)$ yielding a digraph $G \circ v$, called the **exclusion** of v from G. Formally, it is defined as

$$G \circ v := G - v + \mathrm{N}_G^-(v) \times \mathrm{N}_G^+(v).$$

Informally speaking, v is removed and every predecessor of v is connected to every successor of v by an arc if that arc does not already exist. An example

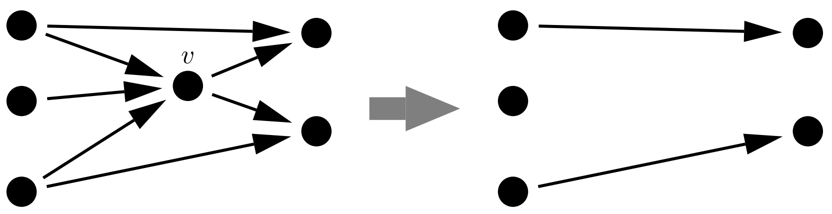

Figure 1.1: The deletion of v.

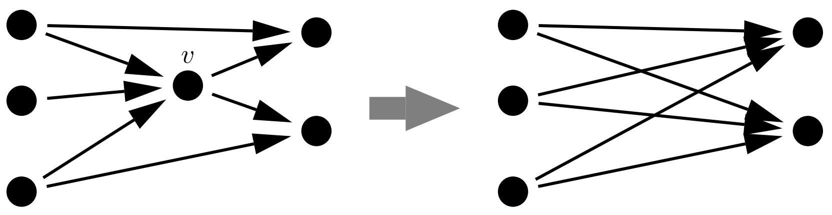

Figure 1.2: The exclusion of v.

is depicted in Figure 1.2. The exclude operation is known by other names in other sources (see e.g. [20, 37, 44]).

As can be seen from the definition, the exclusion of v is only feasible if v is loop-free, while the deletion and all other operations on G are always feasible. Thus, the exclusion as defined here might look artificial, but in the next section we will see a close relation to feedback set problems.

1.2 Basic problems and concepts

This section will present the basic problems treated in this thesis. In addition, some problem-based concepts will be introduced in order to facilitate the presentation of the ideas. Let us start with the definition of the feedback vertex and the feedback arc set of a digraph $G = (V, A)$.

Definition 1.1 (feedback vertex (resp. arc) set, redundancy). *Let $G = (V, A)$ be a digraph.*

1. *A set $F \subset V$ is called a **feedback vertex set (FVS)** of G if $G - F$ is acyclic.*
2. *A set $f \subset A$ is called a **feedback arc set (FAS)** of G if $G - f$ is acyclic.*

*A feedback vertex (resp. arc) set $F \subset V$ is said to be **minimal** if there is no proper subset $F' \subsetneq F$ such that F' is also a feedback vertex (resp. arc) set. Otherwise, if $F' \subsetneq F$ is a feedback vertex (resp. arc) set, any $x \in F \setminus F'$ is called **redundant**.*

Feedback vertex sets and feedback arc sets are closely related. To see this, consider the **line digraph** $L(G) = (V_L, A_L)$ of a digraph $G = (V, A)$. It is constructed as follows: We set the vertex set V_L of $L(G)$ as the arc set of G, i.e. $V_L := A$. Then, two arcs $a, b \in V_L$ are connected by an arc in $L(G)$ if the target of a is equal to the source of b, i.e.

$$A_L := \{(a, b) \in V_L \times V_L : a = (u, v) \vee b = (v, w)\}.$$

Having constructed the line digraph, one can see that any feedback arc set f of G is a feedback vertex set of $L(G)$. In other words, any feedback arc set problem can be seen as a feedback vertex set problem. This is the reason why the focus of this thesis will be on the latter problems.

But which problems will be considered here? The focus shall be on the **minimum feedback vertex set problem**, i.e. the problem of determining a feedback vertex set of minimal cardinality. In addition, the **weighted minimum feedback vertex set problem** is treated marginally. An instance of this problem is a **(vertex-)weighted digraph** $G = (V, A, w)$ with a positive **weight function** $w : V \to \mathbb{R}_+$. The task is to determine a feedback vertex set F of minimal **weight** $w(F) := \sum_{v \in F} w(v)$. In both cases a solution for these problems is called an **optimal** feedback vertex set.

The problems mentioned suggest the definition of the set $\mathcal{C}_G$ as

$$\mathcal{C}_G := \{C : C \text{ is a cycle in } G\}.$$

A cycle C is called **minimal** if there is no cycle C' with $V(C') \subsetneq V(C)$. One can easily see that a cycle $C = (v_0, \ldots, v_k, v_0)$ is minimal if and only if it is simple and there is no arc $(v_i, v_j) \in A$ $(i, j \in \{0, \ldots, k\})$ such that $(v_i, v_j) \notin A(C)$. Hence, a subset $\mathcal{M}_G \subset \mathcal{C}_G$ can be identified, namely

$$\mathcal{M}_G := \{C \in \mathcal{C}_G : C \text{ is minimal}\}.$$

Now, these definitions permit a characterisation of feedback vertex sets as the following proposition shows:

Proposition 1.2. *Let $G = (V, A)$ be a digraph. Then, $F \subset V$ is a feedback vertex set if and only if*

$$\forall C \in \mathcal{M}_G : F \cap V(C) \neq \emptyset.$$

This characterisation will be of use on several occasions throughout this thesis. A related characterisation also involving the set $\mathcal{M}_G$ is that of redundant vertices.

Proposition 1.3. *Let $F \subset V$ be a FVS of a digraph $G = (V, A)$. A vertex $v \in F$ is not redundant if and only if*

$$\exists C \in \mathcal{M}_G : F \cap V(C) = \{v\}.$$

Proof. Suppose $v \in F$ is not redundant, meaning that the digraph $G' := G - (F \setminus \{v\})$ is cyclic. So there is a minimal cycle $C \in \mathcal{M}_{G'} \subset \mathcal{M}_G$ for which $F \cap V(C) = \{v\}$ holds as F is a FVS of G.

Now, assume $v \in F$ to be redundant, which means $F' := F \setminus \{v\}$ is a FVS of G. Let $C \in \mathcal{M}_G$ be an arbitrary minimal cycle in G with $v \in V(C)$. Because F' is a FVS of G we have $F' \cap V(C) \neq \emptyset$ in view of Proposition 1.2. Hence

$$F \cap V(C) = (F' \dot\cup \{v\}) \cap V(C) = \underbrace{(F' \cap V(C))}_{\neq \emptyset} \dot\cup \underbrace{(\{v\} \cap V(C))}_{=\{v\}} \supsetneq \{v\}$$

holds and the assertion follows. □

Now, let us examine how the deletion and the exclusion affect the set $\mathcal{M}_G$. First, consider $\mathcal{M}_{G-v}$ for a vertex $v \in V$. Then, one immediately sees that $\mathcal{M}_{G-v} \subset \mathcal{M}_G$. More precisely,

$$\mathcal{M}_{G-v} = \{C \in \mathcal{M}_G : v \notin V(C)\}.$$

The situation is different when considering $\mathcal{M}_{G \circ v}$. Let $C = (v_0, \ldots, v_k, v_0) \in \mathcal{M}_G$ be a minimal cycle. The only interesting case is where $\mathrm{N}^-_G(v) \cap V(C) \neq \emptyset \neq \mathrm{N}^+_G(v) \cap V(C)$. Otherwise, C is also a cycle in $G \circ v$ and it remains minimal, thus $C \in \mathcal{M}_{G \circ v}$. So, consider the case $\mathrm{N}^-_G(v) \cap V(C) \neq \emptyset \neq \mathrm{N}^+_G(v) \cap V(C)$. If $v \in V(C)$, w.l.o.g. $v = v_k$ can be assumed, then because of the minimality of C we have $\mathrm{N}^-_G(v) \cap V(C) = \{v_{k-1}\}$ and $\mathrm{N}^+_G(v) \cap V(C) = \{v_0\}$. It follows that $C' := (v_0, \ldots, v_{k-1}, v_0)$ is a cycle in $G \circ v$ and moreover it is minimal, hence $C' \in \mathcal{M}_{G \circ v}$. Now, suppose $v \notin V(C)$. This scenario is depicted in Figure 1.3. After excluding v the cycle C is still a cycle in $G \circ v$ but not necessarily minimal anymore, i.e. there is a cycle $C' \in \mathcal{M}_{G \circ v}$ with $V(C') \subseteq V(C)$.

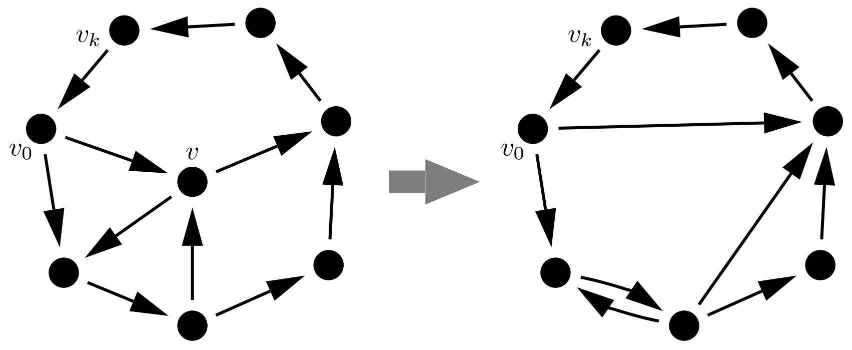

Figure 1.3: A minimal cycle before and after the exclusion of v.

These considerations can be summarised by the following proposition:

Proposition 1.4. *Let $G = (V, A)$ be a digraph and $v \in V$.*

1. *If F' is a feedback vertex set of $G - v$, then $F := F' \cup \{v\}$ is a feedback vertex set of G.*

2. *If $G \circ v$ is feasible and F a feedback vertex set of $G \circ v$, then F is also a feedback vertex set of G.*

Having defined the set $\mathcal{C}_G$ an equivalence relation $\sim$ on V can be defined as follows:

$$u \sim v \quad :\Longleftrightarrow \quad u = v \vee \exists C \in \mathcal{C}_G : \{u, v\} \subset V(C)$$

The digraphs induced by the corresponding equivalence classes are called **strongly connected components (SCCs)** of G. A component that has no entering arcs from other components is called **source SCC** whereas one that has no leaving arcs is called **sink SCC**. Let us denote the SCCs of G by $S_1, \ldots, S_r$. If $r = 1$, then G is said to be **strongly connected**. Because of the definition of $\sim$ for any $C \in \mathcal{M}_G$ there is a strongly connected component S_i $(1 \leq i \leq r)$ such that $V(C) \subset V(S_i)$. This means that in order to find an optimal feedback vertex set of G, each strongly connected component can be treated separately. So, throughout this thesis G will be assumed to be strongly connected unless otherwise stated.

Note that from the examination of $\mathcal{M}_{G\circ v}$, which lead to Proposition 1.4, it follows that $G \circ v$ is strongly connected if G is.

Another concept is the **cycle-domination** introduced in [55]. We will say a vertex $w \in V$ **cycle-dominates** another vertex $v \in V$ (written as $v \rightsquigarrow w$) if

$$\forall C \in \mathcal{C}_G : \quad (v \in V(C) \Rightarrow w \in V(C)).$$

If $v \rightsquigarrow w$ and $w \rightsquigarrow v$, then v and w are said to be **mutually cycle-dominated** and we shall write $v \leftrightsquigarrow w$. If v is dominated by w, by definition every cycle broken by v is also broken by w. This means that in any minimal FVS F of G with $v \in F$, the vertex v may be replaced by w and F remains a FVS of G. Note that because F is assumed to be minimal, v and w cannot both be part of F. So, this replacement leaves the cardinality of F unchanged. It follows that in order to solve the minimum feedback vertex set problem, dominated vertices may be excluded without impairing optimality. Note however that this does not apply to the weighted minimum feedback vertex set problem because $w(F)$ may increase when replacing v with w. Also note that vertices possessing a loop are never dominated, so this concept is in sync with the feasibility of the exclude operation.

1.3 A linear programming formulation for the weighted minimum feedback vertex set problem

In the previous section, the weighted minimum feedback vertex set problem was introduced. To state the problem as a linear program, let $G = (V, A, w)$ be an instance of this problem. We have seen that the weighted minimum feedback vertex set problem is a special weighted set cover problem where the sets consist of the vertex sets $V(C)$ for a cycle $C \in \mathcal{M}_G$. Thus, the standard zero-one integer linear programming (ILP) formulation

$$\begin{aligned}
\textbf{minimise} \quad & \sum_{v \in V} w(v) x_v & & \\
\textbf{subject to} \quad & \sum_{v \in V(C)} x_v \geq 1 & \forall C \in \mathcal{M}_G & \qquad (1.1)\\
& x_v \in \{0,1\} & \forall v \in V &
\end{aligned}$$

is tempting. Then, any optimal solution $(x_v)_{v \in V}$ of the ILP yields an optimal feedback vertex set F via

$$F := \{v \in V : x_v = 1\}. \tag{1.2}$$

However, there is a problem with this formulation. The size of $\mathcal{M}_G$ may grow exponentially leading to an exponential number of constraints in the ILP formulation. But there is an alternative ILP formulation requiring only a linear number of constraints which we cite from [11].

Let $n := |V|$. It is well known that acyclic digraphs $G = (V, A)$ can be sorted topologically, i.e. there is a linear ordering $d_v \in \{1, \ldots, n\}$ of the vertices $v \in V$ of G, such that the arcs of G are directed from left to right, i.e.

$$d_v - d_u \geq 1 \quad \forall (u,v) \in A. \tag{1.3}$$

Regardless whether or not (1.3) is satisfied, the vector $(x_v)_{v \in V}$ can be defined by

$$x_v := \begin{cases} 1 & \text{if } \exists u \in V : d_v - d_u < 1 \wedge (u,v) \in A \\ 0 & \text{else} \end{cases}. \tag{1.4}$$

Then, the set $F \subset V$ defined by (1.2) is obviously a FVS of G. Furthermore, because $d_v \in \{1, \ldots, n\}$ for all $v \in V$ we have

$$d_v - d_u \geq 1 - n \quad \forall (u,v) \in A. \tag{1.5}$$

Thus, by combining (1.4) and (1.5) we get

$$d_v - d_u + n x_v \geq 1 \quad \forall (u,v) \in A. \tag{1.6}$$

In [11] the converse direction is also shown: For an arbitrary FVS F of G and the vector $(x_v)_{v \in V}$ defined by

$$x_v := \begin{cases} 1 & \text{if } v \in F \\ 0 & \text{else} \end{cases}$$

there is a linear ordering $d_v \in \{1, \ldots, n\}$ of the vertices $v \in V$ satisfying (1.6). Thus the ILP

$$\begin{aligned}
\textbf{minimise} \quad & \sum_{v \in V} w(v) x_v & & \\
\textbf{subject to} \quad & d_v - d_u + n x_v \geq 1 & \forall (u,v) \in A & \\
& d_v \in \{0, \ldots n\} & \forall v \in V & \qquad (1.7)\\
& x_v \in \{0,1\} & \forall v \in V &
\end{aligned}$$

is an ILP formulation for the weighted minimum feedback vertex set problem instance $G = (V, A, w)$ which needs only $2|V| + |A|$ constraints. Actually, the constraint (1.7) can be replaced by $d_v \geq 0$ which leads to a mixed integer linear program (MILP) formulation.

Solving ILPs is NP-hard [28]. To determine a lower bound on the optimum value of the above ILP in polynomial time, the relaxation of the ILP can be solved. Unfortunately, this lower bound is always ≤ 1. But by adding some constraints of the form (1.1) better lower bounds can be obtained.

Chapter 2

The state of the art

Nothing learned is ever forgotten. . . we hold all our life experience deep within the recesses of our multifaceted minds. Our latent insight inexplicably surfaces and astonishes.

Chantell Van Erbe

In order to give an overview of the existing theory concerning the feedback set problems on digraphs, the first fact to be mentioned is certainly the NP-hardness of the minimum feedback vertex set problem which applies to both the undirected and the directed version [28]. As seen in section 1.2, the minimum feedback arc set problem on digraphs can be polynomially reduced to the directed version of the minimum feedback vertex set problem. In fact, there is also a polynomial reduction in the opposite direction [19] implying that the minimum feedback arc set problem on digraphs is NP-hard, too. The situation is different for the minimum feedback arc set problem for undirected graphs. This problem is the inverse of the maximum spanning tree problem which admits determining a solution in polynomial time. Thus, the minimum feedback arc set problem for undirected graphs can be solved in polynomial time, too.

Because the feedback vertex set problem on digraphs is NP-hard, any exact algorithm for this problem is expected to have an exponential runtime for general digraphs. Two of them are presented in section 2.1. However, when restricting the input digraphs conveniently, polynomial time algorithms for determining optimal FVSs can be obtained. This is described in section 2.1.2.

If computation time for determining a FVS of an arbitrary digraph is bounded by a polynomial, e.g., we have to restrict to approximation algorithms, typically coming up with suboptimal solutions. To evaluate their quality, the performance-ratio comes into play.

Definition 2.1 (r-approximation). *For the problem to minimise a function $w : \mathcal{D} \to \mathbb{R}_{\geq 0}$ let $O \in \mathcal{D}$ be an optimal solution. Then, a solution $X \in \mathcal{D}$ is a **r-approximation** of the minimisation problem if*

$$\frac{w(X)}{w(O)} \leq r.$$

Definition 2.2 (performance-ratio). *Let A be an approximation algorithm for the problem to minimise a function $w_I : \mathcal{D}_I \to \mathbb{R}_{\geq 0}$ given an input $I \in \mathcal{I}$ of A. Let $A(I)$ denote the solution determined by A with input I. Then, A is said* *to have **performance-ratio** r if for any instance $I \in \mathcal{I}$ the solution $A(I)$ is an r-approximation.*

The Definitions 2.1 and 2.2 will be mainly applied to feedback set problems: the function w returns the cardinality (resp. weight) of a given feedback set. The term r need not to be a real constant. It can be any real-valued function that depends on the particular (weighted) digraph.

In this thesis, heuristics are taken to mean approximation algorithms which either have no good performance-ratio (better than $O(|V|)$) or no good performance-ratio could yet be proved. Despite that, they typically produce near-optimal solutions. Accordingly, section 2.2 describes the existing approximation algorithms with reasonable performance-ratio while section 2.3 is devoted to heuristics.

2.1 Exact algorithms

As mentioned, the minimum feedback vertex set problem on digraphs is NP-hard. So, one can only hope to obtain polynomial time algorithms for this problem if one restricts the input instances to certain classes. This section discusses exact algorithms for both general digraphs and special classes.

For the minimum feedback arc set problem on a digraph $G = (V, A)$ there is an interesting note due to Lawler [34] in which he shows how to solve this problem in time $O(2^{|V|})$. His idea relies on the following dynamic programming approach: For a subset $U \subset V$ let f be an optimal FAS of $G[U]$. Because $G[U] - f$ is acyclic, there must be a vertex $u \in U$ with $\mathrm{d}^+_{G[U]-f}(u) = 0$ which means $g := \{(u, v) : v \in \mathrm{N}^+_{G[U]}(u)\} \subset f$. Hence, $f' := f \setminus g$ is an optimal FAS of $G[U] - g$. But f' is also an optimal FAS of $G[U] - u = G[U \setminus \{u\}]$ since $\mathrm{d}^+_{G[U]-g}(u) = 0$. Thus, if $\mathrm{fas}(U)$ denotes the cardinality of an optimal FAS of $G[U]$, we obtain the following equation:

$$\mathrm{fas}(U) = |f| = |f'| + |g| = \mathrm{fas}(U \setminus \{u\}) + \mathrm{d}^+_{G[U]}(u).$$

This leads to the identity

$$\mathrm{fas}(U) = \min\{\mathrm{fas}(U \setminus \{u\}) + \mathrm{d}^+_{G[U]}(u) : u \in U\}.$$

Thus, by computing $\mathrm{fas}(U)$ for all subsets $U \subset V$ in increasing order of cardinality of U using the above identity, it is possible to determine $\mathrm{fas}(V)$ in time $O(|\mathcal{P}(V)|) = O(2^{|V|})$.

2.1.1 General algorithms

2.1.1.1 Reductions

Algorithms for feedback set problems use digraph reductions in order to speed up the computation. In this context, by digraph reduction we mean a transformation of the digraph $G = (V, A)$ which results in a digraph $G' = (V', A')$ such that either $|V'| < |V|$ or $|A'| < |A|$ and that the knowledge of an optimal

FVS F' of G' permits a fast determination of an optimal FVS F of G, i.e. in polynomial time.

The five most straight forward reductions are described by Levy and Low in [36]. In the following we give a brief description of them whereby we assume $v \in V$ to be a vertex in G:

LOOP(v) If v has a loop, it must be member of any FVS of G. Thus, G is transformed into $G - v$ as for any FVS F' of $G - v$ the set $F' \cup \{v\}$ is a FVS of G.

IN0(v) If v is loop-free and has indegree 0, it cannot be part of any cycle. Hence, G is transformed into $G - v$ since both digraphs have the same minimal FVSs.

OUT0(v) If v is loop-free and has outdegree 0, G is transformed into $G - v$ with the same argumentation as in IN0(v).

IN1(v) If v is loop-free and has indegree 1, it has a unique predecessor u. Then, v is obviously cycle-dominated by u. As explained in section 1.2 any optimal FVS F of $G \circ v$ is also an optimal one of G. Thus, G is transformed into $G \circ v$.

OUT1(v) If v is loop-free and has outdegree 1, G is transformed into $G \circ v$ with a symmetric argumentation as in IN1(v).

In [36] it is also shown that these five reductions have the finite Church-Rosser property which is to say: if these reductions are applied in arbitrary order until no further reductions are possible, the resulting digraph is unique up to isomorphism.

Another reduction that decreases the cardinality of V is presented in [37]. There, a vertex $v \in V$ is called a core of a diclique if $G[v \cup \mathrm{N}_G(v)]$ is a diclique. Then, the CORE(v) reduction works as follows:

CORE(v) If v is a core of a diclique, $\mathrm{N}_G(v)$ is part of an optimal FVS. Hence, G is transformed into $G - \mathrm{N}_G(v)$ as any optimal FVS F' of $G - \mathrm{N}_G(v)$ yields the optimal FVS $F' \cup \mathrm{N}_G(v)$ of G.

In Appendix A new reductions will be presented which in some sense subsume all the above mentioned reductions except LOOP(v).

In addition, there are reductions which cause a decrease of $|A|$ only. Let $(u, v) \in A$. Then, a trivial one is the following:

ACYCLIC(u, v) If $(u, v) \in A_{acyc}(G)$, transform G into $G - (u, v)$. By definition of $A_{acyc}(G)$ the SCCs of G remain unchanged. Thus, G and $G-(u, v)$ share the same FVSs (see section 1.2).

All of the following arc deletion reductions base on the idea of detecting an arc $a \in A$ that is not contained in arc sets of minimal cycles. If a is such an arc, by Proposition 1.2 it can be deleted from G without impairing optimality. The first reduction of this kind is mentioned in [44]:

DOUBLE(u, v) If (u, v, u) is a cycle in G, define

$$B := \left(\bigcup_{\substack{C \in \mathcal{C}_G \\ \{u,v\} \in V(C)}} A(C) \right) \setminus \left(\bigcup_{C \in \mathcal{M}_G} A(C) \right).$$

The set B contains arcs of (nonminimal) cycles passing both u and v. On the other hand, the arcs in B are not part of any minimal cycles. Therefore G is transformed into $G - B$ because both have the same FVSs.

In [44] it is also shown how the arc set B can be determined in linear time. A similar idea is presented in [37]. There, for a given digraph G, the arc set $\mathrm{PIE} := \{(u, v) \in A : (v, u) \in A\}$ is defined. In other words, PIE is the union of all arc sets of cycles having length 2. The so called PIE reduction is the following:

PIE Let $B := A_{acyc}(G - \mathrm{PIE})$. An arc $a \in B$ cannot belong to a minimal cycle in G because by definition of B for any cycle $C \in \mathcal{C}_G$ with $a \in A(C)$ we have $A(C) \cap \mathrm{PIE} \neq \emptyset$ which means there is a cycle $C' = (x, y, x)$ in G with $A(C') \subset A(C) \cap \mathrm{PIE}$. Hence, as in DOUBLE($u, v$), G is transformed into $G - B$.

In [37], there is also the DOME(u, v) reduction described. As the previous reductions, it attempts to prove that the arc (u, v) cannot belong to a minimal cycle. The DOME(u, v) reduction presented in [37] works as follows:

DOME(u, v) An arc $(u, v) \in A \setminus \mathrm{PIE}$ is dominated if $\mathrm{N}^-_{G-\mathrm{PIE}}(u) \subset \mathrm{N}^-_G(v)$ or $\mathrm{N}^+_{G-\mathrm{PIE}}(v) \subset \mathrm{N}^+_G(u)$. It is proven that dominated arcs are never part of an arc set of a minimal cycle. So, if (u, v) is dominated, it is deleted, i.e. G is transformed into $G - (u, v)$.

Basically, all the mentioned reductions are admissible for the weighted minimum feedback vertex set problem with a weight function w. However, the reductions have to be adapted to cope with the weighted case. The main reason for this is that for two vertex sets $F, F' \subset V$, the fact $|F| \leq |F'|$ does not imply $w(F) \leq w(F')$. For a survey of reductions for weighted minimum feedback set problems we refer to [30].

2.1.1.2 The Smith-Walford algorithm

An early exact algorithm to solve the unweighted minimum feedback vertex set problem on digraphs was the Smith-Walford algorithm [55]. Basically, it attempts to determine an optimal FVS for an induced subgraph which can be safely added to the global FVS. Then, the locally optimal FVS is removed from the digraph and the process is repeated until the digraph becomes empty. To describe the algorithm in more detail, we will follow [38]. Some preliminary notation is needed.

So, let $G = (V, A)$ be the digraph for which an optimal FVS is to be determined. W.l.o.g. G may be assumed to be strongly connected. For an arbitrary subset $H \subset V$ let

$$A_{SW}(G, H) := G[H \cup V_{acyc}(G - H)]$$

```
function SmithWalford(digraph $G = (V, A)$, integer $k$) do

    integer $i$;
    vertex set $FVS$;
    vertex set $F$;

    $FVS \leftarrow \emptyset$;
    DoReductions($G$, $FVS$);

    for all SCCs $S$ of $G$ with $|S| > 1$ do
        for $i$ = $k$ to |V| do
            for each $F \subset V(S)$ with $|F| = i$ do
                if $F$ is W$_i$-set in $S$ do
                    $FVS \leftarrow FVS \cup F$;
                    $FVS \leftarrow FVS \cup$ SmithWalford($S - F$, 1);

                    process next SCC $S$ of $G$;
                od;
            od;
        od;
    od;

    return $FVS$;
od;
```

Algorithm 2.1: The Smith-Walford algorithm.

be the SW-associated digraph of H. It is induced by H and all vertices not belonging to cycles of $G - H$. Then, by definition, H is a FVS of $A_{SW}(G, H)$. If moreover H is an optimal FVS of $A_{SW}(G, H)$, it is said to be essential. In this case we know that H must be part of an optimal FVS of G. To see this, let F be an optimal FVS of $G_R := G - V(A_{SW}(G, H))$. If $F \cup H$ is a FVS of G, it is also optimal because $A_{SW}(G, H)$ and G_R have disjoint vertex sets. And indeed, $F \cup H$ is a FVS of G. Suppose it is not. Then, there is a cycle $C \in \mathcal{C}_G$ such that $(F \cup H) \cap V(C) = \emptyset$. It follows that $C \in \mathcal{C}_{G-H}$. Because F is a FVS of G_R we have $V_{acyc}(G - H) \cap V(C) \neq \emptyset$ which is a contradiction. If H is essential and $|H| \leq i$, it is called a W$_i$-set.

So, the Smith-Walford algorithm systematically tests every possible subset $H \subset V$ for being a W_i-set in ascending order of i. If H is detected to be essential it is added to the global FVS and $V(A_{SW}(G, H))$ is deleted from G. After each deletion the resulting digraph is simplified by applying the Levy-Low reductions (see section 2.1.1.1). After the simplification the digraph is decomposed into its strongly connected components and each component is resolved recursively. The complete algorithm is shown in Algorithm 2.1. The procedure `DoReductions(`G`, `FVS`)` modifies G according to the Levy-Low reduction rules and adds vertices to FVS if necessary. The algorithm is invoked by the call `SmithWalford(`G`, 1)`.

Regarding the runtime of the Smith-Walford algorithm, the worst case appears if the input digraph $G = (V, A)$ is a diclique. To see this, first note that none of the Levy-Low reductions is applicable in G (provided $|V| > 2$) before an essential set F has been found. Furthermore, from section 2.1.2 we know that F to be essential it must have cardinality $|V| - 1$ and this condition is also sufficient. According to [38] testing all vertex sets for being $\mathrm{W}_{|V|-2}$-sets takes time $O(|V|^{2|V|-3}|A|)$ which is also the total runtime as the time to confirm that F is a $\mathrm{W}_{|V|-1}$-set is negligible.

2.1.1.3 A branch and bound algorithm

The Smith-Walford algorithm – at least in its proposed form – is not capable to solve the weighted minimum feedback vertex set problem. In order to solve it exactly, one has to rely on branch and bound algorithms. Variants of this technique are proposed in [11, 37, 44]. We will only describe the main parts while elaborating on differences concerning the details only marginally.

The algorithm starts by decomposing the entire digraph G into its SCCs and treating each SCC separately. Then, a vertex $v \in V$ is chosen heuristically and an optimal FVS F_v with the constraint that v is included in F_v is determined recursively. After that, the branch and bound algorithm attempts to prove that F_v is indeed an optimal solution. To do so, another optimal FVS F is determined, but now with the restriction that v is not a member of F. If $w(F_v) \leq w(F)$, the attempt was successful and F_v is returned as an optimal FVS, otherwise F is returned. To determine F_v and F, according to Proposition 1.4, optimal FVSs of $G - v$ and $G \circ v$ are determined, respectively. Furthermore, after each deletion of a vertex the resulting digraph is decomposed into its SCCs. Recall that this decomposition is not necessary after an exclusion of a vertex since the digraph remains strongly connected as noted in section 1.2. In some situations the determination of F can be skipped. This is the case when a lower bound on $w(F)$ can be computed which proves that F_v is optimal. In order to obtain a payoff for the additional computation of the lower bound, it must be reasonable fast compared to the determination of F. The complete branch and bound algorithm is shown in Algorithm 2.2. It is invoked by the call `BranchAndBound_SCC(`G`)`.

One way to determine a lower bound of $w(F)$ is the standard approach based on the relaxation of an ILP formulation which is mentioned in section 1.3 and described in [11]. A more promising bounding method is described in [37]. First, the five Levy-Low reductions together with the reductions ACYCLIC(u, v), CORE(v), PIE and DOME(u, v) are applied to G as long as possible. During that process the lower bound, which was initially set to 0, is increased by 1 for each vertex added to the partial FVS. After that, a diclique D of size $k := |V(D)| \geq 3$ is attempted to be determined heuristically. In case of success, $V(D)$ is deleted from the digraph and the lower bound is increased by $k - 1$ (because at least $k - 1$ vertices are necessary to break all cycles of D – see Proposition A.7). In case of failure of the heuristic, a shortest cycle C is determined, $V(C)$ is deleted from the digraph and the lower bound is increased by 1. This process is repeated until the entire digraph becomes empty. Fundamentally, the same bounding technique can also be applied for the weighted

```
function BranchAndBound_SCC(weighted digraph $G = (V, A, w)$) do

    vertex set $F$;

    if $V = \emptyset$ do
        return $\emptyset$;
    od;

    $F$ ← $\emptyset$;

    for all SCCs $S$ of $G$ do
        $F$ ← $F$ ∪ BranchAndBound($S$);
    od;

    return $F$;
od;

function BranchAndBound(weighted digraph $G = (V, A, w)$) do

    vertex $v$;
    vertex set $F_v$;
    vertex set $F$;

    if $V = \emptyset$ do
        return $\emptyset$;
    od;

    for all $v \in V$ do
        if $(v, v) \in A$ do
            return $\{v\}$ ∪ BranchAndBound_SCC($G - v$);
        od;
        else if $\min(\mathrm{d}^-_G(v), \mathrm{d}^+_G(v)) = 0$ do
            return $\emptyset$;
        od;
        else if $\mathrm{d}^-_G(v) = 1$ and $w(v) \geq w(\mathrm{N}^-_G(v))$ do
            return BranchAndBound($G \circ v$);
        od;
        else if $\mathrm{d}^+_G(v) = 1$ and $w(v) \geq w(\mathrm{N}^+_G(v))$ do
            return BranchAndBound($G \circ v$);
        od;
    od;

    $v$ ← SelectVertex($G$);

    $F_v$ ← $\{v\}$ ∪ BranchAndBound_SCC($G - v$);

    if LowerBound($G \circ v$) < $w(F_v)$ do
        $F$ ← BranchAndBound($G \circ v$);
        if $w(F) < w(F_v)$
            return $F$;
    od;

    return $F_v$;
od;
```

Algorithm 2.2: The branch and bound algorithm.

minimum feedback vertex set problem on a weighted digraph $G = (V, A, w)$. Yet, concerning the update of the lower bound, one has to take into account the weight function w. While it is straight forward to increase the lower bound by $w(v)$ if LOOP(v) is applied, one has to be careful when dealing with a diclique D of size k. In this situation, the bound is increased by $w(F)$ where $F \subset V(D)$ is a minimum-weight vertex set with $|F| = k - 1$. Similarly, when removing $V(C)$ for a cycle $C \in \mathcal{C}_G$, the lower bound is raised by $\min\{w(v) : v \in V(C)\}$.

