BACHELOR THESIS

Richard Hladík

Combinatorial Algorithms for Flow Problems

Computer Science Institute of Charles University

Supervisor of the bachelor thesis: Mgr. Martin Koutecký, Ph.D.
Study programme: Computer Science
Study branch: General Computer Science

Prague 2021

I am extremely grateful to my supervisor Mgr. Martin Koutecký, Ph.D. for all his invaluable help and his supportiveness during our work on this project and beyond. I would also like to thank Swati Gupta, Steffen Borgwardt and Sebastian Pokutta for the help they provided over email. Lastly, I thank my family, as well as my friends Tom and Pája, for their constant support and patience.

Title: Combinatorial Algorithms for Flow Problems

Author: Richard Hladík

Institute: Computer Science Institute of Charles University

Supervisor: Mgr. Martin Koutecký, Ph.D., Computer Science Institute of Charles University

Abstract: The multicommodity flow problem (MCF) and the length-bounded flow problem (LBF) are two generalisations of the maximum flow problem. Both can be solved using linear programming and approximated using fully polynomial-time approximation schemes (FPTASs). However, there are no known algorithms for them that are at the same time 1) exact, 2) polynomial, and 3) combinatorial and/or not relying on general methods like linear programming. Multicommodity flow is sometimes called "the easiest problem with no combinatorial algorithm". In this thesis, we summarise problem-specific as well as general methods used to solve these problems. We propose two new combinatorial algorithms, one based on the Frank-Wolfe method for convex optimisation (for MCF and LBF), and the other one based on most helpful cycle cancelling (for MCF), and prove that in networks with polynomial demands they both run in $\mathrm{poly}(\textit{input size}, 1/\varepsilon)$ time. We also present some results from polyhedral theory, examining the circuits of MCF and LBF polyhedra. On the one hand, we prove that the existence of a circuit-like set consisting of vectors of small norm would make both algorithms nearly-exact (i.e., with $\mathcal{O}(\log 1/\varepsilon)$ convergence). On the other hand, we prove that exponential circuits exist for both MCF and LBF. The existence of a circuit-like set other than the circuit set which only contains small vectors remains an open question.

Keywords: algorithm, flow, combinatorial, frank-wolfe, most helpful cycles

Název práce: Kombinatorické algoritmy pro tokové problémy

Autor: Richard Hladík

Institut: Informatický ústav Univerzity Karlovy

Vedoucí práce: Mgr. Martin Koutecký, Ph.D., Informatický ústav Univerzity Karlovy

Abstrakt: Problém multikomoditního toku (MCF) a problém K-omezeného toku (LBF) jsou dvě zobecnění problému maximálního toku. Oba problémy jdou řešit pomocí lineárního programování a aproximovat plně polynomiálními aproximačními schématy (FPTAS). Není však pro ně znám žádný algoritmus, který je zároveň 1) exaktní, 2) polynomiální a 3) kombinatorický a/nebo nepoužívající obecné metody jako lineární programování. O multikomoditním toku se někdy mluví jako o „nejsnazším problému, který nemá kombinatorický algoritmus“. V této práci shrnujeme specializované i obecné metody pro řešení obou problémů. Přinášíme dva nové kombinatorické algoritmy, první založený na metodě Frank-Wolfe pro konvexní optimalizaci (pro MCF i LBF), druhý založený na most helpful cycle cancelling (pro MCF), a dokazujeme, že v sítích s polynomiálními poptávkami oba algoritmy běží v čase $\mathrm{poly}(\textit{délka vstupu}, 1/\varepsilon)$. Také přinášíme výsledky v polyhedrální teorii, zejména v souvislosti s kružnicemi MCF- a LBF-mnohostěnů. Na jednu stranu dokazujeme, že existence množiny zobecňující množinu kružnic sestávající se pouze z vektorů malé normy by už zaručila „skoro-exaktnost“ obou algoritmů (resp. konvergenci v $\mathcal{O}(\log 1/\varepsilon)$ krocích). Na druhou stranu dokazujeme existenci exponenciálně velkých kružnic pro MCF i LBF. Existence množiny zobecňující množinu kružnic jiné než množina kružnic, která obsahuje pouze malé vektory, zůstává otevřenou otázkou.

Klíčová slova: algoritmus, tok, kombinatorický, frank-wolfe, most helpful cycles

Contents

Introduction

Network flows is one of the most prominent fields of combinatorial optimisation [AMO88]. Flow networks are used to model real-world situations in which a commodity (oil, network packets, electricity, ...) is transported in a network consisting of vertices (junctions, routers, logical gates, ...) and edges (pipes, networking cables, wires, ...) connecting them. Usually each edge has a *capacity* signifying the amount of the commodity that can be transported through it. There are usually also two special vertices, the *source* and the *sink*, in which the commodity is created or consumed, respectively. In all the other vertices, the amount of the commodity coming in must equal the amount of the commodity going out (the so-called *flow conservation property*). A *flow* then determines the amount of the commodity transported by each edge so that the assumptions on capacities and flow conservation are met.

The most famous of network flow problems is the maximum flow problem [FF56], in which the task is to find a flow transporting the maximum amount of one commodity from the source to the sink. In this thesis, we study two generalisations of this problem, namely the multicommodity flow problem (MCF) [FF58] and the length-bounded flow problem (LBF) [AK71]. At the first sight, they seem quite unrelated, but we will uncover multiple similarities between them.

In MCF, we have multiple commodities, each with its own source-sink pair and a demand on how much of this commodity must be transported. The task is to find a collection of flows for each commodity – a multicommodity flow or a *multiflow* for short – given that all commodities share a single network and its capacities.

In LBF, we have a single commodity (e.g., network packets, perishable goods) that we want to transport from the source to the sink as quickly as possible; namely, each unit of the commodity is allowed to travel through at most K edges before reaching the sink. Formally, the task is to find a flow satisfying the demand that can be decomposed into paths of length at most K.[1]

Both of these problems have been extensively studied, and several FPTASs exist for both the MCF family of problems [Fle99; GK07; Mąd10] and LBF [Vob16].[2] Both problems can also be solved exactly in polynomial time using linear programming (LP) [FF58]. However, it is not known whether there exists an exact algorithm which is of a combinatorial nature and/or which does not rely on LP or related methods. In fact, MCF has been sometimes called "the easiest problem with no combinatorial algorithm" [Ste12].

The term "combinatorial algorithm" is inherently ambiguous and has to do with taste and style; the term falls into similar categories as "elegant", "intuitive" or "beautiful"; and "I know it when I see one" is a valid and often-heard answer to the question of what constitutes a combinatorial algorithm [VO21]. One possible

[1]Although we talk about units of goods for simplicity, the amount of flow on each path does not have to be integral. That is, LBF is still a continuous problem, not discrete.

[2]An FPTAS, or a fully polynomial-time approximation scheme, is, for a maximisation problem, an approximation algorithm whose running time is, for a chosen approximation error ε, polynomial in the input size and $1/\varepsilon$, and which finds a solution whose objective value is within a factor of $(1 - \varepsilon)$ of the optimal one.

characterisation is that such algorithms rely on the combinatorial structure of the problem and their inner workings can be explained in terms of this structure. This is opposed to algorithms using general methods such as linear programming or convex optimisation, which indeed do solve the problem, but usually offer little meaningful insight into how the problem is "really" solved.

Consider, for example, the shortest paths problem in a graph with positive lengths. It can be certainly formulated as an LP – see, for example, Ahuja, Magnanti, and Orlin [AMO88, Section 11.7]. Since LP is solvable in polynomial time by e.g. the ellipsoid method or interior-point methods, we have obtained a polynomial algorithm for the shortest paths problem.

However, if we try to interpret it and understand why it works – as opposed to "just" proving its correctness from the correctness of LP – we will most likely fail miserably. For example, interior-point methods walk inside an n-dimensional polytope of feasible solutions, trying to find a point which is the farthest with respect to some predefined direction, but at the same time avoiding getting too close to polytope's boundary. It is not at all clear how these steps in a continous, n-dimensional world relate to the discrete problem of finding the shortest path in a graph.

On the other hand, the algorithms of Dijkstra or Bellman and Ford not only solve the same problem more efficiently, they are also much easier to understand, and one can not only see that they work, but also *why* they work.

The pursuit of combinatorial algorithms for problems so far only solvable via LP has been an important topic for a at least a few decades now:

> *[The pursuit of combinatorial algorithms] has led to superior algorithms for a variety of optimization problems, including shortest path, maximum flow, minimum spanning tree, matching, and minimum cost flow.* —Kevin D. Wayne [Way02]

Finding combinatorial algorithms for MCF and LBF is important because they can leverage the specific structure of the problem, and are therefore likely to be faster than general methods. Any such algorithm will also necessarily provide new insights into the problem structure.

On the other hand, any negative results help us better understand what makes these flow problems so different from other generalisations which still allow combinatorial algorithms, such as generalised flow or balanced flow. Also, they force us to refine what we mean by a "combinatorial algorithm" – it is one thing to show such an algorithm, and another thing entirely to (try to) *disprove* its existence. An "I know it when I see it" approach is not really helpful in this case if there is nothing to *see.*

Our Contributions

In this thesis, we summarise problem-specific as well as general methods used to solve MCF and LBF, and interpret some of them combinatorially. In particular, we review several variants of the Frank-Wolfe algorithm (FW) that is used for convex optimisation, and interpret their operation with respect to our specific problems. We also propose a combinatorial algorithm for MCF based on Weintraub's combinatorial algorithm for the minimum-cost maximum flow problem.

Additionally, we conjecture a geometric property of the MCF and LBF polyhedra, under which both methods converge in time weakly polynomial in input size and $\mathcal{O}(\log 1/\varepsilon)$, and, in the case of FW, the demands. (The convergence rate of $\log(1/\varepsilon)$ is also referred to as *linear convergence.*)

Algorithms from the Frank-Wolfe family are in themselves quite simple and the steps they make admit a combinatorial interpretation, which makes them a promising candidate for a general method of constructing combinatorial algorithms. However, until recently, there have been several obstacles to this, which explains why there have been few such attempts in this area. Vanilla FW has provably at best a $\Theta(1/\varepsilon)$-style convergence in the general case, and the fast convergence of improved FW variants in a general enough setting was shown only recently [LJ15]. Even then, FW requires the problem formulation to satisfy certain conditions; their verification for our problems is also a part of our contribution.

In this work, we show that the Away-Steps Frank-Wolfe algorithm applied to a penalty-based formulation of MCF and LBF reduces to iteratively solving the shortest path problem with weights changing based on the current flow, and that it achieves a penalty of ε in time polynomial in the input size, $1/\varepsilon$ and the sum of demands.

We also propose a combinatorial algorithm for the penalty-based formulation of MCF based on the most helpful cycle augmentation algorithm for the minimum-cost maximum flow problem; from now on, we refer to this algorithm as Helpful-MCF. We prove that Helpful-MCF also achieves a penalty of ε in time polynomial in the input size, $1/\varepsilon$ and the sum of demands, although experimental results suggest that the algorithm in fact converges linearly on public instances.

Although both of these algorithms come from a completely different background, we have, to our surprise, discovered that their linear convergence would follow from the same structural properties of the MCF and LBF polyhedra. This dependence is depicted in Figure 1. The most important is the notion of what we call the *inflation rate*, a property of the polyhedron closely related to the so-called Hoffman constant. Intuitively, if the inflation rate is small, it means that if we "inflate" the polyhedron by increasing the right-hand side of its inequality constraints, the newly added points of the inflated polyhedron cannot lie very far from the original polyhedron.

Assuming the inflation rate is polynomial in input size, we directly prove the linear convergence of Helpful-MCF. We also notice that the bound on the inflation rate is equivalent to the so-called Hölder condition (also known as the Hölder error bound, HEB) holding with appropriate parameters for our FW formulation, which would guarantee a linear convergence of the FW algorithm (although polynomial in the demands). In short, HEB states that solutions with a near-optimal penalty value cannot lie very far from some optimal solution.

Although we do not provide explicit bounds on the inflation rate, we show that it can be lower bounded by the norm of the *circuits* of the MCF matrix, where circuits are certain integer vectors associated with a polyhedron's constraint matrix. They have many interesting properties; for example, the set of circuits of a matrix contains all possible edge directions for an arbitrary choice of the right-hand side. They are also a *universal test set*, meaning that whenever we are at a point $\mathbf{x} \in P$ in the polyhedron that is suboptimal with respect to a linear[3] cost function, there always exists a suitably scaled circuit $\varepsilon\mathbf{g}$ such that $\mathbf{x}+\varepsilon\mathbf{g} \in P$ and the cost improves. Perhaps surprisingly, we show that exponential-sized circuits exist for both MCF and LBF. However, the bound on the inflation rate would be implied by any universal test set (actually, even weaker conditions suffice), and the existence of such a set is an open question.

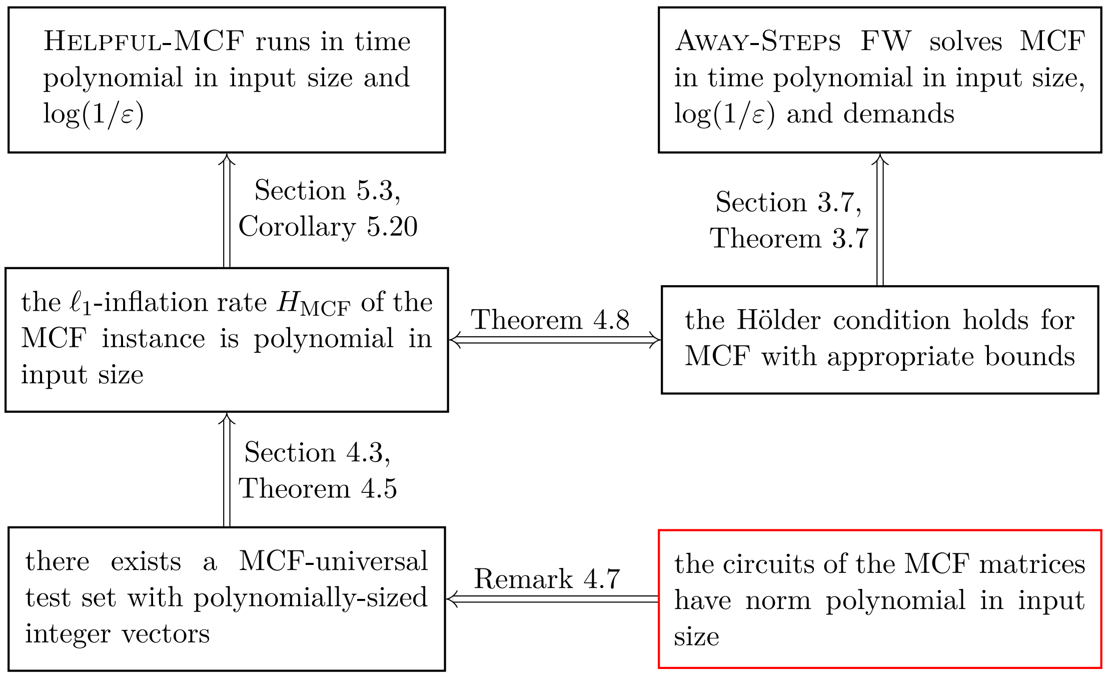
Helpful-MCF runs in time polynomial in input size and $\log(1/\varepsilon)$
Away-Steps FW solves MCF in time polynomial in input size, $\log(1/\varepsilon)$ and demands
Section 5.3, Corollary 5.20
Section 3.7, Theorem 3.7
the ℓ_1-inflation rate H_{MCF} of the MCF instance is polynomial in input size
Theorem 4.8
the Hölder condition holds for MCF with appropriate bounds
Section 4.3, Theorem 4.5
there exists a MCF-universal test set with polynomially-sized integer vectors
Remark 4.7
the circuits of the MCF matrices have norm polynomial in input size

Figure 1: An implication diagram showing the dependence between fast convergence of our algorithms and structural properties of MCF. The statement in red has been proven false, all other statements remain undecided. The diagram for LBF is practically the same, except that there is no Helpful-LBF algorithm.

Organisation of the Thesis

Overall, Chapters 1 and 2 introduce the needed terminology and notation and review existing results. Our original results are contained in Chapters 3 to 5; namely, each of the chapters first introduces the existing results and then builds upon them to present our contribution. The introduction of each chapter details the split between our and existing results, as well as the overall organisation of the chapter. In order to make the thesis more self-contained, we have reproved some simple and helpful statements, at times somewhat differently than was previously done in the literature.

In Chapter 1, we briefly review basic theory from flow networks, graph theory and optimisation, and introduce most of our notation and terminology.

In Chapter 2, we review the current state of the art. We begin by formulating the problems in Section 2.1, and restate them as linear and convex programs in Section 2.2. In Section 2.3, we review current approaches to MCF and LBF,

[3]Actually, even separable convex.

including several recent FPTASs. We also briefly mention general methods for linear and convex programming and discuss their application and interpretation with respect to MCF and LBF.

In Chapter 3, we describe the Frank-Wolfe algorithm used for convex optimisation, and interpret its operation when applied to the convex formulations of MCF and LBF. In Chapter 4, we define circuits and describe their basic properties, define the previously mentioned inflation rate and describe its relationship to circuits. Lastly, we show that exponential-sized circuits exist for MCF and LBF. In Chapter 5, we describe Weintraub's algorithm for solving the minimum-cost maximum flow problem based on cycle cancelling and present its adaptation for MCF. We analyse its convergence and show that it is weakly polynomial in the input size and $\mathcal{O}(\log(1/\varepsilon))$ if the inflation rate of the MCF polyhedron is small.

Related Work

In minimum-cost maximum flow (MCMF), each edge has a cost per each unit of flow transported, and the task is to find a maximum flow that has additionally the lowest possible cost. Many efficient and combinatorial approaches exist, for example, most helpful cycle cancelling [BT89], minimum mean cycle cancelling [GDL15] or cost scaling [GT90]. See also Shigeno, Iwata, and McCormick [SIM00].

Perhaps the most general flow problem that still admits a combinatorial algorithm is the *generalised minimum-cost maximum flow problem.* The setting is the same as in MCMF, but each edge also may have a positive multiplier called a *gain factor* $\gamma(e)$ associated with it. For each unit of flow that enters edge e, $\gamma(e)$ units exit. For a long time, the problem was only solvable by linear programming methods, or by algorithms that were combinatorial, but not polynomial. It was not until 2002 that Wayne [Way02] presented a polynomial combinatorial algorithm based on a generalisation of the idea of minimum mean cycle cancelling. A version of this problem with no costs is admittedly easier, but it still was not until 1991 that the first combinatorial polynomial algorithm appeared [GPT91], and the first combinatorial strongly polynomial algorithm was given only recently [Vég17; OV20].[4]

We postpone the area-specific related work into the respective relevant chapters and sections, where, after having discussed the needed context, it can be discussed in a more qualified manner. For example, all the related work for Frank-Wolfe is discussed in Chapter 3, while the circuit-related work appears in Chapter 4. Specifically, relevant modifications of MCF and LBF are discussed in Section 2.1, while Section 2.3 deals with the problem-specific approaches for MCF and LBF. Now we will only mention two results that bear the most similarity to our work.

[4]Slightly inaccurately, an algorithm runs in strongly polynomial time if it runs in time polynomial only in the number of numbers on the input, regardless of their size. If it is only polynomial in input's encoding length in bits, it is called *weakly polynomial* instead.

MCF and LBF

The formulation of MCF that does not have capacity constraints and penalises exceeding the capacity by a quadratic penalty instead is by no means new. The same formulations were used in the work of Schneur and Orlin [SO98] and Liu [Liu19]; the general idea of using a penalty in place of capacity constraints appeared in many other works, some of which are given in Section 2.3. As far as we know, the work of Liu is the only one besides ours that uses the Frank-Wolfe algorithm to construct a combinatorial algorithm.

Schneur and Orlin [SO98] use an approach similar to ours. Their algorithm starts with a multiflow that does not satisfy capacity constraints and iteratively improves it by changing it along cycles such that the penalty decreases. In each phase, it has a step size δ, and tries to improve the flow by changing it by δ along some cycle, until no such cycle exists in any commodity. Then, a next phase begins with $\delta := \delta/2$. This approach is in contrast with that of Helpful-MCF: while Helpful-MCF finds the most improving combination of cycle and step size in each step, this method of Schneur and Orlin [SO98] has the step size fixed and picks *any* cycle that is improving with respect to the current step size. Furthermore, the number of cycles found in each phase of their algorithm may be large – proportional to $1/\delta$ and the sum of demands in the worst case – and the total time complexity is polynomial in input size, $1/\varepsilon$ and the sum of demands.

