Eötvös Loránd University
Institute of Mathematics

Lydia Mirabel Mendoza Cadena

The Minimum Feasibility Blocker Problem

MSc Thesis

Supervisor: Kristóf Bérczi

Department of Operations Research

Budapest 2020

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr Kristóf Bérczi, who always supported me. I am very grateful for his guidance and insightful feedback during my research, given with patience and humour.

I also wish to thank Prof. István Ágoston, who brought me a lot of help in all moments.

From the bottom of my heart, I thank my beloved husband, Mario, for all his love and spiritual support on the most difficult times.

Finally, to my mother, and Pille and Queta, who always encourage me to follow my dreams and never give up, but always with rational thinking. Thank you for all the comforting talks. To all my family and friends who had been there cheering at me.

Contents

Chapter 1

Introduction

Imagine that we have a linear program that represents a real-life problem and we want to obtain its solution. If we can find the optimal solution in a reasonable time, then we present such a solution. But what would happen if we find that the problem is infeasible? In most cases, it is not enough to say that there is no solution; we have to obtain some "useful" result. One natural question is: what if we delete some inequalities in order to get a feasible system? Thinking about it, one can notice that we would like to delete as few inequalities as possible because this is going to change our original problem. This is the so called Minimum Feasibility Blocker problem.

Formally, the Minimum Feasibility Blocker (MinFB) problem takes as input a system $\mathcal{S}$ of linear inequalities $Ax \leq b$ and asks for a smallest subset $\mathcal{I}$ such that $\mathcal{S} \setminus \mathcal{I}$ is feasible. As $\mathcal{I}$ "blocks" the feasibility of $\mathcal{S}$, we refer to $\mathcal{I}$ as a feasibility blocker; we avoid calling $\mathcal{I}$ a "solution", to avoid confusion with the solution of the linear system $\mathcal{S} \setminus \mathcal{I}$.

Observe that the infeasibility of a linear problem is not necessarily related to the description of the real-life problem: the data that is used in these systems may be subject to inaccuracies and uncertainties, and therefore may lead to systems which are infeasible. Another source of infeasibility may be modelling errors. Infeasibility itself allows for little conclusions; for a large system of millions of inequalities, infeasibility may stem from a very small subset of data.

Jayaram K. Sankaran [1] proved that MinFB is an NP-hard problem; furthermore, as the feasibility of a linear system can be tested in polynomial time (e.g., by the ellipsoid method [2]) the MinFB problem is NP-complete. On account of this, the problem has been studied with different approaches that will be discussed in Chapter 3. Here we are going to use parameterized complexity theory, which is a branch of complexity theory that was developed in the 1990s by Downey and Fellows, according to Jörg Flum and Martin Grohe [3]. In parameterized complexity theory, the problem input of size n is additionally equipped with one or more integer parameters k and one measures the problem complexity in terms of both n and k. The goal is to solve such an instance I by fixed-parameter tractable algorithms (FPT), which run in time $f(k) \cdot poly(|I|)$ for some computable function f. FPT algorithms give us a new perspective of a problem because, as Jörg Flum and Martin Grohe [3] point out, "*measuring complexity only in terms of the input size means ignoring any structural information about the input instances in the resulting complexity theory* ".

Nevertheless, for FPT problems it is not always obvious which parameter is going to work correctly; in this context, Marek Cygan et. al. [4] declare that "*selecting the right parameter(s) for a particular problem is an art*". For the next example, we recall that the $Kenny - width$ of a digraph D is the maximum number of distinct (not necessarily disjoint) directed st-paths in D over all pairs of distinct vertices, and the $DAG - width$ measures how tree-like the directed graph is. Robert Ganian et. al. [5] proved that Hamiltonian Path problem, whose goal is to find a directed path that visits each vertex of a digraph exactly once, has an FPT algorithm if the parameter is the $Kenny - width$, meanwhile the problem is polynomial if the parameter chosen is the $DAG - width$. On the other hand, DFVS-Number problem, which looks in a directed graph D for the minimum cardinality of a subset of vertices S such that $D - S$ is acyclic, is an open problem if the parameter is the $Kenny - width$, but it is polynomial with respect to the $DAG - width$ parameter. For further details and other examples on the choice of parameters, we refer to [5].

For the MinFB problem, arguably the most natural parameter is the minimum size k of a feasibility blocker $\mathcal{I}$. In addition, we work with other parameters like the number of positive and negative entries in the vector b for certain type of systems, and we also work with the pathwidth associated to the graph of the system, which we will introduce in Chapter 2.

This thesis is based on the work of Alexander Göke, Lydia Mirabel Mendoza Cadena and Matthias Mnich [6]. The proofs rely on combinatorial optimization theory and parameterized complexity, consequently we first present the required preliminaries in those areas in Chapter 2. Chapter 3 exhibits some solutions to the MinFB among different areas, namely, heuristics, approximation algorithms, mixed-integer programming, and polyhedral combinatorics. Chapter 4 gives results about the hardness of the problem using the parameter size of a blocker. In Chapter 5, we present two FPT algorithms for instances with difference constraints and b-values in -1 and $+1$, with two parameters each. Both algorithms are parameterized with the size of a blocker, but the first algorithm also includes the number of -1 entries in b as parameter, meanwhile the second algorithm uses the number of $+1$ entries instead. In addition, two straight-forward algorithms obtained from the FPT algorithms are shown. Finally, in Chapter 6 we discuss the complexity of the MinFB with respect to the parameter pathwidth. As corollary, the treewidth is also considered in this chapter. A condensed table of the results can be found in Chapter 7.

Chapter 2

Technical background

Among this thesis, we use notions from graph theory, combinatorial optimization and complexity theory. We introduce the main definitions and results that are going to be used.

2.1 Notation

We list the general notation that is used in this thesis.

- $[n] = \{1, 2, ..., n\}$.
- $\mathbb{Z}_+$ denotes the set of non-negative integers.
- $\mathbb{Q}_+$ denotes the set of non-negative rationals.
- $\mathbb{R}_+$ denotes the set of non-negative reals.
- $|a|$ denotes the absolute value of the number a.
- $|X|$ denotes the number of members of the set X.
- A matrix of *difference constraints* contains exactly one $+1$ and one -1 in each row.
- A^T is the transpose of the $n \times m$ matrix A, with $n, m \geq 1$.
- $[A|B]$ denotes an $n \times m$ matrix that contains the $n \times m'$ matrix A on the left and the $n \times m''$ matrix B on the right, with $n, m', m'' \geq 1$ and $m = m' + m''$.
- $P[s, t]$ is the st-subpath of the path P.
- $P \circ Q$ it the concatenation of the walks P and Q.
- $G - S$ denotes the graph where all vertices (arcs) in S are deleted from G, where S is a subset of vertices (arcs) of G.
- $supp(u)$ is the support of vector u.
- $conv(C)$ is the convex hull of the set of points C.
- (x, k) denotes the instance x that is parameterized by k.
- (P, k) denotes the problem P that is parameterized by k.
- Big O notation and related:
  - $\mathcal{O}(1)$ describes an algorithm that will always execute in the same time regardless of the size of the input data set.
  - $\mathcal{O}(g(n))$ describes an algorithm whose performance will grow bounded above by $g(n)$ asymptotically.
  - $\Omega(g(n))$ describes an algorithm whose performance will is bounded below by $g(n)$ asymptotically.

2.2 Graph Theory and Combinatorial optimization

We use finite and loopless directed graphs. Let G be a digraph, whose vertex set we denote by $V(G)$ and arc set by $A(G)$. If G is a non-directed graph, we denote the edge set by $E(G)$. Let $w : A(G) \to \mathbb{Q}$ be a weight function over the arcs and let P be a walk on the digraph G, then the weight of P is $w(P) = \sum_{e \in P} w(e)$. We call a cycle negative (non-negative, positive) if its weight is negative (non-negative, positive). For $S \subseteq V(G)$ ($S \subseteq A(G)$), $G - S$ denotes the graph G where all vertices (arcs) in S are deleted. Note that the deletion of a vertex implies the deletion of all the incident arcs.

For two walks P, Q where the last vertex of P equals the first vertex of Q, let $P \circ Q$ be the *concatenation* of P and Q, which is the sequence of all vertices in P followed by all vertices in Q except the first.

In the MinFB problem, given a system $Ax \leq b$ and an integer k, with coefficient matrix $A \in \mathbb{Q}^{m \times n}$, and a right-hand side vector $b \in \mathbb{Q}^m$, the goal is to find a set $\mathcal{I} \subseteq [m]$ of size at most k such that $(a_{i,\bullet} \cdot x \leq b_i)_{i \in [m] \setminus \mathcal{I}}$ is feasible for some $x \in \mathbb{Q}^n$. By multiplying rows of A and the corresponding entries of b with (-1) the MinFB problem also covers in a parameter equivalent way the case where some inequalities are of the type $a_{i,\bullet} x \geq b_i$ if there is no sign restriction on the entries of A and b. Furthermore, equations of the form $a_{i,\bullet} \cdot x = b_i$ can be written as two inequalities $a_{i,\bullet} \cdot x \leq b_i$ and $-a_{i,\bullet} \cdot x \leq -b_i$ (equivalent to $a_{i,\bullet} \cdot x \geq b_i$). In a feasible solution x^* violating such an equation, at most one of the above inequalities is violated. Thus, the MinFB problem with equations can be reduced to the formulation of the MinFB as formulated above without changing the parameter. Conversely though, MinFB as presented above in general cannot be expressed by the MinFB problem having only equations. However, if we allow additionally inequalities with only one variable or require all variables to be non-negative, those are representable by adding slack variables and (if necessary) splitting variables into two non-negative parts x_i^+ and x_i^-.

Returning to graph definitions, a linear ordering $\{v_1, v_2, ..., v_n\}$ of the node-set of a digraph is called a *topological ordering* if all the arcs point forward, that is, for each arc (v_i, v_j), $i < j$. A topological ordering is only possible for acyclic digraphs. Observe that the first node in a topological ordering has no incoming arcs, and the last node has no leaving arcs.

We can efficiently find negative cycles using the Moore-Bellman-Ford algorithm which runs in time $\mathcal{O}(nm)$ on a digraph with n vertices and m arcs. This algorithm can be found in the book of Jørgen Bang-Jensen and Gregory Z. Gutin [7, Section 2.3.4]. Here, we will need an algorithm that finds a shortest negative cycle of length at most ℓ, where ℓ is a positive integer. Recall that *shortest* refers to the number of arcs. We provide such an algorithm (see Algorithm 1).

Lemma 2.1. *There is an algorithm that, given a digraph G, arc weights $w : A(G) \to \mathbb{Q}$ and an integer $\ell \in \mathbb{Z}_+$, in time $\mathcal{O}(\ell nm)$ either finds a shortest negative cycle C of length at most ℓ in G, or decides that none exists.*

Proof. We show that Algorithm 1 fulfils the statement of the lemma. The run time follows from

Algorithm 1: DetectShortestNegativeCycle

Input : A digraph G, arc weights $w : A(G) \to \mathbb{Q}$ and $\ell \in \mathbb{Z}_+$.
Output: A shortest negative cycle of length at most ℓ, or `false` if no such cycle exists.

```
1  Set d_0(v,v) = 0 for v ∈ V(G) and d_0(v,w) = ∞ for v,w ∈ V(G), v ≠ w.
2  for i = 1 to ℓ do
3  |  Set d_i(v,w) = d_{i-1}(v,w) for v,w ∈ V(G).
4  |  foreach s ∈ V(G) do
5  |  |  foreach (v,w) ∈ A(G) do
6  |  |  |  if d_i(s,w) > d_{i-1}(s,v) + w((v,w)) then
7  |  |  |  |  Set d_i(s,w) = d_{i-1}(s,v) + w((v,w)) and p_i(s,w) = v.
8  |  |  if d_i(s,s) < 0 then
9  |  |  |  Set x_i = s and x_{j-1} = p_j(s,x_{j+1}) for j = 0,...,i-1.
10 |  |  |  return the cycle induced by {(x_{j-1},x_j) | j = 1,...i}.
11 return false.
```

the fact that the innermost function can be realized in constant time and the loops are repeated ℓmn times. For correctness, we claim that $d_i(s,t)$ is always the weight of a minimum-weight s-t-walk with at most i arcs and $p_i(s,t)$ is the second-to-last vertex on such a walk, or $d_i(s,t) = \infty$ if no such walk exists. We prove the claim by induction on i. For $i = 0$ that is true by initialization. Consider now the set of minimum-weight s-t-walks of length at most i; among all such walks, let P be a shortest one such that the last edge of P is the first in our iteration over $A(G)$ among all such walks. Let (u,t) be this last arc. Now we want to show that $d_i(s,t) = w(P)$ and $p_i(s,t) = u$.

Assume that $d_i(s,t) < w(P)$. Then either (i) already $d_{i-1}(s,t) < w(P)$, or (ii) there is an $x \in V(G)$ with $d_{i-1}(s,x) + w((x,t)) = d_i(s,t)$. In case (i), by the induction hypothesis, there exists an s-t-walk of weight strictly less than $w(P)$ and at most $i-1$ arcs; this contradicts the choice of P as the shortest such walk. In case (ii), by induction hypothesis, $d_{i-1}(s,x)$ corresponds to an s-x-walk Q with weight $w(Q)$ and at most $i-1$ arcs. Consider the s-t-walk $Q' = Q \circ (x,t)$ for which we have

$$w(Q') = w(Q) + w((x,t)) = d_{i-1}(s,x) + w((x,t)) = d_i(s,t) < w(P) \ .$$

As Q' has at most i arcs, this contradicts the choice of P. Thus, we have $d_i(s,t) \geq w(P)$.