In Algorithm 2.2 only the application of the five Levy-Low reductions and the ACYCLIC(u, v) reduction is demonstrated. But in the same fashion any other reduction mentioned in section 2.1.1.1 can be incorporated into the branch and bound algorithm. For example, in [37] also the reductions CORE(v), PIE and DOME(u, v) are used along with the ones in Algorithm 2.2. In [44], the DOUBLE$(u,\ v)$ reduction is used together with a number of efficient digraph partitioning techniques which are beneficially applied to sparse digraphs only.

Apart from the bounding strategy, the vertex selection (`SelectVertex(`G`)` in Algorithm 2.2) is the other crucial part of the branch and bound algorithm. Generally, to select a vertex for branching, a rank r_v is computed for every $v \in V$ and a vertex with the highest rank is selected. In the weighted case a vertex $v \in V$ is picked that maximises the ratio $\frac{r_v}{w(v)}$. Concerning the choice of r_v, mainly local criterions are considered in literature. For example, in [22, 35] $r_v = \mathrm{d}^-_G(v) + \mathrm{d}^+_G(v)$ and $r_v = \mathrm{d}^-_G(v) \cdot \mathrm{d}^+_G(v)$ have been tried while in [37] the modification $r_v = \mathrm{d}^-_G(v) + \mathrm{d}^+_G(v) + 2\,\mathrm{d}^=_G(v)$ has been used. One exception of these locality-based ranks is presented in [46]. There, for every $v \in V$ a (random) minimal cycle $C_v \in \mathcal{M}_G$ is determined with $v \in V(C_v)$. Then, the rank $r_v = |\{C_u : v \in V(C_u)\}|$ is used.

2.1.2 Polynomial algorithms for special digraph classes

The exact algorithms considered so far are able to solve the (weighted) minimum feedback vertex set problem for general digraphs at the cost of an exponential runtime. However, if the input is restricted to special classes of digraphs, there are exact algorithms which run in polynomial time. Such classes are usually defined by the applicability of certain digraph reductions until the digraph under consideration becomes acyclic and thus an optimal FVS is determined.

The class of completely contractible graphs [36] is an example. This class consists of digraphs which can be reduced by the five Levy-Low reductions LOOP, IN0, OUT0, IN1 and OUT1 (cf. section 2.1.1.1) to the empty digraph. Because the Levy-Low reductions possess the finite Church-Rosser property, a straight forward algorithm which solves the minimum feedback vertex set problem for completely contractible graphs is to apply the Levy-Low reductions in arbitrary order until an (optimal) FVS is determined. In [36] it is shown how this algorithm can be implemented to run in time $O(|A| \log |V|)$ for a digraph $G = (V, A)$.

Another classic example of a class that is defined by particular digraph reductions represents the class of Smith-Walford-i reducible (SW$_i$ reducible) graphs. These digraphs are defined to be those for which the Smith-Walford algorithm

– modified in such a way that only W$_i$-sets (cf. section 2.1.1.2) are looked for – may come up with an optimal FVS. This definition does not automatically lead to a polynomial time algorithm for the minimum feedback vertex set problem on SW$_i$ reducible graphs because it is not clear that any admissible computation of the modified Smith-Walford algorithm will determine an optimal FVS. However, exactly that is shown to be true in [38]. There, it is also argued that the (modified) Smith-Walford algorithm runs in time $O(i|V|^{2i}|A|)$ on any SW$_i$ reducible graph $G = (V, A)$. Particularly, for Smith-Walford-one reducible graphs an optimal FVS can be determined in time $O(|V|^2|A|)$.

The SW$_i$ reducible graphs are a proper subset of the SW$_{i+1}$ reducible graphs. To see this, suppose $G = (V, A)$ to be a diclique of size $i + 2$. Let $H \subset V$ be any vertex set of cardinality $i + 1$. Then, we have $A_{SW}(G, H) = G$ and in view of Proposition A.7 H is a W$_{i+1}$-set and thus G is SW$_{i+1}$ reducible. Now, let $H' \subset V$ have a cardinality no more than i. In this case we get $A_{SW}(G, H') = G[H']$ which implies that H' is not an optimal FVS of $A_{SW}(G, H')$. So, G is not SW$_i$ reducible implying that the class of SW$_{i+1}$ reducible graphs and the class of SW$_i$ reducible graphs are indeed distinct.

The first nontrivial class of digraphs which permit a determination of an optimal FVS in polynomial time has been defined by Shamir [54]. In this context, *nontrivial* is to be understood in the sense that the definition of the class does not rely on any digraph reductions. Shamir denotes the class as reducible flow graphs. To understand its definition a basic knowledge of the depth-first search and related notation is assumed and can be reviewed in [14].

Definition 2.3 (flow graph). *A digraph $G = (V, A)$ is said to be a **flow graph** if there is a root vertex $v_0 \in V$ such that for every vertex $v \in V \setminus \{v_0\}$ there is a path from v_0 to v in G.*

Definition 2.4 (reducible flow graph). *A digraph $G = (V, A)$ is a **reducible flow graph** if it is a flow graph with root vertex v_0 and every depth-first search on G starting in v_0 produces the same set of back arcs.*

Shamir [54] presents a linear time algorithm that solves the minimum feedback vertex set problem on reducible flow graphs. His algorithm is equivalent to Algorithm 2.3. If v is the head of a back arc with maximal preorder number d_v with respect to a depth-first search starting in the root vertex v_0 of G, Speckenmeyer [56] shows that $\{v\}$ is a W$_1$-set (see section 2.1.1.2). So, Shamir's algorithm adds v to the (partial) FVS of G. Let $G' := G - v$ and let $H \subset G'$ be the reachability component of v_0. Because v has maximal preorder number d_v, the digraph $G - V(H)$ is acyclic. Furthermore, H is also a reducible flow graph with root vertex v_0 since it is the reachability component of v_0. From these two facts the correctness of Shamir's algorithm follows.

Finally, we mention the class of cyclically reducible graphs defined by Wang et al [60]. It can be defined in the following way: Let $G = (V, A)$ be a digraph. For a vertex $v \in V$ let $Y(G, v) \subset V(G)$ be the set of vertices $u \in V(G)$ such that there is no path in $G - v$ from u to a vertex w belonging to a cycle $C \in \mathcal{C}_{G-v}$. The convention is that always $v \in Y(G, v)$.

```
function Shamir(digraph $G = (V, A)$, root vertex $v_0$) do

    vertex set $F$;
    vertex $u$;
    vertex $v$;

    $F$ ← $\emptyset$;
    while $G$ is cyclic do
        $(d_v)_{v \in V(G)}$ ← preorder numb. of a DFS from $v_0$;
        $(u, v)$ ← back arc with maximal $d_v$;
        $F$ ← $F \cup \{v\}$;
        $G$ ← $G - v$;
    od;

    return $F$;
od;
```

Algorithm 2.3: Shamir's algorithm for the determination of an optimal FVS of a reducible flow graph.

Definition 2.5 (cyclically reducible graph). *A digraph $G = (V, A)$ is said to be* ***cyclically reducible*** *if there is a sequence $v_1, \ldots, v_k \in V$ such that the sequence of digraphs G_i $(0 \leq i \leq k)$ defined by $G_0 := G$ and $G_{i+1} := G_i - Y(G_i, v_{i+1})$ $(0 \leq i < k)$ satisfies the following:*

(i). $G[Y(G_i, v_{i+1})]$ is cyclic for $0 \leq i < k$

(ii). G_k is acyclic

Wang et al show that if G is cyclically reducible, then for any $u \in V$ with $G[Y(G, u)]$ being cyclic, there is a sequence $v_1, \ldots, v_k \in V$ meeting the above conditions such that $v_1 = u$. Based on that, they present a $O(|V|^2|A|)$ algorithm for determining an optimal FVS of a cyclically reducible graph. Because of their definition, cyclically reducible digraphs are also Smith-Walford-one reducible. One can moreover show that this inclusion is proper [60].

Figure 2.1 sums up the inclusion relations between the considered polynomial classes. The relations not mentioned here can be reviewed in [60].

2.2 Approximation Algorithms

The best approximation algorithm for the minimum feedback vertex set problem on digraphs $G = (V, A)$ achieves a performance-ratio of $O(\log |V| \log \log |V|)$, being in sharp contrast to the undirected case for which approximation algorithms with performance-ratio 2 exist [4, 6]. The $O(\log |V| \log \log |V|)$ performance-ratio is due to Even, Naor, Schieber and Sudan [19]. They apply a theoretical result of Seymour [53] who proves the following theorem:

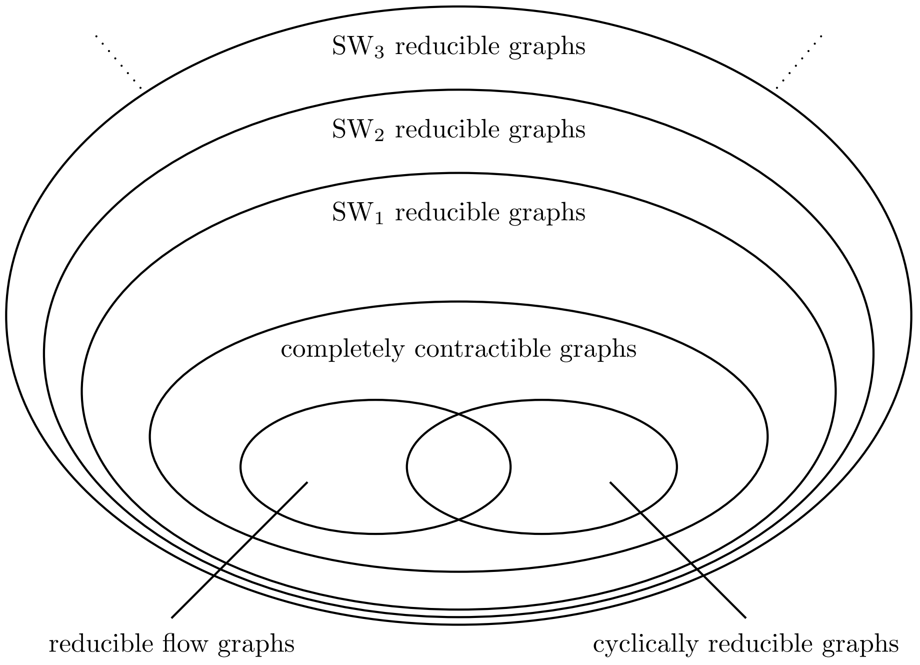

Figure 2.1: The inclusion relations between the polynomial classes.

Theorem 2.6. *Let $G = (V, A)$ be a digraph and let $w : V \to \mathbb{R}_+$ be a function such that*

$$\sum_{v \in V(C)} w(v) \geq 1 \quad \forall C \in \mathcal{C}_G.$$

Let τ be a real number and let

$$\sum_{v \in V} w(v) \leq \tau.$$

Then there exists a FVS F of G with

$$|F| \leq 4\tau \log(4\tau) \log\log_2(4\tau).$$

The achievement of Even et al was to realize that all existence arguments in Seymour's proof of Theorem 2.6 can be made constructive. Founding on this observation and refining Seymour's ideas, they have devised an approximation algorithm for the weighted minimum feedback vertex set problem which determines a FVS F of a weighted digraph $G = (V, A, w)$ having a weight no more than $4\tau^* \log(4\tau^*) \log\log_2(4\tau^*) \leq 4|V| \log(4|V|) \log\log_2(4|V|)$, where τ^* is the weight of an optimal fractional FVS of G (cf. section 1.3). The algorithm has a runtime of $O(|V|^2 A)$.

By restricting the digraphs to particular classes, it is possible to construct constant-factor approximation algorithms for the minimum feedback vertex set problem. The symmetric digraphs are a trivial example. Any arc (u, v) of a symmetric digraph $G = (V, A)$ induces a minimal cycle $(u, v, u) \in \mathcal{M}_G$. Hence, one sees that any FVS F of G is a vertex cover of the underlying graph of G and vice versa. Therefore, by repeatedly selecting an arbitrary arc $(u, v) \in A$, deleting u and v from G and adding them to the partial FVS – thus simulating the well-known approximation algorithm for the minimum vertex cover problem on the underlying graph – one gets a 2-approximation of an optimal FVS of G.

Tournaments and planar digraphs are less trivial examples. Constant-factor approximation algorithms for these classes are considered next.

2.2.1 Tournaments

A tournament is a digraph $G = (V, A)$ such that for any two vertices $u, v \in V$ either $(u, v) \in A$ or $(v, u) \in A$. So, G has exactly $\binom{|V|}{2}$ arcs. Speckenmeyer [56] has shown that the problem of determining an optimal FVS of a tournament is NP-hard. The same is true for the minimum feedback arc set problem on tournaments which was established independently in [2, 12].

One can show that a cyclic tournament G has always a minimal cycle $(u, v, w, u) \in \mathcal{M}_G$. Such a cycle is called a (cyclic) triangle. To obtain a constant-factor approximation of an optimal FVS of G, one can simply repeatedly determine a triangle in G, remove its vertex set and add it to the partial FVS. This has been done by Speckenmeyer [56] and yields a 3-approximation algorithm for the minimum feedback vertex set problem on tournaments.

This algorithm has been superseeded by Cai, Deng and Zang [10] who give a 2.5-approximation algorithm for the problem. Their approach consists of characterising tournaments for which a min-max theorem holds, more precisely, those tournaments, for which the cardinality of an optimal FVS and the triangle-packing number coincide. It turns out that for these tournaments the triangle-packing number is equal to the P_3-packing number[1]. For the problem of determining the latter number a polynomial algorithm is devised for this particular class of tournaments.

This class is characterised in terms of forbidden subgraphs. Finally, by applying the local ratio technique of Bar-Yehuda and Even [5] to the forbidden subgraphs, a 2.5-approximation algorithm for the minimum feedback vertex set problem on general tournaments is obtained.

2.2.2 Planar digraphs

The minimum feedback vertex set problem remains NP-hard on both directed and undirected planar graphs. On the other hand, the weighted minimum feedback arc problem is polynomially solvable on planar digraphs $G = (V, A)$. We refer to [27] for an excellent treatment of this subject. Using an ILP framework, Jünger shows that the separation problem for the polytope associated with the problem to determine a maximum acyclic subdigraph (which is the inverse of the weighted minimum feedback arc problem) of G can be solved in time $O(|V|^3)$.

[1] P_3 denotes a directed path consisting of 3 vertices.

The best approximation algorithm for the weighted minimum feedback vertex set problem has a performance ratio of $\frac{9}{4}$ and is due to Goemans and Williamson [23]. It is based on the so called primal-dual method which is developed in [24]. Generally, for a given minimisation problem this method simultaneously constructs a feasible solution X to the ILP formulation of the problem and a solution Y feasible for the relaxation of the dual of the ILP formulation. Then, the ratio between the weight of X and Y, respectively, is obviously a upper bound on the performance ratio of an approximation algorithm using the primal-dual method. If it can be shown that this ratio is at most α, the approximation algorithm also has performance ratio α. In [24] a generic formula is presented by means of which a performance ratio α can be derived.

This general reasoning is applied for the weighted minimum feedback vertex set problem on planar digraphs in [23]. Using the generic formula and exploiting the planarity of the digraph, the performance ratio $\frac{9}{4}$ is obtained which is shown to be tight.

2.3 Heuristics

2.3.1 GRASP

Greedy randomised adaptive search procedures (GRASP) [48] have been successfully applied to a variety of combinatorial optimisation problems [21, 31, 40]. They represent a multistart method consisting of two phases, a construction phase and a local search phase. In the construction phase, a feasible solution is constructed. This is done by randomly selecting each element of the solution from a restricted candidate list (RCL). The RCL contains elements whose rank is above or equal to a given threshold with respect to some greedy function. To obtain local optimality according to the adopted neighbourhood definition, the local search phase tries to improve the constructed solution and produces a locally optimal solution. This twostage process is applied repeatedly and the best solution found is kept. This is the general GRASP idea as described in [48].

Resende et al [45] have adapted GRASP for the minimum feedback vertex set problem. Experiments with various greedy functions were made, where the product of in- and outdegree turned out to be best. So, this greedy function is used in [45] and the RCL consists – in default settings - of all vertices having maximal product of in- and outdegree. After a vertex is selected from the RCL, added to the FVS and removed from the digraph, the remaining digraph is decomposed into its SCCs. In addition, the Levy-Low reductions (cf. section 2.1.1.1) are applied to simplify the remaining digraph and thus to speed up the algorithm. The local search phase is kept quite simple. It consists only of identifying redundant vertices and removing them from the FVS. Thereby, the vertices of the FVS are tested for being redundant in reversed order of addition to the FVS.

As described in [45], this process of constructing a FVS and transforming it into a minimal one is repeated. In default settings this is done 2048 times.

2.3.2 The combinatorial algorithm of Demetrescu and Finocchi

The approximation algorithm of Demetrescu and Finocchi [15] for the weighted minimum feedback vertex set problem is based on the local-ratio technique [5]. Actually, [15] discusses the weighted minimum feedback arc set problem. Still, the described algorithm can be adapted straight forwardly for the weighted minimum feedback vertex set problem as mentioned by Demetrescu and Finocchi.

In the spirit of the local-ratio technique the adapted version progressively reduces the vertex weights of a given weighted digraph $G = (V, A, w)$ and adds the vertices to the FVS whose weights become equal to 0. More precisely, it starts by randomly selecting a minimal cycle $C \in \mathcal{M}_G$ and determining the minimum $\epsilon = \min\{w(v) : v \in V(C)\}$ of all weights of vertices belonging to C. After that, the weights of all vertices in $V(C)$ are decreased by ϵ and those vertices v with $w(v) = 0$ are added to the FVS F and simultaneously removed from G. This process continues as long as a minimal cycle C can be determined, i.e. until G becomes acyclic.

In a second step the FVS F is transformed into a minimal one. This is achieved by testing all vertices of F for redundancy and removing them from F in case of being redundant. The complete procedure is presented in Algorithm 2.4.

In [15] it is argued that the algorithm for the weighted minimum feedback arc set problem has a runtime of $O(|V||A|)$. This runtime is achieved using the data structure in [16] for maintaining reachability information in digraphs subject to deletion and insertion of arcs. So, using the same data structure we conclude that WFVS can be implemented to run in time $O(|V|^2)$.

2.3.3 Schwikowski's enumeration algorithm

Schwikowski [51] shows how to enumerate all minimal FVSs of a digraph $G = (V, A)$ with polynomial delay, i.e. such that only a polynomial number of steps are performed before the first and between successive outputs. Any enumeration algorithm is expected to have an exponential runtime. Yet, by altering the termination condition of the enumeration algorithm other polynomial heuristic algorithms can be obtained.

The enumeration algorithm of Schwikowski starts with a minimal FVS $F \subset V$. Then, for any $v \in F$ the set $(F \setminus \{v\}) \cup \mathrm{N}^+(v)$ is also a FVS of G but not necessarily minimal. So, consider a successor function $\mu_G : \mathcal{P}(V) \times V \to \mathcal{P}(V)$ such that $\mu_G(F, v) \subset (F \setminus \{v\}) \cup \mathrm{N}^+(v)$ and $\mu_G(F, v)$ is a minimal FVS. In that way, for any minimal FVS F of G and any $v \in F$, the function μ_G assigns a fixed minimal FVS to the pair (F, v). The minimal FVS $\mu_G(F, v)$ is called the v-successor of F with respect to μ_G. $\mu_G(F, v)$ is also called a μ_G-successor of F.

Having the notation of v-successors, the superstructure digraph $\Phi(G, \mu_G)$ can be defined. Its vertex set consists of all minimal FVSs of G and (F, F') is

```
function WFVS(weighted digraph $G = (V, A, w)$) do

    vertex set $F$;
    cycle $C$;
    vertex $v$;
    real $\epsilon$;

    $F \leftarrow \emptyset$;

    while ($G - F$ is cyclic) do
        $C \leftarrow$ minimal cycle in $G - F$;
        $\epsilon \leftarrow \min\{w(v) : v \in V(C)\}$;

        forall $v \in V(C)$ do
            $w(v) \leftarrow w(v) - \epsilon$;

            if ($w(v) = 0$) do
                $F \leftarrow F \cup \{v\}$;
            od;
        od;
    od;

    forall $v \in F$ do
        if ($G - (F \setminus \{v\})$ is acyclic) do
            $F \leftarrow F \setminus \{v\}$;
        od;
    od;

    return $F$;
od;
```

Algorithm 2.4: The combinatorial algorithm of Demetrescu and Finocchi.

```
procedure GenerateMFVS(digraph $G = (V, A)$, succ. func. $\mu_G$) do

    dictionary $D$
    queue $Q$;
    vertex set $F$;
    vertex set $F'$;

    $D$ ← $\emptyset$;
    $Q$ ← $\emptyset$;

    $F$ ← arbitrary minimal FVS of $G$;
    $D$.insert($F$);
    $Q$.insert($F$);

    while ($Q \neq \emptyset$) do
        $F$ ← $Q$.extract();
        output $F$;

        for each $\mu_G$-successor $F'$ of $F$ do
            if ($D$.search($F'$) = nil) do
                $D$.insert($F'$);
                $Q$.insert($F'$);
            od;
        od;
    od;
od;
```

Algorithm 2.5: Schwikowski's enumeration algorithm.

an arc of $\Phi(G, \mu_G)$ if F' is a μ_G-successor of F. In [51] it is shown that $\Phi(G, \mu_G)$ is strongly connected. So, by exploring $\Phi(G, \mu_G)$ by a well-known search procedure like breadth-first search, all minimal FVSs of G are determined. The corresponding algorithm is given in Algorithm 2.5.

Algorithm 2.5 can be slightly modified to run in polynomial time and to determine a heuristic approximation of the optimal FVS of G. This is depicted in Algorithm 2.6 with an appropriate termination criterion to be chosen. In [50] various termination criterions are proposed and the resulting algorithms are studied empirically.

For several enumeration problems like determining all minimal dicuts, all minimal strongly connected digraphs [7] or all minimal triconnected spanning subgraphs [8], the idea of searching a conveniently defined superstructure digraph has been successfully applied in recent time.

```
function GenerateMFVS(digraph $G = (V, A)$, succ. func. $\mu_G$) do

    dictionary $D$
    queue $Q$;
    vertex set $F$;
    vertex set $F'$;
    vertex set $F_{\texttt{min}}$;

    $D$ ← $\emptyset$;
    $Q$ ← $\emptyset$;

    $F_{\texttt{min}}$ ← arbitrary minimal FVS of $G$;
    $D$.insert($F_{\texttt{min}}$);
    $Q$.insert($F_{\texttt{min}}$);

    while (some condition holds) do
        $F$ ← $Q$.extract();

        for each $\mu_G$-successor $F'$ of $F$ do
            if ($D$.search($F'$) = nil) do
                $D$.insert($F'$);
                $Q$.insert($F'$);

                if ($|F'|$ < $|F_{\texttt{min}}|$) do
                    $F_{\texttt{min}}$ ← $F'$;
                od;
            od;
        od;
    od;

    return $F_{\texttt{min}}$;
od;
```

Algorithm 2.6: A prototype of a heuristic algorithm based on Schwikowski's enumeration algorithm.

Chapter 3

Basic facts about Markov chains

What you need to know about the past is that no matter what has happened, it has all worked together to bring you to this very moment. And this is the moment you can choose to make everything new. Right now.

When Angels Speak: Inspiration From Touched By an Angel, Martha Williamson

This chapter is meant as a brief introduction to Markov chains. For reasons of completeness the notion of stochastic processes together with related terms is also included. A further purpose of this chapter is to fix the notation related to Markov chains. This is necessary because all the algorithms developed here are based on Markov chains and so the notation of which will be heavily used, particularly in Chapters 4, 5 and 6. Also, to comprehend the correctness of the algorithms as well as for runtime issues, a basic knowledge of the presented theory is needed. The definitions and theorems of this chapter can be reviewed in any text book dealing with Markov chains. We recommend [43].

3.1 Basic definitions

We begin with the definition of a probability space.

Definition 3.1 (probability space). *A* ***probability space*** *is a measure space $(\Omega, \mathcal{F}, P)$ with a set Ω, a σ-algebra $\mathcal{F} \subset \mathcal{P}(\Omega)$ of measurable sets and a measure $P : \mathcal{F} \to [0, 1]$ satisfying $P(\Omega) = 1$. The set Ω is called the* ***sample space****, $\mathcal{F}$ is the set of* ***events*** *and P is said to be a* ***probability measure****.*

Formally, P is a measure on the probability space $(\Omega, \mathcal{F}, P)$. In probability theory $P(A)$ with $A \in \mathcal{F}$ is – as the name suggests – interpreted as the probability of the event A. Furthermore, having an event $B \in \mathcal{F}$ such that $P(B) > 0$, one can define another probability measure by

$$P(A \mid B) := \frac{P(A \cap B)}{P(B)} \quad \text{(we will write } P(A, B) \text{ for } P(A \cap B)).$$

It is viewed as the probability of event A, given the occurrence of event B. If $P(B) = 0$, then $P(A \mid B)$ is undefined. Note that there might be problems in trying to assign a probability $P(A)$ to an arbitrary set $A \in \mathcal{P}(\Omega)$. However, if Ω is countable, this is not an issue. $\mathcal{F}$ can be set to the power set of Ω and in that case we will write (Ω, P) instead of $(\Omega, \mathcal{P}(\Omega), P)$.

Having specified the notion of a probability space, we are entitled to define a **random variable** X on the probability space $(\Omega, \mathcal{F}, P)$. Formally, X is a measurable function $X : \Omega \to S$ from the sample space Ω to a countable state space S (so $(S, \mathcal{P}(S))$ is a measurable space). But again, in the probability theory framework it is interpreted as a variable which takes a value $i \in S$ (or is in state i) with probability $P(X^{-1}(i))$. In sync with this interpretation, it will be written $X = i$ for the set $X^{-1}(i)$ and similarly $X \neq i$ for $\Omega \setminus X^{-1}(i)$. A real-valued vector d_X can be associated with X which is indexed by S:

$$d_X = (P(X = i))_{i \in S}.$$

This vector is called the **probability distribution vector** (or loosely **distribution**) of X. Obviously, the entries of d_X are all nonnegative and $\|d_X\|_1 = 1$. Any vector possessing these two properties will be called **normalised**.

If the random variable X is real-valued (i.e. $S \subset \mathbb{R}$),l the **expected value** of X is given by

$$E(X) = \sum_{i \in S} i \cdot P(X = i).$$

Hence, $E(X)$ is the asymptotic average value of X in a long-term observation. Like for the probability measure P, one can define the **conditional expected value** of X, given the event $A \in \mathcal{F}$, by

$$E(X \mid A) = \sum_{i \in S} i \cdot P(X = i \mid A).$$

Given a probability space $(\Omega, \mathcal{F}, P)$, a **stochastic process** with countable **state space** S is a sequence $(X_0, X_1, X_2, \ldots)$ of S-valued random variables, defined on $(\Omega, \mathcal{F}, P)$. The indexation of the random variables is interpreted as discrete time. So, in the case of $X_n = i$, we will say that the process is in state i at time n. A Markov chain is a stochastic process having the so called Markov property:

Definition 3.2 (Markov chain). *A stochastic process $(X_0, X_1, X_2, \ldots)$ with state space S on a probability space $(\Omega, \mathcal{F}, P)$ is said to be a* ***(homogeneous) Markov chain*** *if it possesses the* ***Markov property***

$$\begin{aligned} &P(X_{n+1} = x_{n+1} \mid X_n = x_n, \ldots, X_0 = x_0) \\ =&P(X_{n+1} = x_{n+1} \mid X_n = x_n) = P(X_1 = x_{n+1} \mid X_0 = x_n) \qquad \forall n \in \mathbb{N} \end{aligned} \tag{3.1}$$

for any fixed $x_0, \ldots, x_{n+1} \in S$.

The notion of Markov chains permits the definition of the matrix

$$P := (p_{ij})_{i,j \in S} \quad \text{with } p_{ij} = P(X_1 = j \mid X_0 = i)$$

for a chain $M = (X_0, X_1, X_2, \ldots)$. The entry p_{ij} of P is called **transition probability** from i to j, while P is said to be the **(1 step) transition matrix** of M. P is a **stochastic matrix** which is to say the entries of P are nonnegative and

$$\sum_{j \in S} p_{ij} = 1$$

for any $i \in S$. For the nonnegativeness of the entries p_{ij} we will write

$$P \geq 0.$$

In that case P is said to be nonnegative. Similarly, for another same sized matrix P', it will be written $P \geq P'$ if $P - P' \geq 0$. The notation $P > 0$ and $P > P'$ is defined accordingly.

Given the Markov chain M, its corresponding **transition digraph** $G_M = G_P$ can be constructed in a natural way by

$$G_M := G_P := (S, A_P) \quad \text{with } A_P := \{(i,j) \in S \times S : p_{ij} > 0\}.$$

As soon will be seen, many properties of the Markov chain M can be derived from G_M.

By now we have defined the notion of a probability distribution vector $d = (d_i)_{i \in S}$ with respect to a random variable as well as of a transition matrix $P = (p_{ij})_{i,j \in S}$ of a Markov chain. The vector-matrix product of ${}^t d$ and P is defined as usual by

$${}^t d \cdot P := {}^t d' \quad \text{with } d' = (d'_i)_{i \in S} \text{ such that } d'_i = \sum_{j \in S} d_j p_{ji}.$$

Although d and P may be of infinite size, the above product is still well defined, i.e. the sums converge, because d is normalised and the entries of P are bounded. With d also d' is a probability distribution vector of some random variable. Likewise, the product of two stochastic matrices P and P', both indexed by S, can be defined, so that $P \cdot P'$ is also a stochastic matrix.

Having said that, we can proceed in defining the n **step transition matrix** P^n by

$$P^n := (p^n_{ij})_{i,j \in S} \quad \text{with } p^n_{ij} = P(X_n = j \mid X_0 = i).$$

It is recursively obtained from P by $P^n = P^{n-1} \cdot P$ using the Chapman-Kolmogorov equations.

If the probability distribution vector of X_n is denoted by d_{X_n}, the probability distribution vectors are related by ${}^t d_{X_n} = {}^t d_{X_{n-1}} P$ or equivalently ${}^t d_{X_n} = {}^t d_{X_0} P^n$ for $n = 1, 2, \ldots$. In the theory of Markov chains one is interested in the long-run behaviour of the chain M, i.e. one is interested in the long-run behaviour of the sequence $(d_{X_0}, d_{X_1}, d_{X_2}, \ldots)$. As $\|d_{X_n}\|_1 = 1$ for $n = 0, 1, 2, \ldots$, from the BolzanoWeierstrass theorem it follows that the sequence has at least one cluster point. We are particularly interested in sufficient conditions for the existence of exactly one cluster point which is to say the convergence of the sequence. To be able to state these conditions, some additional definitions are needed.

3.2 Irreducibility and periodicity

We start with the irreducibility of a Markov chain.

Definition 3.3 (irreducibility). *A Markov chain $M = (X_0, X_1, X_2, \ldots)$ on a probability space $(\Omega, \mathcal{F}, P)$ with state space S is called **irreducible** if for any two states $i, j \in S$, there is an integer $n \in \mathbb{N}$ such that $P(X_n = j \mid X_0 = i) > 0$.*

The irreducibility of a Markov chain M directly translates into the strong connectivity of its transition digraph G_M. The set of states related to a SCC of G_M is called **communicating class**. The spectrum of the transition matrix P contains some information about the communicating classes of M, as the next lemma shows.

Lemma 3.4. *Let $M = (X_0, X_1, X_2, \ldots)$ be a finite Markov chain with transition matrix P. Then, the geometric multiplicity of the eigenvalue 1 of P is equal to the number of sink SCCs of the digraph G_M.*

Another important concept is the period of a state.

Definition 3.5 (period). *For a Markov chain $M = (X_0, X_1, X_2, \ldots)$ on a probability space $(\Omega, \mathcal{F}, P)$ the greatest common divisor*

$$p(i) := \gcd\{n \in \mathbb{N} : P(X_n = i \mid X_0 = i) > 0\}$$

*is called the **period** of the state i from the state space S. If $p(i) > 1$, then i is called **periodic** with period $p(i)$, while otherwise it is called **aperiodic**.*

Informally speaking, $p(i)$ is the greatest common divisor of all lengths of cycles which pass through i in the transition digraph G_M. It is known that within a communicating class any two states have the same period. Thus, one can also speak of the period of a communicating class. Hence, an irreducible Markov chain is said to be periodic or aperiodic if there is a state that is periodic or aperiodic, respectively.