Liu [Liu19] also uses an approach that resembles ours. The main difference is that while our approach finds the combination of the step size and cycle that decreases the penalty the most, his approach finds *any* improving cycle and then finds the best step size for this particular cycle. The author does not analyse the time complexity of this algorithm; we furthermore suspect that there exists an instance such that a bad choice of negative cycles may lead to very slow convergence to an optimum (or even no convergence), in a way similar to how the Ford-Fulkerson algorithm may not converge on an instance with irrational capacities. Additionally, the author applies the vanilla Frank-Wolfe algorithm to MCF in a formulation similar to ours, but again, analysis and important details are missing and the variant used is known to generally converge slowly. As a consequence of results presented in Chapter 3, this algorithm is at best polynomial in $1/\varepsilon$, the input size and demands, and the linear dependence on $1/\varepsilon$ inherent to this variant of Frank-Wolfe cannot be improved without problem-specific knowledge.

1. Preliminaries

In this chapter, we introduce terminology used throughout the thesis. Some more problem-related notation is introduced later in Section 2.2, especially in Section 2.2.3.

1.1 Linear-Algebraic Minimum, Notation

We write vectors in bold font (e.g., $\mathbf{x}$) and their entries in regular font (e.g., x_1, x_2). The coordinates of a vector may be indexed by any set, for example, $\mathbf{c} \in \mathbb{R}^E$ has an entry c_e for each $e \in E$. In all other respects, vector spaces $\mathbb{R}^E$ and $\mathbb{R}^n$ for $n = |E|$ behave identically. We will sometimes even assume that $E = \{1, \ldots, n\}$. Superscript is used for indexing, i.e., we may have a collection $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^K$ of vectors, then x^i_e is the e-th entry of the vector $\mathbf{x}^i$. We will also use the standard notation $\mathbf{x} \leq \mathbf{y}$ to signify that $x_i \leq y_i$ for all i; similarly for $\mathbf{x} = \mathbf{y}$, $\mathbf{x} < \mathbf{y}$, etc.

We will never distinguish between row and column vectors and treat all vectors as column vectors. The dot product of vectors $\mathbf{x}$ and $\mathbf{y}$ is denoted by $\mathbf{x} \cdot \mathbf{y}$, even if both vectors are column vectors.

Unless specified otherwise, we assume that matrices are integer. The i-th row of matrix $A \in \mathbb{Z}^{m \times n}$ is a vector $A_{i*} \in \mathbb{Z}^n$. Similarly, its j-th column is a vector $A_{*j} \in \mathbb{Z}^m$. Given a set $I \subseteq \{1, \ldots, m\}$ of row indices, A_I is the matrix created by leaving out the rows of A whose indices do not belong to I, and $A_{\overline{I}}$ is the matrix created by leaving out the rows of A whose indices *do* belong to I; similarly for vectors.

The *i-th unit vector* of $\mathbb{R}^n$ is the vector $\mathbf{e}^i$ defined as

$$(\mathbf{e}^i)_j = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

The *identity matrix* $I_n \in \mathbb{R}^{n \times n}$ is a matrix with $I_{i*} = \mathbf{e}^i$.

The *kernel* of a matrix $A \in \mathbb{R}^{m \times n}$ is the set $\ker(A) = \{\, \mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{0} \,\}$

Usually, $\|\cdot\|$ denotes an arbitrary norm, i.e., any function $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ satisfying, for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$:

- $\|\mathbf{x}\| \geq 0$ and $\|\mathbf{x}\| = 0$ only for $\mathbf{x} = \mathbf{0}$,
- $\|\alpha \mathbf{x}\| = |\alpha| \cdot \|\mathbf{x}\|$,
- $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$.

The last of the three properties is called the *triangle inequality.* $\|\cdot\|_1$ denotes the absolute-value norm $\|\mathbf{x}\|_1 = |x_1| + \cdots + |x_n|$ and $\|\cdot\|_2$ denotes the standard Euclidean norm $\|\mathbf{x}\|_2 = \sqrt{x_1^2 + \cdots + x_n^2}$.

A set $X \subseteq \mathbb{R}^n$ is *convex*, if for any $\mathbf{x}, \mathbf{y} \in X$ and $\alpha \in [0, 1]$, it holds that $(1 - \alpha)\mathbf{x} + \alpha\mathbf{y} \in X$, that is, for any two points in X, the line segment connecting them is also contained in X. A *convex hull* $\operatorname{conv}(X)$ of a set $X \subseteq \mathbb{R}^n$ is the smallest convex set (with respect to inclusion) containing X as a subset. A function $f : \mathbb{R}^n \to \mathbb{R}$ is *convex*, if the set $\{\, (\mathbf{x}, y) \in \mathbb{R}^{n+1} \mid y \geq f(\mathbf{x}) \,\}$ is convex. It is *concave* if $-f$ is convex.

A half-space is a set $\{\, \mathbf{x} \in \mathbb{R}^n \mid \mathbf{a} \cdot \mathbf{x} \leq b \,\}$ for $\mathbf{a} \in \mathbb{R}^n$, $b \in \mathbb{R}$. A *polyhedron* is any set $P \subseteq \mathbb{R}^n$ expressible as a finite intersection of half-spaces. A *polytope* is a bounded polyhedron; equivalently it is a set $Q \subseteq \mathbb{R}^n$ expressible as a convex hull of a finite set of points. Both polyhedra and polytopes are convex.

A *face* of a polytope $P = \{\, \mathbf{x} \in \mathbb{R}^n \mid B\mathbf{x} \le \mathbf{b} \,\}$ is a set of points $P' = \{\, \mathbf{x} \in \mathbb{R}^n \mid B_I\mathbf{x} = \mathbf{b}_I \wedge B_{\overline{I}}\mathbf{x} \le \mathbf{b}_{\overline{I}} \,\}$ where $I \subseteq \{1, \ldots, m\}$, i.e., we have changed some inequalities to equalities. It can be shown that a face is also a polyhedron and that a face of a face of P is also a face of P. The dimension of a face (and generally of any set of points) is defined as the dimension of the smallest affine space containing it. A face of dimension 0 is a *vertex*, a face of dimension 1 an *edge*, and a face of dimension $n-1$ a *facet*.

We use the notation $[x]_+$ to stand for $\max(0, x)$. We will also use it vectorwise, that is, $[\mathbf{x}]_+ := \mathbf{y}$ such that $y_i = [x_i]_+$ for all i.

We use $\mathrm{poly}(n)$ to stand for $n^{\mathcal{O}(1)}$; similarly $\mathrm{poly}(n, m, \ldots) = (nm\ldots)^{\mathcal{O}(1)}$.

1.2 Flows

Now we turn our attention to basic terminology of flow networks. We loosely follow the notation of Mareš [Mar17, Chapter 14].

A *network* is a tuple $G = (V, E, \mathbf{c})$ where (V, E) is a directed graph and $\mathbf{c} \in \mathbb{R}^E, \mathbf{c} \ge \mathbf{0}$ is the vector of *edge capacities*. Formally, V is a set of vertices and $E \subseteq \{\, (u, v) \mid u, v \in V \wedge u \ne v \,\}$ is a set of edges. We use the notation uv for the edge (u, v) and say that the edge goes from u to v and that it is adjacent to u and v.

Given (V, E), a vector $\mathbf{f} \in \mathbb{R}^E$ assigning real numbers to edges, and a vertex v, we define:

$$f_+(v) = \sum_{uv \in E} f_{uv}, \quad f_-(v) = \sum_{vu \in E} f_{vu}, \quad \mathbf{f}_\delta(v) = f_+(v) - f_-(v) \tag{1.1}$$

and call it the *inflow*, the *outflow* and the *excess flow* of v (with respect to $\mathbf{f}$), respectively.

Given (V, E) and vertices $s, t \in V$ (usually called *source* and *sink*), an *st-flow* is a vector $\mathbf{f} \in \mathbb{R}^E$ satisfying

$$\mathbf{f} \ge \mathbf{0}, \tag{1.2}$$

$$\forall v \in V \setminus \{s, t\} : \mathbf{f}_\delta(v) = 0, \tag{1.3}$$

that is, the flow is non-negative and does not emerge or vanish in any vertex except for the source and sink. To emphasise the nonnegativity of the flow, we will sometimes call it a *feasible flow* (as opposed to a hypothetical *infeasible flow*, which satisfies (1.3) but not (1.2)).

An *st*-flow $\mathbf{f}$ is *satisfying* (with respect to capacities $\mathbf{c}$) if it also satisfies

$$\mathbf{f} \le \mathbf{c}, \quad \text{i.e.,} \quad \forall e \in E : f_e \le c_e. \tag{1.4}$$

Constraints (1.2)–(1.4) are also called *flow nonnegativity constraints*, *flow conservation constraints* and *capacity constraints*, respectively. Whenever s and t are clear from the context, we shall use the terms *(feasible) flow* and *satisfying flow*. Note that it makes sense to talk about flows – as opposed to satisfying flows – even if no capacities are given, and we will sometimes do so.

Let us also remark that “flow” is sometimes used for what we call “satisfying flow”, and “pseudoflow” is used for what we call “(feasible) flow”; the terminology in the literature is however not consistent.

The *value* of the flow $\mathbf{f}$ is defined as $|\mathbf{f}| = -\mathbf{f}_\delta(s)$ and it can be shown that $|\mathbf{f}| = -\mathbf{f}_\delta(s) = \mathbf{f}_\delta(t)$. Informally, $|\mathbf{f}|$ measures the amount of goods transported from the source to the sink.

1.3 Circulations

Definition 1.1 (Circulation)**.** A *circulation* is a vector $\mathbf{\Delta} \in \mathbb{R}^E$ satisfying:

$$\forall v \in V : \mathbf{\Delta}_\delta(v) = 0,$$

i.e., for every vertex, the inflow is equal to the outflow. A *positive circulation* further satisfies $\mathbf{\Delta} \geq \mathbf{0}$.

Circulations can be regarded as a “neutral” variant of flows, since they do not have a source and a sink and satisfy flow conservation everywhere; we also allow them to be negative. As we will see later, if we write the flow conservation constraints (also including s and t, with $-\mathbf{f}_\delta(s) = \mathbf{f}_\delta(t) = d > 0$ for some d) in a matrix form, $A\mathbf{x} = \mathbf{a}$, then circulations are exactly the elements of $\ker(A)$, i.e., the solutions to $A\mathbf{x} = \mathbf{0}$.

Adding a circulation to a flow always results in a flow, unless the result would be negative on some edge. A circulation $\mathbf{\Delta}$ is called *feasible* with respect to flow $\mathbf{f}$ if $\mathbf{f} + \mathbf{\Delta}$ is a (feasible) flow, i.e., $\mathbf{f} + \mathbf{\Delta} \geq \mathbf{0}$.

Lemma 1.1. *Let $\mathbf{f}$, $\mathbf{f}^1$, $\mathbf{f}^2$ be st-flows, let $\mathbf{\Delta}$, $\mathbf{\Delta}^1$, $\mathbf{\Delta}^2$ be circulations and let $\alpha \in \mathbb{R}$. Then*

- *$\alpha \cdot \mathbf{\Delta}$, $\mathbf{\Delta}^1 + \mathbf{\Delta}^2$ and $\mathbf{0}$ are circulations;*
- *$\alpha \cdot \mathbf{f}$, $\mathbf{f}^1 + \mathbf{f}^2$ and $\mathbf{0}$ are st-flows and $|\alpha \cdot \mathbf{f}| = \alpha \cdot |\mathbf{f}|$, $|\mathbf{f}^1 + \mathbf{f}^2| = |\mathbf{f}^1| + |\mathbf{f}^2|$;*
- *if $\mathbf{f} + \mathbf{\Delta} \geq \mathbf{0}$, then $\mathbf{f} + \mathbf{\Delta}$ is an st-flow and $|\mathbf{f} + \mathbf{\Delta}| = |\mathbf{f}|$;*
- *if $|\mathbf{f}^1| = |\mathbf{f}^2|$, then $\mathbf{f}_1 - \mathbf{f}_2$ is a circulation; if further $\mathbf{f}^2 \leq \mathbf{f}^1$, then it is a positive circulation.*

Proof sketch. By definition of $\mathbf{x}_\delta$ in Equation (1.1), observe that $(\alpha \cdot \mathbf{x})_\delta(v) = \alpha \cdot \mathbf{x}_\delta(v)$ and $(\mathbf{x} + \mathbf{y})_\delta(v) = \mathbf{x}_\delta(v) + \mathbf{y}_\delta(v)$. The rest follows from a substitution into the definition of flows and circulations. □

1.4 Linear Programming

Linear programming is such a versatile technique that its study has throughout the years developed into a field of its own. In this thesis, we use it mostly as a theoretical and practical tool; see [Sch99] or [GLS12] for an in-depth exposition.

A *linear program* (LP) is any optimisation problem that can be expressed in the form:

Linear Programming (LP)

Input: $A \in \mathbb{R}^{m_A \times n}$, $\mathbf{a} \in R^{m_A}$, $B \in \mathbb{R}^{m_B \times n}$, $\mathbf{b} \in \mathbb{R}^{m_B}$, $\mathbf{c} \in \mathbb{R}^n$
Maximise: $\mathbf{cx}$ for $\mathbf{x} \in \mathbb{R}^n$
Subject to: $A\mathbf{x} = \mathbf{a}$
$B\mathbf{x} \leq \mathbf{b}$

Given an LP formulated in this way, its *feasible region* is a polyhedron $P = \{\, \mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{a} \wedge B\mathbf{x} \leq \mathbf{b} \,\}$.

An LP can be solved using a number of arithmetic operations which is polynomial in both n, m and the encoding length of $A, B, \mathbf{a}, \mathbf{b}, \mathbf{c}$ using for example the ellipsoid method [GLS12] or interior-point methods [NN94]. Such a complexity is known as *weakly polynomial*, because it depends on the magnitude of the numbers involved in the input. A strongly polynomial algorithm for LP, i.e., an algorithm using a number of arithmetic operations depending only on n and m, is currently unknown. Tardos [Tar86] gave an algorithm which is only weakly polynomial in the numbers of A, and Megiddo [Meg84] gave a strongly polynomial algorithm for LP if the dimension is a constant.

In all the linear programs in this thesis, A, B and $\mathbf{c}$ will be integer and $\mathbf{a}$ and $\mathbf{b}$ will be rational.

Although we will use LP quite sparingly, most of the algorithms presented in this thesis can also be viewed through the lens of LP (and it is often helpful to do so): each solution corresponds to a point in the polyhedron and the algorithms walk through the polyhedron, trying to find the best solution.

1.5 Conformality

The main reference for this section is [Onn10, Subsection 2.3.4].

Definition 1.2 (Sign-compatibility)**.** Vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ are *sign-compatible*, if they belong to the same orthant[1] of $\mathbb{R}^n$, that is, $x_i \cdot y_i \geq 0$ for all $1 \leq i \leq n$.

Definition 1.3 (Conformality, conformal sum)**.** Let $\sqsubseteq$ be a partial ordering on $\mathbb{R}^n$ defined as follows: $\mathbf{x} \sqsubseteq \mathbf{y}$ if and only if

$$\forall 1 \leq i \leq n, \quad x_i \cdot y_i \geq 0 \wedge |x_i| \leq |y_i|.$$

Informally, $\mathbf{x} \sqsubseteq \mathbf{y}$ if and only if $\mathbf{x}$ would fit in an n-dimensional box with two opposite corners in $\mathbf{0}$ and $\mathbf{y}$. We say that $\mathbf{x}$ is *conformal* to $\mathbf{y}$.

A finite sum $\mathbf{y} = \sum_i \mathbf{x}^i$ is called a *conformal sum* or a *conformal decomposition of* $\mathbf{y}$ if $\mathbf{x}^i \sqsubseteq \mathbf{y}$ for all i.

Fact 1.2. Conformality has the following properties:

a) For nonnegative vectors $\mathbf{x}, \mathbf{y} \geq \mathbf{0}$, saying $\mathbf{x} \sqsubseteq \mathbf{y}$ is equivalent to $\mathbf{x} \leq \mathbf{y}$. This is true, e.g., for flows.
b) For all $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{0} \sqsubseteq \mathbf{x}$ and $\mathbf{x} \sqsubseteq \mathbf{x}$.

[1] *Orthant* is a generalisation of the terms *quadrant* (in $\mathbb{R}^2$) and *octant* (in $\mathbb{R}^3$) for higher dimensions.

c) If $\mathbf{x} \sqsubseteq \mathbf{y} \sqsubseteq \mathbf{z}$, then $\mathbf{y} - \mathbf{x} \sqsubseteq \mathbf{z}$. This can be seen by considering the cases $z_i \geq 0$ and $z_i \leq 0$ for each $1 \leq i \leq n$. Specially, $\mathbf{z} - \mathbf{y} \sqsubseteq \mathbf{z}$.
d) If $\mathbf{\Delta}^2 \sqsubseteq \mathbf{\Delta}^1$ are circulations and $\mathbf{f}$ and $\mathbf{f} + \mathbf{\Delta}^1$ are feasible flows, then $\mathbf{f} + \mathbf{\Delta}^2$ is also a feasible flow. If $\mathbf{f}$ and $\mathbf{f} + \mathbf{\Delta}^1$ are satisfying flows, then $\mathbf{f} + \mathbf{\Delta}^2$ is also a satisfying flow.

We will revisit conformality in Chapter 4, where we will see its usefulness in discrete optimisation, particularly in the context of *circuits*.

1.6 Walks, Paths, Cycles

Definition 1.4 (Directed path, walk, cycle)**.** A *(directed) walk* in graph (V, E) is a sequence $w = (v_0v_1, v_1v_2, \dots, v_{\ell-1}v_\ell)$ such that $v_0, \dots, v_\ell \in V$, $v_0v_1, \dots, v_{\ell-1}v_\ell \in E$. Vertices v_0 and v_ℓ are called the *start vertex* and *end vertex* of w, and we can also say that w is a v_0v_ℓ-walk. The *length* of the walk is then defined as $|w| = \ell$.

A *(directed) path* is a walk $p = (v_0v_1, v_1v_2, \dots, v_{\ell-1}v_\ell)$ such that all v_i are pairwise distinct (which also implies all edges are pairwise distinct).

A *(directed) cycle* is a walk $C = (v_0v_1, v_1v_2, \dots, v_{\ell-1}v_\ell)$ such that $|C| \geq 3$ and all v_i are pairwise distinct, with the only exception being that $v_0 = v_\ell$.

Remark 1.3. It is a well-known fact that every st-walk can be simplified into an st-path only by removing a subsequence of its edges. It can be done by iteratively finding a subsequence of the walk that forms a cycle and removing it until there are no cycles left.

Although we will exclusively work with directed graphs throughout the thesis, we will sometimes need to use paths and cycles that "forget" edges' orientation. This notion is formalised in the following definition.

Definition 1.5 (Undirected path, walk, cycle)**.** An *undirected walk* is a sequence $w = (e_0, \dots, e_{\ell-1})$ such that $e_0, \dots, e_{\ell-1} \in E$ and there exist vertices $v_0, \dots, v_\ell$ such that either $e_i = v_iv_{i+1}$ or $e_i = v_{i+1}v_i$ for $0 \leq i \leq \ell - 1$. We may again say that w is an undirected v_0v_ℓ-walk.

An *undirected path* is an undirected walk p such that all v_i from the definition are pairwise distinct. Similarly, an *undirected cycle* is an undirected walk C such that $|C| \geq 3$ and all v_i from the definition are pairwise distinct except that $v_0 = v_\ell$.

Whenever we talk about a walk/path/cycle, we implicitly mean a directed walk/path/cycle, unless specified otherwise. Also note that a walk/path/cycle is automatically also an undirected walk/path/cycle, but not vice versa.

Without a knowledge of the underlying vertex sequence, the orientation of an undirected walk may be ambiguous – take for example an undirected path of length 1; on the other hand, a cycle has multiple representations, differing by which vertex they start with. For those reasons, we will almost completely abandon the previous definitions and instead use the following one, which takes orientation into account.

Definition 1.6 (Incidence vector)**.** Let $p = (e_0, \dots, e_{\ell-1})$ be an undirected path and $v_0, \dots, v_\ell$ its underlying vertex sequence. An *incidence vector* of p is a vector $\mathbf{p}$ such that:

$$p_e = \begin{cases} 1 & \text{if } e = v_i v_{i+1} \text{ for some } i \in \{0, \dots, \ell - 1\}, \\ -1 & \text{if } e = v_{i+1} v_i \text{ for some } i \in \{0, \dots, \ell - 1\}, \\ 0 & \text{otherwise.} \end{cases}$$

An incidence vector $\mathbf{C}$ of an undirected cycle C is defined in the same manner.

From now on, we will simply say that $\mathbf{p}$ is an (un)directed path and that $\mathbf{C}$ is an (un)directed cycle, only explicitly falling back to the sequence-based notation when needed.