If at any point in time $d_i(s,t) > w(P)$, $d_i(s,t)$ would be set to $w(P)$ when considering the term $d_{i-1}(s,u) + w((u,t))$ as $P[s,u]$ is a minimum-weight s-u-walk with at most $i-1$ arcs. Thus, $d_{i-1}(s,u) = w(P[s,u])$ and $d_{i-1}(s,u) + w((u,t)) = w(P)$.

It remains to show that $p_i(s,t) = u$. If already $d_{i-1}(s,t) = w(P)$, the entry of $p_i(s,t)$ is set to $p_{i-1}(s,t)$ in the beginning and not changed afterwards. Therefore, by induction and choice of P as arc-wise shortest possible, we get that $p_i(s,t) = p_{i-1}(s,t) = u$. Otherwise, $p_i(s,t)$ gets replaced by another vertex after it is set to $p_{i-1}(s,t)$. The last time it is changed is when $d_{i-1}(s,v) + w((v,t)) = w(P)$ for an arc (v,t) for the first time. By the choice of the last arcs of P, this is exactly the arc (u,t) and thus $p_i(s,t) = u$.

By the above claim, after the execution of the innermost loop $d_i(s,s)$ equals the minimum weight of an s-s-walk with at most i arcs. If there is a negative cycle of length at most ℓ, the value of $d_\ell(v,v)$ will be negative for all $v \in V(C)$. Therefore, if we return "false", we do so correctly.

Otherwise, we reach the point where $d_i(s,s) < 0$ for the first time. As $d_{i-1}(v,v) \geq 0$ for all $v \in V$ there is no cycle of length at most $i-1$. But $d_i(s,s)$ corresponds to a closed walk O of negative weight of length at most i. This walk contains a cycle C of negative weight and has length at most i (otherwise it could not have negative weight). Thus, $O = C$, and we output C correctly by using the backtracking mechanism of the p_j's. □

The *support* of a vector u is the set of coordinates at which the vector is nonzero. We denote it by $supp(u)$. A set of vectors C is called *convex* if it satisfies: if $x, y \in C$ and $\lambda \in [0,1]$, then $\lambda x + (1-\lambda)y \in C$. The smallest convex set containing C is called *convex hull*, that is, $conv(C) = \{\lambda_1 x_1 + ... + \lambda_t x_t | t \geq 1; x_1, ..., x_t \in C; \lambda_1, ..., \lambda_t \geq 0; \lambda_1 + ... + \lambda_t = 1\}$

2.3 Complexity Theory

Our goal is to find an algorithm that solves the Minimum Feasibility Blocker problem with respect to a parameter, or to show that the problem is not FPT with such parameter. We introduce some useful definitions to prove hardness results.

Some results rely on the complexity class coNP/poly. We proceed by defining two classes that will help us to understand this complexity class. First, the class coNP contains the languages that are complements of languages in NP. It is unknown whether coNP is different from NP. Second, let NP/poly be the set of languages that are decided by polynomial size non-deterministic circuits. Then the complexity class coNP/poly contains languages that are complements of languages that are decided by polynomial size non-deterministic circuits. We refer to Sanjeev Arora and Boaz Barak [8] to further details in complexity classes.

We formally define an FPT problem, so we require the following definition: a *parameterization* of a decision problem is a function that assigns an integer parameter k to each input instance x. Observe that k can also consider a finite set of integers, and the parameter can be defined explicitly in the input or implicitly in the structure of the problem.

Definition 2.1. *A parameterized problem is called* fixed-parameter tractable (FPT) *if there is an $f(k) \cdot |(n,k)|^c$ time algorithm for some constant c, where $f : \mathbb{Z}_+ \to \mathbb{Z}_+$ is a non-decreasing computable function, n is the size of an instance and k the parameter. The complexity class containing all fixed-parameter tractable problems is called FPT.*

We would like to provide tools in such a way that "*if problem A has a certain type of algorithm, then problem B has a certain type of algorithm as well*" as Marek Cygan et. al. [4] mentioned. Regarding FPT problems, we can use three different approaches to show that a problem belongs or not to this class. Proving that a problem belongs to FPT requires the following definition:

Definition 2.2. *A* parameterized reduction *from problem P to problem Q is an algorithm that, given an instance (x,k) of P, outputs an instance (x',k') of Q such that*

1. *(x,k) is a* `yes`*-instance of $P \iff (x',k')$ is a* `yes`*-instance of Q,*
2. *$k' \leq g(k)$ for some computable function g, and*

3. the running time is $f(k) \cdot |x|^{\mathcal{O}(1)}$ for some computable function f.

It can be shown that if there is a parameterized reduction from P to Q and Q is FPT, then P is FPT as well. Thus, it is common to show that a problem belongs to the FPT class by making a parameterized reduction from another problem known to be FPT.

A remark is that not every parameterized reduction is a polynomial-time reduction and if there is a parameterized reduction from (P, k) to (Q, k') such that P is NP-hard, then this does not necessarily imply that Q is NP-hard, where k and k' are the parameters of P and Q respectively.

On the other hand, proving that a problem does not have an FPT algorithm involves the keywords 'W[1]-hardness' and 'polynomial compression'. Each one has a characteristic type of reduction between problems.

Showing that a problem P belongs to the class W[1]-hard provides enough evidence that P is not in the class FPT. The definition of W-hierarchy is associated with the following notions:

- A *Boolean circuit* with n-inputs is an acyclic digraph where the nodes are partitioned into four disjoint sets:
  - Input nodes. Set of n nodes, where each node has indegree 0.
  - Constant nodes. Each node has indegree 0 and must be labelled either by 0 or 1.
  - Gate nodes. Each node is called an AND gate, OR gate or NOT gate, with label $\wedge$, $\vee$ and $\neg$ respectively. The AND gates and OR gates have indegree 2, meanwhile the NOT gates have indegree 1.
  - Output node. Exactly one of the nodes with outdegree 0.
- Weighted Circuit Satisfiability problem (WCS): Given a boolean circuit C, decide if C has a satisfying assignment of weight exactly k, where the weight of an assignment is the number of input gates receiving value 1.
- WCS[$\mathcal{C}$] denotes the restriction of the problem where the input circuit C belongs to $\mathcal{C}$.
- A *small node* is a node with indegree at most 2.
- A *large node* is a node with indegree greater to 2.
- The *weft* of a circuit is the maximum number of large nodes on a path from an input node to the output node.
- The *depth* of a circuit is the maximum length of a path from an input node to the output node.
- $\mathcal{C}_{t,d}$ is the class of circuits with weft at most t and depth at most d.

Finally, the *W-hierarchy* is defined as follows:

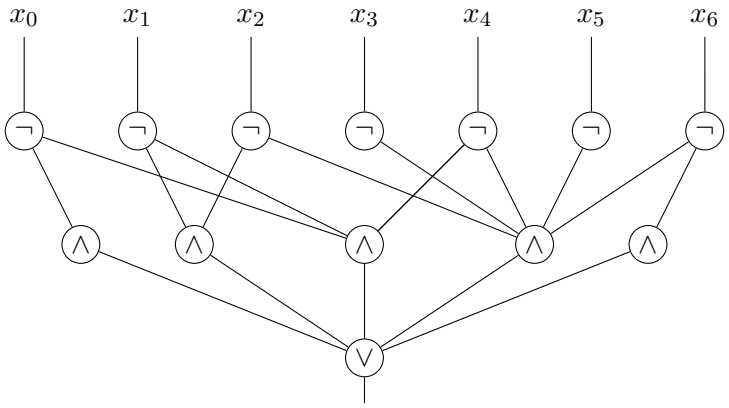

Figure 2.1: *Boolean Circuit with weft 1 and depth 2.*

Definition 2.3. *For $t \geq 1$, a parameterized problem P belongs to* W[t] *if there is a parameterized reduction from P to* WCS$[\mathcal{C}_{t,d}]$ *for some $d \geq 1$.*

We show an example of the previous definition: the boolean circuit from Figure 2.1 belongs to WCS$[\mathcal{C}_{1,3}]$ because any path from an input node to the output node (except the leftmost path) goes through the nodes $x_i \rightarrow$ NOT-gate $\rightarrow$ AND-gate $\rightarrow$ OR-gate, and it has 3 nodes (depth) and only 1 of them has in-degree at least two (weft).

It is known that $FPT \subseteq W[1] \subseteq W[2] \subseteq ...$, and the usual expectation is that the W-hierarchy is a proper hierarchy: W[t] $\neq$ W[t+1] for all $t \geq 1$.

Definition 2.4. *Let $t \in \mathbb{Z}_+$. A parameterized decision problem P is* W[t]-hard *if for every parameterized decision problem Q in $W[t]$, there is a parameterized reduction from Q to P. P is* W[t]-complete *if $P \in W[t]$ and P is $W[t]$-hard.*

Thus, if we have a parameterized problem (P, k) it is enough to give a parameterized reduction from a known W[1]-hard parameterized problem to show that P does not have an FPT algorithm with respect to k. Additionally, there is another concept that "transfers" the hardness from one problem to another: polynomial compressions.

Definition 2.5. *A* polynomial compression *of a parameterized language (P, k) into a language R is an algorithm that takes as input an instance (x, k), works in time polynomial in $|x| + k$, and returns a string y such that*

- *$|y| \leq p(k)$ for some polynomial $p(\cdot)$, and*
- *$y \in R \iff (x, k) \in P$.*

Marek Cygan et. al. [4] explain that " *whenever we talk about the existence of a polynomial compression and we do not specify the target language R, we mean the existence of a polynomial compression into any language R* ".

Many natural parameterized problems do not admit polynomial compressions, under the hypothesis that NP$\subsetneq$ coNP/poly. For this reason, not admitting a polynomial compression is

a complementary result for some FPT algorithms, that is, if we state that an FPT problem does not admit a polynomial compression, then it means that we need the parameter(s) in order to get such FPT algorithm.

We state a useful transformation that conserves the hardness of a language.

Definition 2.6. *Let P, Q be two parameterized problems. An algorithm $\mathcal{A}$ is called a* polynomial parameter transformation (PPT) *from P to Q if, given an instance (x, k) of a problem P, $\mathcal{A}$ works in polynomial time and outputs an equivalent instance $(\hat{x}, \hat{k})$ of problem Q, that is, $(x, k) \in P$ if and only if $(\hat{x}, \hat{k}) \in Q$, such that $\hat{k} \leq p(k)$ for some polynomial $p(\cdot)$. Two parameterized problems are* parameter-equivalent *if there are PPTs in both directions and the transformations additionally fulfil that $\hat{k} = k$.*

Further, a PPT from P to Q together with a polynomial compression for Q yields a polynomial compression for P. This can be used to rule out polynomial compressions:

Proposition 2.2 ([9])**.** *Let P, Q be parameterized problems. If there is a polynomial parameter transformation from P to Q and P admits no polynomial compression, then neither does Q.*

To summarize, suppose that there is a parameterized problem (Q, k') which does not admit a polynomial compression and is W[1]-hard. Then, if we want to show that a parameterized problem (P, k) is not in FPT it is enough to give a PPT from (Q, k') to (P, k).

In the literature, the term 'polynomial compression' is not commonly used. Instead, is more frequent to use 'kernelization'. We first define this term in order to explain the differences between kernelization and polynomial compression.

Let P be a parameterized problem and (x, k) an instance of it. A *kernelization algorithm* or *kernel* for P is an algorithm $\mathcal{A}$ that works in polynomial time and returns an equivalent instance (x', k') of P. It is required that there exists a computable function $g : \mathbb{Z}_+ \to \mathbb{Z}_+$ such that whenever (x', k') is the output for an instance (x, k), then it holds that $|x'| + k' \leq g(k)$. If g is a polynomial (linear) function of the parameter, then we say that P admits a *polynomial (linear) kernel*.

A polynomial kernel is also a polynomial compression: the output of the kernel can be treated as an instance of the unparameterized version of the problem. The main difference between polynomial compression and kernelization is that the polynomial compression is allowed to output an instance of any language R, even an undecidable one. Nevertheless, it is an open question if there exists a problem that admits a polynomial compression but it does not admit a polynomial kernel.

One of the most popular parameters for FPT algorithms is the treewidth, which is a parameter that measures "how tree-like" is a graph. We also define the pathwidth since it is correlated with the treewidth and it is required in another chapter. For an undirected graph G, a *tree decomposition* (*path decompostion*) is a pair $(T, \mathcal{B})$ where T is a tree (path) and $\mathcal{B}$ a collection of bags $B_v \subseteq V(G)$,

each bag corresponding to some node $v \in V(T)$. The bags have the property that for any edge of G both its endpoints appear in some common bag in $\mathcal{B}$, and for each vertex $v \in V(G)$ the bags containing v form a subtree (subpath) of T.

More precisely, in a pathwidth decomposition, the sequence $\mathcal{B} = (B_1, B_2, ..., B_r)$ of bags, $B_i \subseteq V(G)$ for each $i \in [r]$, has the following conditions:

- Every vertex is in at least one bag B_i, for some $i \in [r]$, that is, $\bigcup_{i=1}^{r} B_i = V(G)$.
- For every edge $uv \in E(G)$, there exists at least one bag B_j such that u and v are contained in, for some $j \in [r]$.
- For every vertex $u \in V(G)$, if u is contained in two bags B_i and B_k for some $i \leq k$, that is, $u \in B_i \cap B_k$, then u is also contained in the bag B_j for each index j such that $i \leq j \leq k$. In other words, the indexes of the bags containing u form an interval (path) in $[r]$.