But how can aperiodic Markov chains be recognised? A partial answer is given by the next theorem:

Theorem 3.6 (Frobenius). *A finite irreducible Markov chain M with transition matrix P is aperiodic if and only if $P^m > 0$ for some $m \in \mathbb{N}$.*

Although Theorem 3.6 answers the question whether or not a Markov chain is aperiodic, the answer is (even from the theoretical point of view) not very useful, as the integer m is not quantified. Wielandt has solved this problem by the following theorem:

Theorem 3.7 (Wielandt). *A n-state irreducible Markov chain M with transition matrix P is aperiodic if and only if $P^{n^2-2n+2} > 0$.*

But even Theorem 3.7 is not useful from the computational point of view. There are algorithms [57] which determine the period of a chain in linear time. The final result of this section relates the period of an irreducible Markov chain to the spectrum of its transition matrix:

Proposition 3.8. *Let M be a finite irreducible Markov chain with transition matrix P and period d. Then, $e^{\frac{2k\pi i}{h}}$ is an eigenvalue of P for $k \in \{0, \ldots, h-1\}$ if and only if $h | d$.*

3.3 Recurrence and transience

In the previous section we have roughly examined the structure of Markov chains by means of communicating classes and their periods. However, this does not provide sufficient conditions for the mentioned convergence of the probability distributions. Recurrence and transience is somehow the next detailed level of investigation. To define these two terms, the first hitting time is needed.

Definition 3.9 (first hitting time). *For a Markov chain $M = (X_0, X_1, X_2, \ldots)$ with state space S on a probability space $(\Omega, \mathcal{F}, P)$ the random variable $T_i : \Omega \to \mathbb{N} \cup \{\infty\}$, defined by*

$$T_i^{-1}(n) = \begin{cases} (X_1 \neq i) \cap \cdots \cap (X_{n-1} \neq i) \cap (X_n = i) & \text{if } n \neq \infty \\ \bigcap_{t=1}^{\infty} (X_t \neq i) & \text{if } n = \infty \end{cases},$$

is called the ***first hitting time*** *of the state $i \in S$.*

By writing T_i loosely as $T_i = \min\{k > 0 : X_k = i\}$, with the convention $T_i = \infty$ when $X_k \neq i$ for all $k > 0$, we see that T_i measures the time the chain M needs to get in state i. Its expected value

$$M_i := E(T_i \mid X_0 = i),$$

referred to as **the mean return time** to state i, will play an important role in the further classification of states. Anyway, we have now all ingredients to introduce the concept of recurrence.

Definition 3.10 (recurrence, transience). *Let $M = (X_0, X_1, X_2, \ldots)$ be a Markov chain with state space S on a probability space $(\Omega, \mathcal{F}, P)$. A state $i \in S$ is said to be* ***recurrent*** *if*

$$P(T_i < \infty \mid X_0 = i) = 1,$$

otherwise i is called ***transient****.*

From the definition it follows that a recurrent state i will be visited infinitely often by the Markov chain, once started in i, while transient states will be visited at most a finite number of times. Note, however, that although a state is recurrent, it does not necessarily imply the finiteness of the mean return time to that state. This suggests a further distinction of recurrent states. Thus, a recurrent state i is called **positive recurrent** if $M_i < \infty$, otherwise i is **null recurrent**. The motivation for this naming will be seen in the next proposition. One can show that (positive) recurrence and transience are class properties, i.e. two states from a communicating class are either both (positive) recurrent or both transient. Accordingly, we can say that an irreducible Markov chain is either positive (resp. null) recurrent or transient.

Proposition 3.11. *Let $M = (X_0, X_1, X_2, \ldots)$ be a Markov chain with state space S. If M admits a stationary distribution vector $\pi = (\pi_i)_{i \in S}$, a state $i \in S$ is positive recurrent if and only if $\pi_i > 0$.*

3.4 Limiting and stationary distribution

At the beginning of this chapter we asked about the long-term behaviour of a Markov chain $M = (X_0, X_1, X_2, \ldots)$. More precisely, we were interested in conditions under which the probability distribution vectors $d_{X_0}, d_{X_1}, d_{X_2}, \ldots$ converge to a common limit. The definitions of the previous sections have entitled us to give the answer by the following theorem. It represents one of the main results of this chapter.

Theorem 3.12. *Let $M = (X_0, X_1, X_2, \ldots)$ be an irreducible Markov chain with state space S, let d_{X_k} be the probability distribution vector of X_k. If M is aperiodic and positive recurrent, then the* ***limiting distribution vector*** *$\pi = \lim_{k\to\infty} d_{X_k}$ exists and does not depend on the initial distribution vector d_{X_0}.*

Irreducible Markov chains which are aperiodic and positive recurrent are called **ergodic**. The limiting distribution vector π in Theorem 3.12 may also exist under weaker conditions depending on the initial distribution vector d_{X_0}. In any case, if the limiting distribution vector π exists, then

$$\sum_{i\in S} \pi_i \cdot P(X_1 = j \mid X_0 = i) = \pi_j \quad \text{and} \quad \sum_{i\in S} \pi_i = 1$$
$$\text{or, equivalently in matrix notation,} \tag{3.2}$$
$${}^t\pi P = {}^t\pi \quad \text{and} \quad \|\pi\|_1 = 1$$

holds for the transition matrix P of the Markov chain. Any vector π satisfying (3.2) is called **stationary distribution vector**. The reason for this naming is that if one assumes $d_{X_k} = \pi$ for a probability distribution vector of X_k, then ${}^t d_{X_{k+1}} = {}^t d_{X_k} P = {}^t\pi P = {}^t\pi$ follows and one sees that the distributions remain stationnary as time passes by. The question under which conditions a unique stationary distribution vector exists is essentially answered by the following theorem. This is another main result.

Theorem 3.13. *Let $M = (X_0, X_1, X_2, \ldots)$ be an irreducible Markov chain with state space S. Then, M is positive recurrent if and only if there is a stationary distribution vector $\pi = (\pi_i)_{i\in S}$ of M. In that case*

$$\pi_i = \frac{1}{M_i} \quad \forall i \in S$$

holds, and hence π is unique.

Note that the assertions in Theorem 3.13 are weaker than in Theorem 3.12. This means that if the limiting distribution exists according to Theorem 3.12, it is equal to the stationary distribution postulated in Theorem 3.13. Furthermore, Theorem 3.13 interprets the stationary distribution vector as the inverse of the mean return time vector, to speak informally. This is exactly the idea behind an algorithm due to Speckenmeyer [56] for the minimum feedback vertex set problem. This algorithm is reviewed in Chapter 4. There, we will present an alternative motivation for the design of that algorithm.

As in the subsequent chapters only Markov chains with finite state space will be considered, the next proposition is of interest.

Proposition 3.14. *Let $M = (X_0, X_1, X_2, \ldots)$ be an irreducible Markov chain with state space S. If S is finite, M is positive recurrent and therefore admits a unique stationary distribution.*

Chapter 4

MFVS and variants

Two are better than one, because they have a good reward for their hard work. For if one of them should fall, the other one can raise his partner up. But how will it be with just the one who falls when there is not another to raise him up? [. . .] And if somebody could overpower one alone, two together could make a stand against him. And a threefold cord cannot quickly be torn in two.

The Bible,
Ecclesiastes 4:9–12

In this chapter we will present some new heuristics for the (weighted) minimum feedback vertex set problem for digraphs based on Markov chains. In this connection, the language and results presented in Chapter 3 will be used.

The pioneering work in this direction has been initiated by Speckenmeyer with the paper [56]. Unfortunately, his idea has fallen into oblivion. The reason for this may be manifold. One aspect might be the large worst case runtime of the resulting algorithm. Another aspect might be the fact that there exist faster heuristics that outperform the Markov chain based heuristic.

Both topics are addressed in this chapter. Section 4.1 contains a brief description of Speckenmeyer's original idea and his algorithm MFVS as presented in [56]. As part of the description, we will improve the runtime of the algorithm in section 4.1.4. Based on a alternative implementation, a runtime is derived that shows that, despite of the high theoretical runtime the algorithm, can come up with a solution for practical instances in reasonable time.

In section 4.2 we will introduce a clever yet natural enhancement of Speckenmeyer's heuristic. With this enhancement the resulting algorithm is able to compete with other algorithms with respect to the performance. However, detailed performance comparisons are presented in Chapter 6.

4.1 The basic MFVS algorithm

4.1.1 Motivation

How to select a vertex v of a digraph $G = (V, A)$ to be part of a FVS of small cardinality? One idea is to select the vertex with the highest probability of belonging to an optimal FVS F of G. But what is the probability that v belongs to F? This question is difficult to answer. However, let us try to give an approximative answer. Suppose v lies on a minimal cycle $C \in \mathcal{M}_G$. By further assuming $|F \cap V(C)| = 1$, the probability of v belonging to F can be assessed as

$$\frac{1}{|V(C)|}. \tag{4.1}$$

Let $C' \in \mathcal{M}_G$ be another minimal cycle containing v with $V(C) \cap V(C') = \{v\}$ and $|F \cap V(C')| = 1$. Then, the probability of $v \in F$ can be reevaluated by

$$\frac{1}{|V(C)|} + \left(1 - \frac{1}{|V(C)|}\right)\frac{1}{|V(C')|} = \frac{1}{|V(C)|} + \frac{1}{|V(C')|} - \frac{1}{|V(C)||V(C')|}. \tag{4.2}$$

One could further consider a third minimal cycle $C'' \in \mathcal{M}_G$ (and so forth) with $V(C) \cap V(C') \cap V(C'') = \{v\}$ and $|F \cap V(C'')| = 1$ and one would obtain estimates for the probability similar to (4.1) and (4.2). Yet, two observations can be made: Firstly, the (estimated) probability of v being part of F increases with the number of cycles passing through v. Secondly, the probabilities (4.1) and (4.2) increase as both the length of C and the length of C' decrease.

Now, suppose G is the transition digraph of a random walk M. We can relate the second observation with the stationary distribution vector $\pi = (\pi_v)_{v \in V}$ of M as follows: Assuming M to be irreducible, Theorem 3.13 in conjunction with Proposition 3.14 tells us that $\frac{1}{\pi_v}$ is equal to the mean return time to v. On the other hand, intuitively, v has a small mean return time if many cycles of small length pass through it. Hence, the probability of v being a member of the optimal FVS F approximatively amounts to π_v.

The assumptions which have led to the mentioned connection are not always satisfied. Particularly, $|F \cap V(C)| = 1$ does not hold for optimal FVSs F in general. Nevertheless, the stationary distribution vector π contains enough information about the number and length of the cycles which pass a vertex v. This is especially the case for the class of symmetric digraphs, i.e. digraphs G for which $G = G^{-1}$ holds. In these digraphs any minimal cycle has length 2, so any two distinct minimal cycles share at most one vertex. Speckenmeyer has shown in [56] that on these digraphs MFVS has a performance ratio of $\log |V|$.

4.1.2 The algorithm

Consider a digraph $G = (V, A)$ for which a FVS is to be determined. We can assume that each vertex of G has in- and outdegree of at least 1. The key of Speckenmeyer's heuristic algorithm is the interpretation of G as a finite-state Markov chain. Accordingly, the Markov chain $M_G = (X_0, X_1, X_2, \ldots)$ with state space V is associated to G. M_G is defined on the probability space $(\Omega, \mathcal{F}, P)$

and has the property

$$P(X_1 = v \mid X_0 = u) = \begin{cases} \frac{1}{\mathrm{d}_G^+(u)} & \text{if } (u,v) \in A \\ 0 & \text{else} \end{cases} . \tag{4.3}$$

Because of the assumptions made on G, the chain M_G is well defined. The property (4.3) defines M_G completely with the exception of the initial distribution. It is not specified because Speckenmeyer's algorithm does not depend on it. So, for any digraph G with the mentioned restrictions, M_G will denote the Markov chain constructed above. M_G has the property that $P(X_1 = v_1 \mid X_0 = u) \neq 0$ and $P(X_1 = v_2 \mid X_0 = u) \neq 0$ imply $P(X_1 = v_1 \mid X_0 = u) = P(X_1 = v_2 \mid X_0 = u)$. Every Markov chain having this property is called **random walk**.

We now describe Speckenmeyer's algorithm which is called Markovian FVS-algorithm or MFVS for short [56]. It starts with decomposing G into its SCCs $S_1, \ldots, S_r$. Then, for each S_i $(i = 1, \ldots, r)$ the algorithm selects a vertex $v \in V(S_i)$ to belong to the FVS and processes $S_i - v$ recursively until a FVS is determined. During that process MFVS attempts to simplify the current digraph by applying the Levy-Low reductions (see section 2.1.1.1).[1] The whole algorithm is depicted in Algorithm 4.1.

The FVS F obtained from Algorithm 4.1 is not guaranteed to be minimal. So, it is transformed into a minimal FVS. This is done by checking the vertices of F for redundancy in reversed order of insertion and removing them if they turn out to be redundant. This is shown in Algorithm 4.2. Thus, the algorithm MFVS is invoked by `MakeMinimal(`G`, MFVS_SCC(`G`))`.[2]

The crucial part of MFVS is its vertex selection heuristic. It chooses the vertex $v \in V$ with the smallest mean return time M_v in M_G to belong to a FVS of G. Consequently, the routine `SelectVertex(`G`)` from Algorithm 4.1 computes the stationary distribution vector $\pi = (\pi_v)_{v \in V}$ of M_G and returns the vertex $v \in V$ with the largest value of π_v. Note that MFVS ensures that G is strongly connected whenever `SelectVertex(`G`)` is called. In other words, M_G is irreducible in this situation and, in accordance with Proposition 3.14, M_G has indeed a unique stationary distribution vector which reflects the mean return times as described by Theorem 3.13. The complete vertex selection heuristic is formally shown in Algorithm 4.3.

4.1.3 Bad behaviour

Despite the motivations for MFVS given in section 4.1.1 it still can perform badly without the Levy-Low reductions. Let us call that algorithm MFVS* in this context. The digraph ${}^{2}_{r}\mathcal{F}$ $(r = 3, 4, \ldots)$ depicted in Figure 4.1 (originally

[1]In the original paper [56] the algorithm MFVS has not applied the Levy-Low reductions. Insteed, W_1-sets (cf. section 2.1.1.2) have been recognised. Here, the recognition of W_1-sets is omitted in favour of the Levy-Low reductions for two reasons. Firstly, most approximation algorithms use the Levy-Low reductions, only. Thus, by pursuing the same philosophy for MFVS, it becomes possible to compare it with other algorithms. Secondly, recognising W_1-sets is computationally expensive while MFVS hardly benefits from this feature.

[2]In the original paper [56] no check for redundancy is done. We include it even though for comparability reasons.

```
function MFVS_SCC(digraph $G = (V, A)$) do

    vertex set $F$;

    if $V = \emptyset$ do
        return $\emptyset$;
    od;

    $F$ ← $\emptyset$;

    for all SCCs $S$ of $G$ do
        $F$ ← $F$ ∪ MFVS($S$);
    od;

    return $F$;
od;

function MFVS(digraph $G = (V, A)$) do

    vertex $v$;

    if $V = \emptyset$ do
        return $\emptyset$;
    od;

    for all $v \in V$ do
        if $(v, v) \in A$ do
            return $\{v\}$ ∪ MFVS_SCC($G - v$);
        od;
        else if $\min(\mathrm{d}^-_G(v), \mathrm{d}^+_G(v)) = 0$ do
            return $\emptyset$;
        od;
        else if $\mathrm{d}^-_G(v) = 1$ do
            return MFVS($G \circ v$);
        od;
        else if $\mathrm{d}^+_G(v) = 1$ do
            return MFVS($G \circ v$);
        od;
    od;

    $v$ ← SelectVertex($G$);

    return $\{v\}$ ∪ MFVS_SCC($G - v$);
od;
```

Algorithm 4.1: The Markovian FVS algorithm.

```
function MakeMinimal(digraph $G = (V, A)$, vertex set $F$) do

    vertex $v$;

    forall $v \in F$ do
        if ($G - (F \setminus \{v\})$ is acyclic) do
            $F$ ← $F$ \ $\{v\}$;
        od;
    od;

    return $F$;
od;
```

Algorithm 4.2: The transformation of a FVS F into a minimal one.

```
function SelectVertex(digraph $G = (V, A)$) do

    matrix $P = (p_{uv})_{(u,v) \in V^2}$;
    vector $\pi = (\pi_v)_{v \in V}$;
    vertex $v$;

    $P$ ← CreateTransitionMatrix($G$);
    $\pi$ ← ComputeStationaryDistributionVector($P$);

    determine $v \in V$ with $\pi_v = \|\pi\|_\infty$;

    return $v$;
od;
```

Algorithm 4.3: The vertex selection heuristic of MFVS.

due to Peter Heusch) is an example of this bad performance. In [50], it is claimed without proof that MFVS* determines the FVS $\{M_1, \ldots, M_r\}$ of ${}^2_r\mathcal{F}$. We will prove this fact. By the way, $\{A_1, B_1\}$ is obviously the only optimal FVS of ${}^2_r\mathcal{F}$.

Lemma 4.1. *MFVS* determines the FVS* $\{M_1, \ldots, M_r\}$ *of* ${}^2_r\mathcal{F}$ *for* $r \geq 3$.

Proof. We want to show that $M_1, \ldots, M_r \in {}^2_r\mathcal{F}$ is successively selected by MFVS*. For this purpose consider the subgraph ${}^2_r\mathcal{F}_k$ (Figure 4.2) of ${}^2_r\mathcal{F}$ for $k = 1, \ldots, r$. We compute the stationary distribution $\pi = (\pi_v)_{v \in V({}^2_r\mathcal{F}_k)}$ of $M_{{}^2_r\mathcal{F}_k}$. Then, we have

- $\pi_{A_i} = \pi_{B_i} = 2^{r-k} z$ for $i = 1, \ldots, k-1$
- $\pi_{A_i} = \pi_{B_i} = 2^{r-i} z$ for $i = k, \ldots, r$
- $\pi_{M_i} = \frac{6^{r+1-i}+4}{5 \cdot 3^{r-i}} z$ for $i = k, \ldots, r$

with $z = \frac{5 \cdot 3^{r-k}}{2(11+5k)6^{r-k} - 10 \cdot 3^{r-k} - 2} > 0$. This can be confirmed by verifying the equations (3.2) defining the stationary distribution vector. We see that

- $\pi_{A_i} \geq \pi_{A_{i+1}}$ for $i = 1, \ldots, r-1$
- $\pi_{M_i} = \frac{6^{r+1-i}+4}{5 \cdot 3^{r-i}} z = \frac{2 \cdot 6^{r-i} + \frac{4}{3}}{5 \cdot 3^{r-1-i}} z > \frac{6^{r-i}+4}{5 \cdot 3^{r-1-i}} z = \pi_{M_{i+1}}$ for $i = k, \ldots, r-1$
- $\pi_{M_k} = \frac{6^{r+1-k}+4}{5 \cdot 3^{r-k}} z > \frac{6^{r+1-k}}{6 \cdot 3^{r-k}} z = 2^{r-k} z = \pi_{A_1}$

It follows that $\pi_{M_k} = \|\pi\|_\infty$, and thus M_k is selected by MFVS* in the digraph ${}^2_r\mathcal{F}_k$. Hence, the assertion follows inductively. □

Lemma 4.1 shows that the performance-ratio of MFVS* is arbitrarily high. It is reasonable to assume that the same applies for MFVS.

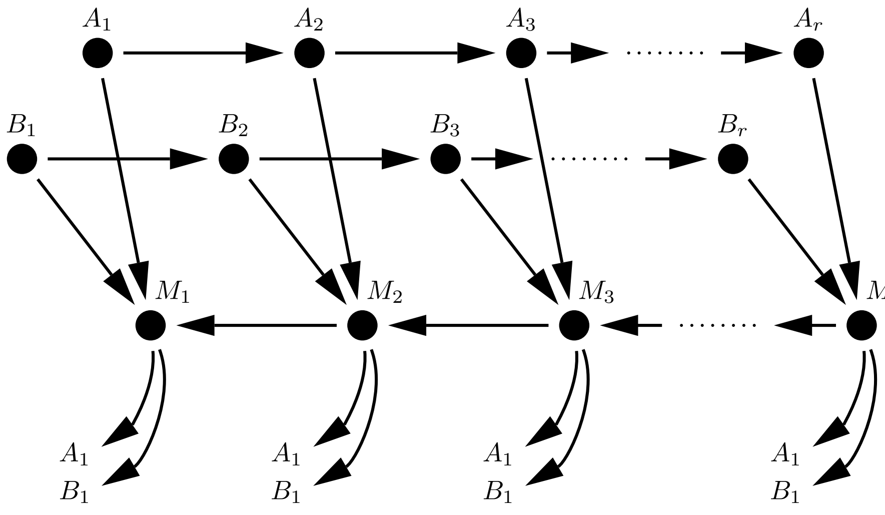

Figure 4.1: The fool graph ${}^2_r\mathcal{F}$ for MFVS.

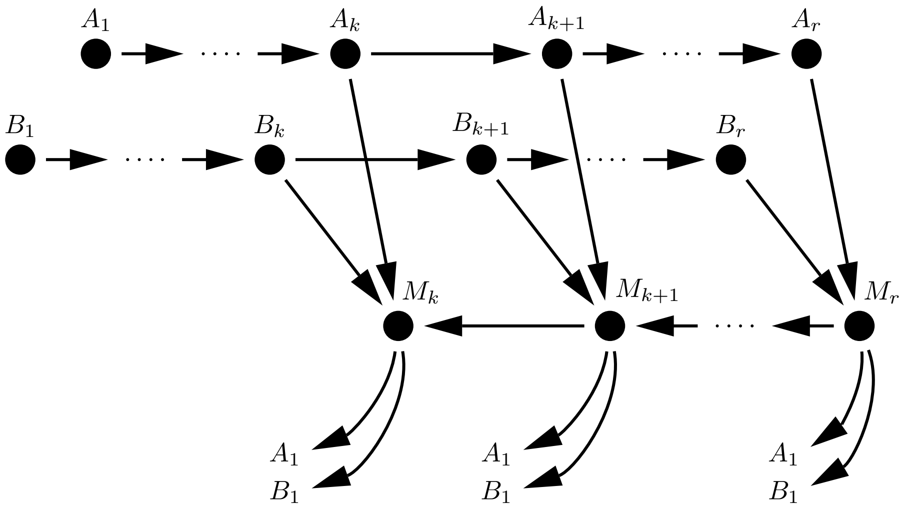

Figure 4.2: The induced subgraph ${}^{2}_{r}\mathcal{F}_k$ of ${}^{2}_{r}\mathcal{F}$. It results from ${}^{2}_{r}\mathcal{F}$ after deleting the vertices $M_1, \ldots, M_{k-1}$.

4.1.4 Runtime and implementation

In Chapter 3 we have seen that the stationary distribution vector $\pi = (\pi_i)_{i\in S}$ of an irreducible Markov chain M with state space S satisfies the equations (3.2) uniquely. These equations can be rewritten as

$$\begin{aligned} ({}^tP - I)\pi &= 0 \\ \sum_{i\in S} \pi_i &= 1, \end{aligned}$$

where P is the transition matrix of M and I is the identity matrix. Because π is the unique solution to the above system of linear equations, replacing any row of the homogeneous system $({}^tP - I)\pi = 0$ by $\sum_{i\in S} \pi_i = 1$ leads to a nonhomogeneous one with the unique solution π. So, computing the stationary distribution vector π of an irreducible Markov chain M consists of solving a nonhomogeneous system of linear equations

$$A\pi = b$$

with a non-singular square matrix $A \in \mathbb{R}^n$. Hence, the stationary distribution π can be determined by performing a matrix inversion of size $|V| \times |V|$. The theoretically best known algorithm for this task has a runtime of $O(|V|^{2.376})$ [13].

Now, recall the algorithm MFVS shown in Algorithm 4.1. Suppose it determines the FVS F of the digraph $G = (V, A)$. At the beginning and after each selection of a vertex to be included in a FVS, MFVS decomposes the digraph into its SCCs and applies the Levy-Low reductions. The former takes linear time [59] while the latter can be implemented to run in time $O(|A| \log |V|)$ [36]. Yet, the dominant operation is the vertex selection itself. For almost each vertex in F the stationary distribution of M_G has to be determined which takes time

$O(|V|^{2.376})$ following the above reasoning. So, one could argue that the total runtime sums up to $O(|V|^{2.376}|F|)$.

However, keep in mind that most of the digraphs arising in practical applications are sparse. In addition, the stationary distribution vector π does not necessarily need to be evaluated to full precision in order to decide which is the largest entry of π. These two facts suggest applying an iterative method to solve the mentioned system of linear equations. By doing so, one can hope to obtain a better bound for the runtime of MFVS.

A basic iterative method for solving a system of linear equations is the power method (see [32] and [61]) which is a direct application of Theorem 3.12 in conjunction with Proposition 3.14. If we write the problem of determining the stationary distribution vector π of the irreducible aperiodic Markov chain M_G again in fixpoint form ${}^tP\pi = \pi$, the power method consists of computing $({}^tP)^k x$ for sufficiently large k with an arbitrary normalised vector x. The idea behind this is that from Theorem 3.12 it follows

$$\lim_{k\to\infty} ({}^tP)^k x = \pi.$$

The periodic case can be easily transformed into an equivalent aperiodic one as follows: The equation

$${}^tP\pi = \pi$$

is equivalent to

$$\underbrace{\tfrac{1}{2}({}^tP + I)}_{{}^tP'}\pi = \pi.$$

So, P' is the transition matrix of an aperiodic Markov chain $M_{G'}$ having the same stationary distribution as M_G. Hence, the power method can be applied to $M_{G'}$.

Now, suppose we want to evaluate π to an accuracy of 10^{-d}. Let λ be a subdominant (second largest) eigenvalue of P. Then, as shown in [32], asymptotically

$$\frac{d}{-\log_{10}|\lambda|}$$

iterations are needed. On the other hand, P has exactly $|A|$ nonzero entries. Hence, the matrix vector product of ${}^tP$ and any vector x can be implemented to run in $O(|A|)$ time using a sparse matrix package. This yields a runtime of $O(\frac{|A|d}{-\ln|\lambda|})$ to compute π. The total runtime of MFVS applying the power method is therefore $O(|A||F|\frac{d}{-\ln|\lambda|})$. Because d is usually bounded — e.g. by the machine precision — it can be regarded as constant. So, the runtime bound reduces to

$$O(|A||F|\tfrac{1}{-\ln|\lambda|}).$$

Thus, the runtime of MFVS is mainly governed by $\frac{1}{-\ln|\lambda|}$. If $|\lambda|$ is well-separated from 1, i.e. there is a $\delta > 0$ with $|\lambda| \leq 1 - \delta$, for each input instance $G = (V, A)$, then MFVS runs in time $O(|A||F|)$, being considerably faster than the approach involving direct matrix inversion with the time bound of $O(|V|^{2.376}|F|)$. Unfortunately, this is not always the case. Even for irreducible aperiodic Markov chains $|\lambda|$ can be arbitrarily close to 1 resulting in an arbitrarily high runtime. Nearly decoupled chains are an example for this phenomenon. In such cases the former method of matrix inversion should be used.

While the closeness of $|\lambda|$ to 1 might theoretically cause runtime problems, for most practical applications this is not an issue and the power method is the algorithm of choice. Although we have implemented it in the simplest form, we have not encountered any runtime problems. Besides, there are ways to accelerate the power method. A survey can be found in [33] where the focus is on the PageRank computation [9]. More general approaches to speedup the computation of the stationary distribution vector are the state compression technique described in [61] and the aggregation/disaggregation technique reviewed in [39]. Since MFVS actually only needs to update the stationary distribution vector after the removal or exclusion of a vertex, the latter algorithm can be used to accomplish this task. This procedure, among others, is outlined in [32].

The algorithm MFVS and all its subsequent variants and extensions have been implemented mainly using the `LEDA` software library [42]. The only part not written using `LEDA` is the power method which has been implemented using the `SparseLib++` class library [18].

4.2 The mean approach

4.2.1 Motivation

The starting point for our considerations, which have led to the mean approach, was the digraph ${}^{2}_{r}\mathcal{F}$ $(r \geq 3)$ from Figure 4.1. As Lemma 4.1 shows ,MFVS* performs bad for this class of digraphs. We tried to understand why this is the case for this particular class and how MFVS* could be fixed to determine the optimal FVS $\{A_1, B_1\}$.

The general idea was to develop a technique to somehow detect and correct at least some nonoptimal vertex selections made by MFVS*. This technique should ideally be 'independent' of the way MFVS* selects vertices, yet roughly having the same quality. For if one of the two heuristics was clearly worse, it would interfere the better one. Of course we cannot expect from a polynomial heuristic the ability to detect all nonoptimal selections made by MFVS*, unless P=NP. Occasionally, the heuristic might even falsely discard an optimal selection in favour of a nonoptimal one. But typically, the heuristic should be a gain.

One might be surprised by the fact that for MFVS* it is straight forward to devise such a heuristic which meets all the mentioned requirements. Just note that the digraphs $G = (V, A)$ and G^{-1} share the same (optimal) FVSs. So, one could consider applying MFVS* on G^{-1}. One evidence that this direction is promising is the next lemma.

Lemma 4.2. *MFVS* determines the optimal FVS* $\{A_1, B_1\}$ *of* ${}^{2}_{r}\mathcal{F}^{-1}$ *for* $r \geq 3$.

Proof. If $\pi = (\pi_v)_{v \in V(^2_r\mathcal{F}^{-1})}$ denotes the stationary distribution of $M_{^2_r\mathcal{F}^{-1}}$, we compute

- $\pi_{A_i} = \pi_{B_i} = \frac{(2r-2i+3)3^{i-1}-1}{2 \cdot 3^{i-1} r} z$ for $i = 1, \ldots, r$
- $\pi_{M_i} = \frac{3^i - 1}{3^{i-1} r} z$ for $i = 1, \ldots, r$

with $z = \frac{3^{r-1} r}{(r^2+5r-3)3^{r-1}+1} > 0$. We see that

- $\pi_{A_i} = \pi_{B_i} = \frac{(2r-2i+3)3^{i-1}-1}{2 \cdot 3^{i-1} r} z = \frac{(2r-2i+3)3^i - 3}{2 \cdot 3^i r} z$
  $= \frac{(2r-2(i+1)+3)3^i + 2 \cdot 3^i - 3}{2 \cdot 3^i r} z$
  $> \pi_{A_{i+1}} = \pi_{B_{i+1}}$ for $i = 1, \ldots, r-1$
- $\pi_{M_i} = \frac{3^i - 1}{3^{i-1} r} z = \frac{3^{i+1}-3}{3^i r} z < \frac{3^{i+1}-1}{3^i r} z = \pi_{M_{i+1}}$ for $i = 1, \ldots, r-1$
- $\pi_{A_1} = \pi_{B_1} = z > \frac{3^r - 1}{3^r} z \geq \frac{3^r - 1}{3^{r-1} r} z = \pi_{M_r}$ for $r \geq 3$

From the above $\pi_{A_1} = \pi_{B_1} = \|\pi\|_\infty$ can be concluded. So, either A_1 or B_1 – say A_1 w.l.o.g – is selected first by MFVS* as a FVS vertex. Furthermore, for $^2_r\mathcal{F}^{-1} - A_1$ there is an optimal FVS consisting of the vertex B_1. It has been shown by Speckenmeyer [56] that in this situation MFVS* determines an optimal FVS of $^2_r\mathcal{F}^{-1} - A_1$. Since $\{B_1\}$ is the unique optimal FVS, MFVS* will determine it.

Hence, we see that MFVS* selects A_1 first and then B_1. □

But how to incorporate the computation of MFVS*(G^{-1}) into that of MFVS*(G)? One possibility is to call `SelectVertex(`G^{-1}`)` each time `SelectVertex(`G`)` is called within Algorithm 4.1. If both calls return the same vertex, we can interpret that as a confirmation of its membership of an optimal FVS. But what to do if the two vertices disagree? As part of the execution of MFVS*(G) and MFVS*(G^{-1}), the stationary distributions $\pi' = (\pi'_v)_{v \in V}$ and $\pi'' = (\pi''_v)_{v \in V}$ of M_G and $M_{G^{-1}}$, respectively, are determined. Our idea is to compute a vector $\pi = (\pi_v)_{v \in V}$ with $\pi_v = h(\pi'_v, \pi''_v)$ $(v \in V)$ and to determine a vertex $v \in V$ such that $\pi_v = \|\pi\|_\infty$. Then, v is selected to belong to the FVS of G.

We have considered different functions $h : \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}_+$. After carrying out experiments with several functions, including $h(x,y) = \frac{1}{2}(x+y)$, $h(x,y) = \sqrt{xy}$ and $h(x,y) = \max\{x,y\}$, the arithmetic mean, i.e. $h(x,y) = \frac{1}{2}(x+y)$, turned out to be best.

4.2.2 The algorithm

Let us denote by MFVS*$_{\text{Mean}}$ the algorithm described in the previous subsection. It differs from MFVS* only in the vertex selection heuristic. The modified heuristic is shown in Algorithm 4.4. By MFVS$_{\text{Mean}}$ the same algorithm will be denoted except that MFVS$_{\text{Mean}}$ also applies the Levy-Low reductions. It follows exactly the framework of Algorithm 4.1, whereby the vertex selection heuristic in Algorithm 4.4 is used.

```
function SelectVertex(digraph $G = (V, A)$) do

    matrix $P = (p_{uv})_{(u,v) \in V^2}$;
    vector $\pi = (\pi_v)_{v \in V}$;
    vector $\pi' = (\pi'_v)_{v \in V}$;
    vector $\pi'' = (\pi''_v)_{v \in V}$;
    vertex $v$;

    $P$ ← CreateTransitionMatrix($G$);
    $\pi'$ ← ComputeStationaryDistributionVector($P$);
    $P$ ← CreateTransitionMatrix($G^{-1}$);
    $\pi''$ ← ComputeStationaryDistributionVector($P$);

    $\pi$ ← $\pi' + \pi''$;

    determine $v \in V$ with $\pi_v = \|\pi\|_\infty$;

    return $v$;
od;
```

Algorithm 4.4: The vertex selection heuristic of MFVS$_{\text{Mean}}$.