Informally, the incidence vector is 0 for edges not on the path (cycle), 1 for edges that are passed in their "correct" direction, and -1 for edges that are passed in the opposite direction. Specifically, incidence vectors of directed paths and cycles are 0 and 1 for all edges.

The notion of incidence vectors can naturally be extended to other objects besides paths and cycles; for example, in Chapter 5, we talk about an incidence vector of a disjoint collection of undirected cycles, which is naturally defined as the sum of the incidence vectors of the individual cycles.

Of course, for some objects there does not exist a meaningful definition – that is why we did not define the incidence vector for general walks, since we would have to deal with things like multiple repetitions of the same edge or the occurrence of both uv and vu in the walk (which cannot happen with paths and cycles).

1.7 Weighted Paths and Cycles

Definition 1.7. A *weighted (st-)path* is an (st-)flow $\mathbf{f}$ such that $\mathbf{f} = w \cdot \mathbf{p}$ where $\mathbf{p}$ is an (st-)path and $w \in \mathbb{R}_0^+$; w is then called the *weight* of the weighted path. All path-related terms such as length are naturally extended to weighted paths; for example, $|w\mathbf{p}| = |\mathbf{p}|$.

Similarly, a *weighted cycle* is a circulation $\mathbf{\Delta} = w \cdot \mathbf{C}$ where $w \in \mathbb{R}_0^+$ and $\mathbf{C}$ is an incidence vector of an **undirected** cycle. A *positive weighted cycle* is a circulation $\mathbf{\Delta} = w \cdot \mathbf{C}$ such that $w \in \mathbb{R}_0^+$ and $\mathbf{C}$ is an incidence vector of a directed cycle; clearly $\mathbf{\Delta} \geq \mathbf{0}$.

Lemma 1.4 (Cycle decomposition of circulations)**.** *Every circulation $\mathbf{\Delta}$ can be expressed as a conformal sum of at most $|E|$ weighted cycles:*

$$\mathbf{\Delta} = \sum_{i=1}^{\ell} \alpha_i \mathbf{C}^i,$$

where $\ell \leq |E|$, $\alpha_i > 0$ and all $\mathbf{C}^i$ are different.

Proof sketch. We prove that z cycles suffice, where $z \leq |E|$ is the number of nonzero edges in $\mathbf{\Delta}$. We proceed by induction on z. If $z = 0$, we are done. Otherwise, one can show that we may always find an undirected cycle $\mathbf{C} \neq \mathbf{0}$ such that $\varepsilon\mathbf{C} \sqsubseteq \mathbf{\Delta}$ for some small $\varepsilon > 0$. (Key observation: due to flow conservation, each vertex that has flow coming in also must have flow coming out.)

Let $\alpha > 0$ be the maximum value such that $\mathbf{\Delta} - \alpha\mathbf{C} \sqsubseteq \mathbf{\Delta}$. Set $\mathbf{\Delta}' := \mathbf{\Delta} - \alpha\mathbf{C}$. Clearly, if $\Delta_e = 0$, then also $\Delta'_e = 0$. Due to the choice of α, there exists an edge e such that $\Delta_e \neq 0$ and $\Delta'_e = 0$. Therefore, we have reduced the number of nonzero edges at least by one and we may use the inductive hypothesis on $\mathbf{\Delta}'$.

Let $\mathbf{\Delta}' = \alpha_1\mathbf{C}_1 + \cdots + \alpha_\zeta\mathbf{C}_\zeta$ be the decomposition of $\mathbf{\Delta}'$ with $\zeta < z$. The sought decomposition is $\mathbf{\Delta} = \alpha_1\mathbf{C}_1 + \cdots + \alpha_\zeta\mathbf{C}_\zeta + \alpha\mathbf{C}$. Its conformality follows from Fact 1.2 and the fact that $\mathbf{\Delta}' \sqsubseteq \mathbf{\Delta}$ and $\alpha\mathbf{C} \sqsubseteq \mathbf{\Delta}$.

See [AMO88, Property 3.6] for a full proof. □

The realisation that if we express a positive vector as a conformal sum, all the summands must also be positive, leads to the following corollary:

Corollary 1.5. Every positive circulation $\mathbf{\Delta}$ can be expressed as a conformal sum of at most $|E|$ positive weighted cycles.

Remark 1.6. As we will see in Chapter 4, if we view the set of all satisfying flows as a polytope $P = \{\mathbf{f} \in \mathbb{R}^E \mid A\mathbf{f} = \mathbf{d} \wedge \mathbf{f} \geq \mathbf{0} \wedge \mathbf{f} \leq \mathbf{c}\}$ where A is the incidence matrix of the network $\mathcal{G}$ and $\mathbf{d}$ is the vector of per-vertex demands, then undirected cycles in $\mathcal{G}$ correspond to so-called *circuits* of A (also discussed in Chapter 4) and circulations in $\mathcal{G}$ correspond to vectors $\mathbf{x} \in \mathbb{R}^E$ such that $A\mathbf{x} = \mathbf{0}$.

The previous lemma can then be seen as a special case of [Onn10, Lemma 2.17], which states that each vector $\mathbf{x} \in \mathbb{R}^n$ satisfying $A\mathbf{x} = \mathbf{0}$ can be expressed as a conformal sum of at most n circuits of A. (Actually, it guarantees even stronger properties that we do not need here.)

Lemma 1.7 (Path decomposition of flows)**.** *Every st-flow can be expressed as a conformal sum of at most $|E|$ weighted st-paths and at most $|E|$ positive cycles.*

Proof. See [AMO88, Theorem 3.5]. The idea is similar to the proof of Lemma 1.4, with the difference that we have to substract several weighted st-paths first till the remaining flow has zero value, turning it into a positive circulation. The rest of the proof is identical. □

Corollary 1.8. For every st-flow $\mathbf{f}$, there exists an st-flow $\mathbf{g}$ such that $|\mathbf{f}| = |\mathbf{g}|$, $\mathbf{g} \sqsubseteq \mathbf{f}$ and $\mathbf{g}$ can be expressed as a conformal sum of weighted st-paths.

Proof. Use Lemma 1.7 to express $\mathbf{f}$ as a conformal sum $\mathbf{f} = \sum_{i=0}^{q} \alpha_i \cdot \mathbf{C}_i + \sum_{i=0}^{r} \beta_i \cdot \mathbf{p}_i$ and set $\mathbf{g} := \sum_{i=0}^{r} \beta_i \cdot \mathbf{p}_i$. We have

$$|\mathbf{f}| = \left|\sum_{i=0}^{q} \alpha_i \cdot \mathbf{C}_i + \sum_{i=0}^{r} \beta_i \cdot \mathbf{p}_i\right| = \sum_{i=0}^{q} |\alpha_i \cdot \mathbf{C}_i| + \sum_{i=0}^{r} |\beta_i \cdot \mathbf{p}_i| = \sum_{i=0}^{q} 0 + \sum_{i=0}^{r} \beta_i = |\mathbf{g}|\,.$$

□

Remark 1.9. The decomposition of circulations (and flows) into cycles (and paths) is not unique. Even a simple flow $\mathbf{f} = 2\mathbf{p}$ consisting of a single weighted path can be decomposed as $\mathbf{f} = 2\mathbf{p}$, $\mathbf{f} = \mathbf{p} + \mathbf{p}$, $\mathbf{f} = 1/4 \cdot \mathbf{p} + 7/4 \cdot \mathbf{p}, \ldots$, however silly this example may be.

We will assume that the procedure given in the proofs of Lemma 1.4 and Lemma 1.7 is deterministic – for a fixed flow, it always gives the same decomposition. It is also "well-behaved", in the sense that no cycle or flow is zero or repeats in the decomposition. We will refer to this particular decomposition as the *canonical decomposition.*

1.8 Convergence

Lastly, we present two useful technical lemmas which we will later use when talking about convergence of algorithms.

Lemma 1.10. *Let (h_n) be a sequence of non-negative numbers such that for some $B > 0$,*

$$h_{t+1} \leq h_t \cdot (1 - h_t/B).$$

Then, for all $t \geq 1$,

$$h_t \leq \frac{2B}{t}.$$

Proof. This lemma is a special case of Kerdreux, d'Aspremont, and Pokutta [KdP18, Lemma B.1]. The recurrence there is of the form $h_{t+1} \leq h_t \cdot \max\{1/2, 1 - h_t/B\}$, but the proof holds even for our recurrence. □

Lemma 1.11. *Let (h_n) be a sequence of non-negative numbers such that for some $\rho \in (0, 1]$ and $\alpha > 0$,*

$$h_t \leq h_0 \cdot (1 - \rho)^{\alpha t}.$$

Then for any $\varepsilon > 0$, the inequality $h_Q \leq \varepsilon$ holds for any Q satisfying

$$Q \geq \log(h_0/\varepsilon) \cdot \frac{\rho}{\alpha}.$$

Proof. This is a direct consequence of the known inequality $(1+1/x)^x \leq e$ applied with $x = -1/\rho$:

$$\begin{aligned} h_Q &\leq h_0 \cdot (1-\rho)^{\alpha Q} \leq h_0 \cdot e^{(-1/\rho)\cdot \alpha Q} \leq h_0 \cdot e^{(-1/\rho)\cdot \alpha \log(h_0/\varepsilon)\cdot \frac{\rho}{\alpha}} \\ &= h_0 \cdot e^{-1\cdot \log(h_0/\varepsilon)} = h_0 \cdot e^{\log(\varepsilon/h_0)} = h_0 \cdot \varepsilon/h_0 = \varepsilon. \end{aligned}$$

□

2. The State of the Art

In this chapter, we review the current state of the art regarding both MCF and LBF. We start by formally stating both problems in Section 2.1, as well as mentioning notable related problems. Then we present two formulations of MCF and LBF in Section 2.2, namely in the form of linear and convex programs. We introduce some more notation and terminology in the process. In Section 2.3, we review problem-specific methods for both problems, namely the traditional algorithms for MCF, which are generally without polynomial-time guarantees, and modern FPTASs for both MCF and LBF. We also briefly touch on the Minimum Length-Bounded Cut problem. We conclude by mentioning some of the methods used for solving linear and convex programs, and their application and (current lack of) interpretation with respect to MCF and LBF.

2.1 Flow Problems

As a warm-up, let us define the Maximum Flow problem on which both of our problems are based.

A *flow network* is a tuple $\mathcal{G} = (G, s, t)$ where $G = (V, E, \mathbf{c})$ is a network and $s, t \in V, s \neq t$.

The Maximum Flow problem is:

Maximum Flow

Input:	A flow network $\mathcal{G} = (G, s, t)$.
Find:	A satisfying *st*-flow of maximum value.

We also define a similar problem which, for our purposes, acts like a decision version of the Maximum Flow problem.

Fixed-Demand Flow

Input:	A flow network $\mathcal{G} = (G, s, t)$, demand $d \in \mathbb{R}_0^+$ (i.e., $d \in \mathbb{R}$, $d \geq 0$).
Find:	A satisfying *st*-flow of value d.

It is trivial to see that whenever there exists a satisfying flow $\mathbf{f}$ of value d, there also exists a satisfying flow of value $d' < d$: one may simply scale $\mathbf{f}$ down by a correct factor.

Both problems can be generalised by introducing a vector $\mathbf{w} \in \mathbb{R}^E$ of costs per one flow unit transported along each edge. Then the task is not only to find a maximum or satisfying flow, but to find one that minimises $\mathbf{w} \cdot \mathbf{f}$ at that:

Minimum-Cost Maximum Flow (MCMF)

Input:	A flow network $\mathcal{G} = (G, s, t)$, weights $\mathbf{w} \in \mathbb{R}^E$.
Find:	A satisfying *st*-flow $\mathbf{f}$ that primarily maximises $\lvert\mathbf{f}\rvert$ and secondarily minimises $\mathbf{w} \cdot \mathbf{f}$.

Minimum-Cost Fixed-Demand Flow (MCFDF)

Input:	A flow network $\mathcal{G} = (G, s, t)$, demand $d \in \mathbb{R}_0^+$, weights $\mathbf{w} \in \mathbb{R}^E$.
Find:	A satisfying st-flow $\mathbf{f}$ of value d that minimises $\mathbf{w} \cdot \mathbf{f}$.

2.1.1 Multicommodity Flow

The Multicommodity Flow problem (MCF) generalises the Maximum Flow problem by considering multiple commodities, each with its own source and sink, sharing the underlying network and capacities.

A *commodity* is defined by a tuple (s_k, t_k, d_k) where $s_k, t_k \in V, s_k \neq t_k$ and $d_k \in \mathbb{R}_0^+$; s_k, t_k and d_k are respectively called the *source*, *sink*, and *demand* of commodity k.

Given a collection of commodities $\mathcal{K} = \Big((s_k, t_k, d_k)\Big)_{k=1}^K$, a *multiflow* is a collection $\mathbf{f}$ of flows $\mathbf{f}^1, \ldots, \mathbf{f}^K$ such that $\mathbf{f}^k$ is an $s_k t_k$-flow and $|\mathbf{f}^k| = d_k$.[1] For each edge $e \in E$, we define the *total flow* along it as:

$$\mathbf{f}^\Sigma(e) = \sum_{k=1}^K f_e^k. \tag{2.1}$$

The notation $\mathbf{f}_e \in \mathbb{R}^K$ denotes the collection of K flows on edge e, while the notation $\mathbf{f}^\Sigma \in \mathbb{R}^E$ naturally denotes the collection of $\mathbf{f}^\Sigma(e)$ for all $e \in E$.

A multiflow $\mathbf{f}$ is *satisfying* if:

$$\forall e \in E : \mathbf{f}^\Sigma(e) \leq c_e,$$

or equivalently,

$$\mathbf{f}^\Sigma \leq \mathbf{c}.$$

A *multiflow network* is a pair $\mathcal{G} = (G, \mathcal{K})$ where $G = (V, E, \mathbf{c})$ is a network and $\mathcal{K} = \Big((s_k, t_k, d_k)\Big)_{k=1}^K$ is a collection of commodities. We will also sometimes use the notation $\mathbf{d} \in \mathbb{R}^K, \mathbf{d} = (d_1, \ldots, d_K)$ for the vector of demands.

Again, we may state the Multicommodity Flow problem:

Multicommodity Flow (MCF)

Input:	A multiflow network $\mathcal{G} = (G, \mathcal{K})$
Find:	A satisfying multiflow.

Remark 2.1. MCF has many variants, some of which are often referred to as "the multicommodity flow problem" in the literature. To avoid confusion, let us briefly describe some of them and point out their differences from our definition.

- *Undirected MCF.* In this variant, the underlying graph is undirected. The flow can travel in both directions along each edge, and the capacity constraints restrict the total flow in both directions combined. Note that this

[1] Notice the subtle difference from single-commodity flow: since we will be only considering the setting with fixed demands, we have incorporated the demands into the definition of the multiflow.

is different from having two directed edges for each undirected edge, since in the latter case the two collections of flows on the opposite edges are unrelated. It is unclear whether there exists a meaningful reduction between directed and undirected MCF. For example, undirected 2-commodity MCF seems to have fundamentally different structure than directed 2-commodity MCF [Ita78].

- *Integer MCF.* It is a well known property of the Maximum Flow that if all the capacities are integer, there exists an optimal flow that is also integer. Surprisingly, this does not hold for MCF with integer demands, even in the case of two commodities and unit capacities [Ita78]. In fact, the problem of finding a 0-1 multiflow in a two-commodity network with unit capacities is $\mathcal{NP}$-hard [EIS76], as is, by extension, the general integer MCF problem. On the other hand, MCF is in $\mathcal{P}$, since it may be formulated as a linear program with polynomially many variables and constraints (see Section 2.2.1).

- *Weighted MCF.* While we have formulated MCF as a decision problem, there are also many optimisation formulations. In weighted MCF, each combination of an edge and commodity has an associated weight $w_e^k \in \mathbb{R}$ and the goal is to find a satisfying multiflow minimising $\sum_{e\in E}\sum_{k\in K} w_e^k \cdot f_e^k$. Some of the methods presented in later chapters may be adapted for this variant. See [AMO88] for the full formulation and an overview of approaches.

- *Concurrent flow.* In this variant, we want to maximise the scaling factor ρ such that if we scale all demands by ρ, the resulting multiflow network still admits a satisfying flow. See [Mąd10, Section 1.1] for a short survey of known results.

- *Other generalisations.* In the most general setting, there are no demands and the optimisation criterion is a function of the current multiflow, e.g., a weighted sum of $|\mathbf{f}^k|$, a quadratic function penalising "nearly saturated" edges etc. The hardness of these variants obviously depends on the choice of the criterion. Some versions are even more general and allow, e.g., to specify additional per-commodity restrictions. For a more detailed overview and a list of hard instances, see [Ope13].

Remark 2.2. The definitions and lemmas we gave in Chapter 1 easily generalise to multicommodity flows. For example, a *multicirculation* $\mathbf{\Delta}$ is a collection of circulations $\mathbf{\Delta}^1, \dots, \mathbf{\Delta}^K$. A (multicommodity) weighted $s_k t_k$-path is defined as a multiflow $\mathbf{f}$ such that $\mathbf{f}^k = w\mathbf{p}$ is a (single-commodity) weighted $s_k t_k$-path and $\mathbf{f}^j = \mathbf{0}$ for $j \neq k$. Similarly, a (multicommodity) weighted cycle is defined as a multicirculation $\mathbf{\Delta}$ such that $\mathbf{\Delta}^k$ is a (single-commodity) weighted cycle and $\mathbf{\Delta}^j = \mathbf{0}$ for $j \neq k$.

Multicommodity analogues of Lemma 1.4 and Lemma 1.7 hold: a multicirculation $\mathbf{\Delta}$ is a conformal sum of at most $|E| \cdot K$ weighted cycles, at most $|E|$ per each commodity, and a multiflow $\mathbf{f}$ is a conformal sum of at most $|E| \cdot K$ weighted $s_k t_k$-paths and $|E| \cdot K$ weighted cycles, again at most $|E|$ of both per each commodity.

2.1.2 Length-Bounded Flow

Other generalisation of the Maximum Flow problem we are interested in is the Length-Bounded Flow problem (LBF). We already know from Section 1.7 that every *st*-flow can be decomposed into *st*-paths.[2] In LBF, the goal is to find a satisfying flow of a given value that can be decomposed into paths of length at most K for some $K \in \mathbb{N}$.

A *K-flow network* is a tuple $\mathcal{G} = (G, s, t, K)$ where $G = (V, E, \mathbf{c})$ is a network, $s, t \in V$ and $K \in \mathbb{N}$. A *K-bounded st-path* is an *st*-path of length at most K. A *K-bounded st-flow* (or just *K-flow*) is an *st*-flow $\mathbf{f}$ such that there exist positive real weights $w_1, \ldots, w_\ell > 0$ and K-bounded *st*-paths $\mathbf{p}_1, \ldots, \mathbf{p}_\ell$ satisfying:

$$\sum_{i=1}^{\ell} w_i \mathbf{p}_i = \mathbf{f}.$$

The Length-Bounded Flow problem may be stated as follows:

Length-Bounded Flow (LBF)

Input:	A K-flow network $\mathcal{G} = (G, s, t, K)$, demand $d \in \mathbb{R}_0^+$.
Find:	A satisfying K-bounded *st*-flow of value d.

Remark 2.3. As with MCF, there are many notable variants of LBF and other related problems:

- *Maximum length-bounded flow.* Instead of finding a K-flow of value d, find a K-flow of maximum value. This is actually the standard formulation of LBF. We decided to formulate LBF as a decision problem instead, mainly for the sake of simplicity and consistency with MCF. As in the case of weighted MCF, many of the algorithms presented here can be adapted to maximum LBF. Note that one can always solve maximum LBF using LBF and binary searching for the maximum value of d for which there still exists a satisfying flow. This introduces a $\mathcal{O}(\log(d/\varepsilon))$ slowdown, where ε is the needed accuracy.

- *General edge lengths.* One can assign a length $\ell(e) \geq 0$ to each edge and define the length of a path as the sum of the lengths of edges on the path. Our definition can then be seen as a special case with $\ell(e) = 1$ for all edges. It turns out that LBF with this generalised notion of path length is $\mathcal{NP}$-hard even for unit capacities [Bai+10].

- *Integrality.* Similarly to integer MCF, LBF with integer demands does not necessarily have an integer solution, even in the case of unit capacities. In fact, the problem of finding a 0-1 K-bounded flow of given value in a network with unit capacities is equivalent to the K-bounded edge-disjoint path problem [Vob16], which is $\mathcal{NP}$-hard [IPS82a].

[2] More precisely, into *st*-paths and positive cycles, but the cycles can be omitted while preserving flow value and satisfaction.