A tree decomposition is a generalization of a path decomposition because the last condition is relaxed to get the form of a tree instead of a path, that is, the following conditions hold for a set of bags $\mathcal{B} = (B_1, B_2, ..., B_r)$ in a tree decomposition:

- Every vertex is in at least one bag B_i, for some $i \in [r]$, that is, $\bigcup_{i=1}^{r} B_i = V(G)$.
- For every edge $uv \in E(G)$, there exists at least one bag B_j such that u and v are contained in, for some $j \in [r]$.
- The nodes in T associated with any vertex of G define a subtree in the nodes of T. Equivalently, $B_i \cap B_k \subseteq B_j$ for every bag j on the path from bag i to bag k in T.

In both decompositions, the *width* of $(T, \mathcal{B})$ is defined as the largest bag size of $\mathcal{B}$ minus one. The *treewidth* (*pathwidth*) of G is the minimum width over all tree (path) decompositions of G. Figure 2.2 shows an example of a tree and a path decomposition for a given graph. The width of both decompositions is 2, therefore the treewidth and the pathwidth are bounded by 2.

Unfortunately, obtaining the treewidth and the pathwidth of a graph is an NP-hard problem, but there are FPT and approximation algorithms for this problem (see Marek Cygan et. al. [4]).

Remark 2.3. *Since path decompositions are a special case of tree decompositions, thus the pathwidth is greater or equal to the treewidth for any graph.*

Furtheremore, as in the example above, a simple observation is that if a tree (path) decomposition is found for a given graph, then the width of such decomposition is a bound for the treewidth (pathwidth) of the graph.

We are ready to proceed to the hardness results and the FPT algorithms.

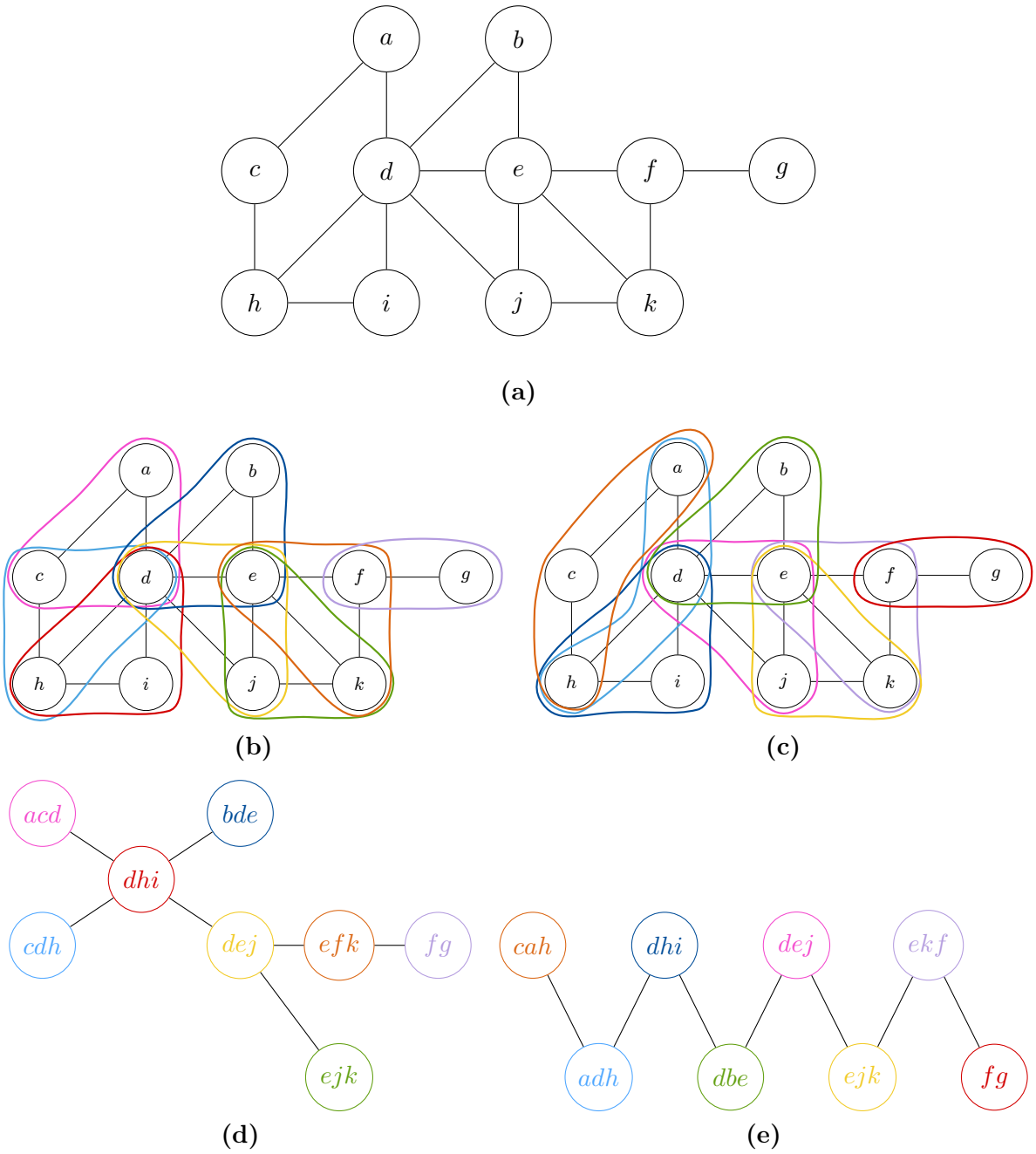

Figure 2.2: ***(a)*** *Original graph.* ***(b)*** *Bags of tree decomposition.* ***(c)*** *Bags of path decomposition.* ***(d)*** *Tree decomposition with width 2.* ***(e)*** *Path decomposition with width 2.*

Chapter 3

Solution approaches

In this section, we describe five different methods used for approximating the blocker set of the MinFB problem. Recall that the goal of the problem is to find the minimum set of inequalities whose removal will lead a feasible system; the input of this problem is the system itself. As an easy observation, there are two natural approaches to find the maximum feasible system. We can either add constraints until we get the maximum number of feasible constraints or delete constraints until we achieve feasibility with as few as possible deletions. In this line, adding inequalities may end in a different subsystem than removing inequalities. Finally, for further details on the methods we refer to the articles cited in each approach.

Heuristics. The heuristic method proposed by John W. Chinneck [10] uses sensitivity filtering and elastic programming in order to get the set of constraints to be eliminated. The sensitivity analysis is a technique that deletes constraints that are not involved in the *irreducible infeasible subsystems* (IIS), where an IIS is an infeasible subsystem of the original for which every proper subsystem is feasible. We call a subset C of the system an *IIS-cover* if it intersects every IIS of the system. The sensitivity filtering works after getting a base from a Phase 1 LP. This base may not be IIS, therefore we eliminate constraints for which the Phase 1 objective is not sensitive. On the other hand, in elastic programming non-negative 'elastic variables' e_i are added deriving in 'elasticized' row constraints. This allows the row constraints to "stretch" against the resistance of an elastic objective function, $\min_i e_i$, which replaces the original objective. This final linear program is called *standard elastic program*. We show the different types of elasticized row constraints in Table 3.1. Observe that the violation of constraints in the original problem can be noticed by the non-zero elastic variables; these elastic variables are also penalized in the objective function. Since a non-zero elastic variable "stretches" the constraint, we refer to stretched constraints instead of violated constraints (the violation occurs in the original).

For simplicity, let us denote by SINF the sum of infeasibilities and by NINF the number of

Table 3.1: *Row constraint conversions for Elastic Programming.*

Nonelastic	Elastic
$\sum_i a_{ij}x_j \geq b_i$	$\sum_i a_{ij}x_j + e_i \geq b_i$
$\sum_i a_{ij}x_j \leq b_i$	$\sum_i a_{ij}x_j + e_i \leq b_i$
$\sum_i a_{ij}x_j = b_i$	$\sum_i a_{ij}x_j + e'_i - e''_i = b_i$

infeasibilities, that is, the sum of the values of the elastic variables and the number of non-zero elastic variables, respectively. Before explaining how the procedure works, we need the following observations:

1. After the Elastic Program is finished, the set of violated constraints is an IIS-cover and the number of violated constraints is an upper bound on the cardinality of an IIS-cover.
2. If there is only one violated constraint, then the cardinality of the IIS-cover is minimum.
3. Removing one constraint of a minimum IIS-cover usually reduces elastic SINF more than removing one constraint of an IIS-cover which is not minimum.
4. Constraints to which the elastic objective function is not sensitive upon termination do not reduce elastic SINF when removed from the model.

Then, the idea of the algorithm is that we solve the standard elastic program derived from the original problem. Using the obtained basis, we can use a greedy algorithm that only considers constraints for which the elastic SINF objective function is sensitive (by Obs. 4), and we select a constraint that gives the minimum value of elastic SINF when it is eliminated from the elastic model (by Obs. 3). By Observations 1 and 2, unless the cardinality is 1, the minimum cardinality of the IIS-cover cannot be guaranteed because the minimum value of elastic SINF may not corresponds to a constraint that belongs to the minimum IIS-cover, but if we minimize over NINF then it may be close to the minimum.

Approximation algorithms. In this field, Edoardo Amaldi and Viggo Kann [11] work with Minimum Unsatisfied Linear Relations Problem (Min ULR), which looks for a solution that violates as few inequalities as possible. We recall the following results:

- **Helly's Theorem**. Let $F = \{K_1, K_2, ..., K_m\}$ be a family of convex sets in $\mathbb{R}^d$. If the intersection of any $d+1$ members of F is non-empty, then the intersection of all members of F is non-empty.
- **Special case of Helly's Theorem**. A finite set $Ax \leq b$ of linear inequalities in d variables has a solution x^* or else there is a subset $A'x \leq b'$ of at most $d+1$ of the inequalities such that $A'x \leq b'$ has no solution.
- **Farkas' Lemma**. The system $Ax \leq b$ is infeasible if and only if there exists vector $y \geq 0$ such that $y^T A = 0$ and $y^T b < 0$.

In their work, they use both Helly's Theorem and Farkas' Lemma to obtain a $(n+1)$-approximation result, where n is the number of variables: their goal is to find a Helly obstruction with at most $(n+1)$ relations, so they look for a vector y with at most $(n+1)$ non-zero entries as in Farkas' Lemma. In order to get y, an auxiliary homogeneous system is proposed, namely $z^T[A|b] = [\underline{0}^T|0]$ where the non-trivial variable z is zero for every component where y is zero. Once y is found, it corresponds to the Helly obstruction. If there are p inequalities in the system, this procedure can be applied at most $\frac{p}{n+1}$ times, and in each iteration we delete at most $(n+1)$ relations while a single one may suffice. This shows that it is a (n+1) approximation algorithm. Additionally,

they also demonstrate that unless P = NP, the MIN ULR problem cannot be approximated in polynomial time within a constant factor.

Mixed-integer programming. The aim of article written by Gianni Codato and Matteo Fischetti [12] is to solve a mixed-integer program (MIP) by identifying a master integer linear problem (master ILP) whose solutions are sent to another linear program which validates and possible returns inequalities to be added to the master ILP. Those inequalities are associated to minimal (or irreducible) infeasible subsystems of a certain linear system and can be separated efficiently when the master solution is integer. The original problem for which the authors work with has the form $\min\{c^T x : Fx \leq g, x \in \{0,1\}^n\}$ with additional conditional constraints. For each inequality i, there is a unique index $j := j(i)$ such that if $x_{j(i)} = 1$ then we add the constraint $a_i^T y \geq b_i$, plus a (possibly empty) set of unconditional constraints $Dy \geq e$. The usual way to solve this problem is to produce the big-M method [1] associated to this problem in order to activate/deactivate some conditional constraints. But instead of this, they used Benders' Decomposition in order to create a work-space with the vector x: a combinatorial Benders' cut $\sum_{i \in C} x_{j(i)} \leq |C| - 1$ is defined, where $C \subseteq I$ induces the minimal (or irreducible) infeasible subsystem (MIS or IIS as in the Heuristic approach). Thus, a Combinatorial Benders' cut is violated by $x^* \in [0,1]^n$ if and only if $\sum_{i \in C} x_{j(i)} > |C| - 1$ or equivalently $1 > \sum_{i \in C}(1 - x_{j(i)})$. The article describes an algorithm that finds the MIS. So, given an infeasible system $\tilde{A}y \geq \tilde{b}$, the MIS problem searchs for an inclusion-minimal set of rows yielding an infeasible system. Consider the system (3.1) with its dual (3.2) as follows

$$\min\{0^T y : \tilde{A}y \geq \tilde{b}\}, \tag{3.1}$$

$$\max\{u^T \tilde{b} : u^T \tilde{A} = 0^T, u \geq 0\}. \tag{3.2}$$

We recall a theorem that can be found in Mokhtar Bazaraa et. al. [13, Section 6.2]. For any primal and dual linear programming problems, exactly one of the following statements is true:

1. Both possess optimal solutions and their optimal value is the same.
2. One problem has an unbounded optimal objective value, in which case the other problem must be infeasible.
3. Both problems are infeasible.

Since $u = 0$ is a feasible solution for (3.2), thus whenever the primal (3.1) is infeasible then the dual is unbounded. In other words, if the primal is infeasible, then there is a vector u^* so that ${u^*}^T \tilde{b} > 0$. The later inequality allows us to replace the dual objective function by the constraint $u^T \tilde{b} = 1$. So, we have a dual problem that finds a linear combination of the rows $\tilde{A}y \geq \tilde{b}$. Multiplying (3.1) by ${u^*}^T$, we obtain a valid inequality ${u^*}^T \tilde{A}y \geq {u^*}^T \tilde{b}$. By using (3.2) and previous observations, this implies that $0^T y = {u^*}^T \tilde{A}y \geq {u^*}^T \tilde{b} > 0$, thus providing the infeasibility of $\tilde{A}y \geq \tilde{b}$. The existence of u^* is guaranteed by the Farkas' Lemma. Finally, we can use the

[1] The big-M method is an initialization method for the Simplex. It is similar to the Two-Phase method, but the big-M method assigns a "big" value to the artificial variables x_a on the original objective function. So, if the problem is $\min\{cx : Ax = b, \quad x \geq 0\}$ we solve $\min\{cx + M1x_a : Ax + x_a = b, \quad x, x_a \geq 0\}$. Then, if there exists an optimal solution, this will not include any artificial variable.

heuristic dual objective function $\sum_i w_i u_i$ to get u^*, where w_i are used to drive the linear problem to get a small support of u because each vertex of the dual polyhedron in (3.1) has a support defining a MIS. In addition, one can identify alternative MIS by iteratively setting to zero some of the u variables.