The expectation that motivated the design of MFVS$_{\text{Mean}}$ was that MFVS$_{\text{Mean}}$ will perform better than MFVS in the average case. Although this issue will be discussed in detail in Chapter 6, we give an example which supports this expectation. We argue that MFVS*$_{\text{Mean}}$ determines an optimal FVS of ${}^2_r\mathcal{F}$ for small instances.

Lemma 4.3. *MFVS*$^*_{Mean}$ *determines the optimal FVS* $\{A_1, B_1\}$ *of* ${}^2_r\mathcal{F}$ *for* $3 \leq r \leq 24$.

Proof. Let $\pi' = (\pi'_v)_{v \in V({}^2_r\mathcal{F})}$ and $\pi'' = (\pi''_v)_{v \in V({}^2_r\mathcal{F}^{-1})}$ be the stationary distribution of $M_{{}^2_r\mathcal{F}}$ and $M_{{}^2_r\mathcal{F}^{-1}}$, respectively. From the proofs of Lemma 4.1 and 4.2 it is known that

- $\pi'_{A_i} = \frac{5 \cdot 2^{r-i} \cdot 3^{r-1}}{32 \cdot 6^{r-1} - 10 \cdot 3^{r-1} - 2}$ for $i = 1, \ldots, r$
- $\pi'_{M_i} = \frac{(6^{r+1-i} + 4) 3^{i-1}}{32 \cdot 6^{r-1} - 10 \cdot 3^{r-1} - 2}$ for $i = 1, \ldots, r$

as well as

- $\pi''_{A_i} = \pi''_{B_i} = \frac{(2r - 2i + 3) 3^{r-1} - 3^{r-i}}{2(r^2 + 5r - 3) 3^{r-1} + 2}$ for $i = 1, \ldots, r$
- $\pi''_{M_i} = \frac{3^r - 3^{r-i}}{(r^2 + 5r - 3) 3^{r-1} + 1}$ for $i = 1, \ldots, r$

With $\pi = \frac{1}{2}(\pi' + \pi'')$ from the proofs of Lemma 4.1 and 4.2 it also follows $\pi_{A_i} > \pi_{A_{i+1}}$ for $i = 1, \ldots, r - 1$. So it remains to be shown that $\pi_{A_1} > \pi_{M_i}$ for $i = 1, \ldots, r - 1$.

With $X := 32 \cdot 6^{r-1} - 10 \cdot 3^{r-1} - 2$ and $Y := (r^2 + 5r - 3)3^{r-1} + 1$ we get

$$\begin{aligned}
\pi_{A_1} &= \frac{1}{2}(\pi'_{A_1} + \pi''_{A_1}) = \frac{1}{2}\left(\frac{5 \cdot 6^{r-1}}{X} + \frac{r3^{r-1}}{Y}\right) \\
&= \frac{1}{2XY}(5 \cdot 6^{r-1}(r^2 + 5r - 3)3^{r-1} + 5 \cdot 6^{r-1} + 32r3^{r-1}6^{r-1} - r3^{r-1} \cdot 10 \cdot 3^{r-1} - 2r3^{r-1}) \\
&= \frac{1}{2XY}(18^{r-1}(5r^2 + 57r - 15) + 5 \cdot 6^{r-1} - 10r9^{r-1} - 2r3^{r-1})
\end{aligned}$$

For $2 \le i \le r$ we get

$$\begin{aligned}
\pi_{M_i} &= \frac{1}{2}(\pi'_{M_i} + \pi''_{M_i}) = \frac{1}{2}\left(\frac{\overbrace{(6^{r+1-i} + 4)3^{i-1}}^{\le 3(6^{r-1}+4)}}{X} + \frac{3^r - 3^{r-i}}{Y}\right) \le \frac{1}{2}\left(\frac{3(6^{r-1} + 4)}{X} + \frac{3^r}{Y}\right) \\
&= \frac{1}{2XY}(\underbrace{3(6^{r-1} + 4)}_{<4 \cdot 6^{r-1}}Y + \underbrace{32 \cdot 3^r 6^{r-1}}_{=96 \cdot 18^{r-1}} - 10 \cdot 3^r 3^{r-1} - 2 \cdot 3^r) \\
&= \frac{1}{2XY}(4 \cdot 6^{r-1}Y + 104 \cdot 18^{r-1} - \underbrace{8 \cdot 18^{r-1}}_{>10r9^{r-1}} - \underbrace{30 \cdot 9^{r-1}}_{>2r3^{r-1}} - 2 \cdot 3^r) \\
&< \frac{1}{2XY}(4 \cdot 6^{r-1}(r^2 + 5r - 3)3^{r-1} + 4 \cdot 6^{r-1} + 104 \cdot 18^{r-1} - 10r9^{r-1} - 2r3^{r-1}) \\
&= \frac{1}{2XY}(18^{r-1}(4r^2 + 20r - 12) + 4 \cdot 6^{r-1} + 104 \cdot 18^{r-1} - 10r9^{r-1} - 2r3^{r-1}) \\
&= \frac{1}{2XY}(18^{r-1}(4r^2 + 20r + 92) + 4 \cdot 6^{r-1} - 10r9^{r-1} - 2r3^{r-1} \\
&= \frac{1}{2XY}(18^{r-1}(4r^2 + 20r + 36r - \underbrace{36r}_{\ge 108} + 92) + 4 \cdot 6^{r-1} - 10r9^{r-1} - 2r3^{r-1} \\
&\le \frac{1}{2XY}(18^{r-1}(4r^2 + 56r - 16) + 4 \cdot 6^{r-1} - 10r9^{r-1} - 2r3^{r-1} \\
&< \pi_{A_1}
\end{aligned}$$

In addition we have

$$\begin{aligned}
\pi_{M_1} &= \frac{1}{2}(\pi'_{M_1} + \pi''_{M_1}) = \frac{1}{2}\left(\frac{6^r + 4}{X} + \frac{3^r - 3^{r-1}}{Y}\right) = \frac{1}{2}\left(\frac{6^r + 4}{X} + \frac{2 \cdot 3^{r-1}}{Y}\right) \\
&= \frac{1}{2XY}((6^r + 4)(r^2 + 5r - 3)3^{r-1} + 6^r + 4 + 64 \cdot 3^{r-1}6^{r-1} - 20 \cdot 3^{r-1}3^{r-1} - 4 \cdot 3^{r-1}) \\
&= \frac{1}{2XY}((6^r + 4)(r^2 + 5r - 3)3^{r-1} + 72 \cdot 18^{r-1} - \underbrace{8 \cdot 18^{r-1}}_{>10r9^{r-1}} + \underbrace{6^r + 4 - 20 \cdot 9^{r-1} - 4 \cdot 3^{r-1}}_{<0}) \\
&< \frac{1}{2XY}(18^{r-1}(6r^2 + 30r - 18) + 3^{r-1}(4r^2 + 20r - 12) + 72 \cdot 18^{r-1} - 10r9^{r-1}) \\
&= \frac{1}{2XY}(18^{r-1}(6r^2 + 30r + 54) + 3^{r-1}\underbrace{(4r^2 + 22r - 12)}_{<3 \cdot 6^{r-1}} - 10r9^{r-1} - 2r3^{r-1}) \\
&< \frac{1}{2XY}(18^{r-1}(6r^2 + 30r + 54) + 3 \cdot 18^{r-1} - 10r9^{r-1} - 2r3^{r-1}) \\
&= \frac{1}{2XY}(18^{r-1}(6r^2 + 30r + 57) - 10r9^{r-1} - 2r3^{r-1}) \\
&< \pi_{A_1}
\end{aligned}$$

for $3 \leq r \leq 24$. The above calculations imply $\pi_{A_1} = \pi_{B_1} = \|\pi\|_\infty$ for $3 \leq r \leq 24$. So w.l.o.g A_1 is selected first by MFVS*$_{\text{Mean}}$ to be part of the FVS. Then $\{B_1\}$ is the unique optimal FVS of $\frac{2}{r}\mathcal{F} - A_1$. With an analogue reasoning as in the proof of Lemma 4.2 we conclude that B_1 is selected next by MFVS*$_{\text{Mean}}$. □

4.3 Weighted versions

Having presented the algorithms MFVS and MFVS$_{\text{Mean}}$ for the minimum feedback vertex set problem, it is reasonable to also develop counterparts for the weighted minimum feedback vertex set problem. These counterparts will be called WMFVS and WMFVS$_{\text{Mean}}$, respectively. But before stating, them it shall be argued that they are natural generalisations of the unweighted versions.

4.3.1 Motivation

Let $G = (V, A, w)$ with a weight function $w : V \to \mathbb{R}_+$ be an instance of the weighted minimum feedback vertex set problem. We will describe a reduction from the weighted to the unweighted minimum feedback vertex set problem that motivates the design of the algorithms WMFVS and WMFVS$_{\text{Mean}}$.

Without loss of generality, let us first make some simple assumptions. Because $\mathbb{Q}$ is dense in $\mathbb{R}$ we can assume all weights to be in $\mathbb{Q}$, i.e. $w : V \to \mathbb{Q}_+$. Moreover, since V is finite, there is an integer $N \in \mathbb{N}$ such that $Nw(v) \in \mathbb{N}$ for all $v \in V$. Thus, by replacing w with Nw, we can assume further $w : V \to \mathbb{N}$. Of course, the value of N may become arbitrarily large. But the integer N will not be used in the final algorithms. It is only an 'auxiliary constant' that serves the purpose to facilitate the description of the reduction. Its potentially large value is not an issue.

4.3.1.1 A reduction to the unweighted case

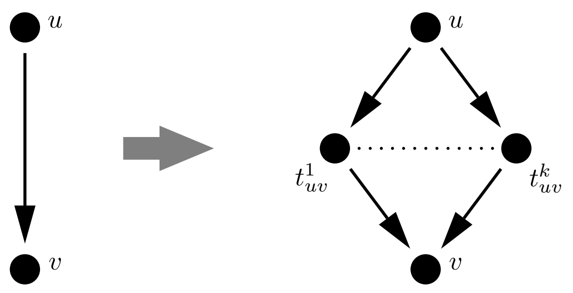

Figure 4.3: The replacement of the arc (u, v) by the paths (u, t^i_{uv}, v) with $i = 1, \ldots, k$.

The reduction itself is divided into two steps. The first step is to construct a weighted digraph $G' = (V', A', w')$ from G. This is done by replacing each arc

$(u, v) \in A$ by a set of paths (u, t^i_{uv}, v) $(i = 1, \dots, k)$ for a fixed integer $k \in \mathbb{N}$ as depicted in Figure 4.3. Formally, we have

$$V' := V \cup \overbrace{\{t^1_{uv}, \dots, t^k_{uv} : (u, v) \in A\}}^{=:V_A}$$

and

$$A' := \{(u, t^i_{uv}), (t^i_{uv}, v) : (u, v) \in A,\, t^i_{uv} \in V_A\}.$$

The weight function $w' : V' \to \mathbb{N}$ is defined as

$$w'(v) := \begin{cases} w(v) & \text{if } v \in V \\ 1 & \text{if } v \in V_A \end{cases}.$$

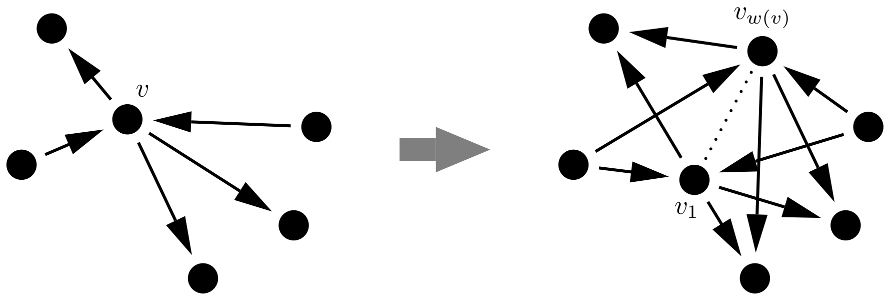

Figure 4.4: The splitting of v into the vertices $v_1, \dots, v_{w(v)}$.

In a second step we construct an unweighted digraph $H = (V'', A'')$ from G' by splitting each vertex $v \in V \cap V'$ into the vertices $v_1, \dots, v_{w(v)}$ as shown in Figure 4.4. More precisely, H is defined by

$$V'' := V_A \cup \overbrace{\{v_1, \dots, v_{w(v)} : v \in V\}}^{=:V_w}$$

and

$$A'' := \{(t^i_{uv}, v_j) : t^i_{uv} \in V_A,\, v_j \in V_w\} \cup \{(v_j, t^i_{vu}) : t^i_{vu} \in V_A,\, v_j \in V_w\}.$$

Having constructed the digraph H, let us examine its relation to G. This will be done by first comparing their FVSs and then the stationary distributions of M_G and M_H.

4.3.1.2 The feedback vertex sets of G and H

We start with a FVS $F_H \subset V''$ of H and want to construct a FVS $F \subset V$ of G. But first, let us examine the structure of F_H in more detail. Assume that $t^i_{uv} \in F_H \cap V_A$ is not redundant. Then, we have $t^1_{uv}, \dots, t^k_{uv} \in F_H$, too. This can be seen as follows. Because t^i_{uv} is not redundant, there is a cycle $C = (u_\ell, t^i_{uv}, v_{\ell'}, {}^1s, \dots, {}^rs, u) \in \mathcal{C}_H$ with $\ell \in \{1, \dots, w(u)\}$, $\ell' \in \{1, \dots, w(v)\}$ and ${}^1s, \dots, {}^rs \in V''$ such that

$$F_H \cap V(C) = \{t^i_{uv}\} \tag{4.4}$$

according to Proposition 1.3. Then, by construction of H, we also have a cycle $C' = (u_\ell, t^j_{uv}, v_{\ell'}, {}^1s, \ldots, {}^rs, u) \in \mathcal{C}_H$ for any $t^j_{uv} \in V_A$ $(j \neq i)$. Because F_H is a FVS of H, $F_H \cap V(C') \neq \emptyset$ holds and on account of (4.4) we get

$$F_H \cap V(C') = \{t^j_{uv}\}.$$

This means that t^j_{uv} is also not redundant. So we see

$$t^i_{uv} \text{ redundant} \Longleftrightarrow t^1_{uv}, \ldots, t^k_{uv} \text{ redundant}.$$

By applying a similar reasoning we also conclude

$$v_i \in V_w \text{ redundant} \Longleftrightarrow v_1, \ldots, v_{w(v)} \in V_w \text{ redundant}.$$

Hence, F_H being minimal, we can assume

$$t^i_{uv} \in F_H \Longrightarrow t^1_{uv}, \ldots, t^k_{uv} \in F_H \quad \text{and} \quad v_j \in F_H \Longrightarrow v_1, \ldots, v_{w(v)} \in F_H. \tag{4.5}$$

Thus, we are entitled to construct a FVS $F' \subset V'$ of G' by

$$F' := (F_H \cap V_A) \cup \{v \in V : v_1 \in F_H \cap V_w\}.$$

Given the minimality of F_H, it is obvious that F' is minimal, too. Moreover, by definition of F', it can be similarly assumed $t^1_{uv}, \ldots, t^k_{uv} \in F'$ if $t^i_{uv} \in F'$ for some $i \in \{1, \ldots, k\}$. Because $\mathrm{N}^+_{G'}(t^i_{uv}) = \{v\}$ for any $i = 1, \ldots, k$, the set

$$F := (F' \cap V) \cup \{v \in V : t^1_{uv} \in F'\}$$

is a FVS of G. It is minimal because F' is assumed so.

In order to compare $w(F)$ and $|F_H|$, note that $|F_H| = w'(F')$. Furthermore, observe that, speaking loosely, the vertices $t^1_{uv}, \ldots, t^k_{uv} \in F' \cap V_A$ are replaced by their common successor v to obtain F. Hence, assuming $k > \max\{w(v) : v \in V\}$ we have $w'(F') \geq w(F)$. To sum up,

$$|F_H| \geq w(F) \quad \text{and} \quad (|F_H| = w(F) \iff F_H \cap V_A = \emptyset)$$

holds.

From F another FVS

$$\widetilde{F}_H := \{v_1, \ldots, v_{w(v)}\} \tag{4.6}$$

of H can be constructed with $w(F) = |\widetilde{F}_H| \leq |F_H|$. Thus, the conclusion is that $F_H \cap V_A = \emptyset$ if F_H is optimal. In this case F is optimal, too. Conversely, given that F is optimal it induces an optimal FVS $\widetilde{F}_H$ of H via (4.6).

4.3.1.3 The stationary distributions of M_G and M_H

Now, consider the relation between G and H from another point of view: compare the stationary distributions π and π' of the Markov chains M_G and M_H, respectively.

Proposition 4.4. *For a strongly connected weighted digraph $G = (V, A, w)$ let the digraph H be defined as in section 4.3.1.1. If $\pi = (\pi_v)_{v \in V}$ and $\pi' = (\pi'_v)_{v \in V_H}$ denote the stationary distributions of the Markov chains M_G and M_H, respectively, then π' has the following form:*

$$\begin{aligned} \pi'_{v_i} &= \frac{1}{2w(v)} \pi_v && \text{for } v_i \in V_w \\ \pi'_{t^i_{uv}} &= \frac{1}{2k\,\mathrm{d}^+_G(u)} \pi_u && \text{for } t^i_{uv} \in V_A \end{aligned}$$

Proof. As G is assumed to be strongly connected, so is the digraph H. Thus the Markov chains M_G and M_H are irreducible, hence due to Proposition 3.14 the stationary distributions π and π' exist and are unique. Thus, it has to be shown that π' in the asserted form has the properties (3.2) of a stationary distribution. But, before showing this, note that by construction of H the following holds for the vertex degrees:

$$\begin{aligned} \mathrm{d}^-_H(v_i) &= k\,\mathrm{d}^-_G(v), \quad \mathrm{d}^+_H(v_i) = k\,\mathrm{d}^+_G(v) && \text{for } v_i \in V_w \\ \mathrm{d}^-_H(t^i_{uv}) &= w(u), \quad \mathrm{d}^+_H(t^i_{uv}) = w(v) && \text{for } t^i_{uv} \in V_A \end{aligned}$$

In the following calculations it will be assumed that M_G and M_H are defined on the probability spaces $(\Omega, \mathcal{F}, P)$ and $(\Omega, \mathcal{F}, P')$, respectively. Now, suppose π is a stationary distribution of M_G, i.e. (3.2) holds. Then, for a fixed $v_i \in V_w$ we have

$$\begin{aligned} \sum_{u \in V_H} \pi'_u \cdot P'(X_1 = v_i \mid X_0 = u) &= \sum_{t^j_{uv} \in V_A} \pi'_{t^j_{uv}} \cdot P'(X_1 = v_i \mid X_0 = t^j_{uv}) \\ &= \sum_{t^j_{uv} \in V_A} \frac{1}{2k\,\mathrm{d}^+_G(u)} \pi_u \frac{1}{\mathrm{d}^+_H(t^j_{uv})} \\ &= \sum_{j=1}^{k} \sum_{(u,v) \in A} \frac{1}{2k\,\mathrm{d}^+_G(u)\,\mathrm{d}^+_H(t^j_{uv})} \pi_u \\ &= \sum_{(u,v) \in A} \sum_{j=1}^{k} \frac{1}{2k\,\mathrm{d}^+_G(u) w(v)} \pi_u \\ &= \frac{1}{2w(v)} \sum_{(u,v) \in A} \sum_{j=1}^{k} \frac{1}{k\,\mathrm{d}^+_G(u)} \pi_u \\ &= \frac{1}{2w(v)} \sum_{(u,v) \in A} \frac{1}{\mathrm{d}^+_G(u)} \pi_u \\ &= \frac{1}{2w(v)} \underbrace{\sum_{u \in V} \pi_u \cdot P(X_1 = v \mid X_0 = u)}_{= \pi_v \text{ in view of (3.2)}} \\ &= \frac{1}{2w(v)} \pi_v = \pi'_{v_i}. \end{aligned}$$

For $t^i_{uv} \in V_A$

$$\begin{aligned}
\sum_{s\in V_H} \pi'_s \cdot P'(X_1 = t^i_{uv} \mid X_0 = s) &= \sum_{(u_j, t^i_{uv})\in A_H} \pi'_{u_j} \cdot P'(X_1 = t^i_{uv} \mid X_0 = u_j) \\
&= \sum_{(u_j, t^i_{uv})\in A_H} \frac{1}{\mathrm{d}^+_H(u_j)} \pi'_{u_j} = \sum_{(u_j, t^i_{uv})\in A_H} \frac{1}{k\,\mathrm{d}^+_G(u)} \frac{1}{2w(u)} \pi_u \\
&= \frac{1}{2k} \sum_{(u_j, t^i_{uv})\in A_H} \frac{1}{\mathrm{d}^+_G(u) w(u)} \pi_u = \frac{1}{2k} \sum_{j=1}^{w(u)} \frac{1}{\mathrm{d}^+_G(u) w(u)} \pi_u \\
&= \frac{1}{2k\,\mathrm{d}^+_G(u)} \pi_u = \pi'_{t^i_{uv}}
\end{aligned}$$

holds. So, finally

$$\begin{aligned}
\|\pi'\|_1 &= \sum_{v\in V_H} \pi'_v = \sum_{v_i\in V_w} \pi'_{v_i} + \sum_{t^i_{uv}\in V_A} \pi'_{t^i_{uv}} = \sum_{i=1}^{w(v)} \sum_{v\in V} \pi'_{v_i} + \sum_{i=1}^{k} \sum_{(u,v)\in A} \pi'_{t^i_{uv}} \\
&= \sum_{v\in V} \sum_{i=1}^{w(v)} \frac{1}{2w(v)} \pi_v + \sum_{(u,v)\in A} \sum_{i=1}^{k} \frac{1}{2k\,\mathrm{d}^+_G(u)} \pi_u \\
&= \frac{1}{2} \underbrace{\sum_{v\in V} \pi_v}_{=1} + \sum_{(u,v)\in A} \frac{1}{2\,\mathrm{d}^+_G(u)} \pi_u = \frac{1}{2} + \frac{1}{2} \sum_{v\in V} \underbrace{\sum_{u\in V} \pi_u \cdot P(X_1 = v \mid X_0 = u)}_{=\pi_v \text{ in view of (3.2)}} \\
&= \frac{1}{2} + \frac{1}{2} \underbrace{\sum_{v\in V} \pi_v}_{=1} = 1
\end{aligned}$$

shows that the equations (3.2) are satisfied for π' and the assertion follows. □

4.3.2 Algorithms

A naive way to apply the Markov chain methodology to the weighted minimum feedback vertex set problem would be to construct the unweighted instance $H = (V'', A'')$ from the weighted digraph $G = (V, A, w)$ as described in section 4.3.1.1 and to determine a FVS F_H of H using the known algorithms for the unweighted case. Then, in section 4.3.1.2 it has been shown how to derive a FVS $F \subset V$ of G from F_H such that $w(F) \leq |F_H|$ provided the parameter k is large enough which is assumed throughout the remainder of this section. So, let us for example consider the behaviour of MFVS* when confronted with the digraph H.

Recall that MFVS* always chooses the vertex $s \in V''$ to belong to the FVS F_H which maximises π'_s, where $\pi' = (\pi'_s)_{s\in V''}$ is the stationary distribution vector of M_H. It can be concluded from Proposition 4.4 that $s \notin V_A$ if k is large enough. Thus, s is of the form $s = v_i \in V_w$ with an appropriate $v \in V$. This implies that $s = v_i$ maximises the ratio $\frac{1}{2w(v)}\pi_v$ due to Proposition 4.4. Now, consider the stationary distribution vector $\pi'' = (\pi''_s)_{s\in V''\setminus\{v_i\}}$ of M_{H-v_i}. From

```
function SelectVertex(weighted digraph $G = (V, A, w)$) do

    matrix $P = (p_{uv})_{(u,v) \in V^2}$;
    vector $\pi = (\pi_v)_{v \in V}$;
    vector $\pi' = (\pi'_v)_{v \in V}$;
    vertex $v$;

    $P$ ← CreateTransitionMatrix($G$);
    $\pi'$ ← ComputeStationaryDistributionVector($P$);
    for all $v \in V$ do
        $\pi_v$ ← $\frac{\pi'_v}{w(v)}$;
    od;

    determine $v \in V$ with $\pi_v = \|\pi\|_\infty$;

    return $v$;
od;
```

Algorithm 4.5: The vertex selection heuristic of WMFVS.

Proposition 4.4 we conclude

$$\pi''_s = \pi'_s \quad \text{for} \quad s \in V'' \setminus \{v_1, \ldots, v_{w(v)}\}$$
$$\text{and}$$
$$\pi''_{v_j} = \frac{w(v)}{w(v)-1}\pi_{v_j} = \frac{1}{2(w(v)-1)}\pi_v \quad \text{for} \quad j \in \{1, \ldots, w(v)\} \setminus \{i\}.$$

So, it can be seen that $\pi''_{v_j} = \|\pi''\|_\infty$ for all $j \in \{1, \ldots, w(v)\} \setminus \{i\}$. Hence, one such $v_j \in V(H - v_i)$ is selected next by MFVS* to belong to the FVS F_H. Thus, eventually each $v_1, \ldots, v_{w(v)} \in V''$ is chosen to be part of F_H and the process repeats until the complete FVS F_H is determined by MFVS*. F_H has the property (4.5) while $F_H \cap V_A = \emptyset$. Consequently, a FVS F of G can be constructed with $w(F) = |F_H|$ as shown in section 4.3.1.2.

Once the FVS F_H is determined, it is not necessarily minimal. If for instance $v_i \in F_H$ (and hence each $v_1, \ldots, v_{w(v)} \in F_H$) is redundant, then the corresponding vertex $v \in F$ is redundant, too, and vice versa. Thus, the behaviour of MFVS* with input H can be simulated without the explicit construction of the digraph H. Just take the framework of MFVS as presented in Algorithm 4.1 and use the modified selection heuristic shown in Algorithm 4.5. The resulting algorithm is called WMFVS, while – as in the case of MFVS – WMFVS* shall denote the same algorithm except that the Levy-Low reductions are not applied. As seen above, MFVS* always selects a vertex $v_i \in V_w$ which maximises the ratio $\frac{\pi_v}{w(v)}$ when determining a FVS of H. By definition of F that is equivalent to the membership of v in F. Hence, WMFVS*, when confronted with G, exactly mimics MFVS* with input H. It determines the same FVS F which has been derived from F_H as explained in section 4.3.1.2.

```
function SelectVertex(weighted digraph $G = (V, A, w)$) do

    matrix $P = (p_{uv})_{(u,v) \in V^2}$;
    vector $\pi = (\pi_v)_{v \in V}$;
    vector $\pi' = (\pi'_v)_{v \in V}$;
    vector $\pi'' = (\pi''_v)_{v \in V}$;
    vertex $v$;

    $P \leftarrow$ CreateTransitionMatrix($G$);
    $\pi' \leftarrow$ ComputeStationaryDistributionVector($P$);
    $P \leftarrow$ CreateTransitionMatrix($G^{-1}$);
    $\pi'' \leftarrow$ ComputeStationaryDistributionVector($P$);
    for all $v \in V$ do
        $\pi_v \leftarrow \frac{\pi'_v + \pi''_v}{w(v)}$;
    od;

    determine $v \in V$ with $\pi_v = \|\pi\|_\infty$;

    return $v$;
od;
```

Algorithm 4.6: The vertex selection heuristic of WMFVS$_{\text{Mean}}$.

In the same way, another algorithm WMFVS$_{\text{Mean}}$ for the weighted minimum feedback vertex set problem can be derived from WMFVS$_{\text{Mean}}$. While sharing the framework with MFVS (see Algorithm 4.1), its vertex selection heuristic looks like in Algorithm 4.6.

Besides being motivated by the reduction of the weighted digraph $G = (V, A, w)$ to the unweighted digraph $H = (V'', A'')$, the algorithms WMFVS and WMFVS$_{\text{Mean}}$ are also natural extensions of the algorithms MFVS and MFVS$_{\text{Mean}}$. To obtain a FVS F of G with a small weight $w(F)$ one might consider successively choosing vertices $v \in V$ with small weight $w(v)$. If moreover v belongs to a large number of minimal cycles in G, that choice is surely reasonable. But what if v belongs to only a few minimal cycles? For the total weight $w(F)$ of F it might be desireable to select a heavier vertex that breaks many cycles in favour of many light vertices which break the same amount of minimal cycles in G. On the other hand, if $w(v)$ is too high, it is never favourable to select v as part of F because the large number of cycles containing v cannot compensate the weight penalty imposed by $w(v)$. However, there is no known way to determine the number of minimal cycles containing a vertex v in polynomial time. But this number can be estimated with the help of the stationary distribution $\pi = (\pi_v)_{v \in V}$ of M_G and $\pi' = (\pi'_v)_{v \in V}$ of $M_{G^{-1}}$. If we have for instance $\pi_v = \alpha \cdot \pi_u$, then intuitively it can be expected that v is passed by roughly α times more (minimal) cycles than u. Similarly, if $\pi_v + \pi'_v = \alpha \cdot (\pi_u + \pi'_u)$, again intuitively, it can be assessed that v belongs to roughly α times more (minimal) cycles than u. Of course, these statements do not hold for any instance. But typically π_v and $\pi_v + \pi'_v$ are reliable estimations for the number of cycles containing v. So, choosing the vertices $v \in V$ to belong to the FVS F of G based on the ratio $\frac{\pi_v}{w(v)}$ (resp. $\frac{\pi_v + \pi'_v}{w(v)}$) is a plausible compromise between selecting light vertices and vertices breaking many minimal cycles as done by WMFVS (resp. $\text{WMFVS}_{\text{Mean}}$).

Chapter 5

Randomised metaheuristics

He who leaves nothing to chance will do few things poorly, but he will do few things.

Edward F. Halifax

In the course of reviewing the state of the art concerning feedback set problems in Chapter 2, we have encountered some metaheuristics. One example is the GRASP algorithm described in section 2.3.1. It basically consists of iterating a randomised algorithm for the determination of minimal FVSs many times and returning the FVS with the smallest cardinality. Another metaheuristic in section 2.3.3 is Schwikowski's enumeration algorithm `GenerateMFVS`. It explores the superstructure digraph $\Phi(G, \mu_G)$ in BFS manner whose vertices correspond to minimal FVSs of a digraph G.

Motivated by these two examples, we have been attempting to develop similar metaheuristics that incorporate the Markov chain techniques elaborated in Chapter 4. For this purpose, $\text{MFVS}_{\text{Mean}}$ has been taken as the core algorithm because it proved to be best among all other deterministic heuristics according to the evidence provided in Chapter 6. The results of our attempt are described in section 5.1 and 5.2. While section 5.1 suggests a GRASP variant of $\text{MFVS}_{\text{Mean}}$, section 5.2 deals with a search procedure guided by $\text{MFVS}_{\text{Mean}}$ which has some similarities to `GenerateMFVS`.

5.1 Randomising $\text{MFVS}_{\text{Mean}}$

Recall the approximation algorithm $\text{MFVS}_{\text{Mean}}$ from section 4.2. Its heart is the vertex selection heuristic. From the technical point of view, the heuristic constructs a stochastic matrix P from the current digraph G and determines the unique stationary distribution vector π. The same is done for G^{-1} and the so obtained stationary distribution vectors π and π' are used as the selection criterion.

5.1.1 The algorithm

How can $\text{MFVS}_{\text{Mean}}$ be randomised? One approach is to perturb the entries of P randomly and independently while preserving nonnegativeness. This can be achieved by multiplying each entry p_{ij} by the factor $(1 + r_{ij} \cdot T)$, where $r_{ij} \in (-1, 1)$ is a random number and $T \in (0, 1]$. The resulting matrix P' is no longer stochastic, i.e. the row elements do not sum up to 1. This property can be forced by scaling the rows of P' appropriately, thereby getting the stochastic matrix Q. As a consequence of the perturbation, $G_P = G_Q$ holds. From the theory presented in Chapter 3, it follows that Q has a unique stationary distribution vector π if P has one. The same perturbation and reasoning can be applied to G^{-1} and we obtain another stationary distribution vector π'. Then, π and π' can be used to select a vertex from G as done in $\mathrm{MFVS}_{\mathrm{Mean}}$. So, the randomised version of $\mathrm{MFVS}_{\mathrm{Mean}}$ — referred to as $\mathrm{RMFVS}_{\mathrm{Mean}}$ — adopts the framework of MFVS given in Algorithm 4.1 while using the vertex selection heuristic shown in Algorithm 5.1. Note that in this context, the function `Random` returns a pseudorandom number from the range $(-1, 1)$.