- *Length-bounded cut.* Similarly to the relation between the MAXIMUM FLOW and MINIMUM CUT problem, the LENGTH-BOUNDED FLOW has a corresponding MINIMUM LENGTH-BOUNDED CUT problem. The objective there is to find the set of edges with the smallest possible sum of capacities such that after their removal, the graph no longer contains an *st*-path of length $\leq K$. Variations exist; the goal may be to remove vertices instead of edges and the size of the cut may be defined as the number of vertices in the cut. MINIMUM LENGTH-BOUNDED CUT has been studied more than LBF and several hardness results, approximation algorithms, and parameterised results (both positive and negative) are known [DK18; CK20]. We will talk about MINIMUM LENGTH-BOUNDED CUT more in Section 2.3.3.

2.2 Formulations

In this section, we present two common formulations of MCF and LBF. First we formulate both problems as linear programs which we then transform into convex programs by relaxing some constraints and bringing them into the objective function. The second approach may seem unnecessarily complicated, given that linear programming is generally easier then convex programming, but it allows us to approach the problems from a different perspective.

2.2.1 Linear Programming

MCF

Formulating MCF as a linear program is straightforward:

$$\begin{aligned} &\text{find } \mathbf{f} \in \mathbb{R}^{E\times K} \\ &\text{subject to } \mathbf{f}_\delta^k(v) = \begin{cases} -d_k & \text{if } v = s_k, \\ d_k & \text{if } v = t_k, \\ 0 & \text{otherwise.} \end{cases} \qquad \forall k = 1,\dots,K \;\; \forall v \in V \\ &\qquad\quad \mathbf{f}^\Sigma \quad \leq \mathbf{c} \\ &\qquad\quad \mathbf{f} \quad\; \geq \mathbf{0} \end{aligned} \tag{MCF-LP}$$

(Of course, we must replace $\mathbf{f}_\delta^k$ and $\mathbf{f}^\Sigma$ with their respective definitions from Equations (1.1) and (2.1). Both of them expand to sums of some entries of $\mathbf{f}$ and the formulation is therefore indeed an LP formulation.) The equalities correspond to the flow conservation constraints for each commodity, while the inequalities are just the capacity and nonnegativity constraints for multicommodity flows.

It immediately follows that MCF is in $\mathcal{P}$, since the LP size is polynomial in the problem size and since linear programs can be solved in weakly polynomial time. What is more, the constraint matrix of MCF has small coefficients, which means that MCF is solvable in *strongly* polynomial time using Tardos' algorithm. The same is true for LBF, as we will shortly see.

LBF

When formulating LBF, it is common not to work directly with the original graph, but to apply the following transformation first [KS06, Claim 1].

For a network $G = (V, E, \mathbf{c})$ and $K \in \mathbb{N}$, define a *layered network* G_K as follows. We take the set of vertices V and make $K+1$ distinct copies $V^0, \ldots, V^K$. Let v^k denote the copy of v in V^k. We then define $G_K = (V_K, E_K, \mathbf{c})$ where:

$$\begin{aligned} V_K &= V^0 \cup \cdots \cup V^K, \\ E_K &= \{\, u^k v^{k+1} \mid k \in \{0, \ldots, K-1\}, uv \in E \,\} \\ &\quad \cup \{\, v^k v^{k+1} \mid k \in \{0, \ldots, K-1\}, v \in V \,\}. \end{aligned}$$

For $e = uv \in E$, let e^k denote the edge $u^k v^{k+1}$. The edges of type $v^k v^{k+1}$ are called *loop edges.*

(Note that we have not yet specified how the capacities $\mathbf{c} \in \mathbb{R}^E$ relate to flows $\mathbf{f} \in \mathbb{R}^{E_K}$ in the layered network, so although it makes sense to talk about flows, it does not yet make sense to talk about satisfying flows.)

Due to practical reasons, we shall abuse the notation and denote the flow on edge e^k as f_e^k instead of f_{e^k}. This notation is also more consistent with MCF.

The idea is that each K-bounded st-path in G can be represented as an s^0t^K-path in G_K and each s^0t^K-path in G_K conversely represents a st-**walk** in G. Using the fact that flows can be decomposed into paths, it can be concluded that s^0t^K-flows correspond to K-bounded st-flows and vice versa. In order to state this formally, we need a few definitions first.

Definition 2.1 (Flattening, expansion)**.** Given a network G and the corresponding layered network G_K, let *flattening* be a mapping π_K from s^0t^K-paths in G_K to K-bounded st-paths in G, where we define $\pi_K(p_K)$ as follows (using the definition of paths as sequences of edges):

First, we remove all loop edges from p_K and replace each edge $e^k \in E^K$ by a corresponding edge $e \in E$. This gives us a walk w in G of length $\leq K$. We then simplify w into a path p by iteratively removing cycles until none are left (using Remark 1.3) and set $\pi_K(p_K) := p$. It can be verified that p is a K-bounded st-path because w is a K-bounded st-walk and the loop removal procedure does neither increase the length of the path nor change its endpoints.

Let *expansion* be a mapping π_K^{-1} from K-bounded st-paths in G to s^0t^K-paths in G_K defined as follows: for a path $p = (e_0, \ldots, e_{\ell-1})$, $\ell \leq K$,

$$\pi_K^{-1}(p) = (e_0^0, \ldots, e_{\ell-1}^{\ell-1}, t^\ell t^{\ell+1}, \ldots, t^{K-1}t^K).^3$$

Note that $\pi_K(\pi_K^{-1}(p)) = p$, but generally $\pi_K^{-1}(\pi_K(p_K)) \neq p_K$.

Definition 2.2 (Flattening for flows)**.** We will extend the definition of π_K and π_K^{-1} for weighted paths and flows: Given a K-bounded weighted st-path $w\mathbf{p}$ in G, we define $\pi_K(w\mathbf{p}) = w\mathbf{p}_K$ where $\mathbf{p}_K \in \mathbb{R}^{E_K}$ is the incidence vector of $\pi_K(p)$. Similarly, given $w\mathbf{q}_K$ in G_K, define $\pi_K^{-1}(w\mathbf{q}_K) = w\mathbf{q}$ where $\mathbf{q}$ is the incidence vector of $\pi_K^{-1}(q)$.

For a flow $\mathbf{f}_K$ in G_K, define $\pi_K(\mathbf{f}_K) = \sum_{i=0}^r \pi_K(\beta_i \mathbf{p}_i)$ where $\mathbf{f}_K = \sum_{i=0}^q \alpha_i \mathbf{C}_i + \sum_{i=0}^r \beta_i \mathbf{p}_i$ is the canonical decomposition (in the sense of Remark 1.9) of $\mathbf{f}_K$ into weighted cycles and paths.

[3]This is why we introduced loop edges in the definition above, since without them, we would be only able to represent st-paths of length exactly K.

Similarly, for a K-bounded st-flow $\mathbf{g}$ in G, define $\pi_K^{-1}(\mathbf{g}) = \sum_{i=0}^{r'} \pi_K^{-1}(\beta'_i \mathbf{p}'_i)$ where $\mathbf{g} = \sum_{i=0}^{q'} \alpha'_i \mathbf{C}'_i + \sum_{i=0}^{r'} \beta'_i \mathbf{p}'_i$ is the canonical decomposition of $\mathbf{g}$ into weighted cycles and paths.

Notice that $|\mathbf{f}_K| = |\pi_K(\mathbf{f}_K)|$ and $|\mathbf{g}| = |\pi_K^{-1}(\mathbf{g})|$.

What about edge capacities and satisfaction? Observe that for a flow $\mathbf{f}$ in G and its expansion $\mathbf{g} = \pi_K^{-1}(\mathbf{f})$ in G, it holds that $f_e = \sum_{k=0}^{K-1} g_e^k$ for every edge $e \in E$, since every weighted path in the decomposition of $\mathbf{f}$ that contributes to f_e gets mapped to a weighted path of same weight that contributes to g_e^k for exactly one k. This motivates the following definition:

Definition 2.3 (Total flow, satisfying flow)**.** Let $\mathbf{f}$ be a flow in $G_K = (V_K, E_K)$, which is a layered representation of $G = (V, E, \mathbf{c})$. We define the total flow along edge $e \in E$ as:

$$\mathbf{f}^{\Sigma}(e) = \sum_{k=0}^{K-1} f_e^k. \tag{2.2}$$

We say that $\mathbf{f}$ is *satisfying* with respect to $\mathbf{c}$ if:

$$\mathbf{f}^{\Sigma} \leq \mathbf{c}.$$

(This is a slight abuse of notation, since we already use the same symbol with MCF. However, the intended meaning is always clear from the context and both of these definitions express a similar concept.)

The previous observations can be summed up in the following lemma:

Lemma 2.4. *A K-bounded flow $\mathbf{f}$ in G is satisfying with respect to $\mathbf{c}$ if and only if the corresponding flow $\pi_K^{-1}(\mathbf{f})$ in G_K is satisfying with respect to $\mathbf{c}$. Furthermore, $|\mathbf{f}| = |\pi_K^{-1}(\mathbf{f})|$.*

Consequently, there exists a K-bounded satisfying flow of value d in G if and only if there exists a satisfying flow of value d in G_K, and given one of those flows, we may construct the other one in a straightforward manner.

Proof. The first part follows directly from the definitions. For the second part, we prove both implications. Given a satisfying flow $\mathbf{f}$ in G of value d, we already know that $\pi_K^{-1}(\mathbf{f})$ is a satisfying flow in G_K and that it has value d. On the other hand, if we have a satisfying $\mathbf{f}_K$ in G_K of value d, we may observe that $\mathbf{g}_K := \pi_K^{-1}(\pi_K(\mathbf{f}_K))$ is also satisfying and of value d, since $\mathbf{g}_K^{\Sigma} \leq \mathbf{f}_K^{\Sigma}$. This can be seen by decomposing $\mathbf{f}_K$ into weighted paths and observing what $\pi_K^{-1} \circ \pi_K$ does to each path. Now we may apply the first part of the lemma on $\pi_K(\mathbf{f}_K)$, which says that $\pi_K(\mathbf{f}_K)$ is satisfying and of value d if and only if $\pi_K^{-1}(\pi_K(\mathbf{f}_K)) = \mathbf{g}_K$ is, but we have just shown that it is. $\square$

We may now finally formulate LBF as a linear program:

$$\begin{aligned}
&\text{find } \mathbf{f} \in \mathbb{R}^{E_K} \\
&\text{subject to } \mathbf{f}_\delta(v) = \begin{cases} -d & \text{if } v^k = s^0, \\ d & \text{if } v^k = t^K, \\ 0 & \text{otherwise.} \end{cases} \quad \forall v^k \in V_K \\
&\qquad \mathbf{f}^{\Sigma} \quad \leq \mathbf{c} \\
&\qquad \mathbf{f} \quad \geq \mathbf{0}
\end{aligned} \tag{LBF-LP}$$

(Again, we have to replace $\mathbf{f}^k_\delta$ and $\mathbf{f}^\Sigma$ with their respective definitions, this time from Equations (1.1) and (2.2). These definitions are again just sums of entries of $\mathbf{f}$ and the formulation is therefore an LP formulation.)

2.2.2 Penalty Function

Constraint relaxation is a general approach in constrained optimisation. The idea is that some constraints are in some sense more difficult to satisfy than others, so we remove them and bring them to the objective function instead. One of common techniques used in linear programming is *Lagrangian relaxation* [AMO88, Chapter 16]. We will use a slightly different approach by introducing a *penalty function*. See e.g. [BV04, Section 6.2] for more insight and related techniques.

In MCF and LBF, we consider the capacity constraints to be the "difficult" constraints, and the non-negativity and flow conservation constraints to be the easy ones. The intuition is that without capacity constraints, we are basically left with several smaller independent and unbounded Fixed-Demand Flow problems (in the case of MCF), or even just one larger unbounded Fixed-Demand Flow problem (in the case of LBF). In both cases, the structure of the remaining problems is much simpler (e.g., the constraint matrix is totally unimodular) and well studied.

To turn the capacity constrains into a penalty function, we consider the following function for both problems:[4]

$$\varphi(\mathbf{f}) := \sum_{e\in E} \varphi_e(\mathbf{f}), \qquad \text{where} \qquad \varphi_e(\mathbf{f}) := \max\{\mathbf{f}^\Sigma(e) - c_e, 0\}^2 = \Big[\mathbf{f}^\Sigma(e) - c_e\Big]_+^2 .$$

In other words, edges that do not exceed capacity do not contribute to the penalty, and the remaining edges contribute the square of their capacity violation. Trivially, satisfying flows (which we want to find in both MCF and LBF), are exactly the flows whose penalty is 0.

It can be seen that φ is a sum of several convex functions and is therefore also convex. We prove this more rigorously in Section 3.7.2.

Relaxing the capacity constraints in (MCF-LP) and (LBF-LP) using this penalty function then yields the following two convex programs:

$$\begin{aligned}
\text{minimise } \varphi(\mathbf{f}) &= \sum_{e\in E} \Big[\mathbf{f}^\Sigma(e) - c_e\Big]_+^2 \text{ for } \mathbf{f} \in \mathbb{R}^{E\times K} \\
\text{subject to } \mathbf{f}^k_\delta(v) &= \begin{cases} -d_k & \text{if } v = s_k, \\ d_k & \text{if } v = t_k, \\ 0 & \text{otherwise.} \end{cases} \quad \forall k = 1,\dots,K \ \ \forall v \in V \\
\mathbf{f} &\geq \mathbf{0}
\end{aligned} \tag{MCF-P}$$

[4]As we have already mentioned, the meaning of $\mathbf{f}^\Sigma$ depends on the problem.

and:

$$\begin{aligned}
&\text{minimise } \varphi(\mathbf{f}) = \sum_{e\in E} \left[\mathbf{f}^{\Sigma}(e) - c_e\right]_+^2 \text{ for } \mathbf{f} \in \mathbb{R}^{E_K} \\
&\text{subject to } \mathbf{f}_\delta(v) = \begin{cases} -d & \text{if } v^k = s^0, \\ d & \text{if } v^k = t^K, \quad \forall v^k \in V_K \\ 0 & \text{otherwise.} \end{cases} \\
&\mathbf{f} \quad \geq \mathbf{0}
\end{aligned} \tag{LBF-P}$$

As mentioned previously, the constraints of (MCF-P) and (LBF-P) are arguably simpler than those of (MCF-LP) and (LBF-LP). Namely:

- (MCF-P) can be decomposed into K unbounded Fixed-Demand Flow problems in G. These subproblems are unrelated, and only interact in the penalty.
- (LBF-P) is a single unbounded Fixed-Demand Flow problem in the layered network G_K.

We will show how some approaches exploit this structure in later chapters.

2.2.3 Notation and Terminology

Since we will be mostly using the relaxed formulations (MCF-P) and (LBF-P) throughout the majority of this thesis, it is useful to introduce some general notation and terminology. In the following, $\mathbf{f}$ is always a (multi)flow, $\mathbf{\Delta}$ a feasible (multi)circulation (with respect to $\mathbf{f}$) and $\mathbf{\Delta}^*$ a feasible (multi)circulation such that $\mathbf{f} + \mathbf{\Delta}^*$ is optimal. The notation is identical for MCF and LBF.

The *optimality gap* is defined as $\mathrm{GAP}(\mathbf{f}) = \varphi(\mathbf{f}) - \varphi(\mathbf{f} + \mathbf{\Delta}^*)$, i.e., it measures by how much the current penalty is greater than the optimal one. *Penalty improvement* with respect to $\mathbf{f}$ is defined as $\mathrm{IMP}_{\mathbf{f}}(\mathbf{\Delta}) = \varphi(\mathbf{f}) - \varphi(\mathbf{f} + \mathbf{\Delta})$. Clearly $\mathrm{GAP}(\mathbf{f}) = \mathrm{IMP}_{\mathbf{f}}(\mathbf{\Delta}^*)$. GAP is convex (since it is just φ shifted by a constant) and $\mathrm{IMP}_{\mathbf{f}}$ is concave for a fixed $\mathbf{f}$ (since then it is a difference of a constant and of a convex function).

A feasible circulation $\mathbf{\Delta}$ is *improving* (with respect to $\mathbf{f}$) if $\mathrm{IMP}_{\mathbf{f}}(\mathbf{\Delta}) > 0$, i.e., if and only if adding $\mathbf{\Delta}$ to $\mathbf{f}$ improves the penalty. A (possibly infeasible) circulation $\mathbf{\Delta}$ is an *improving direction* if there exists some $\varepsilon > 0$ such that $\varepsilon\mathbf{\Delta}$ is feasible and improving. The properties of φ (namely convexity) guarantee that whenever $\varepsilon\mathbf{\Delta}$ for $\varepsilon > 0$ is improving, then $\varepsilon'\mathbf{\Delta}$ for $\varepsilon > \varepsilon' > 0$ is also improving.

One-dimensional penalty (1D penalty) with respect to $\mathbf{f}$ and a feasible $\mathbf{\Delta}$ is a function $h_{\mathbf{f},\mathbf{\Delta}} : [0,1] \to \mathbb{R}$ defined as $h_{\mathbf{f},\mathbf{\Delta}}(\gamma) := -\mathrm{IMP}_{\mathbf{f}}(\gamma\mathbf{\Delta}) = \varphi(\mathbf{f} + \gamma\mathbf{\Delta}) - \varphi(\mathbf{f})$. It will be useful in contexts where we are figuratively standing at point $\mathbf{f}$ knowing that $\mathbf{\Delta}$ is an improving direction and are interested in how adding different quantities of $\mathbf{\Delta}$ changes the penalty. We will talk about $h_{\mathbf{f},\mathbf{\Delta}}$ later, now it suffices to say that it is convex and continuously differentiable (because φ is), $h_{\mathbf{f},\mathbf{\Delta}}(0) = 0$ and that its first derivative is nondecreasing. It follows from our discussion of improving directions that $\mathbf{\Delta}$ is improving if and only if there exists $\varepsilon > 0$ such that $h_{\mathbf{f},\mathbf{\Delta}}(\varepsilon) < 0$. Applying the convexity of $h_{\mathbf{f},\mathbf{\Delta}}$, its relation to $h'_{\mathbf{f},\mathbf{\Delta}}$ and the fact that $h_{\mathbf{f},\mathbf{\Delta}}(0) = 0$ gives us the following characterisation.

Lemma 2.5. *Let $\mathbf{f}$ be a flow and $\mathbf{\Delta}$ a circulation. Then $\mathbf{\Delta}$ is an improving direction if and only if $h'_{\mathbf{f},\mathbf{\Delta}}(0) < 0$.*

We will also use some of this notation even for penalties different from φ and problems other than MCF and LBF, especially GAP($\mathbf{x}$) (which denotes the difference between the current penalty and the optimal one) and $\mathrm{IMP}_{\mathbf{x}}(\mathbf{y})$ (which denotes the difference between the penalty at $\mathbf{x}$ and the penalty at $\mathbf{x} + \mathbf{y}$).

2.3 Prior Work

Both MCF and LBF have been widely studied and many algorithms have been developed for them. For example, LBF has been long thought solvable by an algorithm proposed by Koubek and Říha [KŘ81], but Altmanová [Alt18] has recently shown that this algorithm is fundamentally flawed and most likely not easily fixable. The earliest algorithms for MCF date back to 1950s [FF58]. In recent years, many FPTASs for MCF have been proposed [SM90; Fle99; GK07; Mąd10] (see [Mąd10] for more references). The FPTAS of Garg and Könemann [GK07] has been adapted for LBF by Voborník [Vob16].

In this section, we briefly describe some of the more traditional methods of solving MCF. We then turn our attention towards modern FPTASs for MCF and LBF. After that, we talk about the Minimum Length-Bounded Cut problem and its relationship to LBF. Lastly, we briefly examine some of the methods used for solving linear and convex programs, and their application and possible combinatorial interpretation with respect to MCF and LBF. This section is meant as a general overview; the reader is advised to see the original articles for a more in-depth description.

2.3.1 Traditional Algorithms for MCF

The main reference for this section is [AMO88, Chapter 17], see also the surveys by Assad [Ass78] and Kennington [Ken78]. Let us emphasise that to our knowledge, none of the presented algorithms have been shown to run in polynomial time [AMO88, Section 17.9], although some of them may run in polynomial time if the capacities are polynomial in input size and/or the required approximation error (if applicable for the algorithm) is constant.

Traditional approaches for MCF usually fall into one of three categories: price-directive decomposition [CST70; CD74], resource-directive decomposition [GG74] and basis partitioning [GM76].