Polyhedral combinatorics. Marc E. Pfetsch [14] looks for the complementary problem of MinFB, the MAXIMUM FEASIBLE SUBSYSTEM PROBLEM (Max FS), whose goal is to find a feasible subsystem containing as many inequalities as possible. In his article, he works with irreducible infeasible subsystems (IIS, mentioned in the heuristics approach), and the polyhedron and facets of them. For ease of notation, let us denote by $\Sigma = \{Ax \leq b\}$ the original system of n variables and m inequalities. We will built in the following results

1. If Σ is an infeasible system, then the IISs of Σ are in one-to-one correspondence with the supports of the vertices of the polyhedron [2] $P(\Sigma) = \{y \in \mathbb{R}^m : y^T A = 0, y^T b = -1, y \geq 0\}$.

2. The vertices of $P(\Sigma)$ are uniquely defined by their supports.

3. Let $S(\Sigma) = S = [m]$ be the set of inequality indexes of the system Σ. For $S' \subseteq S$, define the polyhedron $P_{S'}(\Sigma) = \{y \in P(\Sigma) : y_i = 0, i \in S'\}$. Then, $C \subseteq S(\Sigma)$ is an IIS cover if and only if $P_C(\Sigma) = \emptyset$.

4. If $P_C(\Sigma) \neq \emptyset$ then there exists v vertex of $P_C(\Sigma)$. This implies that $supp(v) \cap C = \emptyset$, and then $supp(v)$ is an IIS that is uncovered. This vertex can be found in polynomial time.

5. The inequality arising from the IIS I defines a facet of the polytope $Q = conv\left\{y \in \{0,1\}^m : \sum_{i \in S} y_i \geq 1 \text{ for all IIS } S\right\}$ as long as $|I| > 1$.

Finally we define the SEPARATION PROBLEM FOR IIS-INEQUALITIES: given $y^* \in [0,1]^m$, check if there exists an IIS I such that the corresponding inequality is violated by y^*, i.e. $\sum_{i \in I} y_i^* < 1$. Then, finding the IIS-cover is an special case of this separation problem when y^* is the incidence vector of the set to be tested. Nevertheless, the SEPARATION PROBLEM FOR IIS-INEQUALITIES is NP-hard.

We explain three heuristic methods and we remark that by its nature, these heuristics may fail to produce a violated IIS-inequality. The main goal of the three methods is to find

$$\lambda := \begin{cases} \min\{\sum_{i \in S} y_i^* : S = supp(v), v \text{ vertex of } P(\Sigma)\} & P(\Sigma) \neq \emptyset \\ \infty & P(\Sigma) = \emptyset, \end{cases}$$

because if $\lambda < 1$, then by the first point of the results, $supp(v)$ provides an IIS whose IIS-inequality is violated; if $\lambda \geq 1$, then no such IIS exists.

[2] Observe that the description of this polyhedron is very similar to the description obtained in the Mixed-integer programming approach. Also, it is strongly related to another formulation of the Farkas' Lemma that establishes $P(\Sigma) \neq \emptyset$ if and only if Σ is infeasible.

- Method 1: "Single". The separation problem is simply approximated by the linear program $\min\{{y^*}^T p : p \in P(\Sigma)\}$, whose solution is a vertex which provides an IIS. This method gives one inequality by iteration that may not be violated necessarily; we iterate until no IIS is found.

- Method 2: "Extend". This method is an extension of Method 1. Let $S = supp(y^*)$. Then a vertex solution can be found using $\min\{{y^*}^T p : p \in P_S(\Sigma)\}$. If the problem is feasible, then it gives a vertex which corresponds to an IIS-cover I. We proceed by greedily enlarging S, that is, we pick an element of I that is contained in the maximum number of IIS's and then we repeat until there is not vertex found by the linear program.

- Method 3: "Round". In contrast to Method 2, in this method the set S is arbitrarily chosen: take $\alpha \in [0, 1]$ and $S = \{i : y_i^* \geq \alpha\}$. In the article, the author starts with $\alpha = 0.1$ and then increases it by 0.1 until S is not longer an IIS-cover. Failure is declared if $\alpha \geq 0.6$.

Chapter 4

Hardness results for MinFB

In this chapter, were are going to show that the Minimum Feasibility Blocker problem (MinFB) is not FPT when it is parameterized by k, the minimum size of a feasible blocker. To do so, we are going to make a reduction to a problem which belongs to the W[1]-hard class. First of all, we state the input and the goal of MinFB when it is parameterized by k.

Minimum Feasibility Blocker (MinFB)

Input: A coefficient matrix $A \in \mathbb{Q}^{m \times n}$, a right-hand side vector $b \in \mathbb{Q}^m$ and an integer k.
Task: Find a set $\mathcal{I} \subseteq [m]$ of size at most k such that $(a_{i,\bullet} \cdot x \leq b_i)_{i \in [m] \setminus \mathcal{I}}$ is feasible for some $x \in \mathbb{Q}^n$.

The main result of this section is the following theorem. We recall that a matrix that contains exactly one $+1$ and one -1 in each row is called *difference constraint matrix*.

Theorem 4.1. *The* MinFB *problem is W[1]-hard when parameterized by the minimum size k of a feasibility blocker, even for difference constraints and right-hand sides $b \in \{-1, +1\}^m$.*

Assuming FPT $\neq$ W[1], this theorem disproves the conjecture of Sylvain Guillemot [15] that finding the minimum number of unsatisfied equations or inequalities is fixed-parameter tractable for linear systems with at most two variables per equation or inequality.

Interestingly, we can actually rule out a polynomial compression for MinFB parameterized by $k + b_-$, where b_- is the number of negative entries in b:

Theorem 4.2. *Assuming NP $\not\subseteq$ coNP/poly,* MinFB *does not admit a polynomial compression when parameterized by $k + b_-$ even for systems A of difference constraints and right-hand sides $b \in \{-1, +1\}^m$.*

The most intriguing open question arising from this result is whether our hardness result can be strengthened to rule out a polynomial compression for MinFB parameterized by $k + b_+$.

Additionally, Till Fluschnik et. al. [16] gave a result in terms of polynomial compression with two parameters, stated in the next proposition.

Proposition 4.3 ([16])**.** *Assuming NP$\not\subseteq$ coNP/poly,* Bounded Edge Directed (s,t)-Cut *does not admit a polynomial compression in $k + \ell$ even when G is a directed acyclic digraph.*

In order to prove Theorems 4.1, and 4.2, we will give a reduction to a problem that is indeed W[1]-hard: the Bounded Edge Directed (s,t)-Cut problem, which is stated below.

Bounded Edge Directed (s,t)-Cut

Input: A digraph G, two distinct vertices s,t, and integers $k,\ell \in \mathbb{Z}_+$.

Task: Find a set $X \subseteq E(G)$ of size at most k such that $G-X$ contains no s-t-paths of length at most ℓ.

Petr A. Golovach and Dimitrios M. Thilikos [17] show that this problem is parameterized intractable for parameter k:

Proposition 4.4. *([17]) The* Bounded Edge Directed (s,t)-Cut *problem is* W[1]*-hard when parameterized by k even for the special case where G is an acyclic digraph.*

The reduction from Bounded Edge Directed (s,t)-Cut to MinFB is going through the following intermediate reductions:

Bounded Edge Directed (s,t)-Cut
$\to$ Directed Small Cycle Transversal
$\to$ Negative DFAS
$\to$ MinFB.

Therefore, let us begin by defining the Directed Small Cycle Transversal problem.

Directed Small Cycle Transversal

Input: A directed graph G and integers k,ℓ.

Task: Find a set $X \subseteq E(G)$ of size at most k such that $G-X$ contains no cycles of length at most ℓ.

Till Fluschnik et. al. showed that Directed Small Cycle Transversal does not admit a kernel of size polynomial in k and ℓ, unless NP$\subseteq$ coNP/poly. Their result can be strengthened to not admitting a polynomial compression. They further observed that Directed Small Cycle Transversal admits a simple branching algorithm that runs in time $\mathcal{O}(\ell^k \cdot n \cdot (n+m))$ when n is the number of vertices and m the number of arcs. Here we argue that a dependence on both parameters k and ℓ is necessary for fixed-parameter tractability:

Lemma 4.5. *There is a polynomial parameter transformation from* Bounded Edge Directed (s,t)-Cut *in acyclic digraphs with parameter k (parameter ℓ, parameter $k+\ell$) to* Directed Small Cycle Transversal *with parameter k (resp. ℓ, resp. $k+\ell$).*

This even holds if Directed Small Cycle Transversal *is restricted to instances where there are vertices $s,t \in V(G)$ such that all cycles of G consist of an s-t-path and an arc of the form (t,s).*

Proof. Let (G,s,t,k,ℓ) be a Bounded Edge Directed (s,t)-Cut instance where G is an acyclic digraph. As G is an acyclic digraph, it admits a topological ordering $v_1,\ldots,v_{|V(G)|}$ of its

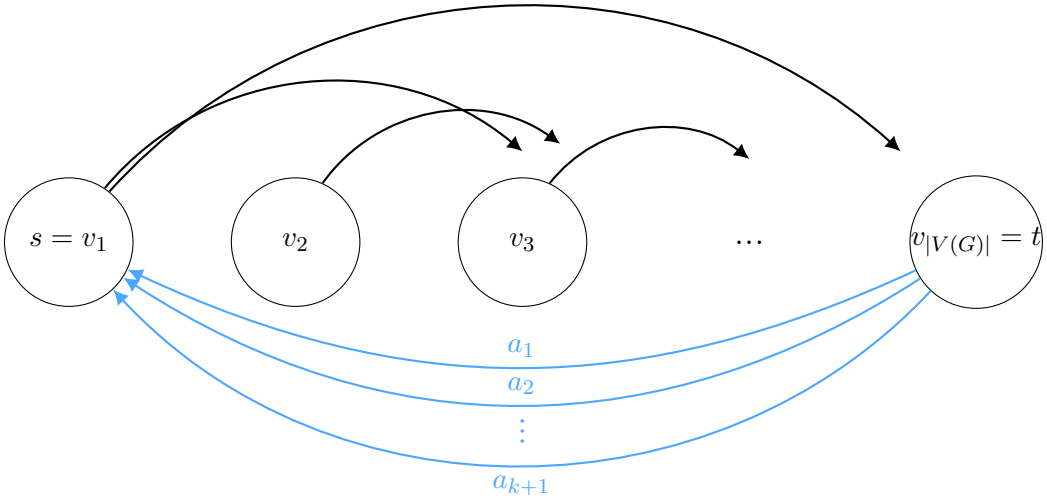

Figure 4.1: Directed Small Cycle Transversal *instance obtained from a* Bounded Edge Directed (s,t)-Cut *instance, after adding* $k+1$ *parallel arcs from* t *to* s.

vertices, so that there are no arcs (v_i, v_j) for $j < i$. Without loss of generality, let $v_1 = s$ and $v_{|V(G)|} = t$, as vertices before s or after t in a topological ordering are never part of any s-t-path.

We now create a digraph G' from G by adding $k+1$ parallel arcs $a_1, \ldots, a_{k+1}$ from t to s. Then every cycle in G' consists of an s-t-path and an arc a_i, as G was acyclic. Set $\ell' = \ell + 1$. Then (G', k, ℓ') is an instance of Directed Small Cycle Transversal (see Figure 4.1).

The above transformation can be done in polynomial time and the parameter increases by at most one (depending on whether ℓ is part of the parameter).

It remains to show that (G', k, ℓ') is a "yes"-instance of Directed Small Cycle Transversal if and only if (G, s, t, k, ℓ) is a "yes"-instance of Bounded Edge Directed (s,t)-Cut.

For the forward direction, let X' be a solution to (G', k, ℓ'), that is, $G' - X'$ has no cycle of length at most ℓ and $|X| \leq k$. Consider $X = X' \setminus \{a_1, \ldots, a_{k+1}\}$. For sake of contradiction, suppose that $G - X$ contains an s-t-path P of length at most ℓ. As $|X'| \leq k$, it can not contain all arcs a_i. Without loss of generality, $a_1 \notin X'$. Then P followed by a_1 is a cycle in $G' - X'$ of length at most $\ell + 1 = \ell'$—a contradiction to X' being a solution to (G', k, ℓ').

For the reverse direction, let X be a solution to (G, s, t, k, ℓ), this is $G - X$ has no s-t-path of length at most ℓ and $|X| \leq k$. Then X is also a solution to (G', k, ℓ') by the following argument. Suppose, for sake of contradiction, that $G' - X$ contains a cycle C of length at most ℓ'. By the structure of G', C consists of an s-t-path P in $G - X$ and an arc a_i. Then $|P| = |C| - 1 \leq \ell$, contradicting that X is a solution to (G, k, ℓ). □

Now we introduce another cycle deletion problem, this time on arc-weighted digraphs.

Negative Directed Feedback Arc Set (Negative DFAS)

Input: A digraph G, a weight function $w : A(G) \to \mathbb{Q}$ and an integer k.