In order to devise a GRASP variant of $\mathrm{MFVS}_{\mathrm{Mean}}$, which will be called $\mathrm{GRASP}_{\mathrm{MFVS}_{\mathrm{Mean}}}$, the randomised algorithm $\mathrm{RMFVS}_{\mathrm{Mean}}$ is iterated t times and the overall minimum cardinality FVS is returned. To ensure that the quality of the determined FVS is at least that of $\mathrm{MFVS}_{\mathrm{Mean}}$, no perturbations are imposed to the transition matrices during the first iteration. In other words, $\mathrm{GRASP}_{\mathrm{MFVS}_{\mathrm{Mean}}}$ consists of one call of $\mathrm{MFVS}_{\mathrm{Mean}}$ and $t - 1$ calls of $\mathrm{RMFVS}_{\mathrm{Mean}}$. The complete procedure is given in Algorithm 5.2.

At first sight, one might conclude that $\mathrm{GRASP}_{\mathrm{MFVS}_{\mathrm{Mean}}}$ violates the common principles of GRASP as fixed in [48] because $\mathrm{RMFVS}_{\mathrm{Mean}}$ does not explicitly build a restricted candidate list (RCL) in order to select a FVS vertex as does the GRASP implementation in section 2.3.1. However, this violation is only superficial. Indeed, one could build a RCL from the vertices obtained from multiple calls of the vertex selection heuristic in Algorithm 5.1. But constructing the RCL that way is inherently nondeterministic, i.e. each construction of a RCL yields a different one, even if the underlying digraph were the same each time. Thus, selecting randomly a vertex from this RCL is conceptionally the same as selecting, say, the first one. Hence, only the first element of the RCL needs to be determined.

What remained unspecified up to now, is the parameter T which controls the amount of perturbation the entries of the matrix are exposed to. If the value of T is too small, too few vertices have the chance to be chosen by the selection heuristic. In the extreme case, only one vertex does have this chance, and so no improvement, compared to $\mathrm{MFVS}_{\mathrm{Mean}}$, is achieved. On the other hand, if the value of T is chosen too large, there is too much diversity, i.e. too many vertices may be chosen. Hence, the heuristic degrades to a pure random selection. Thus, a compromise has to be found. After some testing, setting T constantly to $\frac{1}{10}$ proved to be reasonable. This simple setting seems unsatisfactory, though. Particularly, adjusting T depending on the matrix to be perturbed is desireable. However, we have not found a way to achieve that.

5.1.2 Runtime and implementation

Because $\mathrm{RMFVS}_{\mathrm{Mean}}$ has obviously the same runtime as $\mathrm{MFVS}_{\mathrm{Mean}}$, the runtime of $\mathrm{GRASP}_{\mathrm{MFVS}_{\mathrm{Mean}}}$ is t times the one of $\mathrm{MFVS}_{\mathrm{Mean}}$. As $\mathrm{MFVS}_{\mathrm{Mean}}$, it is implemented in `C++` using the software libraries `LEDA` [42] and `SparseLib++` [18]. For the perturbation the pseudorandom number generator `Mersenne Twister`

```
function SelectVertex(digraph $G = (V, A)$) do

     matrix $P = (p_{uv})_{(u,v) \in V^2}$;
     vector $\pi = (\pi_v)_{v \in V}$;
     vector $\pi' = (\pi'_v)_{v \in V}$;
     vector $\pi'' = (\pi''_v)_{v \in V}$;
     vertex $v$;

     $P$ ← CreateTransitionMatrix($G$);
     $P$ ← PerturbMatrix($P$);
     $\pi'$ ← ComputeStationaryDistributionVector($P$);
     $P$ ← CreateTransitionMatrix($G^{-1}$);
     $P$ ← PerturbMatrix($P$);
     $\pi''$ ← ComputeStationaryDistributionVector($P$);

     $\pi$ ← $\pi' + \pi''$;

     determine $v \in V$ with $\pi_v = \|\pi\|_\infty$;

     return $v$;
od;

function PerturbMatrix(matrix $P = (p_{uv})_{(u,v) \in V^2}$) do

     matrix $Q = (q_{uv})_{(u,v) \in V^2}$;
     vertex $u$;
     vertex $v$;
     real s;

     for all $u \in V$ do
          $s$ ← 0;
          for all $v \in V$ do
               $q_{uv}$ ← $p_{uv}$ * (1 + Random() * $T$);
               $s$ ← $s$ + $q_{uv}$;
          od;

          for all $v \in V$ do
               $q_{uv}$ ← $q_{uv}$ / $s$;
          od;
     od;

     return $Q$;
od;
```

Algorithm 5.1: The vertex selection heuristic of RMFVS$_{\text{Mean}}$.

```
function GRASP(digraph $G = (V, A)$, integer $t$) do

    integer i;
    vertex set $F_{\mathtt{min}}$;
    vertex set $F$;

    $F_{\mathtt{min}}$ ← $\mathtt{MFVS}_{\mathtt{Mean}}(G)$;

    for $i$ = 2 to $t$ do
        $F$ ← $\mathtt{RMFVS}_{\mathtt{Mean}}(G)$;

        if ($|F| < |F_{\mathtt{min}}|$) do
            $F_{\mathtt{min}}$ ← $F$;
        od;
    od;

    return $F_{\mathtt{min}}$;
od;
```

Algorithm 5.2: The GRASP variant of $\mathrm{MFVS}_{\mathrm{Mean}}$.

`MT19937` [41] has been used.

5.2 The Markovian search procedure

In section 2.3.3 we have seen an example of an uninformed search. While the version in Algorithm 2.5 is exhaustive, i.e each minimal FVS of a given digraph G is eventually generated, its heuristic variants typically perform poor with respect to their runtimes because of being uninformed. Our idea is to alter the search to be informed and that way being able to effort substantial pruning of the search tree. Of course, the resulting algorithm MarkovSearch is not guaranteed to find the optimal FVS, even if it is not time constrained. Nevertheless, it typically delivers better FVSs than Algorithm 2.6 within the same amount of computation time.

5.2.1 The algorithm

Schwikowski's Algorithm 2.5 implicitly constructs the superstructure digraph $\Phi(G, \mu_G)$ whose vertices represent minimal FVSs of a digraph $G = (V, A)$. If (F, F') is an arc of that superstructure digraph, then we have

$$\delta(F, F') := |F \setminus F'| = 1.$$

Let us interpret $\delta(F, F')$ as the distance[1] between F and F'. With this illustration in mind, we can say that Schwikowski's algorithm traverses the digraph $\Phi(G, \mu_G)$ in steps of width 1. Our idea is to allow MarkovSearch to move in one step from the current FVS F to more distant FVSs F'. Suppose we allow

[1]not in the strict mathematical sense, as $\delta(\ ,\ )$ is not symmetric

```
function GenSucc(digraph $G = (V, A)$, vertex set $F$, integer $d$) do

    vertex set $F'$;

    $F'$ ← $F$;
    Remove randomly $d$ vertices from $F'$;
    $F'$ ← $F'$ ∪ $\mathtt{MFVS}_{\mathtt{Mean}}(G - F')$;

    F' ← MakeMinimal($G$, $F'$);

    return $F'$
od;
```

Algorithm 5.3: The generation of a successor FVS of F based on step width d.

a step width of at most d.

How can a FVS F' be derived from F with $\delta(F, F') \leq d$? Take a subset $X \subset F$ of cardinality d and set $\widetilde{F} := F \setminus X$. Because of the minimality of F, the set $\widetilde{F}$ is not a FVS of G, i.e. $\widetilde{G} := G - \widetilde{F}$ is cyclic. Let $\widetilde{\widetilde{F}} := \mathrm{MFVS}_{\mathrm{Mean}}(\widetilde{G})$ be the FVS of $\widetilde{G}$ determined by $\mathrm{MFVS}_{\mathrm{Mean}}$. Then, $\widetilde{F} \cup \widetilde{\widetilde{F}}$ is a FVS of G with $\delta(F, \widetilde{F} \cup \widetilde{\widetilde{F}}) \leq d$. But it is not necessarily minimal. So, it is transformed into a minimal one by removing redundant vertices using Algorithm 4.2. The FVS obtained that way is denoted by F'. As a result, $\delta(F, F') \leq d$ possibly does not hold because some vertices of $\widetilde{F}$ might be redundant. Thus, actually it turns out that d represents a reference value only. However, this does not impair the algorithm itself because the notion of step width only served as an illustration.

Concerning the above description some questions remain open. How many FVSs F' are expanded from F? Because of runtime issues there cannot be explored all possibilities of selecting X. Most of them have to be pruned. This trade-off between completeness and information of the search is a design choice which results in an overall good performance. Accordingly, only k sets X are chosen by which k FVSs F' are derived from F. So, how to choose the subsets X? After having applied several heuristics, it turned out that determining X randomly achieves the best results. The process of deriving a FVS F' from a given FVS F is outlined in Algorithm 5.3.

Which of the generated nonexpanded FVS should be expanded preferably? In this respect we follow Schwikowski's reasoning and expand always the FVS having the smallest cardinality. This is a common principle called best first search [49]. With respect to the termination criterion, MarkovSearch attempts to perform t expansions. In other words, at most $k \cdot t + 1$ minimal FVSs are generated as shown in Algorithm 5.4.

Hence, only the step width d remains to be specified which turned out to be a difficult task. Generally, we are not able to present an explicit formula

```
function MarkovSearch(digraph $G = (V, A)$, integer $k$, $r$, $t$) do

    dictionary $D$
    priority queue $Q$;
    vertex set $F_{\mathtt{min}}$;
    vertex set $F$;
    vertex set $F'$;
    integer i;

    $D$ ← $\emptyset$;
    $Q$ ← $\emptyset$;

    $F$ ← $\mathtt{MFVS}_{\mathtt{Mean}}(G)$;
    $D$.insert($F$);
    $Q$.insert($F$);

    $F_{\mathtt{min}}$ ← $F$;

    while ($Q \neq \emptyset$ and $t > 0$) do
        $t$ ← $t - 1$;

        $F$ ← $Q$.extract_min();

        for $i$ = 1 to $k$ do
            $F'$ ← GenSucc($G$, $F$, $r$);

            if ($D$.search($F'$) $\neq$ nil) do
                $D$.insert($F'$);
                $Q$.insert($F'$);

                if ($|F'| < |F_{\mathtt{min}}|$) do
                    $F_{\mathtt{min}}$ ← $F'$;
                od;
            od;
        od;
    od;

    return $F_{\mathtt{min}}$;
od;
```

Algorithm 5.4: The local search procedure MarkovSearch.

that works well for every possible input instance $G = (V, A)$. Several plausible formulas, which involve the density of G, have been tried and none proved significantly better than the other ones. Yet, we are able to give some hints on how to choose d.

Firstly, if d is chosen too small, it is likely that $\delta(F, F')$ is close to zero and the search degenerates due to the lack of diversity. On the other hand, if d is chosen too large, $\delta(F, F')$ tends to be large. If F is assumed to be near-optimal, this means that F' is 'far' away from a good solution. Intuitively, it does not make much sense to set d larger than $\frac{|F|}{2}$ because opting for the replacement of more than the half of the vertices of F would imply that the entire FVS F was far from being optimal. Hence, the optimal setting is somewhere between these two extremes. Secondly, the optimal step width d increases when so does the density of G. A plausible explanation for this behaviour is that as G becomes denser, the size of an optimal FVS F of G increases. Obviously, $|F|$ is bounded by $|V|$. Hence, any approximation algorithm has less chances to make suboptimal decisions in the course of determining a near-optimal FVS. So, it does not suffice to attempt to improve the given FVS locally — a more global improvement is necessary, one has to increase the step width.

All these claims about the choice of the step width d are supported by the following experiments: We have run MarkovSearch with a varying number of expansions on random digraphs (see section 6.1.2) of different densities, whereas in each run always generating 2 successors. The results are shown in Figures 5.1–5.4. First, it can be seen that there is an optimal setting of d. Moreover, this optimum is clearly below $\frac{|F|}{2}$ if F is the FVS determined by $\text{MFVS}_{\text{Mean}}$. Furthermore, it is also evident that the optimum increases as the instances get denser.

5.3 Runtime and implementation

Let $\Lambda(n)$ be the runtime of $\text{MFVS}_{\text{Mean}}$ on a digraph with n vertices. During each generation of a successor FVS, $\text{MFVS}_{\text{Mean}}$ determines a FVS of a digraph with d vertices (see Algorithm 5.3), where d is the step width. From Algorithm 5.4 it can be seen that $k \cdot t$ such successors are generated, where k is the number of successors generated in one expansion and t the number of expansions. In addition, MarkovSearch is initialised by a FVS obtained from $\text{MFVS}_{\text{Mean}}$. So, the total runtime of MarkovSearch for determining a FVS of a digraph $G = (V, A)$ is $\Lambda(|V|) + kt\Lambda(d)$.

As in the case of all other Markov chain based algorithms presented so far, MarkovSearch is implemented in `C++` using the software libraries `LEDA` [42] and `SparseLib++` [18]. `Mersenne Twister MT19937` [41] has been taken as the pseudorandom number generator.

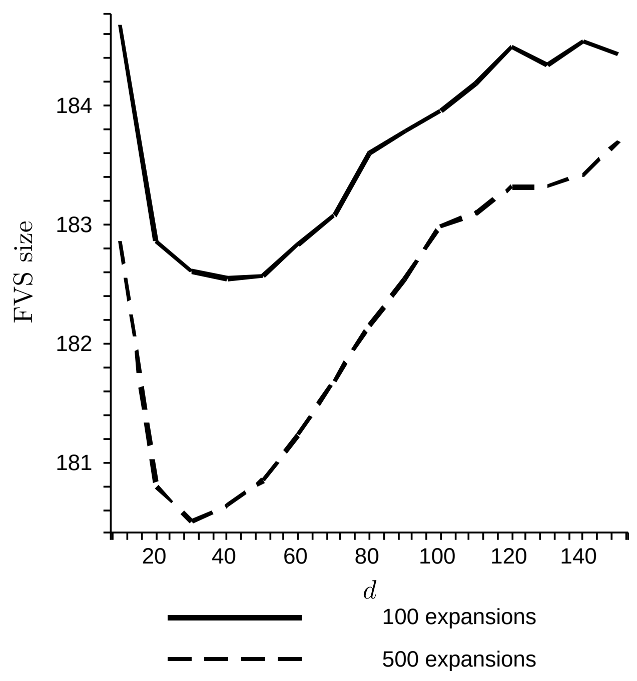

Figure 5.1: The results of experimenting with MarkovSearch with varying step width d on 100 random digraphs ($p = 0.05$) consisting of 300 vertices.

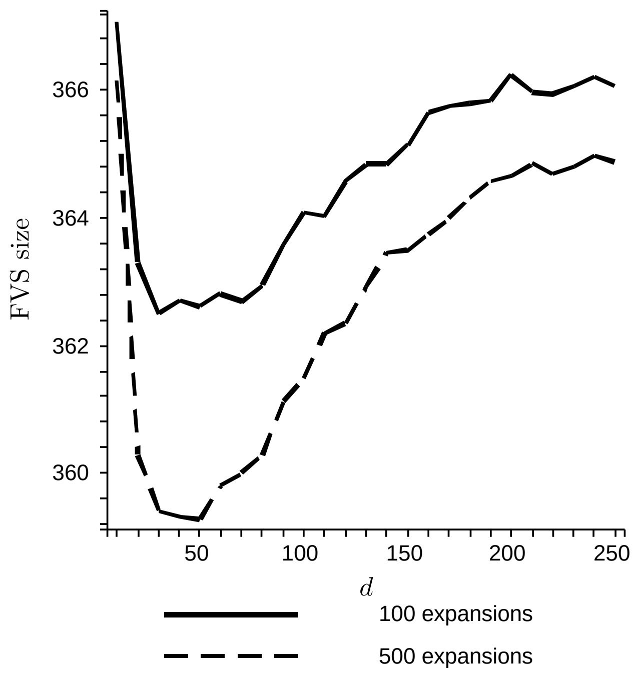

Figure 5.2: The results of experimenting with MarkovSearch with varying step width d on 100 random digraphs ($p = 0.05$) consisting of 500 vertices.

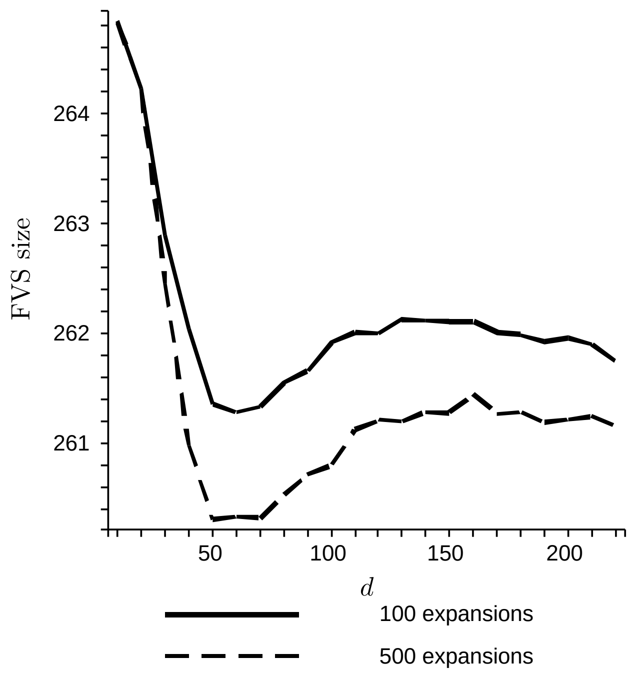

Figure 5.3: The results of experimenting with MarkovSearch with varying step width d on 100 random digraphs ($p = 0.2$) consisting of 300 vertices.

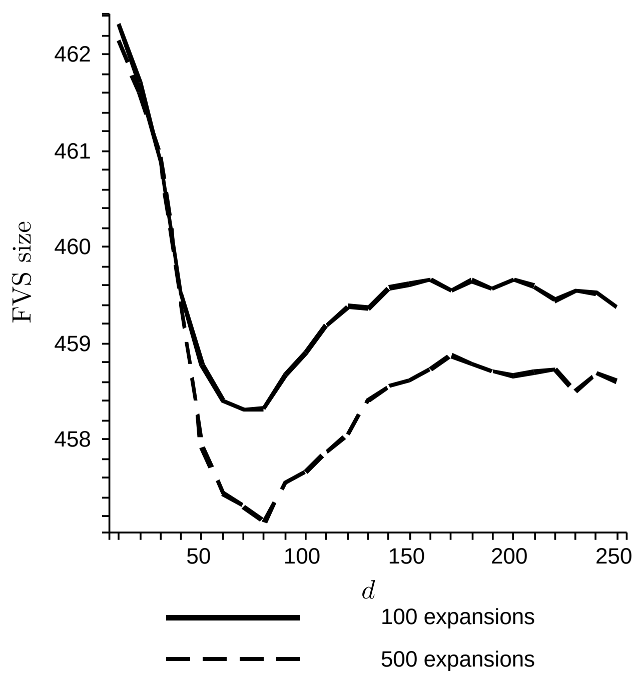

Figure 5.4: The results of experimenting with MarkovSearch with varying step width d on 100 random digraphs ($p = 0.2$) consisting of 500 vertices.

CHAPTER 6

Experimental results and discussion

To pry into the secrets of this world, we must make experiments. But experiment is a clumsy instrument, afflicted with a fatal determinacy which destroys causality.

Banesh Hoffman

Only when you free yourself to be a mere beginner again, which implies experimentation, do you progress to the next level of excellence...

Stella Reinwald

Before presenting performance results of the algorithms developed in Chapters 4 and 5, the algorithms which will be referred to in this chapter, shall be reviewed. This is done in section 6.1.1. Furthermore, section 6.1.2 describes the classes of digraphs on which the various approximation algorithms are run. The results of these runs are presented in section 6.2. We close the chapter with concluding remarks in section 6.3.

6.1 Experimental environment

6.1.1 Algorithms

The approximation algorithms considered in the present chapter are summarised in Figure 6.1. To have a clue on the performance of our algorithms we introduce the reference algorithm Simple. It uses the framework of Algorithm 4.1 but iteratively selects the next FVS vertex v which maximises the function

$$\sum_{v \in \mathrm{N}^-(v)} \mathrm{d}^-(v) \cdot \sum_{v \in \mathrm{N}^+(v)} \mathrm{d}^+(v).$$

In the course of our experiments it turned out that Simple performs better than the algorithm which iteratively selects the vertex v maximising the function $\mathrm{d}^-(v) \cdot \mathrm{d}^+(v)$.

The original algorithm $\mathrm{GRASP}_{\mathrm{Resende}}$ [22] has a flaw which is pointed out in [25]. The flaw leads $\mathrm{GRASP}_{\mathrm{Resende}}$ to miss a lot of redundant feedback vertices resulting in larger FVSs. This flaw has been removed and the corrected version of $\mathrm{GRASP}_{\mathrm{Resende}}$ is used for comparisons.

Simple The reference algorithm. It selects a vertex v that maximises the product

$$\sum_{v \in \mathrm{N}^-(v)} \mathrm{d}^-(v) \cdot \sum_{v \in \mathrm{N}^+(v)} \mathrm{d}^+(v).$$

MFVS The original Markovian FVS algorithm of Speckenmeyer. See Algorithm 4.1.

MFVS$_{\mathbf{Mean}}$ The variant of MFVS that determines the stationary distribution of the Markov chain associated with the digraph and its inverse digraph in order to select a vertex. See section 4.2 and Algorithm 4.4.

WMFVS The weighted variant of MFVS. See section 4.3 and Algorithm 4.5.

WMFVS$_{\mathbf{Mean}}$ The weighted version of MFVS$_{\mathrm{Mean}}$ See section 4.3 and Algorithm 4.6.

GRASP$_{\mathbf{MFVS_{Mean}}}$ A GRASP variant driven by MFVS$_{\mathrm{Mean}}$. See Algorithm 5.2.

MarkovSearch The local search procedure with similarities to the one of Schwikowski (section 2.3.3) which is guided by MFVS$_{\mathrm{Mean}}$. See Algorithm 5.4. For each experiment, exactly two successors per expansion are generated.

GRASP$_{\mathbf{Resende}}$ The GRASP algorithm for the minimum feedback vertex set problem as described by Resende et al. See section 2.3.1.

WFVS The algorithm of Demetrescu and Finocchi for the weighted minimum feedback vertex set problem. See Algorithm 2.4.

Figure 6.1: The approximation algorithms referred to in this chapter.

6.1.2 Test instances

We perform the experiments on samples of various classes. They include the following:

Random digraphs A random digraph G is constructed by adding an arc (u, v) to G with uniform probability $p \in [0, 1]$. These digraphs are introduced and studied in [29].

Regular digraphs A digraph $G = (V, A)$ is regular if for each vertex $v \in V$ we have $\mathrm{d}^-_G(v) = \mathrm{d}^+_G(v) = k$ for a fixed $k \in \mathbb{N}$. To obtain strong connectivity, we construct such digraphs by determining k arc disjoint Hamiltonian cycles and adding their arcs to G.

Resende digraphs The Resende digraphs consist of 40 randomly generated digraphs of varying density [45].

ISCAS89 The ISCAS89 benchmark for VLSI circuit testing [3] consists of 31 circuits. From these circuits their s-graphs can be constructed which serve as test instances for feedback vertex set problems. For the relevance of the s-graphs with respect to circuit testing and their relation to the minimum feedback vertex set problem, see [58].

6.2 Experimental results

In the following tables, the presented experimental results for random and regular digraphs are always the arithmetic mean of 100 randomly generated instances. The standard deviation of these results is denoted by σ.

6.2.1 Algorithms for the unweighted minimum feedback vertex set problem

6.2.1.1 Deterministic algorithms

In the present section, the deterministic algorithms Simple, MFVS and MFVS$_{\text{Mean}}$ are considered. First, we carry out experiments on random digraphs. The results are presented in Tables 6.1–6.8. When comparing the results of the algorithms Simple and MFVS, it can be seen that the latter outperforms the former. This observation surprises on account of the fact that Simple uses an ingenuous heuristic whereas MFVS uses a sophisticated one. The situation changes when comparing the results of Simple and MFVS$_{\text{Mean}}$. It can be seen that the sophisticated algorithm performs better than Simple. Thus, we conclude that MFVS$_{\text{Mean}}$ is a clear improvement over MFVS. This assessment is also confirmed by the standard deviations. It can be seen that the standard deviations of the results of MFVS$_{\text{Mean}}$ are typically smaller than those of the other results, which indicates that the results of MFVS$_{\text{Mean}}$ are more reliable.

vertices	Simple	MFVS	MFVS$_{\text{Mean}}$
50	0.14 (σ=0.35)	0.14 (σ=0.35)	0.14 (σ=0.35)
100	1.17 (σ=1.02)	1.17 (σ=1.02)	1.17 (σ=1.02)
150	4.64 (σ=1.88)	4.65 (σ=1.89)	4.64 (σ=1.88)
200	13.13 (σ=2.61)	13.15 (σ=2.61)	12.99 (σ=2.59)
250	26.92 (σ=3.67)	27.18 (σ=3.56)	26.65 (σ=3.56)
300	45.07 (σ=3.88)	46.00 (σ=4.18)	44.60 (σ=3.96)
350	66.01 (σ=4.23)	67.45 (σ=3.96)	65.09 (σ=3.95)
400	91.21 (σ=5.04)	92.93 (σ=5.35)	90.21 (σ=4.63)
450	118.80 (σ=4.63)	121.13 (σ=4.45)	116.50 (σ=4.23)
500	147.85 (σ=5.92)	150.89 (σ=5.57)	146.34 (σ=5.36)

Table 6.1: Test results for random graphs with $p = 0.01$.

vertices	Simple	MFVS	MFVS$_{\text{Mean}}$
50	0.88 (σ=0.79)	0.88 (σ=0.79)	0.88 (σ=0.79)
100	8.22 (σ=1.99)	8.23 (σ=1.97)	8.21 (σ=1.97)
150	23.66 (σ=2.75)	23.64 (σ=2.86)	23.34 (σ=2.82)
200	48.24 (σ=2.99)	49.08 (σ=3.15)	47.47 (σ=3.10)
250	76.40 (σ=3.47)	78.25 (σ=3.43)	75.62 (σ=3.38)
300	107.84 (σ=3.63)	110.10 (σ=4.09)	106.88 (σ=3.65)
350	144.37 (σ=4.09)	147.00 (σ=4.44)	141.44 (σ=3.74)
400	180.41 (σ=4.16)	183.73 (σ=4.05)	177.97 (σ=3.75)
450	219.81 (σ=4.39)	223.30 (σ=4.68)	216.81 (σ=4.08)
500	259.16 (σ=4.04)	263.69 (σ=4.64)	256.43 (σ=3.85)

Table 6.2: Test results for random graphs with $p = 0.02$.

vertices	Simple	MFVS	MFVS$_{\text{Mean}}$
50	2.83 (σ=1.27)	2.84 (σ=1.28)	2.83 (σ=1.27)
100	17.46 (σ=2.19)	17.69 (σ=2.19)	17.34 (σ=2.07)
150	42.80 (σ=2.77)	43.36 (σ=2.82)	41.94 (σ=2.68)
200	74.45 (σ=2.96)	75.51 (σ=3.13)	73.12 (σ=3.05)
250	110.00 (σ=3.63)	111.93 (σ=3.85)	108.49 (σ=3.34)
300	148.10 (σ=3.67)	150.78 (σ=3.62)	146.13 (σ=3.57)
350	187.99 (σ=3.82)	190.98 (σ=3.61)	185.59 (σ=3.27)
400	229.58 (σ=3.17)	233.24 (σ=3.95)	226.97 (σ=3.41)
450	272.83 (σ=3.38)	277.57 (σ=3.87)	269.70 (σ=3.44)
500	315.87 (σ=3.49)	320.59 (σ=4.16)	312.26 (σ=3.25)

Table 6.3: Test results for random graphs with $p = 0.03$.

vertices	Simple	MFVS	MFVS$_{\text{Mean}}$
50	5.09 (σ=1.57)	5.06 (σ=1.54)	5.06 (σ=1.56)
100	25.65 (σ=2.17)	26.05 (σ=1.99)	25.23 (σ=2.09)
150	56.17 (σ=2.86)	57.01 (σ=2.70)	55.24 (σ=2.60)
200	92.64 (σ=2.84)	94.22 (σ=3.02)	91.50 (σ=2.55)
250	132.63 (σ=3.21)	134.60 (σ=3.11)	130.67 (σ=2.68)
300	173.46 (σ=2.78)	176.35 (σ=3.13)	171.47 (σ=2.61)
350	216.65 (σ=3.09)	220.08 (σ=3.19)	214.18 (σ=2.99)
400	260.50 (σ=3.37)	264.68 (σ=3.78)	257.83 (σ=3.35)
450	305.34 (σ=3.17)	309.65 (σ=3.33)	302.38 (σ=3.16)
500	350.45 (σ=3.19)	355.04 (σ=3.25)	347.45 (σ=2.44)

Table 6.4: Test results for random graphs with $p = 0.04$.

vertices	Simple	MFVS	$\text{MFVS}_{\text{Mean}}$
50	7.53 (σ=1.77)	7.53 (σ=1.72)	7.48 (σ=1.70)
100	32.67 (σ=2.42)	33.38 (σ=2.31)	32.36 (σ=2.27)
150	67.27 (σ=2.88)	68.30 (σ=2.86)	66.33 (σ=2.50)
200	106.60 (σ=2.40)	108.39 (σ=2.73)	105.44 (σ=2.34)
250	148.53 (σ=2.62)	150.33 (σ=2.73)	146.81 (σ=2.66)
300	191.47 (σ=3.08)	194.32 (σ=3.14)	189.65 (σ=2.80)
350	236.39 (σ=2.98)	239.36 (σ=3.16)	234.04 (σ=2.51)
400	281.29 (σ=3.00)	285.54 (σ=2.89)	279.16 (σ=2.74)
450	327.58 (σ=2.94)	332.14 (σ=3.30)	325.04 (σ=2.64)
500	374.43 (σ=2.59)	379.16 (σ=3.19)	371.74 (σ=2.56)

Table 6.5: Test results for random graphs with $p = 0.05$.

vertices	Simple	MFVS	$\text{MFVS}_{\text{Mean}}$
50	17.73 (σ=1.68)	18.05 (σ=1.72)	17.49 (σ=1.62)
100	55.41 (σ=1.95)	56.03 (σ=1.97)	54.86 (σ=1.92)
150	98.44 (σ=2.16)	99.58 (σ=2.30)	97.26 (σ=1.96)
200	143.16 (σ=2.31)	145.09 (σ=2.34)	142.05 (σ=2.27)
250	189.52 (σ=2.23)	191.51 (σ=2.17)	188.26 (σ=2.09)
300	236.71 (σ=1.95)	239.31 (σ=2.30)	235.34 (σ=1.83)
350	284.80 (σ=1.94)	286.81 (σ=2.26)	283.03 (σ=2.08)
400	332.73 (σ=2.18)	335.77 (σ=2.10)	331.00 (σ=2.09)
450	380.77 (σ=2.17)	384.31 (σ=2.15)	378.96 (σ=1.98)
500	429.69 (σ=1.96)	432.65 (σ=1.98)	427.46 (σ=1.66)

Table 6.6: Test results for random graphs with $p = 0.1$.

vertices	Simple	MFVS	$\text{MFVS}_{\text{Mean}}$
50	24.35 (σ=1.38)	24.76 (σ=1.48)	24.12 (σ=1.39)
100	66.77 (σ=1.70)	67.71 (σ=1.73)	65.91 (σ=1.48)
150	112.83 (σ=1.67)	113.58 (σ=1.62)	111.77 (σ=1.72)
200	159.29 (σ=1.76)	161.36 (σ=1.64)	158.50 (σ=1.75)
250	207.41 (σ=1.62)	208.93 (σ=1.72)	206.40 (σ=1.46)
300	255.88 (σ=1.67)	257.83 (σ=1.71)	254.52 (σ=1.60)
350	304.30 (σ=1.68)	306.67 (σ=1.87)	303.21 (σ=1.54)
400	353.29 (σ=1.83)	355.41 (σ=1.92)	352.24 (σ=1.80)
450	402.22 (σ=1.57)	404.39 (σ=1.84)	401.16 (σ=1.70)
500	451.48 (σ=1.81)	453.83 (σ=1.83)	450.01 (σ=1.69)

Table 6.7: Test results for random graphs with $p = 0.15$.

vertices	Simple	MFVS	$\text{MFVS}_{\text{Mean}}$
50	29.31 (σ=1.29)	29.58 (σ=1.42)	28.83 (σ=1.39)
100	73.45 (σ=1.33)	74.44 (σ=1.47)	73.13 (σ=1.27)
150	120.62 (σ=1.51)	121.60 (σ=1.62)	119.83 (σ=1.39)
200	168.99 (σ=1.57)	170.50 (σ=1.42)	167.87 (σ=1.65)
250	217.34 (σ=1.55)	218.81 (σ=1.41)	216.57 (σ=1.37)
300	266.08 (σ=1.42)	267.98 (σ=1.46)	265.24 (σ=1.47)
350	315.09 (σ=1.44)	317.16 (σ=1.68)	314.52 (σ=1.37)
400	364.31 (σ=1.53)	366.33 (σ=1.64)	263.75 (σ=1.56)
450	413.80 (σ=1.29)	415.86 (σ=1.62)	412.92 (σ=1.38)
500	463.39 (σ=1.56)	465.01 (σ=1.59)	462.59 (σ=1.45)

Table 6.8: Test results for random graphs with $p = 0.20$.