Price-directive methods relax the capacity constraints and bring them to the objective function, so that the problem decomposes into several single-commodity Minimum-Cost Fixed-Demand Flow problems. We can view this as "charging" the commodities for using the shared capacity of each edge. There are several methods for finding the appropriate prices, e.g. Lagrangian relaxation [AMO88, Chapter 11] or Dantzig-Wolfe decomposition [DW60]. While both can be viewed as general-purpose approaches for decomposing problems consisting of "easy" and "hard" constraints, the latter is more special in that it is applicable for linear programs with a special block structure.

Resource-directive methods use a different approach: instead of allowing each commodity to independently use as much capacity as it needs but "discouraging" congestion using costs, each commodity is instead allocated a portion of each edges' capacity, which it can use as it pleases, but which it cannot exceed. The method then consists of alternating between solving independent (unweighted) single commodity flow problems and using their solution to reallocate the capacities between commodities.

Basis partitioning methods are inspired by the simplex method and the network simplex. They maintain a linear programming basis that is composed of bases of the individual single commodity flow problems, as well as additional edges.

Notable is also a scaling algorithm developed by Schneur [Sch91, Section 3.3][SO98], which we have already mentioned in the introduction. It works with the (MCF-P) formulation, i.e., it has no capacity constraints and a quadratic penalty function. Initially it finds any multiflow satisfying demands. Then it performs a series of scaling steps: it picks δ and repeatedly tries to find a weighted cycle of weight δ that, if added to the flow in some commodity, decreases the penalty. When such a cycle no longer exists, the procedure is repeated with $\delta/2, \delta/4, \ldots$, until it is guaranteed that the optimality gap is below some $\varepsilon > 0$. The running time is polynomial in the input size, $1/\varepsilon$ and the sum of demands, which means that this algorithm is an FPTAS if the capacities are polynomially bounded.

2.3.2 Modern FPTASs

Starting from 1990s, numerous FPTASs for MCF and the related concurrent flow problem have been proposed. We briefly review some of them, see [Mąd10] for a more comprehensive view. In practice, these FPTASs may be preferred to LP, since solving LP might generally be slower and one is often only interested in an approximate solution.

Many of the earlier FPTASs [Kle+94; Lei+95; Gol92] are based on Lagrangian relaxation. They find an initial feasible but unsatisfying flow and redistribute it with the help of a penalty function. Young [You95] deviated from this pattern and presented an oblivious rounding algorithm for the concurrent flow problem that does not reroute the flow and builds it from scratch instead. It uses an exponential edge length function that models the congestion of edges. At each step, it augments the flow along the shortest (i.e., relatively uncongested) path, and updates the length function. At the end, the final solution is obtained by scaling the flow down to honour capacities.

A similar approach was taken by Garg and Könemann [GK07], who also developed a general framework for solving multicommodity flow problems. Fleischer [Fle99] subsequently used this framework to develop faster algorithms for various multicommodity flow problems. Recently, Mądry [Mąd10] proposed an approach that builds upon the previous two results and additionally uses dynamic graph algorithms and data structures along with randomisation to obtain faster running times. In the case of MCF, it runs in time $\widetilde{\mathcal{O}}(\varepsilon^{-2} \cdot |V| \cdot |E|)$, where $\widetilde{\mathcal{O}}(\cdot)$ is a variation of the $\mathcal{O}$-notation that hides polylogarithmic factors.[5]

For LBF, the list of known results is – at least to our knowledge – much shorter, probably because MCF has been more widely studied. Apart from the flawed algorithm of Koubek and Říha [KŘ81], an FPTAS for LBF, even for the

[5]Formally, $g(n)$ is $\widetilde{\mathcal{O}}(f(n))$ if and only if it is $\mathcal{O}(f(n) \cdot \log^k f(n))$ for some constant k.

case with general edge lengths, was given by Baier [Bai04]. However, it relies on linear programming. Baier also mentioned, without going into detail, the possibility of adapting the exponential length function approach of Garg and Könemann and Fleischer for LBF. Such an adaptation was recently given by Voborník [Vob16]; its running time is $\mathcal{O}(\varepsilon^{-2} \cdot |E|^2 \cdot K \log K)$, which makes it an FPTAS.

2.3.3 Minimum Length-Bounded Cut

As already mentioned in Section 2.1.2, the Minimum Length-Bounded Cut problem (LBC) is in some sense a dual problem to LBF. Specifically, the LP relaxation of its integer linear program formulation is dual to an LP formulation of LBF [AKV20].[6] However, this is where the analogy with Maximum Flow and Minimum Cut ends: for example, it is not the case that the value of the minimum K-bounded cut must always equal the size of the maximum K-bounded flow [AK71]; Baier et al. [Bai+10] have shown that there exist infinitely many instances where the ratio between the two values is $\mathcal{O}(|V|^{2/3})$, and at the same time, that this is the worst possible ratio. LBC is also much harder than LBF: it is solvable in polynomial time for $K \leq 3$ ($K \leq 4$ in the vertex-cut version), but $\mathcal{NP}$-hard for $K \geq 4$ ($K \geq 5$) [CK20]; see Itai, Perl, and Shiloach [IPS82b] for both the polynomial algorithm and $\mathcal{NP}$-hardness result for edge-cut LBC.

LBC is fixed parameter tractable (FPT)[7] when parametrised by K and the *tree-width* of the input graph, it is also FPT when parametrised by the *tree-depth* of the graph; see Dvořák and Knop [DK18]. Also look there for additional hardness results.

LBC also admits an approximation algorithm; the best result known to us is an $(0.44K + \mathcal{O}(1))$-approximation for the general case. At the same time, it is known that finding a 1.1715-approximation is $\mathcal{NP}$-hard; see Chlamtáč and Kolman [CK20] for both results and a more detailed overview.

2.3.4 General Methods

Probably the most straightforward way of tackling MCF and LBF is to formulate them as linear or convex programs, for example using the formulations (MCF-LP), (LBF-LP), (MCF-P), (LBF-P) we have given in Section 2.2, and then using general methods for solving linear and convex programs on these formulations. The obvious upside is that any advances in these general methods automatically lead to better algorithms for our problems, not to mention the relative simplicity of this approach if we view the methods as black boxes. Furthermore, this approach is so far the only one that is known to produce both exact and polynomial-time algorithms.

[6]Namely, to a so-called *path formulation*, which is different from (LBF-LP) and generally can have a superpolynomial number of variables – one per each K-bounded *st*-path. The LP sums them up to both calculate the total flow and to constrain the amount of flow through each edge.

[7]FPT is, informally, a class of problems for which there exists an algorithm running in time $n^{\mathcal{O}(1)} \cdot f(k)$ where n is the input size, f is a computable function and k is a parameter depending on the input.

A great downside is that using those techniques as black boxes usually does not provide much insight into the problem structure, nor allows us to utilise this structure to speed up the algorithms. To the best of our knowldge, the combinatorial interpretation of these algorithms is unclear, with the notable exception of the Frank-Wolfe algorithm. We have already spoken at great lengths about the importance of combinatorial algorithms in the introduction and all the points apply here as well. For these reasons, we are mentioning these methods only in passing.

For LPs, one standard approach is to use the simplex method. It is fast in practice, but many of its pivoting rules have exponential-time counterexamples and the existence of a polynomial-time pivoting rule is an open question. Another possibility is to use the ellipsoid method, which is provably (weakly) polynomial (first such algorithm was given by Khachiyan [Kha80]), or some of the interior-point methods which are also (weakly) polynomial (with the first such algorithm given by Karmarkar [Kar84]); for both approaches, many improvements have been made in the recent years. See [Sch99] or [GLS12] for an in-depth exposition to all three approaches and to LP in general.

For convex programs, one may also use interior-point methods, it is known that many polynomial algorithms exist [NN94]. Another approach is the Frank-Wolfe algorithm, which will be the subject of the next chapter. Of course, linear and convex optimisation are vast fields and there are many more approaches, but they are out of the scope of this thesis. The reader is advised to refer to [BV04] or the aforementioned books.

3. Frank-Wolfe

The Frank-Wolfe algorithm (FW, also known as *conditional gradient*) is a general method for smooth convex optimisation. Discovered in the 1950s [FW+56], it has recently regained popularity, especially in sparse optimisation and machine learning. Although the vanilla FW algorithm generally converges in time proportional to $1/\varepsilon$, where ε is the approximation error, multiple fast-converging variants have been proposed in the recent years [Wol70; LJ15; BPZ19].

FW solves a general convex optimisation problem of the form

$$\arg\min_{\mathbf{x}\in P} f(\mathbf{x}), \tag{3.1}$$

where $P \subseteq \mathbb{R}^n$ is a compact and convex *feasible region* and $f : P \to \mathbb{R}$ is convex and differentiable. We will denote the gradient of f at point $\mathbf{x}$ as $\nabla f(\mathbf{x})$; the differentiability of f guarantees that $\nabla f(\mathbf{x})$ exists for all $\mathbf{x} \in P$. The algorithm only needs to access P and f using an *LP oracle* and a *first-order oracle*, respectively:

LP Oracle

Input:	$\mathbf{c} \in \mathbb{R}^n$
Output:	$\mathbf{x} \in P$ maximising $\mathbf{c} \cdot \mathbf{x}$

First-Order Oracle

Input:	$\mathbf{x} \in \mathbb{R}^n$
Output:	$\nabla f(\mathbf{x})$ and $f(\mathbf{x})$

Before reviewing notable variants of FW, we will need a few definitions to be able to compare their performance and assumptions on P and f. We will also informally describe the general idea behind FW-based algorithms.

For simplicity's sake, from now on we assume that P is a bounded convex polytope. Throughout this chapter, $\mathbf{x}^*$ always denotes some fixed optimal solution to (3.1). Recall the definition of the *optimality gap:* $\mathrm{GAP}(\mathbf{x}) = f(\mathbf{x}) - f(\mathbf{x}^*) \geq 0$.

The chapter is organised as follows: in Section 3.1, we (re)define the needed mathematical terminology. The general outline of FW-based algorithms is given in Section 3.2 and the most common variants are discussed in Section 3.3. We then describe two of them, Pairwise FW and Away-Steps FW, in more detail in Section 3.4. We briefly sketch the known result about the convergence of Away-Steps FW in Section 3.5. Our original contribution begins in Section 3.6, where we describe our interpretation of Pairwise FW and Away-Steps FW on MCF, and later, LBF. This is followed by Section 3.7 where we use the general machinery from Section 3.5 to prove concrete bounds on convergence of Away-Steps FW applied to MCF and LBF. We conclude this chapter with Section 3.8 where we prove the main technical theorem from Section 3.5.

3.1 Strong Convexity and Smoothness

We start by formulating the terms we have already used in the description of FW above, and some more that will be useful in the subsequent convergence analysis.

Definition 3.1 (Differentiability, gradient)**.** Given $P \subseteq \mathbb{R}^n$ and $f : P \to \mathbb{R}$, we say that f is *differentiable* if, for every $\mathbf{x}_0 \in P$, f can be locally approximated by some affine mapping $L_{\mathbf{x}_0} : \mathbb{R}^n \to \mathbb{R}$, formally,

$$\lim_{\|\mathbf{x}-\mathbf{x}_0\| \to 0} \frac{f(\mathbf{x}) - L_{\mathbf{x}_0}(\mathbf{x})}{\|\mathbf{x} - \mathbf{x}_0\|} = 0.$$

Another interpretation is that $L_{\mathbf{x}_0}$ determines a hyperplane tangent to f at $\mathbf{x}_0$.

For a differentiable $f : P \to \mathbb{R}$ and a fixed $\mathbf{x}_0 \in P$, we define $\nabla f(\mathbf{x}_0) \in \mathbb{R}^n$ as a row vector whose i-th element is the partial derivative of f at $\mathbf{x}_0$ with respect to the i-th coordinate. Then, notation-wise, $\nabla f(\mathbf{x}_0) \cdot \mathbf{y}$ is a dot product of $\nabla f(\mathbf{x}_0)$ and $\mathbf{y}$. It can be shown that if f is differentiable, then the affine mapping locally approximating f at $\mathbf{x}_0$ is uniquely defined as $L_{\mathbf{x}_0}(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0)$.

Definition 3.2 (Convexity, strong convexity)**.** A differentiable function $f : P \to \mathbb{R}$ is *convex*, if:

$$\forall \mathbf{x}, \mathbf{y} \in P : \quad f(\mathbf{y}) - f(\mathbf{x}) \geq \nabla f(\mathbf{x})(\mathbf{y} - \mathbf{x}).$$

Furthermore, f is *μ-strongly convex* for $\mu > 0$, if:

$$\forall \mathbf{x}, \mathbf{y} \in P : \quad f(\mathbf{y}) - f(\mathbf{x}) \geq \nabla f(\mathbf{x})(\mathbf{y} - \mathbf{x}) + \frac{\mu}{2} \cdot \|\mathbf{x} - \mathbf{y}\|_2^2.$$

Clearly, μ-strong convexity implies regular convexity.

Definition 3.3 (Smoothness)**.** A convex function $f : P \to \mathbb{R}$ is *L-smooth* for $L > 0$, if:

$$\forall \mathbf{x}, \mathbf{y} \in P : \quad f(\mathbf{y}) - f(\mathbf{x}) \leq \nabla f(\mathbf{x})(\mathbf{y} - \mathbf{x}) + \frac{L}{2} \cdot \|\mathbf{x} - \mathbf{y}\|_2^2.$$

One can get an intuitive understanding of convexity by moving $f(\mathbf{x})$ to the right-hand side:

$$\forall \mathbf{x}, \mathbf{y} \in P : \quad f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})(\mathbf{y} - \mathbf{x}),$$

This equation says that if we fix any $\mathbf{x} \in P$, the tangent hyperplane $L_{\mathbf{x}}(\mathbf{y}) = f(\mathbf{x}) + \nabla f(\mathbf{x}) \cdot (\mathbf{y} - \mathbf{x})$ bounds f from below. A μ-strong convex function has an even stronger property: it is not only fully above any tangent hyperplane, it is above any "tangent μ-parabola", where μ controls how steep this parabola is.[1]

On the other hand, an L-smooth function lies fully *below* any "tangent L-parabola". Therefore, if a function is both μ-strongly convex and L-smooth, necessarily $\mu \leq L$. Functions that are strongly convex and smooth are important when it comes to desiging faster FW algorithms, since informally, they do not contain "too flat plateaus" (due to strong convexity) or "too sharp bumps" (due to smoothness).

[1]It may be not immediately clear that the right-hand side indeed is an (n-dimensional) parabola and not for example some "skewed" parabola-like shape. Nevertheless, one can verify that for a fixed $\mathbf{x}$, it can be written as $\mathbf{c} \cdot \mathbf{y} + \mu/2 \cdot \left(\sum_{i=1}^{n} y_i^2\right) + C$ for some $\mathbf{c}$ and C.

3.2 Algorithm Sketch

The most basic variants of the Frank-Wolfe algorithm work roughly as follows: in each iteration, we take the current solution $\mathbf{x}^i$ and try to improve $f(\mathbf{x}^i)$ as much as we can based on the information locally available at $\mathbf{x}^i$. To that end, we approximate f by its tangent hyperplane $L_{\mathbf{x}^i}(\mathbf{x}) = f(\mathbf{x}^i) + \nabla f(\mathbf{x}^i)(\mathbf{x} - \mathbf{x}^i)$ and then find its minimum over P:

$$\mathbf{y} = \arg\min_{\mathbf{x}\in P} L_{\mathbf{x}^i}(\mathbf{x}) = \arg\min_{\mathbf{x}\in P} \nabla f(\mathbf{x}^i) \cdot \mathbf{x}.$$

This can be done using one call to First-Order Oracle (to obtain the vector $\nabla f(\mathbf{x}^i)$) and one call to LP Oracle with $\mathbf{c} = \nabla f(\mathbf{x}^i)$. Then we jump from $\mathbf{x}^i$ in the direction of $\mathbf{y}$. Ideally, $f(\mathbf{y})$ would be the minimum, but we have to remember that $\mathbf{y}$ only solves a linear approximation, and not the original problem. Therefore we do a *line search* to pick the best point on the line segment $\operatorname{conv}(\mathbf{x}^i, \mathbf{y})$: we set $\mathbf{d} = \mathbf{y} - \mathbf{x}^i$ and then pick $\gamma \in [0, 1]$ such that $\mathbf{x}^{i+1} := \mathbf{x}^i + \gamma\mathbf{d} = (1-\gamma)\mathbf{x}^i + \gamma\mathbf{y}$ has the smallest possible value of f. There are multiple methods of picking γ: ideally the function $h_{\mathbf{x}^i,\mathbf{d}} : [0, 1] \to \mathbb{R}$ defined as

$$h_{\mathbf{x}^i,\mathbf{d}}(\gamma) = f(\mathbf{x}^i + \gamma\mathbf{d})$$

is simple enough that we can find $\gamma^* = \arg\min_{\gamma\in[0,1]} h_{\mathbf{x}^i,\mathbf{d}}(\gamma)$ algebraically. Otherwise, we can approximate it, for example by trying $\gamma = 1, \beta^1, \beta^2, \ldots, \beta^k$ for a global constant $\beta \in (0, 1)$ and k. It is known that both approaches yield asymptotically equal convergence for virtually all FW variants.

Observe how this approach contrasts with gradient descent: there, we always make a step in the direction of the gradient, which is, in a theoretical sense "the best possible direction". However, this may lead us out of the feasible region, and therefore we need to project our solution back into the polyhedron after each step. In contrast, Frank-Wolfe only takes the gradient direction as guide in the form of a cost function, finding the best direction within the polyhedron that maximises this cost. Therefore, no projection is needed.

3.3 Common Variants

There are many variants of FW. The original formulation, which we call Classic FW, dates back to 1956 [FW+56], and is nearly identical to what we have just described. For a fixed P and a fixed smooth and strongly convex f, it provably achieves the convergence rate of $1/\varepsilon$, i.e., it requires $t = \mathcal{O}(1/\varepsilon)$ iterations to converge to a solution $\mathbf{x}^t$ with optimality gap $\mathrm{GAP}(\mathbf{x}^t) \leq \varepsilon$.[2] Later, it was shown that it has actually a convergence rate of $\log(1/\varepsilon)$ under the same conditions on f and P, provided that the optimal solution $\mathbf{x}^*$ lies in the strict interior of P [GM86], i.e., it does not lie in any proper face of P. A convergence rate of $\log(1/\varepsilon)$ is also known as *geometric* or *linear convergence*, the latter probably because the order of magnitude of the optimality gap decreases linearly with the number of iterations.

[2]Of course, this disregards the dependence on P, f and the dimension n. We will see that for our problems and some FW variants, it is reasonable, but it generally needs not be.

The reason why Classic FW converges slowly if $\mathbf{x}^*$ lies on the boundary of P is called the *zig-zagging phenomenon* [LJ15]. When $\mathbf{x}^*$ lies in a face of dimension ≥ 2, the algorithm, after getting close enough to the face, alternates between moving towards several vertices of the face, making very little progress towards the optimum lying in its middle. That is because LP Oracle always returns a vertex of P and Classic FW can only move in its direction. To see the problem, imagine an isosceles triangle with the optimum lying in the middle of its base and f being an "amphitheatre" (e.g., $f(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}^*\|_2^2$). See Figure 3.1 for illustration.

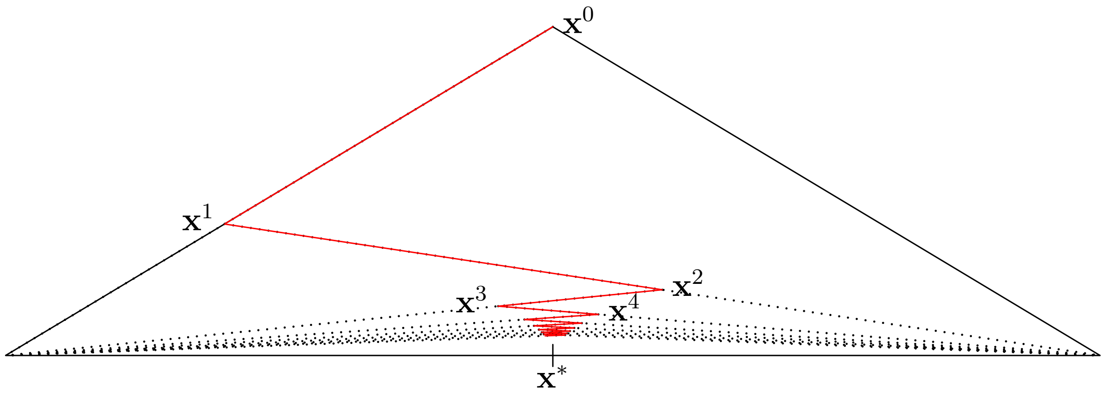

Figure 3.1: Illustration of the zig-zagging phenomenon. The dotted lines represent the directions on which line search is performed.