Task: Find a set $X \subseteq A(G)$ of size at most k such that $G - X$ has no negative cycles.

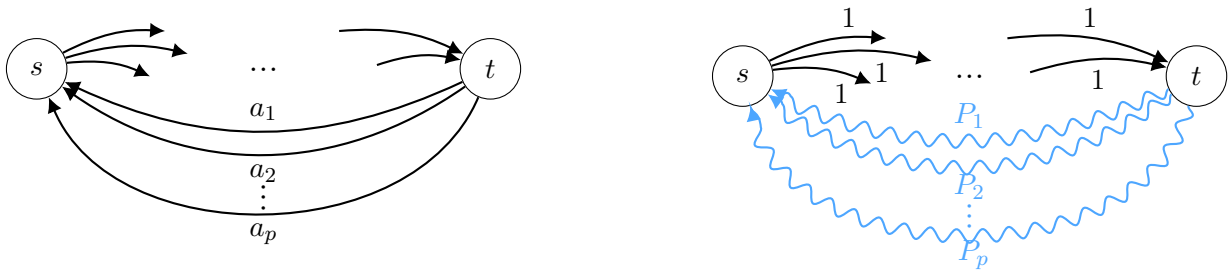

Figure 4.2: Directed Small Cycle Transversal *instance (left) and* Negative DFAS *instance after changing the* (t, s)*-arcs to paths (rigth). The paths are drawn as curved-patterns and each arc of it has weight* -1*. The rest of the arcs have weight* $+1$*.*

Lemma 4.6. *There is a PPT from* Directed Small Cycle Transversal *on instances where every cycle uses an arc of type* (t, s) *when parameterized by* k *(by* $k + \ell$*) to* Negative DFAS *parameterized by* k *(resp.* $k + w_-$*, where* w_- *is the number of arcs with negative weight); this even holds in the case where* $w : A(G) \to \{-1, +1\}$*.*

Proof. We start with a Directed Small Cycle Transversal instance (G, k, ℓ) as described in the lemma. Let $a_1, \ldots, a_p$ be the arcs of the form (t, s). For any $p \geq k+1$ there is always an arc which survives the deletion of some arc set of at most k elements. So we can assume that $p \leq k+1$ as deleting superfluous arcs does not change the solution. Set $A_{+1} = A(G) \setminus \{a_1, \ldots, a_p\}$. Now replace each a_i by mutually disjoint (except for s and t) paths P_i of length ℓ. Call the resulting directed graph G' and let $A_{-1} = \cup_{i=1}^{p} A(P_i)$. Finally, define $w(a) = 1$ for $a \in A_{+1}$ and $w(a) = -1$ for $a \in A_{-1}$. As $A(G') = A_{-1} \uplus A_{+1}$, the function $w : A(G') \to \{-1, +1\}$ is well defined.The instance (G', w, k) has the required form (see Figure 4.2).

Also, the transformation can be made in polynomial time. As k remains unchanged and $w_- = |A_{-1}| = \ell \cdot p \leq \ell \cdot (k + 1)$ is bounded by a polynomial in $k + \ell$, the parameter restrictions of PPTs are fulfilled. It remains to prove that (G', w, k) is a "yes"-instance of Negative DFAS if and only if (G, k, ℓ) is a "yes"-instance of Directed Small Cycle Transversal.

For the forward direction, let X' be a solution of (G', w, k). Let X be the set where every arc of X' which is part of some P_i is replaced by a_i (and duplicates are removed). Clearly, $|X| \leq |X'| \leq k$. Suppose there is a cycle C of length at most ℓ in $G - X$. Then C contains a unique arc a_j. Let C' be the cycle in G' resulting from the replacement of a_j by P_j in C. Then C' is also in $G' - X'$ by choice of X and contains at most $\ell - 1$ arcs in A_{+1} and ℓ arcs in A_{-1}. This yields the contradiction $w(C') = |A(C') \cap A_{+1}| - |A(C') \cap A_{-1}| \leq \ell - 1 - \ell = -1$.

For the reverse direction, let X be a solution of (G, k, ℓ). Let X' the set X where every arc a_i is replaced by the first arc of P_i. By definition of G', $X' \subseteq A(G')$ and $|X'| = |X| \leq k$ holds. Now suppose there is a cycle C' in $G' - X'$ with $w(C') < 0$. As the paths P_i are mutually disjoint (except the end vertices) every inner vertex of each P_i has in-degree and out-degree one. Thus, if there is an arc of some P_i inside C' the whole path P_i is. Replace each such P_i by a_i to obtain a cycle C. By construction of G' and X', this cycle C is in $G - X$. Each cycle in G and therefore also C contains exactly one arc of type a_i. Therefore, C' contains exactly one path P_i and from $0 > w(C') = |A(C') \cap A_{+1}| - |A(C') \cap A_{-1}| = |A(C') \cap A_{+1}| - \ell$ we get that $|A(C') \cap A_{+1}| < \ell$. As $|A(C') \cap A_{+1}|$ is integral we can sharpen the bound to $|A(C') \cap A_{+1}| \leq \ell - 1$.

By $A(C) = (A(C') \cap A_{+1}) \uplus \{a_i\}$, we get that $|A(C)| = |A(C') \cap A_{+1}| + 1 \leq \ell$—a contradiction. □

Theorem 4.7. *The* Negative DFAS *problem and the* MinFB *problem for difference constraints are parameter-equivalent. Additionally, the equivalence can be constructed such that there is a one-to-one correspondence between constraints and arc weights with* $b_a = w(a)$.

Proof. Let (G, w, k) be a Negative DFAS problem instance, and let $n = |V(G)|$ and $m = |A(G)|$. Fix an arbitrary ordering $v_1, \ldots, v_n$ of the vertices of G and $\alpha_1, \ldots, \alpha_m$ of the arcs of G.

As matrix A, we take the incidence matrix of G which is defined as matrix $A = (a_{i,j}) \in \mathbb{R}^{m \times n}$ with entries $a_{i,j} = +1$ for $\alpha_i \in \delta^-(v_j)$, $a_{i,j} = -1$ for $\alpha_i \in \delta^+(v_j)$, and $a_{i,j} = 0$ otherwise. Furthermore, let $b_i = w(\alpha_i)$. The resulting tuple (A, b, k) is an instance of the MinFB problem with A being a matrix of difference constraints.

The construction is bijective by the following reverse construction. Define a directed graph on n vertices $v_1, \ldots, v_n$, then for every constraint $a_{i,\bullet} \cdot x \leq b_i$ add an arc as follows: let j^- be the unique index with $a_{i,j^-} = -1$, and let j^+ be the unique index with $a_{i,j^+} = +1$. Add an arc $\alpha = (v_{j^+}, v_{j^-})$ with weight $w(\alpha) = b_i$ to the current digraph. Let G be the resulting digraph after all arcs are added. Then (G, w, k) is the constructed Negative DFAS instance. It is easy to verify that this indeed reverses the first construction.

Now we want to compare solutions of the two problems. Intuitively, deleted constraints and arcs have a one to one correspondence, but we will formally prove the equivalence here. For this we need the notion of "feasible potentials". A *feasible potential* (with respect to G and w) is a function $\pi : V(G) \to \mathbb{Q}$ such that, for every arc $a = (x, y) \in A(G)$, the following inequality holds: $w(a) - \pi(x) + \pi(y) \geq 0$. It is well-known that a weighted digraph has a feasible potential if and only if it has no cycle of negative weight (see, for example, the book of Alexander Schrijver [18, Theorem 8.2]).

In the following, for each $X \subseteq A(G)$ denote by $\mathcal{I}_X$ the corresponding indices of the constraints and vice-versa. Then the following equivalences hold:

$(G - X, w)$ contains no negative cycles with respect to w.

$\Leftrightarrow$ $(G - X, w)$ has a feasible potential $\pi : V(G) \to \mathbb{Q}$.

$\Leftrightarrow$ There is some $\pi : V(G) \to \mathbb{Q}$ such that $\pi(u) \leq \pi(v) + w(\alpha)$ for all $\alpha = (u, v) \in A(G) \setminus X$.

$\Leftrightarrow$ There is some $x \in \mathbb{Q}^{V(G)}$ such that $x_u - x_v \leq w(\alpha)$ for all $\alpha = (u, v) \in A(G) \setminus X$.

$\Leftrightarrow$ There is some $x \in \mathbb{Q}^n$ such that $a_{i,\bullet} \cdot x \leq b_i$ for all $i \in \{1, \ldots, m\} \setminus \mathcal{I}_X$.

Furthermore, as X and $\mathcal{I}_X$ have the same cardinality, the last statement is equivalent to the statement that X is a solution to (G, w, k) if and only if $\mathcal{I}_X$ is a solution to (A, b, k). □

Concatenating all reductions above, we obtain the following corollary:

Corollary 4.8. *There is a PPT from* Bounded Edge Directed (s,t)-Cut *parameterized by* k *(in* $k + \ell$*) to* MinFB *parameterized by* k *(in* $k + b_-$*). This even holds for instances of* MinFB *where* A *is a system of difference constraints and* $b \in \{-1, +1\}^m$.

With the last corollary, our two hardness results can be proven:

Proof of Theorem 4.1. With Proposition 2.2, we can use the W[1]-hardness of Bounded Edge Directed (s,t)-Cut from Proposition 4.4 and the PPT from Corollary 4.8 to get the W[1]-hardness of MinFB. Also the structure of the instance follows from this. □

Proof of Theorem 4.2. This follows by combining Proposition 4.3 and Corollary 4.8 with the help of Proposition 2.2. The structure of the MinFB instance follows as in Theorem 4.1. □

We already showed that using only the parameter minimum size of a feasible blocker is not enough to obtain an FPT algorithm for the MinFB problem. Additionally, we need at least another parameter in order to get such FPT algorithm. In the next section, we will give two FPT algorithms which are parameterized using two parameters each. As we mentioned in Chapter 2, knowing that MinFB does not admit a polynomial compression with the parameters k and b_- is a complementary result for the FPT algorithm: both parameters are required.

Chapter 5

FPT algorithms: parameters for systems of difference constraints

In the last chapter, we proved that the MinFB problem has no FPT algorithm when it is parameterized by the size of the blocker (see Theorem 4.1). Is it possible to find an FPT algorithm using additionally another parameter? If the system is such that $b \in \{-1, +1\}^m$ and the constraint matrix is a difference constraint, the answer is yes, indeed. Recall that a difference constraint matrix contains in each row exactly one $+1$ and one -1. In this chapter, we are going to use two extra parameters: the number of positive entries in b, and the number of negative entries in b. With the help of these two parameters, we will get two FTP algorithms, one for each of them.

Let A be a $n \times m$ matrix of difference constraints, that is, A contains exactly one $+1$ and one -1 in each row, and let $b \in \{-1, +1\}^m$. Define w_- as the number of negative entries from b and let w_+ as the number of positive entries from b, that is, the number of -1's and the number of $+1$'s in the b vector respectively. We write w_- and w_+ as concordance with the Theorem 4.7, since there is a 1-to-1 correspondence between b and w.

We will require a subroutine that finds a negative cycle for both algorithms. We will use Algorithm 1, which finds a shortest negative cycle C of length at most ℓ in $\mathcal{O}(\ell nm)$ time, where ℓ is an integer. Recall that *shortest* refers to the number of arcs.

5.1 Few negative right-hand sides

We exhibit an FPT algorithm for the MinFB problem for constraint matrices and with right-hand side $b \in \{-1, +1\}^m$, when parameterized by the size k of the deletion size of a blocker and the number w_- of negative right-hand sides. To do so, we will give a simple algorithm for the Negative DFAS problem with $w : A(G) \to \{-1, +1\}$, parameterized by k and w_-.

The general idea of the algorithm is the following: it first guesses the negative arcs in the solution and then it branches on the positive arcs of a negative cycle (as long as such cycle exists). The algorithm keeps track of the already deleted arcs in the set S_0. See Algorithm 2.

Lemma 5.1. *Algorithm 2 is correct and runs in time* $(k+1)w_-^{k+1}\mathcal{O}(nm)$.

Algorithm 2: SimpleNegativeCycleDeletion

Input : A digraph G, arc weights $w : A(G) \to \mathbb{Q}$ and $k \in \mathbb{Z}_+$.
Output: A set $S \subseteq A(G)$ of at most k arcs such that $G - S$ has no negative cycles or `false` if no such set exists.

```
1 for every k_-, k_+ ∈ Z_{≥0} with k_- + k_+ = k do
2     for every subset S_- of w^{-1}(-1) ⊆ A(G) with |S_-| = k_- do
3         S_0 = S_-.
4         while G - S_0 contains a negative cycle C and |S_0| ≤ k do
5             Branch on adding an arc a ∈ A(C) with w(a) = +1 to S_0.
6         if |S_0| ≤ k then
7             return S_0.
8 return false.
```

Proof. The algorithm works in three steps: The first **for** loop guesses how many of the deleted arcs have weight -1 with the variable k_-. The second **for** loop then iterates over every (k_-)-element subset of these arcs. The last procedure then tries to fix the negative cycle by only deleting arcs of weight $+1$.

As we enumerate all choices of k_- and the subsets of negative arcs we only need to argue correctness for the last procedure. The procedure only returns a value other than "false" if this value is a solution. So we only need to argue that if there is a solution we will find it. So let S be a solution and S_0 contain all k_- arcs of weight -1 in S. We get S_0 by correct guessing of the **for** loops. If $S_0 = S$, then $|S_0| \leq k$ and the graph $G - S_0$ will contain no negative cycle, so we correctly return S_0. Otherwise, there is a negative cycle C in $G - S_0$. By choice of S_0 there must be an arc $a \in A(C) \cap S$ with weight $+1$. Branching on all possible choices in line 5, one of the branches must have found the right arc and added it to S_0. Thus, in each recursive call we find an additional element of S until $S_0 = S$.