Next, we perform the experiments for the three mentioned algorithms on regular digraphs, the results of which are given in Tables 6.9–6.16. Anything that has been said for the results on random digraphs applies for the results on regular digraphs. Particularly, $\text{MFVS}_{\text{Mean}}$ outperforms the other two algorithms. Note that regular digraphs are sparser than random digraphs. Thus, $\text{MFVS}_{\text{Mean}}$ proves superior on different classes of digraphs showing its overall superiority. The good performance is somehow surprising because all three algorithms select the first vertex of the FVS randomly. For Simple this is obvious, as its selection criterion are the degrees of the vertices which are entirely equal for any two vertices. Concerning MFVS and $\text{MFVS}_{\text{Mean}}$, it can be shown that all vertices of a regular digraph have the same stationary distribution, so both algorithms select the first vertex of the FVS randomly. As more and more vertices are selected, the digraph becomes less and less regular, thus the selection heuristics of the algorithms become more effective.

vertices	Simple	MFVS	$\text{MFVS}_{\text{Mean}}$
50	16.74 (σ=0.93)	16.46 (σ=0.97)	16.29 (σ=0.94)
100	30.83 (σ=1.23)	30.88 (σ=1.17)	29.77 (σ=1.22)
150	44.64 (σ=1.80)	44.60 (σ=1.57)	43.21 (σ=1.43)
200	57.90 (σ=1.88)	57.95 (σ=1.68)	56.23 (σ=1.41)
250	71.33 (σ=2.03)	71.46 (σ=1.86)	69.29 (σ=1.48)
300	85.23 (σ=2.07)	85.23 (σ=1.74)	82.11 (σ=1.80)
350	98.30 (σ=1.96)	98.12 (σ=2.21)	95.18 (σ=1.88)
400	111.99 (σ=2.71)	111.86 (σ=2.29)	107.99 (σ=1.91)
450	124.69 (σ=2.52)	125.05 (σ=2.46)	120.46 (σ=2.32)
500	138.28 (σ=2.65)	138.18 (σ=2.70)	134.03 (σ=2.50)

Table 6.9: Test results for regular digraphs with $k = 3$.

vertices	Simple	MFVS	$\text{MFVS}_{\text{Mean}}$
50	20.90 (σ=0.91)	20.69 (σ=0.98)	20.29 (σ=0.95)
100	38.67 (σ=1.30)	38.70 (σ=1.24)	37.69 (σ=1.19)
150	56.75 (σ=1.66)	56.75 (σ=1.54)	54.94 (σ=1.48)
200	74.39 (σ=1.90)	73.59 (σ=1.71)	71.57 (σ=1.69)
250	91.98 (σ=2.04)	91.44 (σ=1.95)	88.94 (σ=1.78)
300	109.41 (σ=2.02)	109.24 (σ=2.22)	105.56 (σ=1.81)
350	126.84 (σ=2.39)	126.30 (σ=2.22)	122.60 (σ=1.97)
400	144.06 (σ=2.47)	143.87 (σ=2.72)	139.84 (σ=2.08)
450	161.64 (σ=2.44)	161.38 (σ=2.76)	155.86 (σ=1.92)
500	178.99 (σ=2.75)	178.42 (σ=2.73)	172.63 (σ=2.68)

Table 6.10: Test results for regular digraphs with $k = 4$.

vertices	Simple	MFVS	MFVS$_{\text{Mean}}$
50	23.93 (σ=1.11)	23.85 (σ=1.02)	23.43 (σ=0.87)
100	45.13 (σ=1.38)	44.69 (σ=1.35)	43.77 (σ=1.36)
150	65.91 (σ=1.47)	65.58 (σ=1.44)	64.16 (σ=1.62)
200	86.74 (σ=1.79)	86.58 (σ=1.81)	84.06 (σ=1.60)
250	107.35 (σ=2.08)	106.83 (σ=1.98)	103.84 (σ=1.95)
300	127.79 (σ=2.35)	127.74 (σ=2.34)	123.89 (σ=1.92)
350	148.27 (σ=2.63)	147.51 (σ=2.09)	143.83 (σ=2.08)
400	168.59 (σ=2.44)	169.16 (σ=2.49)	163.56 (σ=2.74)
450	189.26 (σ=2.78)	188.94 (σ=2.79)	183.35 (σ=2.43)
500	209.74 (σ=2.99)	209.33 (σ=3.14)	203.08 (σ=2.42)

Table 6.11: Test results for regular digraphs with $k = 5$.

vertices	Simple	MFVS	MFVS$_{\text{Mean}}$
50	26.51 (σ=1.07)	26.33 (σ=1.04)	25.98 (σ=1.13)
100	49.75 (σ=1.41)	49.88 (σ=1.51)	48.84 (σ=1.43)
150	73.54 (σ=1.81)	73.39 (σ=1.68)	71.63 (σ=1.58)
200	96.72 (σ=1.98)	96.29 (σ=1.94)	93.97 (σ=1.60)
250	119.78 (σ=2.04)	119.57 (σ=2.20)	116.60 (σ=1.92)
300	142.86 (σ=2.17)	142.33 (σ=2.38)	138.31 (σ=1.96)
350	165.85 (σ=2.58)	165.10 (σ=2.46)	161.22 (σ=2.16)
400	189.15 (σ=2.82)	188.44 (σ=2.43)	183.40 (σ=2.44)
450	211.70 (σ=2.84)	211.67 (σ=3.09)	205.76 (σ=2.39)
500	234.55 (σ=2.96)	234.04 (σ=3.02)	227.88 (σ=2.49)

Table 6.12: Test results for regular digraphs with $k = 6$.

vertices	Simple	MFVS	MFVS$_{\text{Mean}}$
50	28.53 (σ=1.19)	28.55 (σ=0.96)	27.99 (σ=0.97)
100	54.12 (σ=1.34)	54.27 (σ=1.29)	52.88 (σ=1.26)
150	79.42 (σ=1.63)	79.51 (σ=1.66)	77.51 (σ=1.34)
200	105.04 (σ=2.00)	104.55 (σ=1.89)	102.24 (σ=1.73)
250	130.20 (σ=2.13)	129.87 (σ=2.08)	126.86 (σ=1.87)
300	155.27 (σ=2.62)	154.27 (σ=2.15)	150.99 (σ=1.97)
350	180.12 (σ=2.14)	180.08 (σ=2.28)	175.23 (σ=2.07)
400	204.98 (σ=2.62)	204.55 (σ=2.60)	199.80 (σ=2.41)
450	230.39 (σ=2.76)	229.39 (σ=2.94)	223.68 (σ=2.28)
500	254.85 (σ=3.05)	254.72 (σ=2.97)	248.23 (σ=2.47)

Table 6.13: Test results for regular digraphs with $k = 7$.

vertices	Simple	MFVS	MFVS$_{\text{Mean}}$
50	30.34 (σ=0.86)	30.14 (σ=1.09)	29.81 (σ=1.01)
100	57.59 (σ=1.50)	57.52 (σ=1.35)	56.38 (σ=1.35)
150	84.58 (σ=1.53)	84.57 (σ=1.61)	82.70 (σ=1.72)
200	111.48 (σ=1.88)	111.47 (σ=1.79)	109.06 (σ=1.67)
250	138.53 (σ=2.12)	138.58 (σ=2.24)	135.33 (σ=2.19)
300	165.41 (σ=2.54)	165.18 (σ=2.45)	161.26 (σ=2.18)
350	191.97 (σ=2.33)	191.88 (σ=2.72)	187.66 (σ=2.12)
400	219.18 (σ=2.44)	218.51 (σ=2.78)	213.49 (σ=2.53)
450	245.52 (σ=2.63)	245.46 (σ=2.93)	239.25 (σ=2.57)
500	271.98 (σ=2.93)	272.17 (σ=3.00)	265.24 (σ=2.87)

Table 6.14: Test results for regular digraphs with $k = 8$.

vertices	Simple	MFVS	MFVS$_{\text{Mean}}$
50	31.72 (σ=0.90)	31.76 (σ=1.02)	31.31 (σ=1.01)
100	60.48 (σ=1.40)	60.57 (σ=1.27)	59.31 (σ=1.29)
150	89.00 (σ=1.48)	89.16 (σ=1.73)	87.36 (σ=1.51)
200	117.29 (σ=1.72)	117.71 (σ=1.92)	114.91 (σ=1.69)
250	145.75 (σ=2.23)	146.04 (σ=2.18)	142.87 (σ=2.02)
300	174.03 (σ=2.52)	174.12 (σ=2.34)	170.16 (σ=2.00)
350	202.12 (σ=2.45)	202.36 (σ=2.84)	197.57 (σ=2.22)
400	230.69 (σ=2.93)	230.66 (σ=2.16)	225.46 (σ=2.08)
450	258.37 (σ=2.56)	258.61 (σ=2.92)	253.00 (σ=2.40)
500	286.29 (σ=3.12)	286.53 (σ=2.96)	280.38 (σ=2.56)

Table 6.15: Test results for regular digraphs with $k = 9$.

vertices	Simple	MFVS	MFVS$_{\text{Mean}}$
50	32.91 (σ=0.92)	33.08 (σ=0.96)	32.67 (σ=0.92)
100	63.07 (σ=1.27)	63.12 (σ=1.41)	61.89 (σ=1.18)
150	92.61 (σ=1.77)	93.16 (σ=1.41)	91.00 (σ=1.46)
200	122.48 (σ=1.90)	122.84 (σ=1.89)	120.49 (σ=1.71)
250	151.99 (σ=2.04)	152.29 (σ=2.22)	148.94 (σ=1.97)
300	181.81 (σ=2.24)	181.87 (σ=2.18)	178.05 (σ=1.99)
350	211.17 (σ=2.47)	211.43 (σ=2.24)	206.85 (σ=2.18)
400	240.98 (σ=2.42)	240.87 (σ=2.64)	235.51 (σ=2.11)
450	269.87 (σ=2.64)	270.62 (σ=2.77)	264.69 (σ=2.17)
500	298.94 (σ=3.09)	299.94 (σ=3.45)	293.54 (σ=2.60)

Table 6.16: Test results for regular digraphs with $k = 10$.

In order to complete the exposition of the experimental results with respect to the deterministic approximation algorithms, we present Table 6.17. It contains the results for the ISCAS89 benchmark. In accordance with the previous tables it can be seen that $\mathrm{MFVS}_{\mathrm{Mean}}$ is superior to the other algorithms. It fails to yield the optimum solution on one occasion.

instance	Optimal	Simple	MFVS	$\mathrm{MFVS}_{\mathrm{Mean}}$	LR
s1196	0	0	0	0	0
s1238	0	0	0	0	0
s13207	59	59	59	59	59
s1423	21	21	21	21	22
s1488	5	5	5	5	5
s1494	5	5	5	5	5
s15850	88	88	90	89	89
s208	0	0	0	0	0
s27	1	1	1	1	1
s298	1	1	1	1	1
s344	5	5	5	5	5
s349	5	5	5	5	5
s35932	306	306	306	306	306
s382	9	9	9	9	9
s38417	374	378	374	374	374
s38584	292	293	293	292	296
s386	5	5	5	5	5
s400	9	9	9	9	9
s420	0	0	0	0	0
s444	9	9	9	9	9
s510	5	5	5	5	5
s526	3	3	3	3	3
s5378	30	30	30	30	30
s641	7	7	7	7	7
s713	7	7	7	7	7
s820	4	4	4	4	4
s832	4	4	4	4	4
s838	0	0	0	0	0
s9234	53	53	53	53	53
s953	5	5	5	5	5
sum	1315	1320	1318	1316	1321

Table 6.17: Experimental results for the ISCAS89 benchmark. The column LR contains the best solution obtained by different heuristics by Lee and Reddy [35], as cited in [44].

6.2.1.2 Randomised algorithms

The randomised algorithms MarkovSearch, $\text{GRASP}_{\text{MFVS}_{\text{Mean}}}$ and $\text{GRASP}_{\text{Resende}}$ are compared in this section. We start with four samples of random digraphs, as shown in Table 6.18. Table 6.19 contains the corresponding runtimes. We see that MarkovSearch outperforms the other two algorithms both in runtime and quality of the solutions. The difference between the runtimes of $\text{GRASP}_{\text{Resende}}$ and the other two algorithms is striking. While MarkovSearch and $\text{GRASP}_{\text{MFVS}_{\text{Mean}}}$ consume only a fraction of time of $\text{GRASP}_{\text{Resende}}$, they deliver better results. The same is true with respect to the results for regular digraphs, as shown in Table 6.20 and 6.21. Nevertheless, the standard deviations of MarkovSearch are typically higher than those of the other two algorithms. This indicates that although MarkovSearch achieves the best results, these results are slightly less reliable than those of $\text{GRASP}_{\text{MFVS}_{\text{Mean}}}$ and $\text{GRASP}_{\text{Resende}}$. However, MarkovSearch still remains superior.

One also observes that $\text{GRASP}_{\text{MFVS}_{\text{Mean}}}$ benefits only little from the doubling of the number of iterations. At this point, we do not investigate how the performance of MarkovSearch depends on the number of expansions. This issue has been addressed in Chapter 5 in Figures 5.1–5.4. It can be seen that the benefit of more expansions depends on the density of the digraphs. It decreases as the density increases.

p	$\text{GRASP}_{\text{MFVS}_{\text{Mean}}}$ 10 iterations	$\text{GRASP}_{\text{MFVS}_{\text{Mean}}}$ 20 iterations	MarkovSearch 100 expansions	$\text{GRASP}_{\text{Resende}}$
0.05	186.93 (σ=2.26)	186.11 (σ=2.11)	183.21 (σ=2.48)	187.27 (σ=2.31)
0.10	232.80 (σ=1.69)	232.30 (σ=1.28)	229.74 (σ=1.43)	232.91 (σ=1.33)
0.15	252.19 (σ=1.25)	251.82 (σ=1.09)	249.74 (σ=1.22)	251.70 (σ=1.05)
0.20	263.30 (σ=0.92)	262.91 (σ=0.96)	261.20 (σ=1.07)	262.59 (σ=0.92)

Table 6.18: Test results for random graphs with 300 vertices and varying p. The step width varies from 30 to 60 in steps of 10 (cf. Figure 5.1 and 5.3).

p	$\text{GRASP}_{\text{MFVS}_{\text{Mean}}}$ 10 iterations	$\text{GRASP}_{\text{MFVS}_{\text{Mean}}}$ 20 iterations	MarkovSearch 100 expansions	$\text{GRASP}_{\text{Resende}}$
0.05	653 sec.	1241 sec.	661 sec.	21238 sec.
0.10	1149 sec.	2186 sec.	1040 sec.	81014 sec.
0.15	1631 sec.	3261 sec.	1419 sec.	164625 sec.
0.20	2109 sec.	4328 sec.	1840 sec.	274136 sec.

Table 6.19: Runtimes for the experiments from Table 6.18.

Finally, the randomised algorithms $\text{GRASP}_{\text{MFVS}_{\text{Mean}}}$, MarkovSearch and $\text{GRASP}_{\text{Resende}}$ are compared by applying them to the Resende digraphs. As in the case of random and regular digraphs, $\text{GRASP}_{\text{MFVS}_{\text{Mean}}}$ and $\text{GRASP}_{\text{Resende}}$ prove inferior to MarkovSearch. This applies to both the quality of the solutions and the runtime. For the instances involving up to 100 vertices, all algorithms yield comparable results. However, for the larger instances differences, in performance and runtime are evident. In such a situation, the huge time consumption

k	$\text{GRASP}_{\text{MFVS}_{\text{Mean}}}$ 10 iterations	$\text{GRASP}_{\text{MFVS}_{\text{Mean}}}$ 20 iterations	MarkovSearch 100 expansions	$\text{GRASP}_{\text{Resende}}$
4	136.18 (σ=1.42)	135.26 (σ=1.38)	132.23 (σ=2.11)	140.67 (σ=0.96)
6	180.05 (σ=1.33)	279.32 (σ=1.28)	175.01 (σ=1.76)	184.59 (σ=1.23)
8	209.95 (σ=1.42)	209.08 (σ=1.27)	204.82 (σ=1.66)	214.17 (σ=1.11)
10	232.12 (σ=1.37)	231.41 (σ=1.63)	226.56 (σ=1.71)	235.77 (σ=1.17)

Table 6.20: Test results for regular graphs with 400 vertices and varying k.

k	$\text{GRASP}_{\text{MFVS}_{\text{Mean}}}$ 10 iterations	$\text{GRASP}_{\text{MFVS}_{\text{Mean}}}$ 20 iterations	MarkovSearch 100 expansions	$\text{GRASP}_{\text{Resende}}$
4	517 sec.	937 sec.	565 sec.	8844 sec.
6	691 sec.	1269 sec.	864 sec.	13429 sec.
8	840 sec.	1625 sec.	1163 sec.	18205 sec.
10	951 sec.	1858 sec.	1458 sec.	24268 sec.

Table 6.21: Runtimes for the experiments from Table 6.20.

of $\text{GRASP}_{\text{Resende}}$ is striking, while in the same time the algorithm delivers the worst overall results. The digraph with 1000 vertices and 20000 arcs is an extreme example for this. The FVS determined by MarkovSearch is smaller than the one determined by $\text{GRASP}_{\text{Resende}}$ by 20 vertices.

vert.	arcs	Opt.	GRASP_R	GRASP_M 10 iterat.	GRASP_M 20 iterat.	M-Search 100 expan.
50	100	3	3	3	3	3
50	150	9	9	9	9	9
50	200	13	13	13	13	13
50	250	17	17	17	17	17
50	300	19	19	19	19	20
50	500	28	28	29	29	28
50	600	31	31	32	32	31
50	700	33	33	33	33	34
50	800	34	34	35	34	34
50	900	36	36	36	36	36
100	200	9	9	9	9	9
100	300	17	17	19	17	17
100	400	23	23	23	24	23
100	500	32	32	33	33	32
100	600	36	38	38	38	37
100	1000	53	54	55	54	54
100	1100	54	55	55	56	55
100	1200	57	58	58	58	57
100	1300	60	61	62	62	60
100	1400	61	62	63	64	61
500	1000	31	32	33	33	31
500	1500	≤ 63	68	69	69	63
500	2000	≤ 101	107	108	102	103
500	2500	≤ 133	146	141	142	138
500	3000	≤ 164	175	175	174	168
500	5000	≤ 238	252	249	251	243
500	5500	≤ 253	270	266	264	261
500	6000	≤ 267	284	277	277	271
500	6500	≤ 279	294	294	290	284
500	7000	≤ 290	304	303	302	296
1000	3000	≤ 128	139	135	136	133
1000	3500	≤ 162	175	173	171	167
1000	4000	≤ 193	206	206	203	202
1000	4500	≤ 229	249	243	243	240
1000	5000	≤ 260	283	278	275	268
1000	10000	≤ 476	508	498	499	493
1000	15000	≤ 591	619	613	611	606
1000	20000	≤ 659	692	681	680	672
1000	25000	≤ 710	733	726	728	723
1000	30000	≤ 749	773	768	768	759
sum		≤ 6601	6941	6877	6863	6751
total time		—	41626 sec.	800 sec.	1466 sec.	733 sec.

Table 6.22: Experimental results for the Resende digraphs. The considered algorithms are $\text{GRASP}_\text{Resende}$ (GRASP_R), $\text{GRASP}_{\text{MFVS}_\text{Mean}}$ (GRASP_M) and MarkovSearch (M-Search). The step width $\min\{\frac{|F|}{2}, 40\}$ has been used for MarkovSearch where F is the FVS to be expanded.

6.2.2 Algorithms for the weighted minimum feedback vertex set problem

p	WFVS	WMFVS	WMFVS$_{\text{Mean}}$
0.05	15106.93 (σ=480.86)	14169.98 (σ=405.86)	14015.86 (σ=396.19)
0.10	17977.80 (σ=504.25)	17270.20 (σ=508.38)	17093.24 (σ=488.02)
0.15	19147.09 (σ=496.23)	18593.98 (σ=495.09)	18474.23 (σ=482.37)
0.20	19865.35 (σ=541.00)	19400.23 (σ=502.07)	19294.91 (σ=503.07)

Table 6.23: Test results for random graphs with 400 vertices and varying p.

p	WFVS	WMFVS	WMFVS$_{\text{Mean}}$
0.05	207 sec.	226 sec.	256 sec.
0.10	231 sec.	275 sec.	325 sec.
0.15	259 sec.	323 sec.	397 sec.
0.20	288 sec.	367 sec.	465 sec.

Table 6.24: Runtimes for the experiments from Table 6.23.

k	WFVS	WMFVS	WMFVS$_{\text{Mean}}$
4	8108.60 (σ=363.77)	7097.14 (σ=276.34)	6991.38 (σ=290.54)
6	11415.75 (σ=388.88)	10114.85 (σ=336.16)	9963.84 (σ=293.30)
8	13787.14 (σ=471.04)	12282.72 (σ=419.54)	12124.60 (σ=424.99)
10	15266.28 (σ=518.08)	13849.74 (σ=390.74)	13653.53 (σ=378.29)

Table 6.25: Test results for random graphs with 500 vertices and varying p.

We now compare the performance of WFVS, WMFVS and WMFVS$_{\text{Mean}}$. As usual, the three algorithms are compared by running them on samples of random and regular digraphs. The weights of the vertices are random integers in the range $[10, 100]$. The results of these runs are shown in Tables 6.23–6.26.

The superiority of WMFVS and WMFVS$_{\text{Mean}}$ over WFVS is evident, whereas WMFVS$_{\text{Mean}}$ performs only slightly better than WMFVS. This superiority is particularly pronounced for the regular digraphs which are sparse compared to the random digraphs. Furthermore, the results of WMFVS$_{\text{Mean}}$ are more reliable than those of the other two algorithms. This is supported by comparing the respective standard deviations.

Concerning the runtime of the three algorithms, it can be noticed that WFVS is the fastest. However, the larger runtime of the WMFVS and WMFVS$_{\text{Mean}}$ is justifiable in view of the performance.

k	WFVS	WMFVS	WMFVS$_{\text{Mean}}$
4	375 sec.	369 sec.	386 sec.
6	367 sec.	381 sec.	401 sec.
8	385 sec.	393 sec.	421 sec.
10	379 sec.	393 sec.	420 sec.

Table 6.26: Runtimes for the experiments from Table 6.25.

6.3 Conclusion

In this thesis we have developed several new approximation algorithms for feedback set problems including MFVS$_{\text{Mean}}$, GRASP$_{\text{MFVS}_{\text{Mean}}}$, MarkovSearch, WMFVS and WMFVS$_{\text{Mean}}$. In section 6.2.1.1 we have carried out experiments with different classes of digraphs all of which consistently demonstrated that MFVS$_{\text{Mean}}$ is currently the best one among the deterministic algorithms. Furthermore, we have seen that the idea behind MFVS$_{\text{Mean}}$ is a major improvement over MFVS, which is the reason for the success of MFVS$_{\text{Mean}}$. The same idea is applied in WMFVS$_{\text{Mean}}$. Accordingly, in section 6.2.2 we have seen that it delivers better results than WFVS.

With respect to the randomised algorithms, it could be seen that both GRASP$_{\text{MFVS}_{\text{Mean}}}$ and MarkovSearch perform better than GRASP$_{\text{Resende}}$, with MarkovSearch being the best of these three. Currently, GRASP$_{\text{Resende}}$ is considered to be the best approximation algorithm for the minimum feedback vertex set problem [20]. The fact that GRASP$_{\text{MFVS}_{\text{Mean}}}$ and MarkovSearch are superior to GRASP$_{\text{Resende}}$ demonstrates that these two algorithms have to be considered the algorithms of choice.

The aforementioned results emphasise the potential of Markov chains in approximating minimum cardinality (resp. weight) feedback vertex sets. But the presented ideas in this thesis are not the last word on the subject. These ideas can probably be combined with other ideas, also involving Markov chains, to obtain even better algorithms. For instance, any of the considered algorithms primarily attempts to select a vertex which shall belong to a FVS. However, it might occasionally be more beneficial to first exclude a vertex from the digraph that shall not belong to an optimal FVS. Another idea for selecting a vertex is to devise a heuristic that a priori chooses promising candidates. Then, in a second step, another heuristic selects a FVS vertex among the candidates. A further topic not mentioned in this thesis is the determination of lower bounds on optimal FVSs using Markov chains. Typically, determining good lower bounds is harder for dense digraphs than for sparse digraphs. In this context, the ability to determine large induced dicliques is substantial, as demonstrated in [37]. The employment of Markov chains for this task is promising. All this is subject of further research.

Appendix A

New digraph reductions

Everything should be made as simple as possible, but not one bit simpler.

Albert Einstein
(attributed)

This chapter is devoted to some new digraph reductions for the minimum feedback vertex set problem. In addition to presenting the new digraph reductions, we will also prove that any order of application of these reductions results in a unique digraph up to isomorphism (Church-Rosser property). But before stating them and proving their correctness, some preliminary work has to be done.

A.1 Preliminary results

We start by recalling some simple observations.

Observation A.1. *Let $G = (V, A)$ be a digraph. Then, for any $u, v \in V$ and any $a, b \in V \times V$ the following holds provided it is well defined:*

(i). $G + u + v = G + v + u$

(ii). $G - u - v = G - v - u$

(iii). $G + a + b = G + b + a$

(iv). $G - a - b = G - b - a$

(v). $G + v + a = G + a + v$

(vi). $G + v - a = G - a + v$

(vii). $G + a - v = \begin{cases} G - v & \textit{if } a = (u, v) \vee a = (v, u) \\ G - v + a & \textit{else} \end{cases}$

(viii). $G - a - v = G - v - a = \begin{cases} G - v & \textit{if } a = (u, v) \vee a = (v, u) \\ G - v - a & \textit{else} \end{cases}$

(ix). $G - v = G \circ v$ *if* $\mathrm{d}^-_G(v) \cdot \mathrm{d}^+_G(v) = 0$

Proof. Only (ix) has to be justified. According to the definition of the exclusion

$$G \circ v = G - v + \mathrm{N}^-_G(v) \times \mathrm{N}^+_G(v)$$

holds. If $\mathrm{d}^-_G(v) \cdot \mathrm{d}^+_G(v) = 0$, then $\mathrm{N}^-_G(v) \times \mathrm{N}^+_G(v)$ is obviously the empty set and hence the assertion follows. □

Observation A.2. *Let $G = (V, A)$ be a digraph and let $u, v \in V$ be two distinct vertices of G. Then*

(i). $\mathrm{N}^-_G(v) = \mathrm{N}^-_{G-u}(v) = \mathrm{N}^-_{G\circ u}(v)$ *if* $u \notin \mathrm{N}^-_G(v)$
$\mathrm{N}^+_G(v) = \mathrm{N}^+_{G-u}(v) = \mathrm{N}^+_{G\circ u}(v)$ *if* $u \notin \mathrm{N}^+_G(v)$

(ii). $\mathrm{N}^-_{G\circ u}(v) = \mathrm{N}^-_{G-u}(v) \cup \mathrm{N}^-_G(u)$ *if* $u \in \mathrm{N}^-_G(v)$
$\mathrm{N}^+_{G\circ u}(v) = \mathrm{N}^+_{G-u}(v) \cup \mathrm{N}^+_G(u)$ *if* $u \in \mathrm{N}^+_G(v)$

(iii). $$\mathrm{N}^=_{G\circ u}(v) = \begin{cases} \mathrm{N}^=_G(v) \cup (\mathrm{N}^-_G(u) \cap \mathrm{N}^+_G(v)) & \textit{if } u \in \mathrm{N}^-_G(v) \setminus \mathrm{N}^=_G(v) \\ \mathrm{N}^=_G(v) \cup (\mathrm{N}^+_G(u) \cap \mathrm{N}^-_G(v)) & \textit{if } u \in \mathrm{N}^+_G(v) \setminus \mathrm{N}^=_G(v) \\ \mathrm{N}^=_G(v) & \textit{if } u \notin \mathrm{N}_G(v) \end{cases}$$

Proof. Assertion (i) follows from the definition of the deletion and the exclusion, respectively. To see (ii), recall the definition of the exclude operation:

$$G \circ u = G - u + \mathrm{N}^-_G(u) \times \mathrm{N}^+_G(u)$$

Assuming $u \in \mathrm{N}^-_G(v)$, i.e $v \in \mathrm{N}^+_G(u)$, we obtain

$$G \circ u = G - u + \mathrm{N}^-_G(u) \times (\{v\} \,\dot\cup\, \mathrm{N}^+_{G-v}(u)) = G - u + \mathrm{N}^-_G(u) \times \{v\} + \mathrm{N}^-_G(u) \times \mathrm{N}^+_{G-v}(u)$$

and the assertion follows. The case $u \in \mathrm{N}^+_G(v)$ is shown analogously.

Finally, (iii) follows from (i) and (ii). If e.g. $u \in \mathrm{N}^-_G(v) \setminus \mathrm{N}^=_G(v)$, then we have

$$\begin{aligned}\mathrm{N}^=_{G\circ u}(v) &= \overbrace{\mathrm{N}^-_{G\circ u}(v)}^{=\mathrm{N}^-_{G-u}(v) \cup \mathrm{N}^-_G(u)} \cap \underbrace{\mathrm{N}^+_{G\circ u}(v)}_{=\mathrm{N}^+_G(v)} = \overbrace{(\mathrm{N}^-_{G-u}(v) \cap \mathrm{N}^+_G(v))}^{=\mathrm{N}^-_G(v) \cap \mathrm{N}^+_G(v) \text{ as } u \notin \mathrm{N}^+_G(v)} \cup (\mathrm{N}^-_G(u) \cap \mathrm{N}^+_G(v)) \\ &= (\mathrm{N}^-_G(v) \cap \mathrm{N}^+_G(v)) \cup (\mathrm{N}^-_G(u) \cap \mathrm{N}^+_G(v)) = \mathrm{N}^=_G(v) \cup (\mathrm{N}^-_G(u) \cap \mathrm{N}^+_G(v)).\end{aligned}$$

In the case of $u \in \mathrm{N}^+_G(v) \setminus \mathrm{N}^=_G(v)$, a similar reasoning yields the assertion, while the remaining case follows from (i) and (ii). □

The next objective is to establish a notation for the exclusion similar to the notation $G - S$ ($S \subset V$) for a digraph $G = (V, A)$. To justify such a notation, it must be proved that the order in which two vertices $u, v \in V$ are excluded does not matter. This is shown in the next lemma:

Lemma A.3. *Let $G = (V, A)$ be a digraph. Let $u, v \in V$ two distinct vertices.*

(i). If $G \circ v$ is feasible on G, then and only then $G - u \circ v$ is also feasible on $G - u$ and $G - u \circ v = G \circ v - u$ holds.

(ii). If $G \circ u$ is feasible on G and $G \circ u \circ v$ on $G \circ u$, then and only then $G \circ v$ is also feasible on G as well as $G \circ v \circ u$ on $G \circ v$ and we have $G \circ u \circ v = G \circ v \circ u$.

Proof.

(i). The exclusion of v is feasible if and only if v is loop-free in the corresponding digraph. On the other hand, if v is loop-free in G, it is surely loop-free in $G - u$. Conversely, if v is loop-free in $G - u$, it is also loop-free in G since the deletion of u does not destroy loops in v. So, $G \circ v$ is feasible on G if and only if so is $G - u \circ v$ on $G - u$.

Therefore, let us assume v to be loop-free. Then:

$$\begin{aligned} G - u \circ v &= G - u - v + \mathrm{N}^-_{G-u}(v) \times \mathrm{N}^+_{G-u}(v) \\ &= G - v + \mathrm{N}^-_{G-u}(v) \times \mathrm{N}^+_{G-u}(v) - u && \text{Obs. A.1, } {}^{\text{ii}}_{\text{vii}} \\ &= G - v + \mathrm{N}^-_{G}(v) \times \mathrm{N}^+_{G}(v) - u && \text{Obs. A.1, vii} \\ &= G \circ v - u \end{aligned}$$

(ii). $G \circ u$ is feasible on G if and only if u is loop-free in G. On the other hand, $G \circ u \circ v$ is feasible on $G \circ u$ if and only if v is loop-free in $G \circ u$, i.e. v is loop-free and v and u do not form a cycle of length 2 in G. Thus, $G \circ u \circ v$ is feasible if and only if u and v are loop-free and do not form a cycle of length 2. Because this condition is symmetric in u and v, $G \circ u \circ v$ is feasible if and only if so is $G \circ v \circ u$.

Therefore, assuming u and v to be loop-free, three cases have to be considered.