Several variants which tackle this problem have been proposed, most notably Away-Steps FW [Wol70] and Pairwise FW [LJ15]. Both of these algorithms combat the zig-zagging phenomenon by having a different method of choosing the improving direction. In short, Away-Steps FW also allows to move *away* from some vertex, picking either a regular step or away-step (whichever is better) in each iteration, while Pairwise FW combines the regular and away direction and always moves in this "compromise" direction. Surprisingly, both of these simple changes are enough to get rid of the zig-zagging phenomenon. In the next Section 3.4, we will describe Pairwise FW and Away-Steps FW in more detail.

In recent years, many advances were made and algorithms proposed, for example, Lazy FW [BPZ19], FW with restarts [KdP18] and many more. Their description is out of the scope of this thesis; see e.g. [KdP18] for references.

3.4 Pairwise and Away-Steps Frank-Wolfe

Pairwise FW (PFW) and Away-Steps FW (AFW) build on the observation that each solution $\mathbf{x}^i$ can be represented as a convex combination of some vertices of P, i.e., $\mathbf{x}^i = \sum_{\mathbf{v}\in\mathcal{A}} \alpha_{\mathbf{v}}\mathbf{v}$. In FW terminology, vertices are called *atoms*[3], the coefficients $\alpha_{\mathbf{v}} \in [0, 1]$ are called *weights* and the set of atoms with currently non-zero weight is called the *active set*. We denote the set of all atoms by $\mathcal{A}$ and

[3]Actually, the term is more general, and it is allowed to have atoms that are not vertices, but we will not distinguish between atoms and vertices for the sake of simplicity.

the active set by $\mathcal{S}$. By the definition of convex combination, the weights sum up to 1.

One can easily modify the Classic FW to maintain the active set and weights: recall that in each step, FW sets $\mathbf{x}^{i+1} := \mathbf{x}^i + \gamma\mathbf{d} = (1-\gamma)\mathbf{x}^i + \gamma\mathbf{y}$. To reflect that, we need to scale the weights of all atoms in $\mathcal{S}$ by $1-\gamma$, add $\mathbf{y}$ to $\mathcal{S}$ (unless it is already there) and increase $\alpha_\mathbf{y}$ (possibly from zero) by γ.

The key idea behind PFW and AFW is to change the way an improving direction is chosen. Besides the *FW direction* $\mathbf{d}_{\text{FW}} = \mathbf{y} - \mathbf{x}^i$ that points towards $\mathbf{y}$, we also calculate the *away direction* $\mathbf{d}_{\text{away}} = \mathbf{x}^i - \mathbf{v}$, where $\mathbf{v}$ is some atom in the active set. Namely:

$$\mathbf{v} = \arg\max_{\mathbf{v}\in\mathcal{S}} L_{\mathbf{x}^i}(\mathbf{v}) = \arg\max_{\mathbf{v}\in\mathcal{S}} \nabla f(\mathbf{x}^i)\cdot\mathbf{v},$$

That is, we find the atom that *maximises* the linearly approximated objective and then point away from it. Note that we do not need LP Oracle for this, since we can manually check all the atoms in $\mathcal{S}$. Also note that $\mathbf{x}^i + \mathbf{d}_{\text{away}}$ does not necessarily lie in the polyhedron.

Pairwise FW combines these two directions: $\mathbf{d} = \mathbf{d}_{\text{FW}} + \mathbf{d}_{\text{away}}$ and carries on with the line search for $\gamma \in [0, \gamma_{\max}]$ that minimises $\mathbf{x}^{i+1} := \mathbf{x}^i + \gamma\mathbf{d}$, where $\gamma_{\max} = \alpha_\mathbf{v}$. Note that $\mathbf{x}^i + \gamma\mathbf{d} = \mathbf{x}^i + \gamma(\mathbf{y} - \mathbf{x}^i + \mathbf{x}^i - \mathbf{v}) = \mathbf{x}^i + \gamma(\mathbf{y} - \mathbf{v})$, which makes the weight changes much simpler: $\alpha_\mathbf{y} := \alpha_\mathbf{y} + \gamma$, $\alpha_\mathbf{v} := \alpha_\mathbf{y} - \gamma$, and all the other weights remain unchanged. This also explains our choice of $\gamma_{\max}$.

Informally, while each step in Classic FW increases one weight at the expense of uniformly diluting all the other weights, in a Pairwise FW step, the weight is only being shifted between two atoms. One can intuitively see how this helps combat the zig-zagging phenomenon: it is much easier to bring an atom's weight to zero and thus remove it from the active set.

Away-Steps FW has a different strategy for picking $\mathbf{d}$. It compares $\mathbf{d}_{\text{FW}}$ and $\mathbf{d}_{\text{away}}$ and picks the one with smaller value of $\nabla f(\mathbf{x}^i)\cdot\mathbf{d}$. Then, if $\mathbf{d} = \mathbf{d}_{\text{FW}}$ was chosen, it sets $\gamma_{\max} = 1$ and the rest is the same as in Classic FW. Otherwise $\mathbf{d} = \mathbf{d}_{\text{away}}$, it sets $\gamma_{\max} = \alpha_\mathbf{x}/(1-\alpha_\mathbf{x})$ and later, the algorithm sets $\mathbf{x}^{i+1} := \mathbf{x}^i + \gamma\mathbf{d} = \mathbf{x}^i + \gamma(\mathbf{x}^i - \mathbf{v}) = (1+\gamma)\mathbf{x}^i - \gamma\mathbf{v}$, which means that the weights are updated as follows: $\alpha_\mathbf{x} := (1+\gamma)\alpha_\mathbf{x}$ for $\mathbf{x} \neq \mathbf{v}$, and $\alpha_\mathbf{v} := (1+\gamma)\alpha_\mathbf{v} - \gamma$. Observe that these updates are correct and that $\alpha_\mathbf{v}$ becomes zero exactly when $\gamma = \gamma_{\max}$.

See also [LJ15] for a more thorough description (including, for example, the stopping conditions, which we omitted here for simplicity), as well as a pseudocode of both Pairwise FW and Away-Steps FW.

3.5 Convergence Analysis

Since the full analysis of Frank-Wolfe, especially of AFW and PFW, is quite cumbersome, we will only summarise the main ideas and state the results in the form into which we can later plug in. For an exposition, we recommend the blog post series of Pokutta [Pok18a], which offers detailed explanation and links to other resources. For the complete analysis of AFW and PFW, see for example the original paper by Lacoste-Julien and Jaggi [LJ15].

From a very high-level perspective, the convergence analysis of Frank-Wolfe algorithms usually has at least two distinct parts. The overall goal is to arrive at some lower bound for the improvement in one iteration, ideally of the form $\mathrm{GAP}(\mathbf{x}^{i+1}) \leq \mathrm{GAP}(\mathbf{x}^i)(1-\rho)$ where ρ on depends on P and f; in that case, the convergence is linear.

On the one hand, one usually uses strong convexity to obtain an upper bound on $\mathrm{GAP}(\mathbf{x}^i) = f(\mathbf{x}^i) - f(\mathbf{x}^*)$ in terms of the current gradient: roughly, since f is above some parabola, $f(\mathbf{x}^*)$ cannot be very small and therefore $\mathrm{GAP}(\mathbf{x}^i)$ cannot be very large. On the other hand, smoothness gives us a lower bound on the progress, $\mathrm{GAP}(\mathbf{x}^i) - \mathrm{GAP}(\mathbf{x}^{i+1})$, in terms of the current gradient: as f is below some parabola, it cannot deviate very quickly from the tangent hyperplane we used as its local approximation, and moving in the direction of the gradient therefore is "reasonably good".

Then, one needs to combine these two bounds together. This step usually requires some additional ingredients, for example the knowledge that the optimum lies in the strict relative interior of P.

As already mentioned, Classic FW converges linearly if the solution lies in the strict relative interior, and there are instances where its convergence is $\Theta(1/\varepsilon)$ when the solution lies on the boundary. For Away-Steps FW, the following result due to [LJ15] is known:

Theorem 3.1 ([LJ15], Theorem 1)**.** *Let P be a polyhedron, let $f : P \to \mathbb{R}$ be μ-strongly convex and L-smooth and define* $\mathrm{diam}(P)$ *as the* diameter *of P, i.e., the maximum distance between any two points in P. Furthermore, let* $\mathrm{PWidth}(P)$ *be the so-called* pyramidal width *(as defined in Lacoste-Julien and Jaggi [LJ15]) of P. Then the sequence $\mathbf{x}^0, \dots, \mathbf{x}^t$ of iterates of the Away-Steps FW algorithm satisfies:*

$$\mathrm{GAP}(\mathbf{x}^t) \leq (1-\rho)^{t/2}\mathrm{GAP}(\mathbf{x}^0), \qquad \textit{where } \rho = \frac{\mu}{4L} \cdot \left(\frac{\mathrm{PWidth}(P)}{\mathrm{diam}(P)}\right)^2.$$

The $t/2$ term emerges to account for so-called drop steps, during which the algorithm does not necessarily make enough progress. For Away-Steps FW, it can be shown that drop steps amount for at most $1/2$ of all the steps, therefore the bound uses $t/2$ instead of t. Unfortunately, Pairwise FW introduces an additional type of steps called swap steps, of which there can be a very large amount; the authors bound it by $\mathcal{O}(|\mathcal{A}|!)$ per one good step, which leads to $t/\Omega(|\mathcal{A}|!)$ appearing instead of t. Until this bound improves, this makes Pairwise FW impractical for our purposes. Although, we should note that PFW seems to behave quite well in practice, so the bound on the number of swap steps could perhaps be significantly improved.

Lastly, we mention a weaker result that holds even for non-strongly convex functions and Classic FW. For a proof, see e.g. [Pok18a]. The result is well-known for Classic FW; for Away-Steps FW, it can be derived with a $1/2$ slowdown due to drop steps, by observing that the good steps improve the objective at least as much as in the Classic FW algorithm.

Theorem 3.2. *Let P be a polyhedron, let $f : P \to \mathbb{R}$ be convex and L-smooth and define* $\mathrm{diam}(P)$ *as the* diameter *of P. Then the sequence $\mathbf{x}^0, \dots, \mathbf{x}^t$ of iterates of* *the* Classic *FW algorithm satisfies:*

$$\mathrm{GAP}(\mathbf{x}^t) = \mathcal{O}\left(\frac{L\,\mathrm{diam}(P)^2}{t}\right),$$

and the same bound holds for Away-Steps *FW.*

3.5.1 Hölder Error Bound

As we will see later, our penalty function φ is smooth and convex, but not strongly convex. Fortunately, there is an alternative to strong complexity which can be used to prove effectively the same bounds.

Definition 3.4 (Hölder Error Bound (HEB), [KdP18])**.** A convex function $f : P \to \mathbb{R}$ is (θ, σ)-HEB, if there exists $\theta \in [0, 1]$ and $\sigma > 0$ such that

$$\min_{\mathbf{x}^* \in X^*} \|\mathbf{x} - \mathbf{x}^*\|_2 \leq \sigma(f(\mathbf{x}) - f^*)^\theta,$$

where f^* is the minimum value of f and $X^* \subseteq P$ is a set of all $\mathbf{x}^*$ where this minimum is attained.

Since we will be mostly interested in $(1/2, \sigma)$-HEB functions, we will use the shortcut "σ-HEB function" to stand for a $(1/2, \sigma)$-HEB function. In that case, the former equation can be rearranged as:

$$f(\mathbf{x}) - f^* \geq \frac{1}{\sigma} \cdot \min_{\mathbf{x}^* \in X^*} \|\mathbf{x} - \mathbf{x}^*\|_2^2.$$

Roughly, if a penalty function is σ-HEB, then the points with a small value of penalty must lie close to some optimal solution, with the strictness of this condition governed by σ.

As already mentioned, HEB is enough to obtain a bound similar to Theorem 3.3:

Theorem 3.3 (Global linear convergence for Away-Steps FW using HEB)**.** *Let P be a polyhedron, let $f : P \to \mathbb{R}$ be σ-HEB and L-smooth and define* $\mathrm{diam}(P)$ *as the diameter of P. Furthermore, let* $\mathrm{PWidth}(P)$ *be the pyramidal width of P. Then*

$$\mathrm{GAP}(\mathbf{x}^t) \leq (1 - \rho)^{t/2}\mathrm{GAP}(\mathbf{x}^0), \qquad \textit{where } \rho = \frac{1}{8\sigma^2 L} \cdot \left(\frac{\mathrm{PWidth}(P)}{\mathrm{diam}(P)}\right)^2.$$

As the proof is very technical, we will give it at the end of this chapter, in Section 3.8.

3.6 Interpreting Frank-Wolfe on MCF and LBF

Now we present our contribution: the interpretation of PFW and AFW applied to MCF and LBF. We will describe its application for (MCF-P) first; modifications for (LBF-P) are small and will be described at the end of the section.

Recall that in (MCF-P), the feasible region P is the set of all (feasible, but not necessarily satisfying) multiflows $\mathbf{f} \in \mathbb{R}^{E \times K}$, $\mathbf{f} \geq \mathbf{0}$ and the penalty is defined as

$$\varphi(\mathbf{f}) \coloneqq \sum_{e \in E} \left[\mathbf{f}^{\Sigma}(e) - c_e\right]_+^2 ,$$

where $\mathbf{f}^{\Sigma}(e) = \sum_{k=1}^{K} f_e^k$.

We need P to be bounded, so we add an additional constraint $\mathbf{f} \leq M \cdot \mathbf{1}$, where $M = \sum_{k=1}^{K} d_k$ is a sufficiently large upper bound on the total flow through an edge. Let us call (MCF-P) with this added constraint (MCF-P*) and observe that this modification does not affect the existence of a solution with $\varphi(\mathbf{f}) = 0$.[4]

Describing the algorithm amounts to describing LP Oracle and First-Order Oracle. Let us start with the latter.

One can easily verify that for $f(x) = [x + a]_+^2$, we obtain:

$$\left([x+a]_+^2\right)' = 2 \cdot [x+a]_+ \cdot [x+a]_+' = 2 \cdot [x+a]_+ ,$$

and thus, using that $\mathbf{f}^{\Sigma}(e) = \sum_{k=1}^{K} f_e^k$,

$$\frac{\partial \varphi}{\partial f_e^k} = \frac{\partial}{\partial f_e^k} \left[\mathbf{f}^{\Sigma}(e) - c_e\right]_+^2 = 2 \cdot \left[\mathbf{f}^{\Sigma}(e) - c_e\right]_+ .$$

Note that $\frac{\partial \varphi}{\partial f_e^k}(\mathbf{f}) \geq 0$, hence $\nabla\varphi(\mathbf{f}) \geq \mathbf{0}$. For a fixed edge e, all $\frac{\partial \varphi}{\partial f_e^k}(\mathbf{f})$ are equal across commodities.

LP Oracle solves the following linear program (we use the knowledge that it is always called with $\mathbf{c} = \nabla\varphi(\mathbf{f}) \geq \mathbf{0}$):

$$\begin{aligned}
&\text{minimise } \mathbf{c} \cdot \mathbf{f} \text{ for } \mathbf{f} \in \mathbb{R}^{E \times K}, \text{ given fixed } \mathbf{c} \in \mathbb{R}^{E \times K}, \mathbf{c} \geq \mathbf{0} \\
&\text{subject to } \mathbf{f}_\delta^k(v) = \begin{cases} -d_k & \text{if } v = s_k, \\ d_k & \text{if } v = t_k, \quad \forall k = 1, \dots, K \;\; \forall v \in V \\ 0 & \text{otherwise.} \end{cases} \\
&\qquad\qquad \mathbf{f} \quad \geq \mathbf{0} \\
&\qquad\qquad \mathbf{f} \quad \leq M \cdot \mathbf{1} \quad \text{for } M = \textstyle\sum_{k=1}^{K} d_k
\end{aligned} \tag{MCF-ORC}$$

First, we may notice that variables from different commodities do not interact with each other and the problem may thus be decomposed into K separate ones, each of which can be solved independently:

$$\begin{aligned}
&\text{minimise } \mathbf{c}^k \cdot \mathbf{f}^k \text{ for } \mathbf{f}^k \in \mathbb{R}^{E}, \text{ given fixed } \mathbf{c}^k \in \mathbb{R}^{E}, \mathbf{c}^k \geq \mathbf{0} \\
&\text{subject to } \mathbf{f}_\delta(v) = \begin{cases} -d_k & \text{if } v = s_k, \\ d_k & \text{if } v = t_k, \qquad \forall v \in V \\ 0 & \text{otherwise.} \end{cases} \\
&\qquad\qquad \mathbf{f}^k \quad \geq \mathbf{0} \\
&\qquad\qquad \mathbf{f}^k \quad \leq M \cdot \mathbf{1} \quad \text{for } M = \textstyle\sum_{k=1}^{K} d_k
\end{aligned} \tag{MCF-ORC-k}$$

[4]This follows from Lemma 1.7 and Corollary 1.8, since every $\mathbf{f}$ can be made to satisfy $\mathbf{f} \leq M \cdot \mathbf{1}$ by decomposing each commodity into weighted paths and positive cycles and then omitting the cycles.

Now observe that each subproblem is exactly a Minimum-Cost Fixed-Demand Flow (MCFDF) problem with weights $\mathbf{c}^k \geq \mathbf{0}$, source-sink pair (s_k, t_k), demand d_k and uniform capacities $M \cdot \mathbf{1}$. At this point, we might finish by saying that MCFDF can be solved combinatorially in polynomial time (one such algorithm will be presented in Chapter 5). However, the situation is even simpler than that:

Lemma 3.4. *Let $\mathbf{p}$ be a shortest $s_k t_k$-path $\mathbf{p}$ with respect to edge costs $\mathbf{c}^k$. Then the weighted path $d_k \cdot \mathbf{p}$ is an optimal solution to* (MCF-ORC-k)*.*

Proof. One can verify that $d_k \mathbf{p}$ is a satisfying solution, since $M > d_k$.[5]

Consider any $s_k t_k$-flow $\mathbf{f}^k$ with $|\mathbf{f}^k| = d_k$. We need to prove that $\mathbf{c}^k \cdot \mathbf{f}^k \geq \mathbf{c}^k \cdot d_k \mathbf{p}$. According to Lemma 1.7, $\mathbf{f}^k$ can be expressed as a conformal sum of weighted $s_k t_k$-paths and positive cycles: $\mathbf{f}^k = \sum_{i=0}^{q} \alpha_i \cdot \mathbf{C}_i + \sum_{i=0}^{r} \beta_i \cdot \mathbf{p}_i$. Hence, $\mathbf{c}^k \cdot \mathbf{f}^k = \sum_{i=0}^{q} \alpha_i \cdot \mathbf{c}^k \cdot \mathbf{C}_i + \sum_{i=0}^{r} \beta_i \cdot \mathbf{c}^k \cdot \mathbf{p}_i$. Without loss of generality, let $\mathbf{c}^k \mathbf{p}_1$ be minimal over all $\mathbf{c}^k \mathbf{p}_i$. Also realise that $\mathbf{c}^k \mathbf{p}'$ is simply the length of path $\mathbf{p}'$ with respect to edge lengths $\mathbf{c}^k$. Then:

$$\begin{aligned}
\mathbf{c}^k \cdot \mathbf{f}^k &= \sum_{i=0}^{q} \alpha_i \cdot \mathbf{c}^k \cdot \mathbf{C}_i + \sum_{i=0}^{r} \beta_i \cdot \mathbf{c}^k \cdot \mathbf{p}_i \geq \sum_{i=0}^{r} \beta_i \cdot \mathbf{c}^k \cdot \mathbf{p}_i \\
&\geq \sum_{i=0}^{r} \beta_i \cdot \mathbf{c}^k \cdot \mathbf{p}_1 = \mathbf{c}^k \cdot \mathbf{p}_1 \cdot \sum_{i=0}^{r} \beta_i = \mathbf{c}^k \cdot \mathbf{p}_1 \cdot d_k \geq \mathbf{c}^k \cdot \mathbf{p} \cdot d_k,
\end{aligned}$$

where we used, respectively, decomposition, nonnegativity of $a_i \mathbf{C}_i$ and $\mathbf{c}^k$, minimality of $\mathbf{c}^k \cdot \mathbf{p}_1$, reordering, the fact that $\sum_i \beta_i = |\mathbf{f}^k| = d_k$, and the fact that $\mathbf{p}$ is a shortest $s_k t_k$-path. □

Therefore, LP Oracle only needs to find a shortest $s_k t_k$-path (with respect to $\mathbf{c}^k$) for each commodity and then concatenate the results. This can be done in $\mathcal{O}(K|E|\log|V|)$ time using Dijkstra's algorithm. This could be further optimised by exploiting the fact that all $\mathbf{c}^k$ are identical and all the calls to Dijkstra's algorithm thus use the same weighted graph (e.g., by switching to all-pair shortest paths algorithms for cases with large K), but we consider this to be beyond the scope of this work.