For the runtime, the first main observation to be made is that any negative cycle C has length at most $2w_- - 1$. Furthermore, at most $w_- - 1$ arcs of it can have weight $+1$. Therefore, we can check by Lemma 2.1 for the existence of a negative cycle in time $\mathcal{O}(w_- nm)$ and iterate over all arcs with weight $+1$ in a cycle in time $\mathcal{O}(w_-)$. As S_0's size increases by one with each branching and we stop (correctly) if $|S_0| > k$ at most $k - k_-$ recursive calls are made by the branching. Thus, the runtime of the inner branching procedure for fixed S_- is at most $(w_-)^{k-k_-+1}\mathcal{O}(nm)$. The inner **for** loop enumerates, for a fixed value of k_-, at most $w_-^{k_-}$ sets S_-. This **for** loop is executed $k+1$ many times by the outer **for** loop. So the algorithm runs in time $(k+1)w_-^{k_-} \cdot (w_-)^{k-k_-+1}\mathcal{O}(nm) = (k+1)(w_-)^{k+1}\mathcal{O}(nm)$. □

We can summarize the results of this section with the following theorem.

Theorem 5.2. *There is an algorithm that solves* MinFB *problem for systems $\mathcal{S}$ of m difference constraints over n variables and right-hand sides $b \in \{-1, +1\}^m$ in time $(k+1)(b_-)^{k+1}\mathcal{O}(nm)$, when parameterized by k the size of the blocker and by b_- the number of negative entries of b.*

Proof. By Theorem 4.7 we can construct, in polynomial time, an instance of Negative DFAS that has the same parameters $k + w_-$, and such that the original instances is a "yes"-instance if and only if the constructed instance is. Lemma 5.1 then allows us to solve this instance in the

claimed time. □

5.2 Few positive right-hand sides

We show an FPT algorithm for the MinFB when parameterized by the number k of the deletion size of a blocker and the number w_+ of positive right-hand sides. As the last section, we consider the MinFB problem for constraint matrices and with right-hand side $b \in \{-1, +1\}^m$.

Observe that the DFAS problem corresponds to the case when $w_+ = 0$, as in this case all arcs have negative weight. In fact, our algorithm makes oracle calls to an algorithm for the more general problem Subset DFAS.

Subset Directed Feedback Arc Set (Subset DFAS)

Input: A digraph G, an arc set $U \subseteq A(G)$ and an integer k.
Task: Find a set $X \subseteq A(G)$ of at most k arcs that intersects each cycle containing some arc of U.

Rajesh Chitnis et al. [19] showed that he Subset DFAS problem is fixed-parameter tractable by parameter k.

Proposition 5.3 ([19])**.** Subset DFAS *is solvable in time* $2^{\mathcal{O}(k^3)} \cdot n^{\mathcal{O}(1)}$.

The main observation for our algorithm is stated below.

Lemma 5.4. *Let G be a digraph with arc weights $w : A(G) \to \{-1, +1\}$. Then either G has a negative cycle of length at most $2(w_+)^2 + 2w_+$, or every negative cycle C has some arc $a \in A(C)$ that lies only on negative cycles of G.*

Proof. First, if G has no negative cycles then we are done. We are also done if there is a negative cycle of length at most $2(w_+)^2 + 2w_+$. Hence, we can assume that G has a negative cycle and all negative cycles have length at least $2(w_+)^2 + 2w_+ + 1$.

Suppose, for sake of contradiction, that every arc a in a negative cycle lies on a cycle C_a of non-negative weight. Let C be a shortest negative cycle in G. By the argumentation above, C has length at least $2(w_+)^2 + 2w_+ + 1$. In particular, C has length at least $w_+(w_+ + 1) + 1$ and contains, by definition of w_+, at most w_+ arcs of weight $+1$; all other arcs of C have weight -1. By pigeonhole principle, there must be a segment P of C consisting of $w_+ + 1$ arcs of weight -1.

By assumption, for every arc $a = (v, w) \in A(P)$ we have a w-v-path R_a of length at most $|C_a| - 1$. As $w(C_a) \geq 0$ and the C_a's contain at most w_+ arcs of weight $+1$ and all other arcs have weight -1 the cycles have length at most $2w_+$, that is, $|C_a| \leq 2w_+$. Thus, the reverse paths R_a have length $|R_a| \leq |C_a| - 1 \leq 2w_+ - 1$. Say P is an s-t-path, then by concatenating the R_a's with $a \in P$ we get an t-s-walk R' which contains a t-s-path R. By definitions of w_+,the path R contains at most w_+ arcs of weight $+1$, thus $w(R) \leq w_+$.

Consider the closed walk $O = P \circ R$. Then $w(O) = w(P) + w(R) = -(w_+ + 1) + w_+ \leq -1$. Thus, O contains a negative cycle C'. We have

$$\begin{aligned}
|C'| &\leq |O| \\
&= |P| + |R| \\
&\leq |P| + \sum_{e \in E(P)} |R_e| \\
&\leq (w_+ + 1) + (w_+ + 1)(2w_+ - 1) \\
&= 2w_+(w_+ + 1) \\
&< 2(w_+)^2 + 2w_+ + 1 \\
&\leq |C|
\end{aligned}$$

yields a contradiction to the fact that C was a shortest negative cycle. □

Now we describe the idea of Algorithm 3. First, the algorithm checks for negative cycles with up to $2(w_+)^2+2w_+$ arcs. It then guesses the arcs contained in a solution like in our first algorithm, Algorithm 2. Let us denote that set by S'. Thus, we are left with a digraph without small cycles. By Lemma 5.4 we know that each negative cycle has an arc that lies only on negative cycles. We call that set of arcs U. Thus, it remains to solve an instance of Subset DFAS for input (G, U, k). Let X be the solution for such an instance. In the end, we will define the minimum feasibility blocker as $S = S' \cup X$.

Algorithm 3: NegativeCycleDeletion

Input : A digraph G with arc weights $w : A(G) \to \mathbb{Q}$ and $k \in \mathbb{N}$.
Output: A set $S \subseteq A(G)$ of at most k arcs such that $G - S$ has no negative cycle, or `false` if no such set exists.

```
if k < 0 then
    return false.
if there is some negative cycle C of length at most 2(w_+)^2 + 2w_+ in G then
    Branch on deleting an arc of C and try to solve with k − 1.
else
    Identify the set U of all arcs which do not lie on a non-negative cycle.
    return SubsetDirectedFeedbackArcSet (G, U, k).
```

We have to show how to detect the set U of all arcs which lie only on negative cycles before proving correctness and runtime of Algorithm 3. Let us begin by demonstrating that this problem is NP-hard even for weights $w : A(G) \to \{-1, +1\}$. To do so, we provide a reduction from the Hamiltonian s-t-Path problem, which is described below. Its NP-hardness was shown by Richard M. Karp [20].

Hamiltonian s-t-Path

Input: A digraph H and vertices $s, t \in V(H)$.
Task: Find s-t-path in H visiting each vertex of H exactly once.

The reduction from Hamiltonian s-t-Path to Subset DFAS works as follows: consider the instance (H, s, t) with n vertices. Add a path P of length $n-1$ from t to s to the graph. Assign weight $+1$ to each arc of H, and weight -1 to each arc of P. Then an arc of P lies on a cycle of non-negative length if and only if there is an Hamiltonian s-t-path in H.

However, for this construction of weights w we have $w_+ \in \Omega(n)$. We will now show that the task is indeed fixed-parameter tractable when parameterized by w_+. For that, the main observation is that every non-negative cycle has length at most $2w_+$ since we have at most w_+ positive arcs and then the number of negative arcs are at most w_+. We now consider the Weighted ℓ-Path problem.

Weighted ℓ-Path

Input: A digraph G with arc weights $w : A(G) \to \mathbb{Q}$ and numbers $W \in \mathbb{Q}, \ell \in \mathbb{N}$.
Task: Find a path of length exactly ℓ and weight at least W in G.

Meirav Zehavi [21] gave a fast algorithm for Weighted ℓ-Path, based on color-coding techniques.

Proposition 5.5 ([21])**.** Weighted ℓ-Path *can be solved in time* $2^{\mathcal{O}(\ell)} \cdot \mathcal{O}(m \log n)$ *on digraphs with* n *vertices and* m *arcs.*

Therefore, given an arc $a = (s, t)$ one can enumerate all path sizes ℓ from 1 to $2w_+ - 1$ and ask whether there is a t-s-path of length ℓ of weight at least $-w(a)$. This way one can detect a non-negative cycle containing a. This is stated in the following straight result.

Corollary 5.6. *Let* G *be a digraph with* n *vertices and* m *arcs, let* $w : A(G) \to \{-1, +1\}$ *and* $(s, t) \in A(G)$*. Then one can detect in time* $2^{\mathcal{O}(w_+)} \cdot \mathcal{O}(m \log n)$ *if* $a = (s, t)$ *is part of some non-negative cycle* C*.*

Now we can prove the correctness and the runtime of Algorithm 3.

Lemma 5.7. *Algorithm 3 is correct and runs in time* $2^{\mathcal{O}(k^3 + w_+ + k \log w_+)} \cdot n^{\mathcal{O}(1)}$.

Proof. Obviously, we return correctly "false" if our set shall contain less than zero items. Otherwise, we detect with help of Lemma 2.1 whether there is a negative cycle in G of length at most $2(w_+)^2 + 2(w_+)$, in time $\mathcal{O}\left(\left(2(w_+)^2 + 2(w_+)\right) mn\right)$.

If there is such a cycle C, we split into $|C|$ instances, one for each $a \in A(C)$, calling our algorithm recursively with parameter k decreased by one on $G - a$. This is correct, as one of the arcs of C needs to be deleted and we just search for every arc if there is a solution containing this arc.

Otherwise, all negative cycles have length at least $2(w_+^2 + 2(w_+) + 1$, and by Lemma 5.4 every negative cycle must have an arc that lies only on negative cycles. Choose one arc of each such cycle and gather them in the set $U_{<0}$. By definition of the set U, in our algorithm we have $U_{<0} \subseteq U$. Subset DFAS for (G, U, k) now asks for a set S of size at most k such that $G - S$ has no cycle containing an arc of U. As $U_{<0} \subseteq U$, we get that a solution for (G, U, k) is also a solution for

$(G, U_{<0}, k)$. We now want to show that also the reverse direction holds and this solves our original problem. For this, let $S_{<0}$ be a solution for $(G, U_{<0}, k)$ and C a cycle in $G - S_{<0}$. If there is no such cycle, we are done. Otherwise, we know that C cannot be a cycle of negative weight as every cycle of negative weight has an arc in $U_{<0}$. But as $w(C) \geq 0$, cycle C cannot contain an arc of U as those are not contained in non-negative cycles. Thus, $S_{<0}$ is also a solution for (G, U, k). Furthermore, all cycles in $G - S_{<0}$ are non-negative, and therefore $S_{<0}$ is also a solution to our Negative DFAS instance (G, k). This shows the correctness of Algorithm 3.

Its runtime can be bounded as follows: detecting cycles in line 4 can be done in time $\mathcal{O}((2(w_+)^2 + 2w_+)nm)$. The branching step then creates up to $2(w_+)^2 + 2w_+$ instances with the parameter k decreased by one. As there is no other recursive call to this algorithm, we have at most $(2(w_+)^2 + 2w_+ + 1)^k$ instances for which this algorithm is called. In the end, we call the algorithm from Corollary 5.6 to compute the set U which takes time $2^{\mathcal{O}(w_+)} \cdot \mathcal{O}(m^2 \log n)$, as it is called for every arc. By Proposition 5.3, the final call to the Subset DFAS oracle takes time $2^{\mathcal{O}(k^3)} \cdot n^{\mathcal{O}(1)}$.

Thus, we obtain an overall runtime of

$$(2w_+^2 + 2w_+ + 1)^k \cdot \left[(w_+^2 + 2w_+) \cdot \mathcal{O}(nm) + 2^{\mathcal{O}(w_+)} \cdot \mathcal{O}(m^2 \log n) + 2^{\mathcal{O}(k^3)} \cdot n^{\mathcal{O}(1)}\right],$$

which simplifies to $(w_+)^{\mathcal{O}(k)} \cdot 2^{\mathcal{O}(k^3 + w_+)} \cdot n^{\mathcal{O}(1)} = 2^{\mathcal{O}(k^3 + w_+ + k \log w_+)} \cdot n^{\mathcal{O}(1)}$. □

To conclude, we summarize this section with the next theorem.

Theorem 5.8. *There is an algorithm that solves* MinFB *for systems $\mathcal{S}$ of m difference constraints over n variables and right-hand sides $b \in \{-1, +1\}^m$ in time $2^{\mathcal{O}(k^3 + b_+ + k \log b_+)} \cdot n^{\mathcal{O}(1)}$, when parameterized by k the size of the blocker and by b_+ the number of positive entries of b.*

Proof. By 4.7 we can construct, in polynomial time, an instance of Negative DFAS that has the same parameters $k + w_-$, and such that the original instances is a "yes"-instance if and only if the constructed instance is.

Lemma 5.7 then allows us to solve this instance in the claimed time. Regarding the run time, note that $m \leq (k+1)n^2 \in 2^{\mathcal{O}(\log k)} \cdot n^{\mathcal{O}(1)}$. □

5.3 Adding zeros to the right-hand sides

In the last two sections, we exposed two FPT algorithms that solve the MinFB problem for the special case when the matrix is a constraint matrix and with right-hand side $b \in \{-1, +1\}^m$. For the first algorithm, the parameters were the size k of the deletion size of a blocker and the number w_- of negative right-hand sides, while in the second algorithm the parameters were k and the number w_+ of positive right-hand sides. The question naturally arises: Is it possible to find an FPT algorithm for $b \in \{-1, 0, +1\}^m$?