The simplest case is (u, v), $(v, u) \notin A$, i.e. $u \notin \mathrm{N}_G(v)$ and $v \notin \mathrm{N}_G(u)$. It follows:

$$\begin{aligned} G \circ u \circ v &= G - u + \mathrm{N}^-_G(u) \times \mathrm{N}^+_G(u) - v + \mathrm{N}^-_{G \circ u}(v) \times \mathrm{N}^+_{G \circ u}(v) \\ &= G - v + \mathrm{N}^-_{G \circ u}(v) \times \mathrm{N}^+_{G \circ u}(v) - u + \mathrm{N}^-_G(u) \times \mathrm{N}^+_G(u) && \text{Obs. A.1, } {}^{\text{iii}}_{\text{vii}} \\ &= G - v + \mathrm{N}^-_G(v) \times \mathrm{N}^+_G(v) - u + \mathrm{N}^-_{G \circ v}(u) \times \mathrm{N}^+_{G \circ v}(u) && \text{Obs. A.2, i} \\ &= G \circ v - u + \mathrm{N}^-_{G \circ v}(u) \times \mathrm{N}^+_{G \circ v}(u) \\ &= G \circ v \circ u \end{aligned}$$

Next, suppose $(u, v) \in A$ and $(v, u) \notin A$, i.e. $u \notin \mathrm{N}^+_G(v)$ or equivalently

$v \notin \mathrm{N}^-_G(u)$. Then:

$$\begin{aligned}
G \circ u \circ v &= G - u + \mathrm{N}^-_G(u) \times \mathrm{N}^+_G(u) - v + \mathrm{N}^-_{G\circ u}(v) \times \mathrm{N}^+_{G\circ u}(v) \\
&= G - u + \mathrm{N}^-_G(u) \times \mathrm{N}^+_{G-v}(u) - v + \mathrm{N}^-_{G\circ u}(v) \times \mathrm{N}^+_{G\circ u}(v) && \text{Obs. A.1, vii} \\
&= G - u + \mathrm{N}^-_{G\circ v}(u) \times \mathrm{N}^+_{G-v}(u) - v + \mathrm{N}^-_{G\circ u}(v) \times \mathrm{N}^+_G(v) && \text{Obs. A.2, i} \\
&= G - v + \mathrm{N}^-_{G\circ u}(v) \times \mathrm{N}^+_G(v) - u + \mathrm{N}^-_{G\circ v}(u) \times \mathrm{N}^+_{G-v}(u) && \text{Obs. A.1, } \substack{\text{ii}\\\text{iii}\\\text{vii}} \\
&= G - v + \big(\mathrm{N}^-_{G-u}(v) \cup \mathrm{N}^-_G(u)\big) \times \mathrm{N}^+_G(v) \\
&\qquad - u + \mathrm{N}^-_{G\circ v}(u) \times \mathrm{N}^+_{G-v}(u) && \text{Obs. A.2, ii} \\
&= G - v + \mathrm{N}^-_{G-u}(v) \times \mathrm{N}^+_G(v) + \mathrm{N}^-_G(u) \times \mathrm{N}^+_G(v) \\
&\qquad - u + \mathrm{N}^-_{G\circ v}(u) \times \mathrm{N}^+_{G-v}(u) \\
&= G - v + \mathrm{N}^-_{G-u}(v) \times \mathrm{N}^+_G(v) + \mathrm{N}^-_{G\circ v}(u) \times \mathrm{N}^+_G(v) \\
&\qquad - u + \mathrm{N}^-_{G\circ v}(u) \times \mathrm{N}^+_{G-v}(u) && \text{Obs. A.2, i} \\
&= G - v + \mathrm{N}^-_G(v) \times \mathrm{N}^+_G(v) \\
&\qquad - u + \mathrm{N}^-_{G\circ v}(u) \times \mathrm{N}^+_{G-v}(u) + \mathrm{N}^-_G(u) \times \mathrm{N}^+_G(v) && \text{Obs. A.1, } \substack{\text{iii}\\\text{vii}} \\
&= G \circ v - u + \mathrm{N}^-_{G\circ v}(u) \times \mathrm{N}^+_{G-v}(u) + \mathrm{N}^-_{G\circ v}(u) \times \mathrm{N}^+_G(v) \\
&= G \circ v - u + \mathrm{N}^-_{G\circ v}(u) \times \big(\mathrm{N}^+_{G-v}(u) \cup \mathrm{N}^+_G(v)\big) \\
&= G \circ v - u + \mathrm{N}^-_{G\circ v}(u) \times \mathrm{N}^+_{G\circ v}(u) && \text{Obs. A.2, ii} \\
&= G \circ v \circ u
\end{aligned}$$

The final case $(v, u) \in A$ and $(u, v) \notin A$ is proven similarly and omitted for readability reasons.

□

Because of Lemma A.3 we are entitled to introduce the notation

$$G \circ E := G \circ v_1 \circ \cdots \circ v_k$$

for a vertex set $E = \{v_1, \ldots, v_k\} \subset V$. Moreover, we can say that the operation $G \circ E$ is feasible on G if $G \circ \cdots \circ v_i$ is feasible on $G \circ \cdots \circ v_{i-1}$ for all $i = 1, \ldots, k$. From Lemma A.3 it follows that this definition does not depend on the order of the v_i, so it is well defined.

Naturally, the question arises: Under which conditions is $G \circ E$ feasible on G? This is answered by the following lemma:

Lemma A.4. *Let $G = (V, A)$ be a digraph and let $E \subset V$. Then, $G \circ E$ is feasible on G if and only if $G[E]$ is acyclic.*

Proof. First, suppose $E = V$ which is to say $G[E] = G$. Assume further that $G \circ E$ is feasible on G. Then, by Proposition 1.4 it follows that $F := V(G \circ E)$ is a FVS of G. As $E = V$ we have $F = \emptyset$, thus G is acyclic. Now, assume conversely that G is acyclic. Hence, there must be a vertex $v_1 \in V$ having outdegree 0. From $\mathrm{d}^+_G(v_1) = 0$ it follows that v_1 is loop-free, hence $G \circ v_1$ is feasible on G. Moreover, $G \circ v_1 = G - v_1$ follows from Observation A.1, ix and we see that $G \circ v_1$ is acyclic, too. This again implies that there must be a vertex $v_2 \in V(G \circ v_1)$ with $\mathrm{d}^+_{G \circ v_1}(v_2) = 0$. As before, we conclude that $G \circ v_1 \circ v_2$ is feasible on $G \circ v_1$ and that $G \circ v_1 \circ v_2$ is acyclic. Inductively, it follows that $G \circ \cdots \circ v_k$ is feasible on $G \circ \cdots \circ v_{k-1}$ for all $k = 1, \ldots, |V|$, i.e. that $G \circ V$ is feasible on G.

Now, suppose $E \neq V$ and let $F := V \setminus E$. First, assume $G[E]$ is acyclic. This means F is a FVS of G. Then, from the first part of the proof we see that $G - F \circ E$ is feasible on $G - F$. By multiple application of Lemma A.3 it follows that $G \circ E$ is feasible on G. Now, assume $G \circ E$ is feasible on G. Proposition 1.4 tells us that F is a FVS of G, i.e. $G - F = G[E]$ is acyclic. □

Corollary A.5. *Let $G = (V, A)$ be a digraph. Then, $F \subset V$ is a FVS of G if and only if $G \circ (V \setminus F)$ is feasible on G.*

Proof. Suppose F is a FVS of G, meaning that $G - F = G[V \setminus F]$ is acyclic. Then, by Lemma A.4 $G \circ (V \setminus F)$ is feasible on G.

Now, suppose $G \circ (V \setminus F)$ is feasible on G. From this feasibility it follows by Lemma A.4 that $G[V \setminus F] = G - F$ is acyclic which means that F is a FVS of G. □

A byproduct of the considerations so far is the following characterisation of redundant vertices. As one is interested in determining as small as possible FVSs, an efficient recognition of redundant vertices is crucial. So, this characterisation is of importance.

Corollary A.6. *Let $G = (V, A)$ be a digraph, let $F \subset V$ be a FVS of G. A vertex $v \in F$ is redundant if and only if v is loop-free in $G \circ (V \setminus F)$.*

Proof. Let $F' := F \setminus \{v\}$ and set $E := V \setminus F$ and $E' := V \setminus F' = E \cup \{v\}$. Suppose v is redundant, i.e. F' is also a FVS of G. Then, according to Corollary A.5, $G \circ E' = G \circ E \circ v$ is feasible on G. Particularly, $G \circ E \circ v$ is feasible on $G \circ E$ which implies that v is loop-free on $G \circ E$.

Now, suppose v is not redundant. Since F is a FVS of G, $G \circ E$ is feasible G. But because F' is not a FVS of G, $G \circ E'$ is not feasible on G, i.e. $G \circ E \circ v$ is not feasible on $G \circ E$. Both of these assertions hold due to Corollary A.5. The infeasibility of $G \circ E \circ v$ on $G \circ E$ implies that v has a loop in $G \circ E$ according to Lemma A.4. □

A.2 The diclique reductions

Now, we are in the position to state four new reductions and to prove their correctness. Subsequently, they will be loosely called **diclique reductions** and include the following:

INDICLIQUE(v) If v is loop-free and $G[\mathrm{N}^-_G(v)]$ is a diclique, G is transformed into $G \circ v$.

OUTDICLIQUE(v) If v is loop-free and $G[\mathrm{N}^+_G(v)]$ is a diclique, G is transformed into $G \circ v$.

DICLIQUE-2(v) If v is loop-free and $\mathrm{N}_G(v) = N_1 \dot\cup N_2$ such that $G[N_1]$ and $G[N_2]$ are dicliques with $\mathrm{N}^{=}_G(v) \subset N_1$, G is transformed into $G \circ v$.

DICLIQUE-3(v) If v is loop-free, $\mathrm{d}^{=}_G(v) = 0$ and $\mathrm{N}_G(v) = N_1 \dot\cup N_2 \dot\cup N_3$ such that $G[N_1]$, $G[N_2]$ and $G[N_3]$ are dicliques, G is transformed into $G \circ v$.

To prove that the diclique reductions do not impair the optimality concerning the minimum feedback vertex set problem, the following simple proposition is needed.

Proposition A.7. *Any FVS of a diclique $G = (V, A)$ consists of at least $|V| - 1$ vertices.*

Proof. Suppose the assertion is not true which is to say there is a FVS F of G with $|F| \leq |V| - 2$. Then, there are vertices $u, v \in V(G - F)$ such that $(u, v), (v, u) \in A(G - F)$ ($\Leftrightarrow (u, v, u) \in \mathcal{C}_{G-F}$). This contradicts the fact that F is a FVS of G. □

Lemma A.8. *If the digraph $G = (V, A)$ is transformed into $G \circ v$ ($v \in V$) by any of the four diclique reductions, then any optimal FVS of $G \circ v$ is also an optimal FVS of G regarding the minimum feedback vertex set problem.*

Proof. In view of Proposition 1.4 the assertion of the lemma can be reformulated in the following way: If the premises of one of the diclique reductions are satisfied for a vertex $v \in V$, there is an optimal FVS $\widetilde{F}$ of G with $v \notin \widetilde{F}$. So, assume $F \subset V$ is an optimal FVS with $v \in F$ and assume further that the premises of one of the diclique reductions are satisfied. We want to construct another optimal FVS $\widetilde{F}$ such that $v \notin \widetilde{F}$. To do so, set $F' := F \setminus \{v\}$ and $G' := G - F'$. The set F' is not a FVS of G because of the minimality of F, so G' is cyclic and $\{v\}$ is an optimal FVS of G'. The objective is to show that there is another (optimal) FVS $\{u\}$ of G'. In that case, $\widetilde{F} := F' \cup \{u\}$ is another optimal FVS of G with $v \notin \widetilde{F}$ as asserted.

Four cases have to be considered:

(i). INDICLIQUE(v) is applicable: As $G[\mathrm{N}^-_G(v)]$ is a diclique, from Proposition A.7 it can be concluded that

$$\mathrm{d}^-_{G'}(v) = |\,\mathrm{N}^-_{G'}(v)| = |\,\mathrm{N}^-_G(v) \setminus F'| \leq 1.$$

But $\mathrm{d}^-_{G'}(v) = 0$ is impossible because it would imply that v is redundant, contradicting the optimality of F. So, $\mathrm{d}^-_{G'}(v) = 1$ and v is loop-free in G', hence the Levy-Low reduction IN1(v) is applicable in G'. This means, there is another FVS of G' consisting of one vertex – for instance $\mathrm{N}^-_{G'}(v)$. Then, as explained, $\widetilde{F} := F' \cup \mathrm{N}^-_{G'}(v)$ is another optimal FVS of G such that $v \notin \widetilde{F}$.

(ii). OUTDICLIQUE(v) is applicable: As in the case of INDICLIQUE(v), we may conclude $\mathrm{d}^+_{G'}(v) = 1$. In addition, v is loop-free in G', so OUT1(v) is applicable in G'. Similarly to the previous case, this implies that $\mathrm{N}^+_{G'}(v)$ is a FVS of G' consisting of one vertex. So, $\widetilde{F} := F' \cup \mathrm{N}^+_{G'}(v)$ is an optimal FVS of G with $v \notin \widetilde{F}$.

(iii). DICLIQUE-2(v) is applicable: As $G[N_i]$ is a diclique, we have $|N_i \cap F'| \geq |N_i| - 1$ in view of Proposition A.7 for $i = 1, 2$. It follows that

$$|\,\mathrm{N}_{G'}(v)| = |\underbrace{\mathrm{N}_G(v)}_{= N_1 \dot\cup N_2} \setminus F'| = |N_1 \setminus F'| + |N_2 \setminus F'| \leq 2.$$

And because $\mathrm{N}_G^{=}(v) \subset N_1$ it also follows $\mathrm{d}_{G'}^{=}(v) \leq 1$ from which we deduce $\mathrm{d}_{G'}(v) \leq 3$. Like in either of the two previous cases, the possibilities $\mathrm{d}_{G'}^{-}(v) = 0$ and $\mathrm{d}_{G'}^{+}(v) = 0$ can be ruled out. Hence, we have $\mathrm{d}_{G'}^{-}(v) = 1$ or $\mathrm{d}_{G'}^{-}(v) = 1$. This means that either IN1(v) or OUT1(v) is applicable in G' due to the fact that v is loop-free in G'. The applicability of either of the two reductions implies that there is a FVS $\{u\}$ of G' other than $\{v\}$. So, $\widetilde{F} := F' \cup \{u\}$ is an optimal FVS of G.

(iv). DICLIQUE-3(v) is applicable: Similarly to the case DICLIQUE-2(v), we conclude from Proposition A.7 that

$$\underbrace{\mathrm{d}_{G'}^{+}(v) = }_{\mathrm{d}_{G'}^{=}(v)=0}|\,\mathrm{N}_{G'}(v)| = |\underbrace{\mathrm{N}_G(v)}_{= N_1 \dot\cup N_2 \dot\cup N_3} \setminus F'| = |N_1 \setminus F'| + |N_2 \setminus F'| + |N_3 \setminus F'| \leq 3.$$

Once again, we see that $\mathrm{d}_{G'}^{-}(v) > 0$ and $\mathrm{d}_{G'}^{+}(v) > 0$. Hence, $\mathrm{d}_{G'}^{-}(v) = 1$ or $\mathrm{d}_{G'}^{-}(v) = 1$ follows. So, either IN1(v) or OUT1(v) can be applied in G' because v is loop-free in G'. This implies that there is a FVS $\{u\}$ of G' other than $\{v\}$. So, $\widetilde{F} := F' \cup \{u\}$ is also an optimal FVS of G.

□

While it is straight forward to test whether a vertex v meets the conditions of the INDICLIQUE and OUTDICLIQUE reduction in a digraph $G = (V, A)$, this is not true for the other two diclique reductions. To see this, consider the following undirected graph induced by G:

$$\overline{G} := (V, E_G) \quad \text{with} \quad E_A := \{\{u, v\} \subset V : (u, v) \notin A \vee (v, u) \notin A\}.$$

With this definition it can be seen that $\mathrm{N}_{G'}(v)$ is covered by two dicliques if and only if $\overline{G}[\mathrm{N}_{G'}(v)]$ is bipartite which can be tested in linear time with breadth first search. Thus, the test for the applicability of DICLIQUE-2(v) takes time $O(|V|^2)$. In contrast to that, we enter a new complexity level with the DICLIQUE-3 reduction. It turns out that DICLIQUE-3(v) (with $\mathrm{d}_G^{=}(v) = 0$) is applicable if and only if $\overline{G}[\mathrm{N}_{G'}(v)]$ tripartite or equivalently 3-colorable. But the problem to decide the 3-colorability of an undirected graph is known to be NP-complete [28]. So, the DICLIQUE-3 reduction is only of theoretical interest.

By recalling that the empty digraph and the digraph consisting of only one vertex represent dicliques of size 0 and 1, respectively, it can be realized that INDICLIQUE(v) and OUTDICLIQUE(v) subsum all the Levy-Low reductions but LOOP(v). Furthermore, they also include the CORE(v) reduction (cf. section 2.1.1.1) as a special case in the following sense: If $v \in V$ is a core of a diclique, the CORE(v) reduction instructs us to add $\mathrm{N}_G(v)$ to the (partial) FVS. On the other hand, given v is a core of a diclique it follows that $\mathrm{N}_G(v)$ is a diclique, thus for instance INDICLIQUE(v) is applicable. Now, INDICLIQUE(v)

advises us to exclude v from the digraph G. But as $G[v \cup \mathrm{N}_G(v)]$ is a diclique, we see that every vertex $u \in \mathrm{N}_G(v)$ has a loop in $G \circ v$. Hence, each vertex $u \in \mathrm{N}_G(v)$ is then added to the (partial) FVS of G by the LOOP(u) reduction.

A.3 The finite Church-Rosser property for the DICLIQUE-1 reductions

For the Levy-Low reductions it has been proven that any order of application of the reductions always leads to the same digraph [36]. Since INDICLIQUE, OUTDICLIQUE and LOOP generalise the Levy-Low reductions this serves as a motivation to prove the same for these three reductions which will be referred to as **DICLIQUE-1 reductions**. But before being able to prove that property, additional notation is required which is adopted from [36] and [52].

A.3.1 Notation

We start with the notion of a **replacement system** being a triple $(S, \longrightarrow, \equiv)$, where $\longrightarrow \subset S \times S$ is a relation and $\equiv \subset S \times S$ an equivalence relation on the set S, respectively. We will write $x \longrightarrow y$ (resp. $x \equiv y$) rather than $(x, y) \in \longrightarrow$ (resp. $(x, y) \in \equiv$).

The **reflexive closure** $\xrightarrow{\#}$ of $\longrightarrow$ is the relation

$$\xrightarrow{\#} := \{(x, x) : x \in S\} \cup \{(x, y) \in S \times S : x \longrightarrow y\}.$$

Furthermore, by $\xrightarrow{*}$ the **reflexive transitive closure** of $\longrightarrow$ is denoted which is to say

$$\xrightarrow{*} := \{(x, y) \in S \times S : x \xrightarrow{\#} \cdots \xrightarrow{\#} y\}.$$

If the system $(S, \longrightarrow, \equiv)$ is finite (see Definition A.9), the **completion** $\xrightarrow{\triangle}$ of $\longrightarrow$ is can be defined as

$$\xrightarrow{\triangle} := \{x \xrightarrow{*} y : y \xrightarrow{*} z \Longrightarrow y = z\}.$$

With the above notation we can define what it means for a replacement system $(S, \longrightarrow, \equiv)$ to be finite and finite Church-Rosser, respectively.

Definition A.9. *A replacement system* $(S, \longrightarrow, \equiv)$ *is*

(i). finite if for each $x \in S$ *there is a constant integer* $k_x \in \mathbb{N}$ *such that* $x \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_k \longrightarrow y$ *implies* $k \leq k_x$.

(ii). finite Church-Rosser (FCR) if it is finite and from $x_1 \xrightarrow{\triangle} y_1$ *and* $x_2 \xrightarrow{\triangle} y_2$ *with* $x_1 \equiv x_2$ *it follows* $y_1 \equiv y_2$.

A.3.2 The proof of the Church-Rosser property

To show that a replacement system $(S, \longrightarrow, \equiv)$ is FCR, the following characterisation will be used which has been proven in [1].

Theorem A.10. *The replacement system $(S, \longrightarrow, \equiv)$ is FCR if and only if it is finite, and for any $x_1, x_2 \in S$ with $x_1 \equiv x_2$, if $x_1 \longrightarrow y_1$ and $x_2 \xrightarrow{\#} y_2$, then there are some $z_1, z_2 \in S$ with $z_1 \equiv z_2$ such that $y_1 \xrightarrow{*} z_1$ and $y_2 \xrightarrow{*} z_2$.*

Now, consider the replacement system $(\mathfrak{D}, \longrightarrow, \cong)$, where $\mathfrak{D}$ is the set of all digraphs. For two digraphs $G = (V, A)$ and $H = (V', A')$ in $\mathfrak{D}$ let $G \longrightarrow H$ if and only if G can be transformed into H by applying one of the DICLIQUE-1 reductions. Furthermore, $G \cong H$ holds if and only if there is a bijection

$$\sigma : V \to V'$$

such that

$$(u, v) \in A \iff (\sigma(u), \sigma(v)) \in A'.$$

In this case σ is called a **digraph isomorphism**. It will be written $\sigma(G)$ for $(\sigma(V), \{(\sigma(u), \sigma(v)) : (u, v) \in A\})$. Moreover, if $V' \subset V$, we will loosely write $\sigma(H)$ instead of $\sigma_{|V'}(H)$.

The aim is to prove that $(\mathfrak{D}, \longrightarrow, \cong)$ is FCR using Theorem A.10. Fortunately, the following corollary shows that for this particular replacement system, it suffices show some weaker conditions than those in Theorem A.10.

Corollary A.11. *The replacement system $(\mathfrak{D}, \longrightarrow, \cong)$ is FCR if $G \longrightarrow G'_1$ and $G \longrightarrow G'_2$ for any $G, G'_1, G'_2 \in \mathfrak{D}$ imply the existence of some digraphs $H_1, H_2 \in \mathfrak{D}$ with $H_1 \cong H_2$ such that $G'_1 \xrightarrow{*} H_1$ and $G'_2 \xrightarrow{*} H_2$.*

Proof. We show that the conditions of Theorem A.10 are satisfied.

The finiteness of $(\mathfrak{D}, \longrightarrow, \cong)$ is clear as the application of any DICLIQUE-1 reduction decreases the number of vertices of the digraph and the considered digraphs are finite.

For the remainder of the proof assume

$$G_1 \cong G_2 \quad \text{and} \quad G_1 \longrightarrow G'_1 \tag{A.1}$$

and let

$$\sigma : V(G_1) \to V(G_2)$$

be the digraph isomorphism with $\sigma(G_1) = G_2$.

Next, we show that $G_2 \longrightarrow G'_2$ in conjunction with (A.1) implies that there are two digraphs $H_1, H_2 \in \mathfrak{D}$ with $H_1 \cong H_2$ and $G'_1 \xrightarrow{*} H_1$ as well as $G'_2 \xrightarrow{*} H_2$. Obviously, from $G_1 \longrightarrow G'_1$ it follows $G_2 = \sigma(G_1) \longrightarrow \sigma(G'_1)$. So, by the assumptions of the corollary, there are isomorphic digraphs $H_1, H_2 \in \mathfrak{D}$ with $\sigma(G'_1) \xrightarrow{*} H_1$ and $G'_2 \xrightarrow{*} H_2$ from which $G'_1 \xrightarrow{*} \sigma^{-1}(H_1)$ follows. In addition, since $H_1 \cong H_2$, we also have $\sigma^{-1}(H_1) \cong H_2$.

Finally, we show that (A.1) implies the existence of isomorphic digraphs $H_1, H_2 \in \mathfrak{D}$ with $G'_1 \xrightarrow{*} H_1$ and $G_2 \xrightarrow{*} H_2$. This is easily seen as from $G_1 \longrightarrow G'_1$ we conclude $G_2 = \sigma(G_1) \longrightarrow \sigma(G'_1)$ and the previous part of the proof shows the existence of the digraphs $H_1, H_2 \in \mathfrak{D}$ with the desired properties.

Thus, the premises of Theorem A.10 are satisfied and it follows that $(\mathfrak{D}, \longrightarrow, \cong)$ is FCR. □

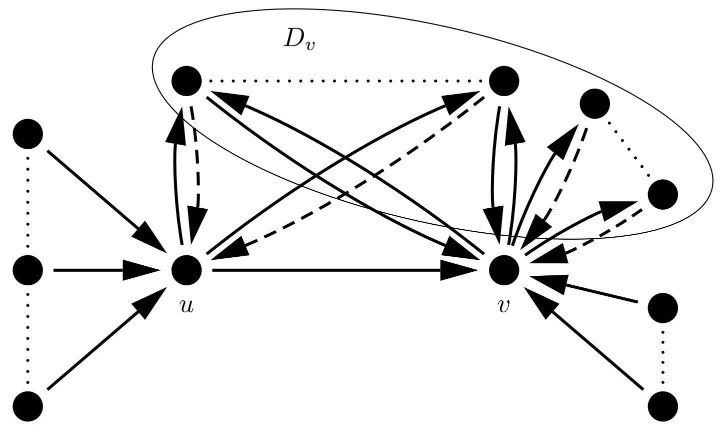

Figure A.1: The case DR_u = OUTDICLIQUE and DR_v = OUTDICLIQUE. The dashed arcs indicate arcs that possibly, but not necessarily, join two vertices.

So, Corollary A.11 will be used in order to prove that $(\mathfrak{D}, \longrightarrow, \cong)$ is FCR. To show that $(\mathfrak{D}, \longrightarrow, \cong)$ satisfies the conditions of Corollary A.11, the following three auxiliary lemmas are needed.

Lemma A.12. *Let $G, G' \in \mathfrak{D}$ be digraphs such that $G \longrightarrow G'$. Let further $G \longrightarrow G - u$ for a vertex $u \in V(G')$. Then, $G - u \longrightarrow G' - u$ and $G' \longrightarrow G' - u$.*

Proof. Because $G \longrightarrow G - u$ we know that u has a loop in G, i.e. LOOP(u) is applicable in G. If G' is of the form $G' = G - v$ ($v \in V(G)$), the same follows for v. This means that u and v also have loops in $G - v = G'$ and $G - u$, respectively. That is to say $G' \longrightarrow G' - u$ and $G - u \longrightarrow G - u - v = G' - u$.

Now, suppose G' is of the form $G' = G \circ v$ which means that G' results of an application of a DICLIQUE-1 reduction in G. Let $D_v \subset G$ be the corresponding diclique. Because D_v is an induced diclique and u has a loop in G, we have $u \notin V(D_v)$ and thus D_v is also a diclique in $G - u$. It follows that the same DICLIQUE-1 reduction is applicable in $G - u$ on v yielding $G - u \longrightarrow G - u \circ v = G' - u$. On the other hand, u having a loop in G also has one in G', so LOOP(u) is applicable in G' and $G' \longrightarrow G' - u$ follows. $\square$

Lemma A.13. *For the replacement system $(\mathfrak{D}, \longrightarrow, \cong)$ let $G \longrightarrow G \circ u$. Let further $G \longrightarrow G \circ v$ for a vertex $v \in V(G) \setminus \mathrm{N}_G^{=}(u)$. Then, there are digraphs $H_1, H_2 \in \mathfrak{D}$ with $H_1 \cong H_2$ such that $G \circ u \xrightarrow{*} H_1$ and $G \circ v \xrightarrow{*} H_2$.*

Proof. As $G \longrightarrow G \circ u$, there must be a diclique reduction

$$\mathrm{DR}_u \in \{\text{INDICLIQUE}, \text{OUTDICLIQUE}\}$$

such that $\mathrm{DR}_u(u)$ is applicable in G, hence let $D_u \subset G$ be the corresponding diclique. The same holds for v, so DR_v and $D_v \subset G$ are defined similarly.

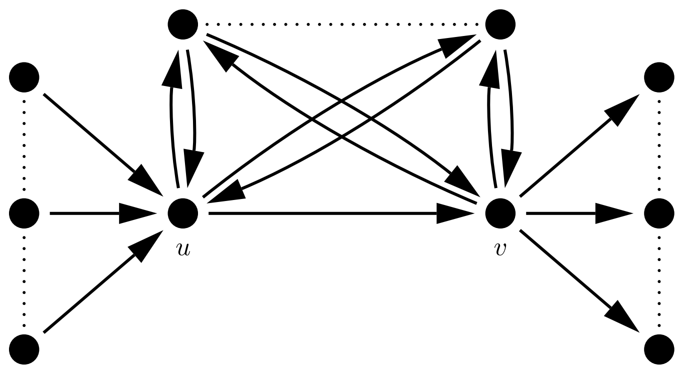

Figure A.2: The case $\mathrm{DR}_u = \mathrm{OUTDICLIQUE}$ and $\mathrm{DR}_v = \mathrm{INDICLIQUE}$.

First, consider the case $u \notin V(D_v)$ and $v \notin V(D_u)$. In this context, let

$$L_u := \mathrm{N}_G^{=}(u) \cap V(D_v) \quad \text{and} \quad L_v := \mathrm{N}_G^{=}(v) \cap V(D_u).$$

Then, each vertex $w \in L_u$ has a loop in $G \circ u$, so $\mathrm{LOOP}(w)$ is applicable and one gets

$$G \circ u \xrightarrow{*} \underbrace{G \circ u - L_u}_{:=G'}.$$

Because $v \notin L_u \subset \mathrm{N}_G^{=}(u)$ we see that $v \in V(G')$ as well as v is loop-free in G'. Moreover, $D'_v := D_v - L_u$ is an induced diclique in G'. Hence, $\mathrm{DR}_v(v)$ can also be applied in G' and we have

$$G \circ u \xrightarrow{*} G' \longrightarrow G' \circ v.$$

Now, each vertex $w \in L_v \cap V(G' \circ v)$ has a loop in $G' \circ v$ and thus

$$G \circ u \xrightarrow{*} G' \longrightarrow G' \circ v \xrightarrow{*} G' \circ v - L_v = \underbrace{G \circ u \circ v - L_u \cup L_v}_{=:H}.$$

Finally, since u and v have symmetric roles and H is symmetric in u and v, we also have $G \circ v \xrightarrow{*} H$.

It remains the case $u \in V(D_v)$ or $v \in V(D_u)$ which implies $u \in \mathrm{N}_G(v)$. Hence, in the remainder of the proof w.l.o.g. it will be assumed $(u, v) \in A(G)$ and $(v, u) \notin A(G)$. There are some subcases to be considered:

(i). $\mathrm{DR}_u = \mathrm{INDICLIQUE}$ and $\mathrm{DR}_v = \mathrm{OUTDICLIQUE}$. As $u \notin \mathrm{N}_G^{=}(v)$, we have $u \notin V(D_v)$ and $v \notin V(D_u)$, so this subcase has been treated in the first part of the proof.

(ii). $\mathrm{DR}_u = \mathrm{OUTDICLIQUE}$ and $\mathrm{DR}_v = \mathrm{OUTDICLIQUE}$. It follows that $D_u = G[\mathrm{N}_G^{+}(u)]$ as well as $D_v = G[\mathrm{N}_G^{+}(v)]$. Since $v \in \mathrm{N}_G^{+}(u)$ it follows $\mathrm{N}_G^{+}(u) \subseteq \mathrm{N}_G^{=}(v) \cup \{v\}$. On the other hand, we have $\mathrm{N}_G^{=}(v) \subset V(D_v)$, thus

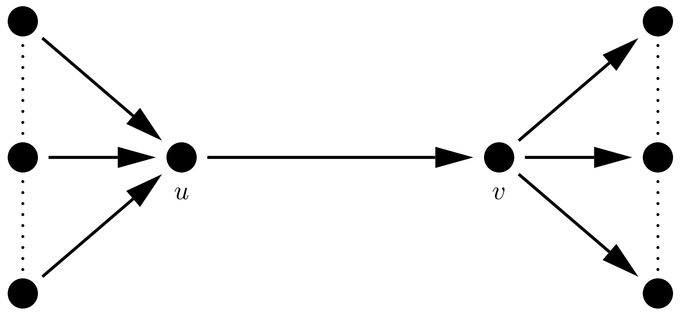

Figure A.3: The digraph from Figure A.2 after the deletion of the vertices $\mathrm{N}^{=}_{G}(u) = \mathrm{N}^{=}_{G}(v)$.

the scenario is the one shown in Figure A.1.