3.6.1 Adaptation for LBF

The adaptation for LBF is very similar. We also augment (LBF-P) with a constraint $\mathbf{f} \leq M \cdot 1$, this time with $M = d$, thus forming (LBF-P*). Although φ and $\mathbf{f}^{\Sigma}(e)$ represent, formally speaking, different concepts than before, they behave the same in virtually all aspects.

Accordingly, $\frac{\partial \varphi}{\partial f_e^k} = 2 \cdot \left[\mathbf{f}^{\Sigma}(e) - c_e\right]_+$ and $\nabla \varphi(\mathbf{f}) \geq \mathbf{0}$. As for LP Oracle, this

[5]This is one of the reasons why (MCF-ORC-k) is much easier than general MCFDF.

time the linear subprogram is a single MCFDF in the layered graph G_K:

$$\begin{aligned}
&\text{minimise } \mathbf{c}\cdot\mathbf{f} \text{ for } \mathbf{f}\in\mathbb{R}^{E_K}\text{, given fixed } \mathbf{c}\in\mathbb{R}^{E_K}, \mathbf{c}\geq\mathbf{0} \\
&\text{subject to } \mathbf{f}_\delta(v) = \begin{cases} -d & \text{if } v^k = s^0, \\ d & \text{if } v^k = t^K, \\ 0 & \text{otherwise.}\end{cases} && \forall v^k \in V_K \\
&\qquad \mathbf{f}^\Sigma(e) \leq c_e && \forall e\in E \\
&\qquad \mathbf{f} \geq \mathbf{0} \\
&\qquad \mathbf{f} \leq M\cdot\mathbf{1} \quad \text{for } M = d
\end{aligned} \tag{LBF-ORC}$$

The same reasoning applies and this LP can be solved by a single call to Dijkstra's algorithm in the layered network G_K. The running time is $\mathcal{O}(|E_K|\cdot\log|V_K|) = \mathcal{O}(K\cdot|E|\log(|V|\cdot K))$.

3.7 Convergence Analysis for MCF and LBF

Now we analyse the convergence of Away-Steps FW adopted for MCF and LBF as described in the previous section. We examine the properties of φ and P and subsequently plug them into Theorem 3.3.

3.7.1 Smoothness

We start by showing that φ is L-smooth for $L = 2n = 2|E|\cdot K$. We go through the calculations with MCF in mind, but they are practically identical for LBF (only notation would change slightly).

Lemma 3.5. *Given $P\subseteq\mathbb{R}^{E\times K}\cong\mathbb{R}^n$ and φ from* (MCF-P)*, the inequality*

$$\varphi(\mathbf{y})-\varphi(\mathbf{x}) \leq \nabla\varphi(\mathbf{x})(\mathbf{y}-\mathbf{x}) + \frac{L}{2}\cdot\|\mathbf{x}-\mathbf{y}\|_2^2$$

holds with $L = 2n$ for every $\mathbf{x},\mathbf{y}\in P$.

Proof. Using the definition of φ and $\nabla\varphi$, we may rewrite the left- and right-hand side as, respectively:

$$\sum_{e\in E}\left[-c_e+\sum_{k=1}^K y_e^k\right]_+^2 - \left[-c_e+\sum_{k=1}^K x_e^k\right]_+^2$$

and

$$\sum_{e\in E} 2\cdot\left[-c_e+\sum_{k=1}^K x_e^k\right]_+\cdot\left(\sum_{k=1}^K y_e^k - x_e^k\right) + \frac{L}{2}\cdot\left(\sum_{k=1}^K\left(x_e^k-y_e^k\right)^2\right).$$

We will show that the inequality holds for every edge in the sum, and summing up those $|E|$ inequalities will prove L-smoothness. From now on, fix $e\in E$. Let $X := -c_e + \sum_{k=1}^K x_e^k$, similarly for Y. Using the new notation, we want to prove that:

$$[Y]_+^2 - [X]_+^2 \leq 2[X]_+(Y-X) + \frac{L}{2}\cdot\|\mathbf{x}_e-\mathbf{y}_e\|_2^2, \quad \text{i.e.,}$$

$$[Y]_+^2 - [X]_+^2 - 2[X]_+(Y-X) \leq \frac{L}{2}\cdot\|\mathbf{x}_e-\mathbf{y}_e\|_2^2. \tag{$*$}$$

Now we will get rid of $[\cdot]_+$ by assuming that $X \geq 0, Y \geq 0$. We will afterwards handle cases where this does not hold. The left-hand side simplifies to:

$$\begin{aligned} Y^2 - X^2 - 2X(Y-X) &= Y^2 - 2XY + X^2 = (X-Y)^2 \qquad (**) \\ &= \left(c_e - c_e + \sum_{k=1}^{K} x_e^k - y_e^k\right)^2 \leq \left(\sum_{k=1}^{K} \left|x_e^k - y_e^k\right|\right)^2 = \|\mathbf{x}_e - \mathbf{y}_e\|_1^2. \end{aligned}$$

Plugging this back into $(*)$ gives us

$$Y^2 - X^2 - 2X(Y-X) \leq \|\mathbf{x}_e - \mathbf{y}_e\|_1^2 \leq \frac{L}{2}\|\mathbf{x}_e - \mathbf{y}_e\|_2^2,$$

which holds with $L = 2n$. This follows from the known relation between ℓ_1- and ℓ_2-norms: $\forall \mathbf{a} \in \mathbb{R}^n,\ \|\mathbf{a}\|_1 \leq \sqrt{n}\|\mathbf{a}\|_2$.

What about the cases when $X < 0$ or $Y < 0$? When both $X < 0$ and $Y < 0$, then $(*)$ asks us to prove that $0 - 0 - 0 \leq \frac{L}{2} \cdot \|\mathbf{x}_e - \mathbf{y}_e\|_2^2$, which holds trivially. When $X < 0$ and $Y \geq 0$, the left-hand side of $(*)$ becomes Y^2. It holds that $0 \leq Y \leq Y - X$, and thus $Y^2 \leq (Y-X)^2 = (X-Y)^2$, and we may carry on from $(**)$ as before. Finally, when $X \geq 0$ and $Y < 0$, the left-hand side simplifies to $-X^2 - 2X(Y-X) \leq Y^2 - X^2 - 2X(Y-X)$ and we may carry on from $(**)$ again. $\square$

3.7.2 Convexity

Next we show that φ is convex. Combined with smoothness, this is enough to guarantee $\mathcal{O}(1/\varepsilon)$ convergence of all three mentioned FW algorithms.

Lemma 3.6. *Given $P \subseteq \mathbb{R}^{E \times K} \cong \mathbb{R}^n$ and φ from* (MCF-P)*, it holds that*

$$\forall \mathbf{x}, \mathbf{y} \in P \quad \varphi(\mathbf{y}) - \varphi(\mathbf{x}) \geq \nabla\varphi(\mathbf{x})(\mathbf{y} - \mathbf{x}).$$

Proof. We will reuse the procedure and notation from the proof of Lemma 3.5. The equation can be written as a sum of $|E|$ similar equations, one per each edge, and we will again prove those and then sum them up to obtain the result. Fix $e \in E$ and reuse the definition of $X \coloneqq -c_e + \sum_{k=1}^{K} x_e^k$, and similarly for Y. Then we need to prove:

$$[Y]_+^2 - [X]_+^2 - 2[X]_+(Y - X) \geq 0$$

This time, the proof is even more straightforward. When $X, Y \geq 0$, we get $(X-Y)^2 \geq 0$, when $X, Y < 0$, we get $0 \geq 0$, for $X \geq 0, Y < 0$, we get $X^2 - 2XY \geq X^2 \geq 0$ and for $X < 0, Y \geq 0$, we get $Y^2 \geq 0$. $\square$

Note that one can also prove the lemma directly by using the well-known properties of convexity, such as that the composition of a convex function with a convex and nondecreasing function is convex.

3.7.3 Strong Convexity

Lastly, we would like to prove μ-strong convexity. Combined with L-smoothness, it would guarantee linear convergence for PAIRWISE FW and AWAY-STEPS FW. However, it turns out in fact, that φ is generally not μ-strongly convex for any

$\mu > 0$. This is because any differentiable μ-strongly convex function needs to have at most one global minimum in the strict interior of P, since any $\mathbf{x}^*$ attaining this minimum has to satisfy $\nabla\varphi(\mathbf{x}^*) = \mathbf{0}$ and the condition on μ-strong convexity translates into all $\mathbf{y} \neq \mathbf{x}^*$ having $\varphi(\mathbf{y}) \geq \varphi(\mathbf{x}^*) + \mu/2 \cdot \|\mathbf{y} - \mathbf{x}^*\|_2^2 > \varphi(\mathbf{x}^*)$.

To be concrete, consider the following example with $V = \{s, u, v, t\}$, $E = \{s \to u, s \to v, u \to t, v \to t\}, \mathbf{c} = \mathbf{1}$, a single commodity with source s, sink t and demand $d = 1$. Then $P = \{\, (\alpha, 1-\alpha, \alpha, 1-\alpha) \mid \alpha \in [0,1] \,\}$ and $\varphi(\mathbf{x}) = 0 \;\; \forall \mathbf{x} \in P$. We can verify that all $\mathbf{x}$ in the strict interior of P have $\nabla\varphi(\mathbf{x}) = \mathbf{0}$, which, along with the observation from the previous paragraph, means that φ can be at most 0-strongly convex.

A possible solution to this problem is to change the penalty in some way, for example by adding a *regulariser*, i.e., chaning $\varphi(\mathbf{f})$ to $\varphi(\mathbf{f}) + \varepsilon \cdot R(\mathbf{f})$, where R is some strongly convex function and $\varepsilon > 0$ is a suitable constant; a sensible option is $R(\mathbf{f}) \coloneqq \|\mathbf{f}\|_2^2$. We can neither confirm nor deny that some sensible choice of R and ε would make the penalty strongly convex without significantly changing the algorithm, as we have tried to avoid this approach due to the complexity it adds to the algorithm. Additionally, it is not immediately clear how to achieve μ-strong complexity for a large enough μ and at the same time not let the $R(\mathbf{f})$ term dominate the penalty.

3.7.4 HEB and Linear Convergence

Theorem 4.8 from Section 4.3 guarantees that if the instance has a solution with $\varphi(\mathbf{x}) = 0$, then φ is an $(H^2|E|)$-HEB function, where H is the so-called *inflation rate* of the MCF / LBF polyhedron. In order to use Theorem 3.3 to prove linear convergence of Away-Steps FW for MCF and LBF, we only need to provide bounds on $\operatorname{diam}(P)$ and $\operatorname{PWidth}(P)$.

We know that P fits in the box with corners in $\mathbf{0}$ and $M \cdot \mathbf{1}$, where $M = \|\mathbf{d}\|_1$ (which equals $\sum_{k=1}^{K} d_k$ for MCF and d for LBF) and its diameter can therefore be bounded by the diameter of this box:

$$\operatorname{diam}(P) \leq \operatorname{diam}([0, M]^{E \times K}) = \|\mathbf{0} - \mathbf{1} \cdot M\|_2 = \|\mathbf{1}\|_2 \cdot M = \sqrt{E \times K} \cdot \|\mathbf{d}\|_1.$$

For $\operatorname{PWidth}(P)$, we use an equivalent characterisation due to Peña and Rodríguez [PR19], which says that the pyramidal width is equivalent to a so-called facial distance of P. This is defined as the minimum distance between a face and the convex hull of all vertices that do not lie on it. As P comes from the relaxed formulation, it is a cartesian product of K (for MCF) or 1 (for LBF) flow polytopes (with capacity M on each edge). It is a known fact that flow polytopes have integer vertices if the capacities are integer, which is the case here (even if $\mathbf{d}$ were not integer, we can set $M \coloneqq \lceil M \rceil$ without changing the problem), and hence P is also integer. This special structure can be exploited: for integer polytopes, it holds that $\operatorname{PWidth}(P)$ is bounded by $1/\operatorname{poly}(M, n)$ [Gup21].

Theorem 3.7. *Let $\mathcal{G} = (G, \mathcal{K})$ be a satisfying MCF instance with $G = (V, E, \mathbf{c})$ and $\mathcal{K} = \Big((s_k, t_k, d_k)\Big)_{k=1}^{K}$. Then Away-Steps FW finds a solution with penalty at most ε in time polynomial in input size, the ℓ_1-inflation rate H_{MCF}, $\|\mathbf{d}\|_1$ and* $\log(1/\varepsilon)$.

The same is true for a satisfying LBF instance $\mathcal{G} = (G, s, t, K)$ with demand d, with H_{LBF} instead of H_{MCF}.

Proof. The proof is the same for MCF and LBF once we define $\|\mathbf{d}\|_1 = d$ for LBF. In both cases, we know from Theorem 3.3 that

$$\mathrm{GAP}(\mathbf{x}^t) \leq (1-\rho)^{t/2}\mathrm{GAP}(\mathbf{x}^0), \quad \text{where } \rho = \frac{\mathrm{PWidth}(P)^2}{8\sigma^2 \cdot L \cdot \mathrm{diam}(P)^2},$$

and plugging the bound on diam(P) from above, the Hölder Error Bound from Theorem 4.8 and the bound on smoothness from Section 3.7.1 gives us:

$$\begin{aligned}\mathrm{GAP}(\mathbf{x}^t) \leq (1-\rho')^{t/2}\mathrm{GAP}(\mathbf{x}^0), \quad \text{where } \rho' &= \frac{\mathrm{PWidth}(P)^2}{8 \cdot H_{\mathrm{MCF}}^4 |E|^2 \cdot 2|E|K \cdot |E|K\|\mathbf{d}\|_1^2} \\ &= \frac{\mathrm{PWidth}(P)^2}{16 H_{\mathrm{MCF}}^4 |E|^4 K^2 \cdot \|\mathbf{d}\|_1^2}.\end{aligned}$$

The initial gap $\mathrm{GAP}(\mathbf{x}^0)$ can be bounded from above by the maximum attainable flow: $\mathrm{GAP}(\mathbf{x}^0) \leq (|E| \cdot K \cdot \|\mathbf{d}\|_1)^2$.

Finally, use Lemma 1.11 to get that we achieve $\varphi(\mathbf{f}^Q) = \mathrm{GAP}(\mathbf{f}^Q) \leq \varepsilon$ after

$$Q = 2 \cdot \frac{\log \mathrm{GAP}(\mathbf{x}^0)}{\rho'} \leq 4 \cdot \frac{16 H_{\mathrm{MCF}}^4 |E|^4 K^2 \cdot \|\mathbf{d}\|_1^2}{\mathrm{PWidth}(P)^2} \cdot \log(|E| \cdot K \cdot \|\mathbf{d}\|_1)$$

iterations. Recalling that $1/\mathrm{PWidth}(P) \in \mathrm{poly}(\|\mathbf{d}\|_1, |E|, K)$ proves the theorem. □

3.8 Proof of Theorem 3.3

Now we prove Theorem 3.3 from Section 3.5.1. As the full proof is heavily technical, we will entrust all the heavy lifting to the original linear convergence article [LJ15] and only point out the changes needed when moving from strong convexity to HEB. All references to lemmas and theorems in the proof will refer to Lacoste-Julien and Jaggi [LJ15] and we expect the reader to follow our progress there. Although we worked out most of the calculations independently (and take the responsibility for errors introduced in the process), all ideas are already contained in Pokutta [Pok18b] and Lacoste-Julien and Jaggi [LJ15].

Proof. We need to prove Theorem 8, the rest follows from its direct application to Theorem 1. The only place where strong convexity comes into question is in Equations (27) and (28) to upper bound the optimality gap. First, use respectively the convexity of φ, rearranging and f being σ-HEB to bound the primal gap; $\mathbf{x}^*$ is the optimal solution minimising $\|\mathbf{x}^* - \mathbf{x}^t\|$, $f^* := f(\mathbf{x}^*)$ and we use the notation $\langle \mathbf{x}, \mathbf{y} \rangle$ for dot product $\mathbf{x} \cdot \mathbf{y}$ for consistency with the original proofs:

$$\begin{aligned} f(\mathbf{x}^t) - f^* &= f(\mathbf{x}^t) - f(\mathbf{x}^*) \\ &\leq \left\langle \nabla f(\mathbf{x}^t), \mathbf{x}^t - \mathbf{x}^* \right\rangle \\ &= \frac{\langle \nabla f(\mathbf{x}^t), \mathbf{x}^t - \mathbf{x}^* \rangle}{\|\mathbf{x}^t - \mathbf{x}^*\|} \|\mathbf{x}^t - \mathbf{x}^*\| \\ &\leq \frac{\langle \nabla f(\mathbf{x}^t), \mathbf{x}^t - \mathbf{x}^* \rangle}{\|\mathbf{x}^t - \mathbf{x}^*\|} \sigma (f(\mathbf{x}^t) - f^*)^{1/2}. \end{aligned}$$

And thus, after dividing by $\sigma(f(\mathbf{x}^t) - f^*)^{1/2}$:

$$\frac{1}{\sigma}(f(\mathbf{x}^t) - f^*)^{1/2} \leq \frac{\langle \nabla f(\mathbf{x}^t), \mathbf{x}^t - \mathbf{x}^* \rangle}{\|\mathbf{x}^t - \mathbf{x}^*\|},$$

and rearranging yields:

$$f(\mathbf{x}^t) - f^* \leq \sigma^2 \left(\frac{\langle \nabla f(\mathbf{x}^t), \mathbf{x}^t - \mathbf{x}^* \rangle}{\|\mathbf{x}^t - \mathbf{x}^*\|} \right)^2 . \tag{3.2}$$

According to Theorem 3, we have

$$\frac{\langle -\nabla f(\mathbf{x}^t), \mathbf{d}_{\text{PFW}} \rangle}{\left\langle -\nabla f(\mathbf{x}^t), \frac{\mathbf{x}^t - \mathbf{x}^*}{\|\mathbf{x}^t - \mathbf{x}^*\|} \right\rangle} \geq \text{PWidth}(P),$$

where $\mathbf{d}_{\text{PFW}} = \mathbf{d}_{\text{FW}} + \mathbf{d}_{\text{away}}$ is the pairwise direction. As both the numerator and denominator are positive, we have:

$$\left\langle -\nabla f(\mathbf{x}^t), \frac{\mathbf{x}^t - \mathbf{x}^*}{\|\mathbf{x}^t - \mathbf{x}^*\|} \right\rangle \leq \frac{\langle -\nabla f(\mathbf{x}^t), \mathbf{d}_{\text{PFW}} \rangle}{\text{PWidth}(P)}.$$

Now we can plug this into (3.2):

$$\begin{aligned} f(\mathbf{x}^t) - f^* &\leq \sigma^2 \left(\frac{\langle \nabla f(\mathbf{x}^t), \mathbf{x}^t - \mathbf{x}^* \rangle}{\|\mathbf{x}^t - \mathbf{x}^*\|} \right)^2 = \sigma^2 \left\langle \nabla f(\mathbf{x}^t), \frac{\mathbf{x}^t - \mathbf{x}^*}{\|\mathbf{x}^t - \mathbf{x}^*\|} \right\rangle^2 \\ &= \sigma^2 \left\langle -\nabla f(\mathbf{x}^t), \frac{\mathbf{x}^t - \mathbf{x}^*}{\|\mathbf{x}^t - \mathbf{x}^*\|} \right\rangle^2 \leq \sigma^2 \frac{\langle -\nabla f(\mathbf{x}^t), \mathbf{d}_{\text{PFW}} \rangle^2}{\text{PWidth}(P)^2}. \end{aligned}$$

Now, observe that g_t that appears in Equation (28) equals $\langle -\nabla f(\mathbf{x}^t), \mathbf{d}_{\text{PFW}} \rangle$. And thus, we have:

$$h_t = f(\mathbf{x}^t) - f^* \leq \frac{g_t^2}{\frac{1}{\sigma^2} \text{PWidth}(P)^2}.$$

Using Theorem 6 on the original bound $h_t \leq g_t^2/(2\mu_f^A)$ gives us that the original bound says that $h_t \leq g_t^2/(2\mu \, \text{PWidth}(P)^2)$. Comparing this with our derived bound gives us that the original bound holds if we choose $\mu := 1/(2\sigma^2)$. We can therefore repeat the rest of the proof with $1/(2\sigma^2)$ instead of μ and $\text{PWidth}(P)^2/\sigma^2$ instead of μ_f^A. Since the rest of the proof does not use strong convexity any more, we can conclude with plugging $\mu = 1/(2\sigma^2)$ into Theorem 1 and obtaining exactly the needed bound. □

4. Circuits and Inflation

The circuits of a polyhedron are an important concept in linear programming and the study of polyhedra. A polyhedron's set of circuits gives us some useful characterisations: for example, the circuits contain all of the polyhedron's edge directions [Onn10, Lemma 2.18]. They are also a *universal test set*: given a linear objective $\mathbf{c}$, any point $\mathbf{x}$ in the polyhedron is optimal with respect to $\mathbf{c}$ if and only if no circuit $\mathbf{g}$ is an improving direction, that is, for no $\mathbf{g}$ does it hold that $\exists \alpha > 0 : \mathbf{x} + \alpha\mathbf{g} \in P$ and $\mathbf{c} \cdot \mathbf{g} < 0$. (The same property holds for any separable convex objective function.) This immediately suggest a simple iterative augmentation scheme: as long as $\mathbf{x}$ is not optimal, set $\mathbf{x} \coloneqq \mathbf{x} + \alpha\mathbf{g}$ for some feasible improving step $\alpha\mathbf{g}$; we shall discuss issues of convergence and other details later on.