The last question is a natural simple extension for the MinFB. We make the following observation. Let a be a 0-arc, then it can be replaced by two consecutive arcs a_- and a_+ with weights -1 and $+1$, respectively (see Figure 5.1). Indeed, by this change of arcs, the

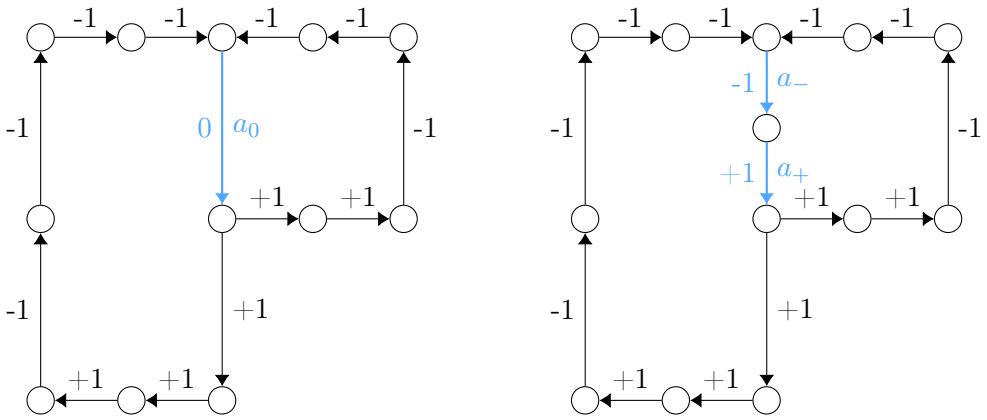

Figure 5.1: *Example of an instance* (G, k) *of the* MinFB *when* $b \in \{-1, 0, +1\}^m$*, with a* 0*-arc* a_0 *(left), and the instance* (G', k) *for* $b \in \{-1, +1\}^m$*, after replacing* a_0 *with arcs* a_- *and* a_+*, with associated weights* -1 *and* $+1$*, respectively (right). If we delete* a_0*, then the two negative cycles of* G *are gone. Analogous if we delete* a_- *or* a_+ *from* G'*; observe that we only require to eliminate one of them to get a graph without negative cycles.*

weight of all walks and cycles remain the same. Using this transformation, we would like to consider the algorithms from the last two sections. Note that the parameters w_- and w_+ are both increased simultaneously by the same number. Let (G, k) be an instance when $b \in \{-1, 0, +1\}^m$ and let (G', k) be the corresponding instance that we obtained with the modification, where $b \in \{-1, +1\}^m$. It is not difficult to see that for any optimal solution of (G, k) we can find a solution of (G', k) having the same size, and viceversa. The only problem derives when we have *too many* 0-arcs, since it increases the running time.

By last observations, we obtain the following straightforward results, derived from Theorems 5.2 and 5.8, respectively.

Corollary 5.9. *There is an algorithm that solves* MinFB *problem for systems* $\mathcal{S}$ *of* m *difference constraints over* n *variables and right-hand sides* $b \in \{-1, 0, +1\}^m$ *in time* $(k+1)(b_-+b_0)^{k+1}\mathcal{O}(nm)$*, when parameterized by* k *the size of the blocker, by* b_- *the number of negative entries of* b*, and by* b_0 *be the number of zero-entries of* b*.*

Corollary 5.10. *There is an algorithm that solves* MinFB *for systems* $\mathcal{S}$ *of* m *difference constraints over* n *variables and right-hand sides* $b \in \{-1, 0, +1\}^m$ *in time* $2^{\mathcal{O}\left(k^3+b_++b_0+k\log(b_++b_0)\right)} \cdot n^{\mathcal{O}(1)}$*, when parameterized by* k *the size of the blocker, by* b_+ *the number of positive entries of* b*, and by* b_0 *be the number of zero-entries of* b*.*

Nonetheless, we remark that the running time of these algorithms is not always good if the parameter b_0 is large. Then, it would be necessary to find other parameters to get a faster FPT algorithm for this case.

Chapter 6

The MinFB and the pathwidth

We are going to consider another parameter in this section: the pathwidth. We will show that using this parameter, the MinFB is NP-hard, even when the pathwith is smaller than 6. This is somehow a surprising result considering that Marthe Bonamy et. al. [22] gave an algorithm that solves DFAS (a particular case of MinFB) for constraint matrices of bounded treewidth. Recall that a matrix with bounded pathwidth is a subclass of matrices with bounded treewidth.

The main result of this chapter is the following.

Theorem 6.1. *The* MinFB *problem is NP-hard even for constraint matrices of pathwidth 6.*

To this end, we reduce Partition to Negative DFAS in digraphs whose underlying undirected graph has pathwidth at most 6. Richard M. Karp [20] showed that Partition is NP-complete.

Partition

Input: A set $\mathcal{A} = \{a_1, \ldots, a_n\}$ of positive integers.

Task: Find $\mathcal{A}' \subset \mathcal{A}$ so that $\sum_{a_i \in \mathcal{A}'} a_i = \sum_{a_i \in \mathcal{A} \setminus \mathcal{A}'} a_i$ or decide that no such set exists.
Equivalently: Let $A = \sum_{i=1}^{n} a_i$. Find $\mathcal{A}' \subset \mathcal{A}$ such that $\mathcal{A}'$ and $\mathcal{A} \setminus \mathcal{A}'$ each sum up to $\frac{A}{2}$.

Proof of Theorem 6.1. Starting from a Partition instance $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ of positive integers, we construct a Negative DFAS instance consisting of a digraph G with arc weights $w : A(G) \to \mathbb{Z}$ and some $k \in \mathbb{Z}_+$, $k \geq 0$. Afterwards, we argue why (G, w, k) has a solution if and only if $\mathcal{A}$ has one.

For every number $a_i \in \mathcal{A}$ construct a gadget G_i as it is shown in Figure 6.1, that is, let $V^{(i)} = \{s_i^{(j)}, t_i^{(j)}, x_i^{(j)}, y_i^{(j)} | j = 1, 2\}$ be the set of vertices in G_i. There are three different types of arcs. The first arcs set $A_1^i = \{(x_i^{(j)}, y_i^{(j)}) | j = 1, 2\}$ with arc weight $-a_i$, contains the arcs we will consider for deletion later. The second set is $A_2^i = \{(y_i^{(j)}, x_i^{(j+1)}), (y_i^{(j)}, x_i^{(j-1)}) | j = 1, 2\}$ with arc weight 0, which enforces the deletion of arcs from the first arc set by inducing negative cycles. The last arc set is $A_3^i = \{(s_i^{(j)}, t_i^{(j)}), (s_i^{(j)}, x_i^{(j)}), (y_i^{(j)}, t_i^{(j)}) | j = 1, 2\}$ with arc weight 0, and this set connects the vertices s_i and t_i to the rest of the gadget.

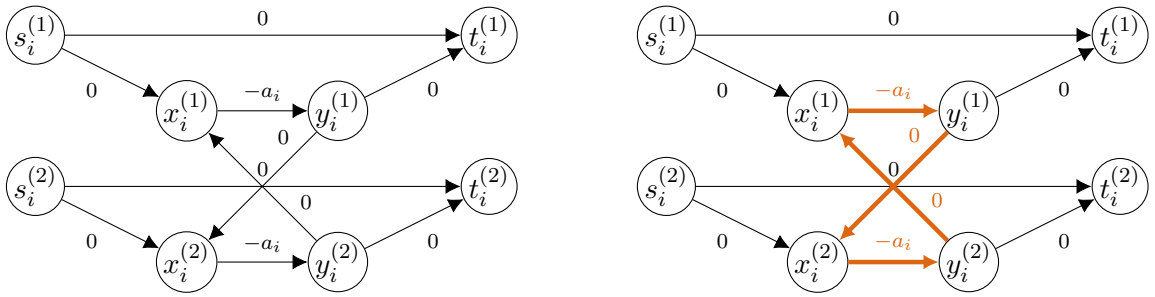

Figure 6.1: *The gadget graph G_i (left) and the negative cycle that G_i contains (right).*

The whole gadget G_i is then defined as $(V^{(i)}, A_1^{(i)} \cup A_2^{(i)} \cup A_3^{(i)})$. Finally, the instance graph G will be defined as the union of all gadgets $1 \leq i \leq n$, where $s_i^{(j)} = t_{i+1}^{(j)}$ for $j = 1, 2$. Additionally, add two vertices s and t with the arcs $(s, s_1^{(j)})$ and $(t_n^{(j)}, t)$ for $j = 1, 2$ with associated weight 0, and also add the arc (t, s) with arc weight $\frac{A}{2}$.

We have to show that $\mathcal{A}$ has a solution if and only if (G, w, n) has one.

Claim 6.2. *If $\mathcal{A}$ has a solution $\mathcal{A}'$, then there is a set S^* such that $G - S^*$ has no negative cycle, and $|S^*| \leq n$.*

Proof. Since $\mathcal{A}$ is a `Yes`-instance, then without loss of generality we can take the elements of $\mathcal{A}$ such that $a_1, a_2 ..., a_{\lfloor n/2 \rfloor} \in \mathcal{A}'$, that is, $\sum_{i=1}^{\lfloor n/2 \rfloor} a_i = \sum_{i=\lfloor n/2 \rfloor + 1}^{n} a_i = \frac{A}{2}$. Thus, we consider the graph as described with this indexing. We want to find S^* such that $G - S^*$ has no negative cycles, and $|S^*| \leq n$. Let us assume for simplicity that n is even.

Observe that if we eliminate one arc from each gadget, then we obtain a gadget that is free from negative cycles, because all gadgets have one negative cycle $x_i^{(1)} \to y_i^{(1)} \to x_i^{(2)} \to y_i^{(2)} \to x_i^{(1)}$ with weight $-2a_i$ (see Fig. 6.1). Nevertheless, it is not enough to delete an arc from the cycles in order to obtain a graph without negative cycles. In order to see this, suppose that we select from each gadget the arc $(x_i^{(2)}, y_i^{(2)})$ and we take $S = \{(x_i^{(2)}, y_i^{(2)}) | 1 \leq i \leq n\}$. Thus, a shortest cycle in $G - S$ is $C' = s \to s_1^{(1)} \to x_1^{(1)} \to y_1^{(1)} \to s_2^{(1)} \to x_2^{(1)} \to y_2^{(1)} \to s_3^{(1)} \to ... \to s_n^{(1)} \to x_n^{(1)} \to y_n^{(1)} \to t_n^{(1)} \to t \to s$, with associated weight

$$
\begin{aligned}
w(C') =& w(s, s_1^{(1)}) + w(s_1^{(1)}, x_1^{(1)}) + w(x_1^{(1)}, y_1^{(1)}) + w(y_1^{(1)}, s_2^{(1)}) + \\
& w(s_2^{(1)}, x_2^{(1)}) + w(x_2^{(1)}, y_2^{(1)}) + w(y_2^{(1)}, s_3^{(1)}) + ... + \\
& w(s_n^{(1)}, x_n^{(1)}) + w(x_n^{(1)}, y_n^{(1)}) + w(y_n^{(1)}, t_n^{(1)}) + w(t_n^{(1)}, t) + w(t, s) \\
=& 0 + 0 + (-a_1) + 0 + \\
& 0 + (-a_2) + 0 + ... + \\
& 0 + (-a_n) + 0 + 0 + \frac{A}{2} \\
=& -\sum_{i=1}^{n} a_i + \frac{A}{2} \\
=& -A + \frac{A}{2} \\
<& 0,
\end{aligned}
$$

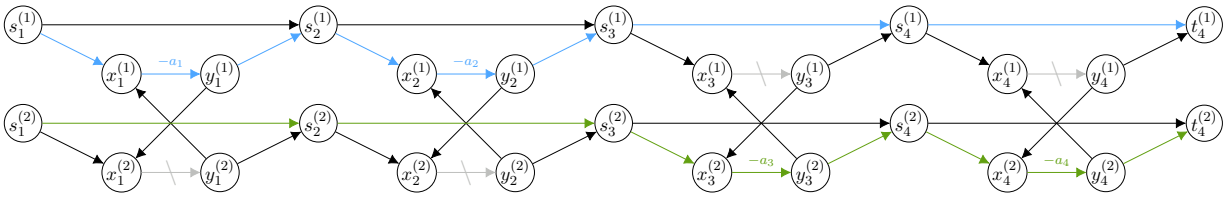

Figure 6.2: *Union of four gadgets after the deletion of negative cycles. Deleted arcs are shown in gray with a backslash. In this case,* $a_1, a_2 \in \mathcal{A}'$. *The blue path shows the shortest path from* $s_1^{(1)}$ *to* $t_4^{(1)}$, *and the green path shows the shortest path from* $s_1^{(2)}$ *to* $t_4^{(2)}$.

which shows that S is not a solution despite the fact that none of the gadgets have a negative cycle.

We consider the deletion of arcs such that for each $1 \leq i \leq n$ we eliminate arc $(x_i^{(2)}, y_i^{(2)})$ from gadget G_i if $a_i \in \mathcal{A}'$, and we eliminate the arc $(x_i^{(1)}, y_i^{(1)})$ otherwise (see Figure 6.2). So, we can define $S^* = \{(x_i^{(2)}, y_i^{(2)}) | a_i \in \mathcal{A}'\} \cup \{(x_i^{(1)}, y_i^{(1)}) | a_i \in \mathcal{A} \setminus \mathcal{A}'\}$.