After the exclusion of u from G, the vertices $w \in \mathrm{N}^{=}_{G}(u)$ have loops in $G \circ u$ as can be concluded from Figure A.1. So, LOOP(w) can be applied and one gets

$$G \circ u \xrightarrow{*} \underbrace{G \circ u - \mathrm{N}^{=}_{G}(u)}_{=:G'}.$$

In G' the subdigraph $D'_v := D_v - \mathrm{N}^{=}_{G}(u)$ is an induced diclique and moreover $\mathrm{N}^{+}_{G'}(v) = V(D'_v)$. Thus, OUTDICLIQUE$(v)$ is applicable in G' and

$$G \circ u \xrightarrow{*} G' \longrightarrow G' \circ v$$

holds. Furthermore, in $G' \circ v$ any vertex $w \in \mathrm{N}^{=}_{G'}(v)$ has a loop, so LOOP(w) is applicable in $G' \circ v$ and

$$G \circ u \xrightarrow{*} G' \longrightarrow G' \circ v \xrightarrow{*} G' \circ v - \mathrm{N}^{=}_{G}(v) = \underbrace{G \circ u \circ v - \mathrm{N}^{=}_{G}(u) \cup \mathrm{N}^{=}_{G'}(v)}_{=:H}$$

follows. Using Observation A.2, iii, a 'simpler' expression for H is obtained:

$$\begin{aligned} H &= G \circ u \circ v - \mathrm{N}^{=}_{G}(u) \cup \mathrm{N}^{=}_{G'}(v) \\ &= G \circ u \circ v - \mathrm{N}^{=}_{G}(u) \cup \underbrace{\mathrm{N}^{=}_{G \circ u - \mathrm{N}^{=}_{G}(u)}(v)}_{=\mathrm{N}^{=}_{G \circ u}(v) \setminus \mathrm{N}^{=}_{G}(u)} \\ &= G \circ u \circ v - \mathrm{N}^{=}_{G}(u) \cup (\mathrm{N}^{=}_{G \circ u}(v) \setminus \mathrm{N}^{=}_{G}(u)) \\ &= G \circ u \circ v - \mathrm{N}^{=}_{G}(u) \cup (\mathrm{N}^{=}_{G}(v) \cup (\mathrm{N}^{-}_{G}(u) \cap \mathrm{N}^{+}_{G}(v)) \setminus \mathrm{N}^{=}_{G}(u)) \quad \text{Obs. A.2, iii} \\ &= G \circ u \circ v - \mathrm{N}^{=}_{G}(u) \cup \mathrm{N}^{=}_{G}(v) \cup (\mathrm{N}^{-}_{G}(u) \cap \mathrm{N}^{+}_{G}(v)) \end{aligned}$$

On the other hand, also in $G \circ v$ each $w \in \mathrm{N}_G^{=}(v)$ has a loop yielding

$$G \circ v \xrightarrow{*} \underbrace{G \circ v - \mathrm{N}_G^{=}(v)}_{=:G''}.$$

Now, the digraph $D'_v := D_v - \mathrm{N}_G^{=}(v)$ is an induced diclique in G'' and moreover $\mathrm{N}_{G''}^{+}(u) = V(D'_v)$ (cf. Observation A.2, ii). Thus, OUTDICLIQUE(u) is applicable in G'' and we obtain

$$G \circ v \xrightarrow{*} G'' \longrightarrow G'' \circ u.$$

In $G'' \circ u$ the vertices in $\mathrm{N}_{G''}^{=}(u)$ have loops which yields

$$G \circ v \xrightarrow{*} G'' \longrightarrow G'' \circ u \xrightarrow{*} G'' \circ u - \mathrm{N}_{G''}^{=}(u) = \underbrace{G \circ u \circ v - \mathrm{N}_G^{=}(v) \cup \mathrm{N}_{G''}^{=}(u)}_{=H \text{ due to Obs. A.2, iii}}.$$

(iii). DR_u = INDICLIQUE and DR_v = INDICLIQUE. This case is analogue to the previous one, so for the sake of readability the proof is omitted.

(iv). DR_u = OUTDICLIQUE and DR_v = INDICLIQUE. It follows that $D_u = G[\mathrm{N}_G^{+}(u)]$. As $v \in \mathrm{N}_G^{+}(u)$ this implies $\mathrm{N}_G^{+}(u) \subseteq \mathrm{N}_G^{=}(v) \cup \{v\}$. On the other hand, from $D_v = G[\mathrm{N}_G^{-}(v)]$ being an induced diclique and $u \in \mathrm{N}_G^{-}(v)$ it follows $\mathrm{N}_G^{-}(v) \subseteq \mathrm{N}_G^{=}(u) \cup \{u\}$. Combining these two inclusions leads to $\mathrm{N}_G^{=}(u) = \mathrm{N}_G^{=}(v)$. This is depicted in Figure A.2.

In $G \circ u$ each $w \in \mathrm{N}_G^{=}(u)$ has a loop which implies

$$G \circ u \xrightarrow{*} \underbrace{G \circ u - \mathrm{N}_G^{=}(u)}_{=:H_1}.$$

Similarly, each $w \in \mathrm{N}_G^{=}(v)$ has a loop in $G \circ v$ which is to say

$$G \circ v \xrightarrow{*} \underbrace{G \circ v - \overbrace{\mathrm{N}_G^{=}(v)}^{=\mathrm{N}_G^{=}(u)}}_{=:H_2}.$$

We claim that the two digraphs

$$H_1 = \underbrace{G - \mathrm{N}_G^{=}(u)}_{=:G'} \circ u \quad \text{and} \quad H_2 = G - \mathrm{N}_G^{=}(u) \circ v$$

are isomorphic, i.e. $H_1 \cong H_2$. This becomes clear when regarding Figure A.3 which depicts the digraph G'. We see that $\mathrm{N}_{G'}^{+}(u) = \{v\}$ and $\mathrm{N}_{G'}^{-}(v) = \{u\}$. So, $H_1 = G' \circ u$ and $H_2 = G' \circ v$ are obviously isomorphic (cf. Observation A.2, ii).

$\square$

Lemma A.14. *Let $G \in \mathfrak{D}$ be a digraph and let $G \longrightarrow G \circ v_1$. Let further $G \longrightarrow G \circ v_2$ for a vertex $v_2 \in \mathrm{N}_G^{=}(v_1)$. Then, there is a digraph $H \in \mathfrak{D}$ such that $G \circ v_1 \xrightarrow{*} H$ and $G \circ v_2 \xrightarrow{*} H$.*

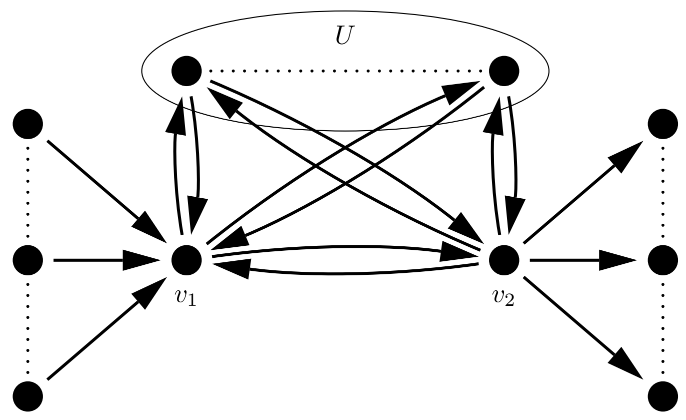

Figure A.4: The situation when $G \twoheadrightarrow G \circ v_1$ and $G \twoheadrightarrow G \circ v_2$ with $v_2 \in \mathrm{N}^{=}_G(v_1)$.

Proof. Let $U := \mathrm{N}^{=}_G(v_1) \cap \mathrm{N}^{=}_G(v_2)$. Note that $\mathrm{N}^{=}_G(v_i)$ $(i = 1, 2)$ induces a diclique because either INDICLIQUE(v_i) or OUTDICLIQUE(v_i) is applicable in G, so obviously, D induces a diclique, too.

We claim that there is no path (w_1, v_i, w_2) in G for $i = 1, 2$ such that $w_1, w_2 \notin U \cup \{v_1, v_2\}$. To see this, note that $G[\mathrm{N}^{-}_G(v_i) \cup \{w_1\}]$ or $G[\mathrm{N}^{+}_G(v_i) \cup \{w_2\}]$ is a diclique because INDICLIQUE(v_i) or OUTDICLIQUE(v_i) is applicable in G. Thus, say $G[\mathrm{N}^{=}_G(v_i) \cup \{w_j\}]$ is a diclique for $j \in \{1, 2\}$. As $v_{3-i} \in \mathrm{N}^{=}_G(v_i)$, one sees $w_j \in \mathrm{N}^{=}_G(v_{3-i})$. But as $v_i \in \mathrm{N}^{=}_G(v_{3-i})$ and $\mathrm{N}^{=}_G(v_{3-i})$ induce a diclique, too, one also sees $w_j \in \mathrm{N}^{=}_G(v_i)$, i.e. $w_j \in \mathrm{N}^{=}_G(v_1) \cap \mathrm{N}^{=}_G(v_2) = U$.

From the claim we conclude

$$\mathrm{N}^{=}_G(v_1) = U \cup \{v_2\} \quad \text{and} \quad \mathrm{N}^{=}_G(v_2) = U \cup \{v_1\}.$$

The complete situation is depicted in Figure A.4.

After the exclusion of v_1 each vertex $w \in \mathrm{N}^{=}_G(v_1)$ has a loop in $G \circ v_1$. So, LOOP(w) can be applied in $G \circ v_1$. Thus, one obtains

$$G \twoheadrightarrow G \circ v_1 \overset{*}{\twoheadrightarrow} G \circ v_1 - \mathrm{N}^{=}_G(v_1) = \underbrace{G - U}_{=:G'} - v_2 \circ v_1.$$

Similarly, after the exclusion of v_2 each vertex in $\mathrm{N}^{=}_G(v_2)$ has a loop in $G \circ v_2$ and one gets

$$G \twoheadrightarrow G \circ v_2 \overset{*}{\twoheadrightarrow} G \circ v_2 - \mathrm{N}^{=}_G(v_2) = G' - v_1 \circ v_2.$$

Now, we claim $G' - v_2 \circ v_1 = G' - v_1 \circ v_2 = G' - v_1 - v_2 =: H$. To confirm this claim, consider the digraph $G' - v_2$. As argued above, in this digraph we have either $\mathrm{d}^{-}_{G'-v_2}(v_1) = 0$ or $\mathrm{d}^{+}_{G'-v_2}(v_1) = 0$. The same holds for v_2 in $G' - v_1$. So, Observation A.1, ix may be applied and the claim follows. Hence, we get

$$G \circ v_1 \overset{*}{\twoheadrightarrow} G \circ v_1 - \mathrm{N}^{=}_G(v_1) = H \quad \text{and} \quad G \circ v_2 \overset{*}{\twoheadrightarrow} G \circ v_2 - \mathrm{N}^{=}_G(v_2) = H.$$

□

Once these three lemmas have been proved, we are able to show the main result of this appendix.

Theorem A.15. $(\mathfrak{D}, \longrightarrow, \cong)$ *is FCR.*

Proof. Let $G \in \mathfrak{D}$ and let further $G_1' := G \bigtriangleup_1 u$ and $G_2' G \bigtriangleup_2 v$ such that $G \longrightarrow G_1'$ and $G \longrightarrow G_2'$, where $\bigtriangleup_1, \bigtriangleup_2 \in \{-, \circ\}$. According to Corollary A.11 it has to be proven that there are digraphs $H_1, H_2 \in \mathfrak{D}$ with $H_1 \cong H_2$ such that $G_1' \xrightarrow{*} H_1$ and $G_2' \xrightarrow{*} H_2$.

The existence of such digraphs H_1, H_2 follows from Lemma A.12 in the case of $\bigtriangleup_1 = -$ or $\bigtriangleup_2 = -$ while in the case of $\bigtriangleup_1 = \circ = \bigtriangleup_2$ it follows from the Lemmas A.13 and A.14. □

A.4 Concluding remarks

Thanks to Theorem A.15 we are entitled to define a new class of digraphs for which the minimum feedback vertex set problem can be solved in polynomial time. This class represents the digraphs which can be transformed into an acyclic digraph (and hence into an empty digraph) by successive application of the DICLIQUE-1 reductions. This class is referred to as the class of **DICLIQUE-1 reducible graphs**. The polynomial algorithm for determining an optimal FVS of these digraphs simply applies the DICLIQUE-1 reductions in arbitrary order until an acyclic digraph is reached. From Theorem A.15 it follows that if there is a particular application order of DICLIQUE-1 reductions that transforms the initial digraph into an acyclic one, then any admissible order will do it.

Concerning the runtime of the algorithm for determining an optimal FVS for a DICLIQUE-1 reducible graph $G = (V, A)$, the dominant operation is the test whether the INDICLIQUE(v) or OUTDICLIQUE(v) reduction can be applied on a vertex $v \in V$. This test consists of checking if a particular vertex set induces a diclique and can be done in time $O(|V|^2)$. If one of the two reductions is applicable, then v has to be excluded from the digraph which takes time $O(|V|^2)$. Since during the execution of the algorithm at most $O(|V|^2)$ such tests and $O(|V|)$ exclusions have to be performed, a total runtime of $O(|V|^4)$ is obtained.

The relatively large time bound of $O(|V|^4)$ of the exact algorithm for the DICLIQUE-1 reducible graphs, when compared with the runtime of $O(|A| \log |V|)$ for the completely contractible graphs which are defined by means of the Levy-Low reductions (cf. section 2.1.2), might suggest that the DICLIQUE-1 reductions are not suited for the use within an approximation algorithm. However, this is not the case for two reasons. Firstly, the time bound of $O(|V|^4)$ is only a theoretical one and a more involved analysis should yield a much better bound. Secondly, the DICLIQUE-1 reductions are more powerful than the Levy-Low reductions. While the latter ones can clearly be applied in sparse digraphs only, the DICLIQUE-1 reductions are also applicable in dense digraphs. By way of an example, consider a diclique D with at least $k \geq 3$ vertices. Apparently, D is not completely contractible, whereas it is indeed DICLIQUE-1 reducible.

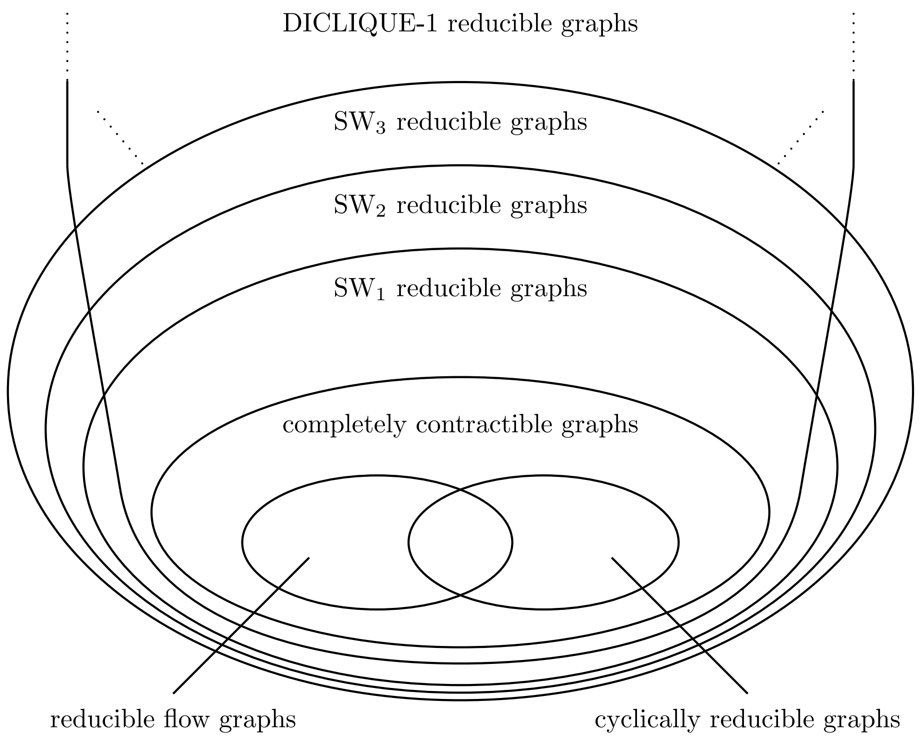

Figure A.5: The DICLIQUE-1 reducible graphs versus other polynomial classes.

This trivial example shows that the DICLIQUE-1 reducible graphs strictly encompass the completely contractible graphs. Moreover, the example shows that the class of DICLIQUE-1 reducible graphs contains digraphs which are SW$_{k-1}$ reducible but not SW$_{k-2}$ reducible (cf. section 2.1.2). Figure A.5 illustrates the inclusion the relation of DICLIQUE-1 reducible graphs to other polynomial classes.

The attempt to also prove the FCR property for all four diclique reductions together with the LOOP reduction is deemed to fail in view of the digraph from Figure A.6. There, only DICLIQUE-3(u) and INDICLIQUE(v) is applicable. After applying DICLIQUE-3(u) the digraph is not reducible anymore by any of the five reductions as can be seen in Figure A.7. Likewise, after applying INDICLIQUE(v) in the digraph of Figure A.6 the resulting digraph is not further reducible (cf. Figure A.8). As the two irreducible digraphs are non-isomorphic, this shows that the diclique reductions together with the LOOP reduction do not possess the FCR property. Figure A.6 is an example in which the DICLIQUE-3 reduction causes problems with respect to the FCR property. Similarly, there are also examples for the DICLIQUE-2 reduction.

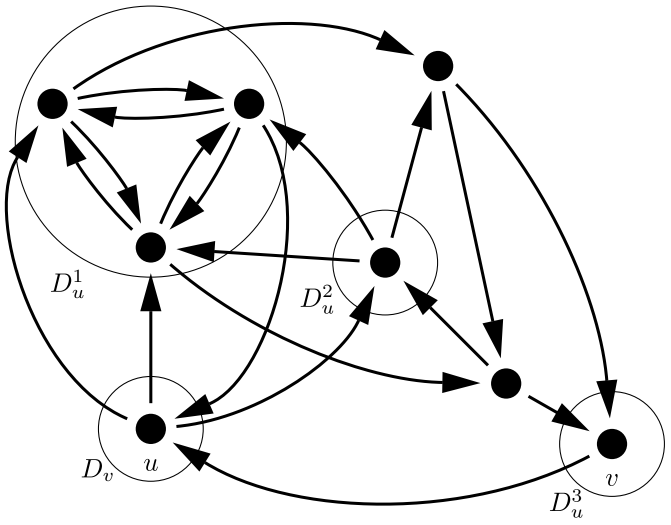

Figure A.6: In the present digraph only DICLIQUE-3(u) and INDICLIQUE(v) are applicable with the corresponding induced dicliques D_u^1, D_u^2, D_u^3 and D_v, respectively.

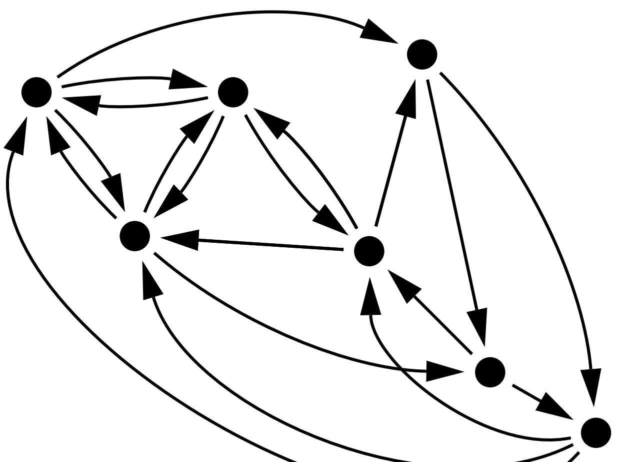

Figure A.7: The digraph from Figure A.6 after the application of DICLIQUE-3(u). The digraph is not further reducible by any of the diclique reductions nor by the LOOP reduction.

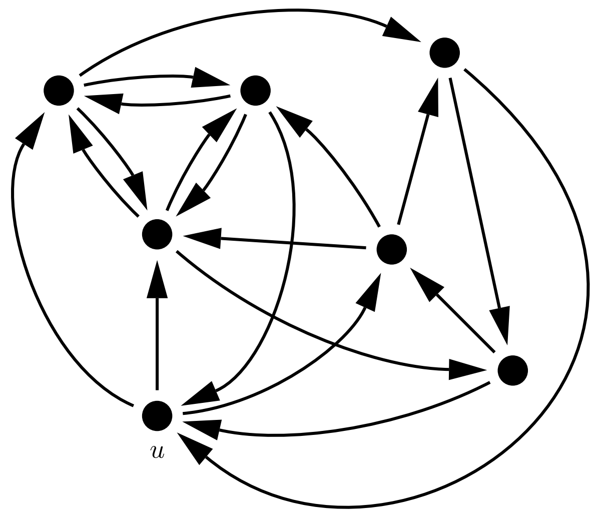

Figure A.8: The digraph from Figure A.6 after applying INDICLIQUE(v). The digraph is not further reducible by any of the diclique reductions nor by the LOOP reduction.

Bibliography

[1] A. V. Aho, R. Sethi, J. D. Ullman. *Code optimization and finite Church-Rosser theorems*, Design and Optimization of Compilers (R. Rustin, Ed.), Prentice-Hall, Englewood Cliffs, N.J., pages 89–105, 1972

[2] N. Alon. *Ranking tournaments*, SIAM Journal on Discrete Mathematics, Vol. 20, No. 1, pages 137-142, 2006

[3] F. Brglez, D. Bryan, K. Kozminski. *Combinational profiles of sequential benchmark circuits*, Proceedings of the IEEE International Symposium on Circuits and Systems, pages 1929–1934, 1989

[4] V. Bafna, P. Berman, T. Fujito *Constant ratio approximations of the weighted feedback vertex set problem for undirected graphs*, ISAAC95, Algorithms and Computation, J. Staples, P. Eades, N. Katoh and A. Moffat Eds., Lecture Notes in Computer Science Vol. 1004, pages 142–151, Springer-Verlag, 1995

[5] R. Bar-Yehuda, S. Even. *A local-ratio theorem for approximating the weighted vertex cover problem*, Annals of Discrete Mathematics 25, pages 27–46, 1985

[6] A. Becker, D. Geiger. *Approximation algorithms for the loop cutset problem*, Proc. of the 10th conference on Uncertainty in Artificial Intelligence, pages 60–68, 1994

[7] E. Boros, K. M. Elbassioni, V. Gurvich, L. Khachiyan. *Enumerating Minimal Dicuts and Strongly Connected Subgraphs and Related Geometric Problems*, 10th International Conference Integer Programming and Combinatorial Optimization (IPCO 2004), Lecture Notes in Computer Science (LNCS) 3064, pages 152–162, 2004

[8] E. Boros, K. Borys, V. Gurvich, G. Rudolf. *Generating* 3*-vertex connected spanning subgraphs*, to appear in Deiscrete Mathematics (DM 14311), 2007

[9] S. Brin, L. Page, R. Motwami, T. Winograd. *The PageRank citation ranking: bringing order to the Web*, Technical Report 1999-0120, Computer Science Department, Stanford University, 1999

[10] M. Cai, X. Deng, W. Zang. *An Approximation Algorithm for Feedback Vertex Sets in Tournaments*, SIAM Journal on Computing, Vol. 30, No. 6, pages 1993–2007, 2001

[11] S. T. Chakradhar, A. Balakrishnan, V. D. Agrawal. *An Exact Algorithm for Selecting Partial Scan Flip-Flops*, Proceedings of the Design Automation Conference, pages 81–86, 1994

[12] P. Charbit, S. Thomassé, A. Yeo. *The minimum feedback arc set problem is NP-hard for tournaments*, Combinatorics, Probability and Computing, Vol. 16, No. 1, pages 1–4, 2006

[13] D. Coppersmith, S. Winograd. *Matrix multiplication via arithmetic progressions*, Journal of Symbolic Computation, Vol. 9, No. 3, pages 251–280, 1990

[14] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. *Introduction to Algorithms*, Second Edition, The MIT Press and McGraw-Hill Book Company 2001

[15] C. Demetrescu, I. Finocchi. 2003. *Combinatorial algorithms for feedback problems in directed graphs*, Information Processing Letters, Vol. 86, No. 3, pages 129–136, 2003

[16] C. Demetrescu, G. F. Italiano. *Fully dynamic transitive closure: Breaking through the $O(n^2)$ barrier*, Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science (FOCS'00), pages 381–389, 2000

[17] R. Diestel. *Graph theory*, Springer-Verlag, New York, 2000

[18] J. Dongarra, A. Lumsdaine, R. Pozo, K. Remington. *A sparse matrix library in C++ for high performance architectures*, Proceedings of the Second Object Oriented Numerics Conference, pages 214–218, 1994

[19] G. Even, J. Naor, B. Schieber, M. Sudan. *Approximating minimum feedback sets and multicuts in directed graphs*, Algorithmica, Vol. 20, No. 2, pages 151–174, 1998

[20] F. Fages, A. Lal. *A Constraint Programming Approach to Cutset Problems*, Computers and Operations Research, Vol. 33, No. 10, pages 2852–2865, 2006

[21] T. A. Feo, M. G. C. Resende. *A probabilistic heuristic for a computationally difficult set covering problem*, Operations Research Letters, Vol. 8, pages 67-71, 1989

[22] P. Festa, P. M. Pardalos, M. G. C. Resende. *Algorithm 815: Fortran subroutines for computing approximate solutions of feedback set problems using GRASP*, ACM Transactions on Mathematical Software, Vol. 27, pages 456–464, 2001

[23] M. X. Goemans, D. P. Williamson. *Primal-Dual Approximation Algorithms for Feedback Problems in Planar Graphs*, Combinatorica, Vol. 18, pages 37–59, 1998

[24] M. X. Goemans, D. P. Williamson. *The primal-dual method for approximation algorithms and its application to network design problems*, D. S. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, pages 144–191. PWS, Boston, 1997

[25] B. Hasselman. *An efficient method for detecting redundant feedback vertices*, CPB Netherlands Bureau for Economic Policy Analysis, Discussion Paper 29, 2004

[26] M. S. Hecht, J. D. Ullman. *Characterization of Reducible Flow Graphs*, Journal of the ACM, Vol. 21, No. 3, pages 167–175, 1974

[27] M. Jünger. *Polyhedral Combinatorics and the Acyclic Subdigraph Problem*, Research and Exposition in Mathematics 7, Heldermann Verlag, Berlin, 1985

[28] R. M. Karp. *Reducibility among combinatorial problems*, R. Miller, J. Thatcher, editors, *Complexity of Computer Communications*, pages 85–103. Plenum Press, 1972

[29] R. M. Karp. *The Transitive Closure of a Random Digraph*, Random Structures and Algorithms, Vol. 1, No. 1, pages 73–93, 1990

[30] H. Koehler. *A contraction algorithm for finding minimal feedback sets*, Proceedings of the Twenty-Eighth Australasian Conference on Computer Science - Vol. 38 (Newcastle, Australia), Australian Computer Society, pages 165–173, 2005

[31] M. Laguna, R. Martí. *GRASP and path relinking for 2-layer straight line crossing minimization*, INFOMRS Journal on Computing, Vol. 11, pages 44–52, 1998

[32] A. N. Langville, C. D. Meyer. *Updating Markov chains with an eye on Google's PageRank*, SIAM Journal on Matrix Analysis and Applications, Vol. 27, No. 4, pages 968–987, 2006

[33] A. N. Langville, C. D. Meyer. *A Survey of Eigenvector Methods for Web Information Retrieval*, SIAM Review, Vol. 47, No. 1, pages 135–161, 2005

[34] E. L. Lawler. *A comment on Minimum Feedback Arc Sets*, IEEE Transactions on Circuit Theory, Vol. 11, No. 2, pages 296-297, 1964

[35] D. Lee, M. Reddy. *On determining scan flip-flops in partial-scan designs*, Proceedings of the International Conference on Computer-Aided Design. pages 322–325, 1990

[36] H. Levy, D. W. Low. *A contraction algorithm for finding small cycle cutsets*, Journal Of Algorithms, Vol. 9, pages 470–493, 1988

[37] Hen-Ming Lin, Jing-Yang Jou. *Computing Minimum Feedback Vertex Sets by Contraction Operations and its Applications on CAD*, IEEE International Conference on Computer Design, pages 364–369, 1999

[38] E. L. Lloyd, M. L. Soffa, C.-C. Wang. *On locating minimum feedback vertex sets*, Journal of Computer and System Sciences, Vol. 37, pages 292–311, 1988

[39] I. Marek, D. B. Szyld. *Local convergence of the (exact and inexact) iterative aggregation method for linear systems and Markov operators*, Numerische Mathematik, Vol. 69, No. 1, pages 61–82, 1994

[40] S. L. Martins, C. C. Ribeiro. *A parallel GRASP for the Steiner problem in graphs*, In Proceedings of the Irregular 98, 1998

[41] M. Matsumoto, T. Nishimura. *Mersenne Twister: A 623-dimensionally equidistributed uniform pseudo-random number generator*, ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, pages 3–30, 1998

[42] K. Mehlhorn, S. Näher. *LEDA: A Platform for Combinatorial and Geometric Computing*, Cambridge University Press, 1999

[43] C. D. Meyer. *Matrix Analysis and Applied Linear Algebra*, SIAM, Philadelphia, 2000

[44] T. Orensten, Z. Kohavi, I. Pomeranz. *An optimal algorithm for cycle breaking in directed graphs*, Journal of Electronic Testing, Vol. 7, No. 1, pages 71–82, 1995

[45] P. M. Pardalos, T. Qian, M. G. C. Resende. *A greedy randomized adaptive search procedure for the feedback vertex set problem*, Journal of Combinatorial Optimization, Vol. 2, pages 399–412, 1999

[46] S. Park, S. Akers. *A graph theoretic approach to partial scan design by k-cycle elimination*, The Proceedings of the International Test Conference, pages 303–311, 1992

[47] J. Perl. *Fusion, propagation and structuring in belief networks*, Artificial Intelligence, Vol. 29, No. 3, pages 241–288, 1986

[48] M. G. C. Resende, C. C. Ribeiro. *Greedy randomized adaptive search procedures*, F. Glover, G. A. Kochenberger, editors, *Handbook of Metaheuristics*, Kluwer Academic Publishers, pages 219–249, 2003

[49] S. J. Russel, P. Norvig. *Artificial Intelligence: A Modern Approach*, Prentice Hall, 1995

[50] B. Schwikowski. *Empirische Analyse von Approximationsalgorithmen für das Feedback-Vertex-Set-Problem in Digraphen*, Diploma thesis, Heinrich-Heine-Universität Düsseldorf, 1994

[51] B. Schwikowski, E. Speckenmeyer. *On computing all minimal solutions for feedback problems*, Discrete Applied Mathematics 117, pages 253–265, 2002

[52] R. Sethi. *Testing for the Church-Rosser property*, Journal of the ACM, Vol. 21, No. 4, pages 671–679, 1974.

[53] P. D. Seymour. *Packing directed circuits fractionally*, Combinatorica, Vol. 15, pages 281–288, 1995

[54] A. Shamir. *A linear time algorithm for finding minimum cutsets in reduced graphs*, SIAM Journal on Computing, Vol. 8, No. 4, pages 645–655, 1979

[55] W. Smith, R. Walford. *The identification of a minimal feedback vertex set of a directed graph*, IEEE Transactions on Circuits and Systems, Vol. 22, No. 1, pages 9–15, 1975

[56] E. Speckenmeyer. *On Feedback Problems in Digraphs*, Graph-Theoretic Concepts in Computer Science, Springer-Verlag, Berlin, Heidelberg, pages 218–231, 1989

[57] W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains*, Princeton University Press, pages 355–358, 1994

[58] S. Tai, D. Bhattacharya. *A three-stage partial scan design method to ease ATPG*, Journal of Electronic Testing: Theory and Applications, Vol. 7, No. 1-2, pages 95–104, 1995

[59] R. E. Tarjan. *Depth-first search and linear graph algorithms*, SIAM Journal on Computing, Vol. 1, No. 2, pages 146–160, 1972

[60] C.-C. Wang, E. L. Lloyd, M. L. Soffa. *Feedback vertex sets and cyclically reducible graphs*, Journal of the ACM, Vol. 32, No. 2, pages 296–313, 1985

[61] A. Xie, P. A. Beerel. *Accelerating Markovian analysis of asynchronous systems using string-based state compression*, Proceedings fourth International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC), IEEE Computer Society Press, pages 247–260, 1998

Acknowledgements

First of all, I would like to thank my supervisor Prof. Dr. Ewald Speckenmeyer. He believed in me and gave me the opportunity to work on a very interesting research field. His guidance and encouragement as well as his constructive criticism and comments are highly appreciated. Furthermore, he has — together with Heinz Heimes from the 'Behindertenvertretung' of the University of Cologne — struggled to overcome bureaucratic hurdles connected with the financing of my employment. Without Prof. Dr. Speckenmeyer the present thesis could not have been come about.

Special thanks go to my office mate Bert Randerath with whom I hit it off so well in the more than three years. He was always open to discussions of all kinds. The atmosphere was very friendly and I can tell you we had much fun.

Many thanks also go to Gabriele Eslamipour, the secretary of the *Lehrstuhl Speckenmeyer*. She assisted me with administrative issues and helped me several times not to drop a clanger. Many thanks also to all other present and former members of the *Lehrstuhl Speckenmeyer* who supported me in all aspects of the everday business, and thus made it very pleasent. I cannot imagine better colleagues.

Special thanks go to Michael Belling who assisted me with the literature research. His affable manner contributed to the pleasent atmosphere of the institute.

I am particularly grateful to David Lewes-Malandrakis for promptly, yet accurately, reviewing parts of this thesis. His feedback helped to correct some clumsy formulations and mistakes, and thus markedly improved the overall readability. The attentive reader will surely notice which chapters have been revised by him.

Finally, I would like to thank all the people in my private life. My mother Vojka Lemaić and my brother Dušan Lemaić have supported me as much as they could during the time of writing this thesis. There are so many other people without which it would have been so much harder to complete this work. I cannot name one person without forgetting ten others. I am very aware of their contributions and owe them many thanks.

Erklärung

Ich versichere, dass ich die von mir vorgelegte Dissertation selbständig angefertigt, die benutzten Quellen und Hilfsmittel vollständig angegeben und die Stellen der Arbeit — einschließlich Tabellen, Karten und Abbildungen —, die anderen Werken im Wortlaut oder dem Sinn nach entnommen sind, in jedem Einzelfall als Entlehnung kenntlich gemacht habe; dass diese Dissertation noch keiner anderen Fakultät oder Universität zur Prüfung vorgelegen hat; dass sie — abgesehen von unten angegebenen Teilpublikationen — noch nicht veröffentlicht worden ist sowie, dass ich eine solche Veröffentlichung vor Abschluss des Promotionsverfahrens nicht vornehmen werde. Die Bestimmungen der Promotionsordnung sind mir bekannt. Die von mir vorgelegte Dissertation ist von Prof. Dr. Ewald Speckenmeyer betreut worden.

Köln, den 21.04.2008