We show that $G - S^*$ has no negative cycles. To do so, let P_1 be the shortest $s_1^{(1)} \to t_n^{(1)}$ path in $G - S^*$, which is defined as

$$\begin{aligned} P_1 =& s_1^{(1)} \to x_1^{(1)} \to y_1^{(1)} \to s_2^{(1)} \to x_2^{(1)} \to y_2^{(1)} \to s_3^{(1)} \to x_3^{(1)} \to y_3^{(1)} \to s_4^{(1)} \to \ldots \to \\ & s_{n/2-1}^{(1)} \to x_{n/2-1}^{(1)} \to y_{n/2-1}^{(1)} \to s_{n/2}^{(1)} \to x_{n/2}^{(1)} \to y_{n/2}^{(1)} \to s_{n/2+1}^{(1)} \to s_{n/2+2}^{(1)} \to \ldots \to \\ & s_{n-1}^{(1)} \to s_n^{(1)} \to t_n^{(1)}. \end{aligned}$$

Thus, a shortest cycle in $G - S^*$ can be defined as $C = s \to P_1 \to t \to s$. The weight of C is

$$\begin{aligned} w(C) =& w(s, s_1^{(1)}) + w(s_1^{(1)}, x_1^{(1)}) + w(x_1^{(1)}, y_1^{(1)}) + w(y_1^{(1)}, s_2^{(1)}) + \ldots + \\ & w(s_{n/2-1}^{(1)}, x_{n/2-1}^{(1)}) + w(x_{n/2-1}^{(1)}, y_{n/2-1}^{(1)}) + w(y_{n/2-1}^{(1)}, s_{n/2}^{(1)}) + \\ & w(s_{n/2}^{(1)}, x_{n/2}^{(1)}) + w(x_{n/2}^{(1)}, y_{n/2}^{(1)}) + w(y_{n/2}^{(1)}, s_{n/2+1}^{(1)}) + w(s_{n/2+1}^{(1)}, s_{n/2+2}^{(1)}) + \ldots + \\ & w(s_{n-1}^{(1)}, s_n^{(1)}) + w(s_n^{(1)}, t_n^{(1)}) + w(t_n^{(1)}, t) + w(t, s) \\ =& 0 + 0 + (-a_1) + 0 + \ldots + \\ & 0 + (-a_{n/2-1}) + 0 + \\ & 0 + (-a_{n/2}) + 0 + 0 + \ldots + \\ & 0 + 0 + 0 + \frac{A}{2} \\ =& -\sum_{i=1}^{n/2} a_i + \frac{A}{2} \\ =& -\sum_{a_i \in \mathcal{A}'} a_i + \frac{A}{2} \\ =& -A/2 + A/2 \\ =& 0. \end{aligned}$$

Therefore, since C is a shortest cycle, there are no negative cycles in $G - S^*$. As the weight of C is zero, by adding any other $(x_i^{(j)}, y_i^{(j)})$-arc we will get a negative cycle, for $1 \leq i \leq n$,

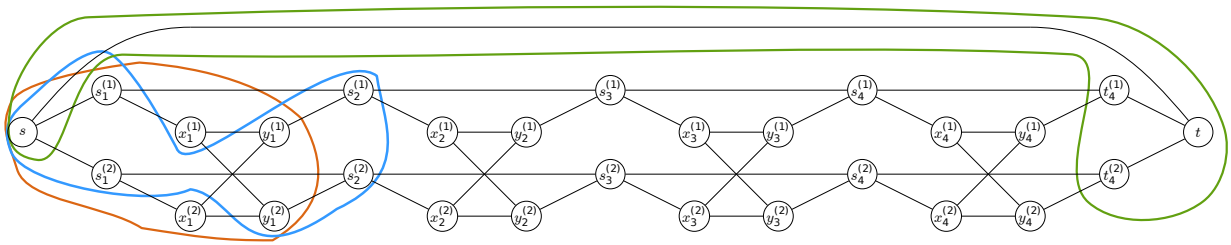

Figure 6.3: *Example of underlying graph with four gadgets. Bags B_1, B_2 and B_9 of its path decomposition are exhibited, where*

$$B_1 = \{s, s_1^{(1)}, s_1^{(2)}, x_1^{(1)}, x_1^{(2)}, y_1^{(1)}, y_1^{(2)}\},\ B_2 = \{s, s_1^{(1)}, s_1^{(2)}, y_1^{(1)}, y_1^{(2)}, t_1^{(1)}, t_1^{(2)}\},$$
$$B_3 = \{s, s_2^{(1)}, s_2^{(2)}, x_2^{(1)}, x_2^{(2)}, y_2^{(1)}, y_2^{(2)}\},\ B_4 = \{s, s_2^{(1)}, s_2^{(2)}, y_2^{(1)}, y_2^{(2)}, t_2^{(1)}, t_2^{(2)}\},$$
$$B_5 = \{s, s_3^{(1)}, s_3^{(2)}, x_3^{(1)}, x_3^{(2)}, y_3^{(1)}, y_3^{(2)}\},\ B_6 = \{s, s_3^{(1)}, s_3^{(2)}, y_3^{(1)}, y_3^{(2)}, t_3^{(1)}, t_3^{(2)}\},$$
$$B_7 = \{s, s_4^{(1)}, s_4^{(2)}, x_4^{(1)}, x_4^{(2)}, y_4^{(1)}, y_4^{(2)}\},\ B_8 = \{s, s_4^{(1)}, s_4^{(2)}, y_4^{(1)}, y_4^{(2)}, t_4^{(1)}, t_4^{(2)}\},$$
$$B_9 = \{s, t_4^{(1)}, t_4^{(2)}, t\}.$$

$j = 1, 2$. Finally, we obtained a set of n arcs such that its deletion give us a graph with no negative cycles. ◁

Claim 6.3. *If (G, w, n) has a solution S^*, then, $\mathcal{A}$ has a solution $\mathcal{A}'$*

Proof. Since $G - S^*$ has no negative cycles, then the solution looks like we described for Figure 6.2 in the last claim. Additionally, we get a corresponding partition $(\mathcal{A}', \mathcal{A} \setminus \mathcal{A}')$ considering $\mathcal{A}' = \{a_i | (x_i^{(2)}, y_i^{(2)}) \in S^*\}$. Let $P^{(1)}$ be a shortest $s_1^{(1)} \to t_n^{(1)}$-path and $P^{(2)}$ be a shortest $s_1^{(2)} \to t_n^{(2)}$-path. Now, since there are no negative cycles in $G - S^*$, the weight of $P^{(1)}$ and $P^{(2)}$ are at most $\frac{A}{2}$, that is, $\sum_{a_i \in \mathcal{A}'} \leq \frac{A}{2}$ and $\sum_{a_i \in \mathcal{A} \setminus \mathcal{A}'} \leq \frac{A}{2}$. Since $\sum_{a_i \in \mathcal{A}} = A$, we finally get that $\sum_{a_i \in \mathcal{A}'} = \sum_{a_i \in \mathcal{A} \setminus \mathcal{A}'} = \frac{A}{2}$ and we conclude that $\mathcal{A}'$ is a solution. ◁

This completes the reduction from Partition to Negative DFAS and therefore shows the NP-hardness of Negative DFAS. It only remains to bound the pathwidth of the generated instances.

We show that the underlying graph of G has pathwidth at most 6, by providing a path decomposition $(P, \mathcal{B})$ of G of width 6. Let P be the path on $2n + 1$ vertices, corresponding to bags $B_1, ..., B_{2n+1} \in \mathcal{B}$ (see Figure 6.3 as an example of P):

- $B_{2i-1} = \{s, s_i^{(1)}, s_i^{(2)}, x_i^{(1)}, x_i^{(2)}, y_i^{(1)}, y_i^{(2)}\}$ for $i = 1, ...n$
- $B_{2i} = \{s, s_i^{(1)}, s_i^{(2)}, y_i^{(1)}, y_i^{(2)}, t_i^{(1)}, t_i^{(2)}\}$ for $i = 1, ...n$
- $B_{2n+1} = \{s, t_n^{(1)}, t_n^{(2)}, t\}$

It is not difficult to see that this is a path decomposition for the underlying graph of G with width 6. □

To conclude this chapter, Theorem 6.1 provides us the following corollary by applying Remark 2.3.

Corollary 6.4. *The* MinFB *problem is NP-hard even for constraint matrices of treewidth 6.*

Chapter 7

Conclusions

Our results, summarized in Table 7.1, provide an example of the FPT algorithms applied to the Minimum Feasibility Blocker problem. Even though some results are focused on simple instances of the problem, this could give an insight into how FPT algorithms work. In general, determining the appropriate parameter is a tough task. In our case, changing the values of b from $\{-1, 1\}$ to $\{-1, 0, 1\}$ is not providing a benefit for the running time when we add the parameter b_0. Thus, we may include another parameter or a completely new set of them. As an example, it will be interesting to verify if the MinFB is FPT tractable with the parameters pathwidth and k.

Table 7.1: *Summary of the fixed parameter results for the* Min Feasibility Blocker.

MinFB estructure	**Parameters**[1]	**Complexity Result**
$A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^n$	k	W[1]-hard (Thm. 4.1)
Difference constraint matrix, $b \in \{-1, +1\}^m$	$k + b_-$	• FPT (Thm. 5.2) • Does not admit a polynomial compression assuming NP $\not\subseteq$ coNP/poly (Thm. 4.2)
Difference constraint matrix, $b \in \{-1, +1\}^m$	$k + b_+$	FPT (Thm. 5.8)
Difference constraint matrix, $b \in \{-1, 0, +1\}^m$	$k + b_+ + b_0$	FPT (Cor. 5.9)
Difference constraint matrix, $b \in \{-1, 0, +1\}$	$k + b_+ + b_0$	FPT (Cor.5.10)
$A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^n$	Pathwidth	NP-hard (Thm. 6.1)
$A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^n$	Treewidth	NP-hard (Cor. 6.4)

Another open question is whether the MinFB problem admits or not a polynomial compression assuming NP $\not\subseteq$ coNP/poly, when it is parameterized with $k+b_+$. Recall that when the parameters are $k + b_-$ no such compression exists. Additionally, it is interesting to know which parameters

[1] k is the maximum size of the blocker, and b_-, b_+ and b_0 are the number of entries in the vector b that are negative, positive and zero, respectively.

can be used to find an FPT algorithm when b has integer values.

Finally, we recall the importance of FPT algorithms. They indeed find an optimal solution in polynomial time given a bounded parameter. This is different from other approaches, when we only get approximated solutions, as we described in Chapter 3.

Bibliography

[1] Jayaram K Sankaran. A note on resolving infeasibility in linear programs by constraint relaxation. *Operations Research Letters*, 13(1):19–20, 1993.

[2] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2. Springer Science & Business Media, 2012.

[3] Jörg Flum and Martin Grohe. Parameterized complexity theory. *Texts in Theoretical Computer Science An EATCS Series*, 2006.

[4] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*, volume 4. Springer, 2015.

[5] Robert Ganian, Petr Hliněný, Joachim Kneis, Alexander Langer, Jan Obdržálek, and Peter Rossmanith. Digraph width measures in parameterized algorithmics. *Discrete applied mathematics*, 168:88–107, 2014.

[6] Alexander Göke, Lydia Mirabel Mendoza Cadena, and Matthias Mnich. Resolving infeasibility of linear systems: A parameterized approach. In Bart M. P. Jansen and Jan Arne Telle, editors, *14th International Symposium on Parameterized and Exact Computation (IPEC 2019)*, volume 148 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[7] Jørgen Bang-Jensen and Gregory Z Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer Science & Business Media, 2008.

[8] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[9] Stefan Kratsch and Magnus Wahlström. Two edge modification problems without polynomial kernels. *Discrete Optimization*, 10(3):193–199, 2013.

[10] John W Chinneck. An effective polynomial-time heuristic for the minimum-cardinality IIS set-covering problem. *Annals of Mathematics and Artificial Intelligence*, 17(1):127–144, 1996.

[11] Edoardo Amaldi, Viggo Kann, et al. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209(1):237–260, 1998.

[12] Gianni Codato and Matteo Fischetti. Combinatorial Benders' cuts for mixed-integer linear programming. *Operations Research*, 54(4):756–766, 2006.

[13] Mokhtar S Bazaraa, John J Jarvis, and Hanif D Sherali. *Linear Programming and Network Flows*. John Wiley & Sons, 2011.

[14] Marc E Pfetsch. Branch-and-cut for the maximum feasible subsystem problem. *SIAM Journal on Optimization*, 19(1):21–38, 2008.

[15] Sylvain Guillemot. FPT algorithms for path-transversal and cycle-transversal problems. *Discrete Optimization*, 8(1):61–71, 2011.

[16] Till Fluschnik, Danny Hermelin, André Nichterlein, and Rolf Niedermeier. Fractals for kernelization lower bounds. *SIAM Journal on Discrete Mathematics*, 32(1):656–681, 2018.

[17] Petr A Golovach and Dimitrios M Thilikos. Paths of bounded length and their cuts: Parameterized complexity and algorithms. *Discrete Optimization*, 8(1):72–86, 2011.

[18] Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24. Springer Science & Business Media, 2003.

[19] Rajesh Chitnis, Marek Cygan, Mohammataghi Hajiaghayi, and Dániel Marx. Directed subset feedback vertex set is fixed-parameter tractable. *ACM Transactions on Algorithms (TALG)*, 11(4):1–28, 2015.

[20] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer, 1972.

[21] Meirav Zehavi. Mixing color coding-related techniques. In *Algorithms-ESA 2015*, pages 1037–1049. Springer, 2015.

[22] Marthe Bonamy, Łukasz Kowalik, Jesper Nederlof, Michał Pilipczuk, Arkadiusz Socała, and Marcin Wrochna. On directed feedback vertex set parameterized by treewidth. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 65–78. Springer, 2018.