Circuits appear in many contexts: from newer results, let us mention the study of matroids and transportation problems [Mil16], the classification of possible definitions of the graph diameter of polyhedra [BDF16], and the study of Hirsch conjecture (which proposes a relationship between the polytope's dimension and the minimum edge-distance between any two of its vertices) and its variants [BSY18]. Very recently, Dadush et al. [Dad+20] design an algorithm which solves LP in strongly-polynomial time essentially if the circuits of the constraint matrix are well-behaved. A curious reader will find further references in the aforementioned papers.

The main reference for this chapter is [BV19]; the reader is also referred to Borgwardt and Viss [BV20] and Onn [Onn10]. As in the whole thesis, all matrices are assumed integer, unless specified otherwise.

The chapter is organised as follows: in Section 4.1, we define circuits and describe their basic properties. Section 4.2 contains the description of the steepest-descent augmentation, which is a general method for solving linear programs using circuits. Our original results begin in Section 4.3. There, we give a new characterisation of polyhedra called the *inflation rate*, which may be seen as a variation on the so-called Hoffman constant. We then give our independently discovered result that a small norm of circuits implies a small inflation rate. In Section 4.4, we look a little closer at the circuits of MCF and single-commodity flows. Lastly, in Section 4.5 we describe a construction of exponential-sized circuits for MCF and sketch its adaptation for LBF.

4.1 Definitions and Basic Properties of Circuits

Definition 4.1 (Support, support-minimality)**.** Given a vector $\mathbf{x} \in \mathbb{R}^n$, its *support* is the set of its nonzero indices, i.e.:

$$\operatorname{supp}(\mathbf{x}) = \{\, i \in \{1, \dots, n\} \mid x_i \neq 0 \,\}.$$

Given a set $X \subseteq \mathbb{R}^n$, a vector $\mathbf{x} \in X$ is *support-minimal* over X if $\operatorname{supp}(\mathbf{y}) \not\subset \operatorname{supp}(\mathbf{x})$ (but possibly $\operatorname{supp}(\mathbf{y}) = \operatorname{supp}(\mathbf{x})$) for all $\mathbf{y} \in X$.

Definition 4.2 (Circuit direction, circuit [BV19])**.** A vector $\mathbf{g}$ is a *circuit direction* of a polyhedron $P = \{\, \mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{a} \wedge B\mathbf{x} \leq \mathbf{b} \,\}$ if $\mathbf{g} \in \ker(A) \setminus \{\mathbf{0}\}$ and

$B\mathbf{g}$ is support-minimal over $\{\, B\mathbf{x} \mid \mathbf{x} \in \ker(A) \setminus \{\mathbf{0}\} \,\}$. A vector $\mathbf{g}$ is a *circuit*, if it is a circuit direction and additionally $g_1, \ldots, g_n$ are all integer and their greatest common divisor is 1.

The set of circuits of a polyhedron $P = \{\, \mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{a} \wedge B\mathbf{x} \leq \mathbf{b} \,\}$ is denoted by $\mathcal{C}(A, B)$. For a polyhedron $P = \{\, \mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{a} \wedge \mathbf{x} \leq \mathbf{b} \,\}$, we will use a shorter notation $\mathcal{C}(A) = \mathcal{C}(A, I_n)$.

Intuitively, circuit directions are such vectors that, if added to some $\mathbf{x} \in \mathbb{R}^n$, they preserve (i.e., do not change the left-hand side of) all equalities and at the same time preserve as many inequalities with respect to inclusion as possible. That is, given a circuit direction, there cannot exist another direction preserving all equalities that preserves a strict superset of inequalities.

There are several ways one may define the set of circuits, because a circuit is fully characterised by its direction, i.e., it is invariant under scaling. Thus, we need some way to normalise circuits in order to obtain a set of some "canonical" circuits. One possible normalisation are the integrality and coprimality conditions given in Definition 4.2.[1] Another possibility is to require $\|\mathbf{g}\| = 1$. Note that in any case, circuits always come in *antipodal* pairs, i.e., if $\mathbf{g}$ is a circuit, then so is $-\mathbf{g}$.

Note that referring to circuits as "circuits of a polyhedron" without specifying how the polyhedron is defined does not make sense, since the circuits depend on the constraint matrices A and B and one polyhedron can have dramatically different sets of circuits depending on which constraints we choose for its definition. To emphasise this point, we will sometimes speak about the *circuits of matrices* A and B instead.

Specially, as $\mathcal{C}(A, B)$ *only* depends on A and B, it means that polyhedra differing only in $\mathbf{a}$ and $\mathbf{b}$ have the same set of circuits. Also note that multiplying the rows of A and B by nonzero coefficients does not change the set of circuits. This means $\mathcal{C}(A) = \mathcal{C}(A, -I_n)$ is also a set of circuits of a standard form polyhedron $P = \{\, \mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{a} \wedge \mathbf{x} \geq \mathbf{0} \,\}$.

An important property of circuits is that they contain as a subset the set of all possible edge directions of P:

Lemma 4.1. *Let $P = \{\, \mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{a} \wedge B\mathbf{x} \leq \mathbf{b} \,\}$ and let X be its set of edge directions, formally*

$$X = \{\, \mathbf{x} \in \mathbb{R}^n \mid \exists \alpha \in \mathbb{R}, \mathbf{p} \in \mathbb{R}^n \text{ s. t. } \operatorname{conv}(\mathbf{p}, \mathbf{p} + \alpha\mathbf{x}) \text{ is an edge of } P \,\}.$$

Let further $C = \{\, \alpha\mathbf{g} \mid \alpha \in \mathbb{R} \setminus \{0\}, \mathbf{g} \in \mathcal{C}(A, B) \,\}$ be the set of circuit directions of P. Then $X \subseteq C$.

Proof. Realise that for every edge direction $\mathbf{e}$, we can choose some row indices I of the matrix B, such that $\mathbf{e}$ is a nonzero solution to the system

$$\begin{aligned} A\mathbf{x} &= \mathbf{0} \\ B_I\mathbf{x} &= \mathbf{0} \end{aligned} \tag{4.1}$$

[1] Actually, here we use the requirement that A and B are integer; with real A and B it may not be possible to scale a circuit direction so that all its entries are integer.

formed from the original system by changing some inequalities to equalities and removing the remaining ones so that rank $\binom{A}{B_I} = n - 1$ (compare with the definition of face and edge in Section 1.1). Among all possible I, pick one that is inclusion-maximal, meaning that keeping any single removed inequality would increase the rank to n. In other words, the set of solutions is a line and keeping any more equalities would leave only a zero solution. Let $\overline{I}$ denote the complement of I, i.e., the indices of rows in B that have not been put into B_I.

We know that $\mathbf{e}$ is a non-zero solution to (4.1). Naturally $\mathbf{e} \in \ker(A) \setminus \{\mathbf{0}\}$, what remains to show is that $B\mathbf{e}$ is support-minimal over $\{ B\mathbf{x} \mid \mathbf{x} \in \ker(A) \setminus \{\mathbf{0}\} \}$. Observe that $\operatorname{supp}(B\mathbf{e}) = \{ i \in \{1, \ldots, n\} \mid (B\mathbf{e})_i \neq 0 \} = \{ i \mid B_{i*}\mathbf{e} \neq 0 \}$. We show that $\operatorname{supp}(B\mathbf{e}) = \overline{I}$: for $i \in I$, we have $B_{i*}\mathbf{e} = 0$ and therefore $i \notin \operatorname{supp}(B\mathbf{e})$. On the other hand, for $i \in \overline{I}$, it must hold that $B_{i*}\mathbf{e} \neq 0$, otherwise i could have been added to I and $\mathbf{e}$ would still remain a nonzero solution to Equation (4.1), which is a contradiction with the maximality of I.

From the same argument, $B\mathbf{e}$ must be support-minimal: if we had $\mathbf{x} \in \ker(A) \setminus \{\mathbf{0}\}$ such that $\operatorname{supp}(B\mathbf{x}) \subsetneq \operatorname{supp}(B\mathbf{e}) = \overline{I}$, we could pick $i \in \overline{I} \setminus \operatorname{supp}(B\mathbf{x})$ and augment I with it. Hence, $\mathbf{e}$ is a circuit direction. $\square$

See also [Onn10, Lemma 2.18] for a different proof.

4.2 Steepest-Descent Augmentation

A direct corollary of Lemma 4.1 is that the set of circuits is a *universal test set* for any linear program over P [BV20]. That is, given a linear program $\min\{ \mathbf{c} \cdot \mathbf{x} \mid \mathbf{x} \in P \}$ and a feasible solution $\mathbf{x}_0$, either $\mathbf{x}_0$ is an optimal solution or there exists a circuit $\mathbf{g} \in \mathcal{C}(A, B)$ and a step size $\alpha > 0$ such that $\mathbf{x}_0 + \alpha\mathbf{g} \in P$ and $\mathbf{c} \cdot (\mathbf{x}_0 + \alpha\mathbf{g}) < \mathbf{c} \cdot \mathbf{x}_0$. This is true since the same property holds for edges, which are always contained in the circuit directions by the previous lemma.

This fact gives rise to a whole family of *augmentation schemes* that start with an arbitrary feasible solution, and, in each iteration, gradually improve the current solution $\mathbf{x}$ by finding (in some sense) the steepest / the best / a good enough circuit direction and moving along it. This is repeated until the optimum is reached or the problem is found to be unbounded. These schemes are in a sense a generalisation of the simplex method, since they too move along edge directions of the polyhedron, but may also use additional directions and traverse its interior.

One such scheme is steepest-descent augmentation. Among strictly feasible circuits, i.e., circuits $\mathbf{g}$ such that $\mathbf{x} + \alpha\mathbf{g} \in P$ for some $\alpha > 0$, it picks the one that minimises $\mathbf{c} \cdot \mathbf{g} / \|B\mathbf{g}\|_1$, that is, the circuit that improves the objective the most per one unit of change in the inequalities. The algorithm then finds the greatest $\alpha > 0$ such that $\mathbf{x} + \alpha\mathbf{g} \in P$ and sets $\mathbf{x} \coloneqq \mathbf{x} + \alpha\mathbf{g}$.

The steepest-descent augmentation scheme has some surprising properties [BV19; BV20]. Firstly, the steepest-descent circuit direction $\mathbf{g}$ does not only minimise $\mathbf{c} \cdot \mathbf{g} / \|B\mathbf{g}\|_1$ over strictly feasible circuits, it minimises that over all strictly feasible directions. Secondly, the steepest-descent directions get less steep over time: if we denote the solutions and improving directions over time as $\mathbf{x}_1, \ldots, \mathbf{x}_t$

and $\alpha_1\mathbf{g}_1, \dots, \alpha_{t-1}\mathbf{g}_{t-1}$ where $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i\mathbf{g}_i$, then it holds that:

$$\frac{\mathbf{c}\mathbf{g}_1}{\|B\mathbf{g}_1\|_1} \le \frac{\mathbf{c}\mathbf{g}_2}{\|B\mathbf{g}_2\|_1} \le \dots \le \frac{\mathbf{c}\mathbf{g}_{t-1}}{\|B\mathbf{g}_{t-1}\|_1} < 0.$$

Furthermore, no $\mathbf{g}_i$ is ever repeated and at least one of n successive inequalities must hold strictly. These two facts imply that the number of steps is limited respectively by $|\mathcal{C}(A, B)|$ and $n \cdot |\{\, \mathbf{c}\mathbf{g}/\|B\mathbf{g}\|_1 \,;\, \mathbf{g} \in \mathcal{C}(A, B) \,\}|$. In other words, if the number of circuits or the number of different steepnesses is polynomial, the steepest-descent augmentation runs in polynomial time. For proofs of and intuitions behind these properties, see Borgwardt and Viss [BV19; BV20].

4.3 Inflating Polyhedra

In this section, we present our original result stating that, roughly, given a polyhedron with small circuits and a point outside of it that violates its constraints only by a small amount, then the point is close to the polyhedron. To the best of our knowledge, this result is original, although it is in spirit similar to a result of Jansen, Lassota, and Rohwedder [JLR20], see discussion below. An interesting property of our result is that its proof uses the steepest-descent algorithm under the hood even though the result itself is purely theoretical.

The general idea that points that "almost" satisfy polyhedron's inequalities are "close" to it is certainly not new: Hoffman [Hof52] proved that for a fixed matrix A and any right-hand side $\mathbf{a}$ such that $P = \{\, \mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} \le \mathbf{a} \,\}$ is nonempty, there exists a constant $c > 0$ (sometimes called *Hoffman constant* or *Hoffman condition number* in the literature) depending only on A, such that if we pick any $\mathbf{x} \notin P$, then the distance between $\mathbf{x}$ and the polyhedron is at most c times the norm of constraint violations of $\mathbf{x}$. Subsequent works have presented alternative proofs and improved bounds on the value of c [GHR95; Man81].

To our knowledge, our work is the first one to explicitly connect Hoffman's bound with circuits. Jansen, Lassota, and Rohwedder [JLR20] mention a similar result, but they state it with the Graver basis instead of circuits, only for n-fold matrices and the equality form formulation $A\mathbf{x} = \mathbf{a}$, $\mathbf{x} \ge \mathbf{0}$, and they do not make an explicit connection to the Hoffman constant. Furthermore, unlike our proof, it is not obvious how to adapt their approach to work for any $(A, \mathbf{a}, B, \mathbf{b})$-universal test set and not only the circuit set (or the Graver basis).

The connection with circuits is especially of importance for families of polytopes with provably small circuits, such as flow polytopes, polytopes defined by n-fold matrices, and generally polytopes defined by matrices with small primal or dual treedepth and small coefficients [KLO18].

We start by formalising Hoffman's constant in our context. As our definition is not equivalent to the usual definition of Hoffmann constant, but may instead be viewed as a granularisation thereof, we use our own terminology grounded in the intuitive notion of "inflating" polyhedra to avoid confusion. The connection between the two terms is discussed in Remark 4.4.

Definition 4.3 (Inflated polyhedron, inflation rate)**.** Let $P = \{\, \mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{a} \wedge B\mathbf{x} \le \mathbf{b} \,\}$, $A \in \mathbb{Z}^{m_A \times n}$, $B \in \mathbb{Z}^{m_B \times n}$ be a nonempty polyhedron and $\boldsymbol{\delta} \in \mathbb{R}^{m_B}$, $\boldsymbol{\delta} \ge \mathbf{0}$. We define the $\boldsymbol{\delta}$*-inflation* of P as the polyhedron $P^{\boldsymbol{\delta}} = \{\, \mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{a} \wedge B\mathbf{x} \le \mathbf{b} + \boldsymbol{\delta} \,\}$. Clearly $P \subseteq P^{\boldsymbol{\delta}}$.

For a norm $\|\cdot\|$, the nonempty polyhedron $P = \{\, \mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{a} \wedge B\mathbf{x} \leq \mathbf{b} \,\}$ has an *inflation rate* $H_{\|\cdot\|}(A, \mathbf{a}, B, \mathbf{b}) \geq 0$ defined as the infimum over all $R > 0$ such that for any $\boldsymbol{\delta} \geq \mathbf{0}$ and any point $\mathbf{x} \in P^{\boldsymbol{\delta}}$, there exists $\mathbf{y} \in P$ such that $\|\mathbf{x} - \mathbf{y}\| \leq R\|\boldsymbol{\delta}\|_1$. Formally:

$$H_{\|\cdot\|}(A, \mathbf{a}, B, \mathbf{b}) = \inf\{\, R > 0 \mid \forall \boldsymbol{\delta} \geq \mathbf{0}\ \forall \mathbf{x} \in P^{\boldsymbol{\delta}}\ \exists \mathbf{y} \in P : \ \ \|\mathbf{x} - \mathbf{y}\| \leq R\|\boldsymbol{\delta}\|_1 \,\}.$$

In other words, if $\mathbf{x}$ is not in P, then it is at most at a distance proportional to $H_{\|\cdot\|}(A, \mathbf{a}, B, \mathbf{b})$ and the magnitude of the violation.

Specially, let $H_1(A, \mathbf{a}, B, \mathbf{b}) \coloneqq H_{\|\cdot\|_1}(A, \mathbf{a}, B, \mathbf{b})$; similarly for $H_2(A, \mathbf{a}, B, \mathbf{b})$ and $H_\infty(A, \mathbf{a}, B, \mathbf{b})$.

Remark 4.2. We had to use infimum instead of minimum in the previous definition, since it is not immediately obvious that the minimum exists. This potentially gives rise to a situation when $\|\mathbf{x} - \mathbf{y}\| > H_{\|\cdot\|}(A, \mathbf{a}, B, \mathbf{b})\|\boldsymbol{\delta}\|_1$ for some $\mathbf{x}$ and $\boldsymbol{\delta}$, but $\|\mathbf{x} - \mathbf{y}\| \leq (H_{\|\cdot\|}(A, \mathbf{a}, B, \mathbf{b}) + \varepsilon)\|\boldsymbol{\delta}\|_1$ for all $\varepsilon > 0$. We can however instantly see that this cannot happen, since a real number cannot be strictly greater than some number, but smaller than all greater numbers. We therefore have

$$\forall \boldsymbol{\delta} \geq \mathbf{0}\ \forall \mathbf{x} \in P^{\boldsymbol{\delta}}\ \exists \mathbf{y} \in P : \ \ \|\mathbf{x} - \mathbf{y}\| \leq H_{\|\cdot\|}(A, \mathbf{a}, B, \mathbf{b})\|\boldsymbol{\delta}\|_1.$$

Remark 4.3. The same caveat as with circuits applies here: even though we are talking about the inflation rate of a *polyhedron*, $H_{\|\cdot\|}(A, \mathbf{a}, B, \mathbf{b})$, as the notation suggests, also depends on how the polyhedron is defined, i.e., it makes no sense to talk about the inflation rate without also specifying A, $\mathbf{a}$, B and $\mathbf{b}$.

Remark 4.4. The traditional Hoffman constant $H(A, B)$ is usually defined not only over all $\boldsymbol{\delta}$, but also over all possible $\mathbf{a}$ and $\mathbf{b}$. Hence, the relationship between our inflation rate and the Hoffman constant is captured by the following equation:

$$H(A, B) = \sup_{\mathbf{a}, \mathbf{b}} H(A, \mathbf{a}, B, \mathbf{b}),$$

where the supremum goes over all $\mathbf{a} \in \mathbb{R}^{m_A}, \mathbf{b} \in \mathbb{R}^{m_B}$ such that $\{\, \mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{a} \wedge B\mathbf{x} \leq \mathbf{b} \,\}$ is nonempty.

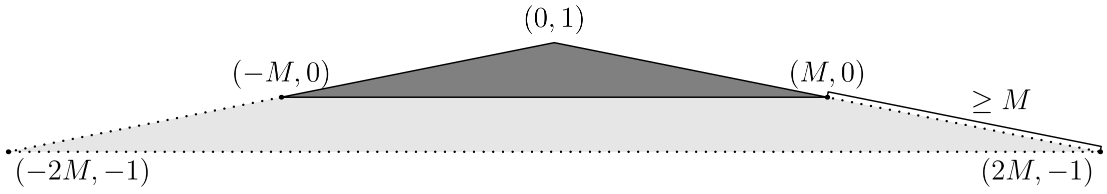

Figure 4.1: An example showing that the inflation rate can be arbitrarily large even for triangles in the plane.

One might think that the inflation rate is a trivial property and even that there has to exist a constant R such that all polyhedra have an inflation rate $\leq R$. This is not true even in the plane, since we may take an arbitrarily large $M \in \mathbb{N}$ and consider a "very obtuse" triangle $T = \operatorname{conv}((-M, 0), (M, 0), (0, 1))$ described by the following inequalities:

$$T = \{(x_1, x_2) \in \mathbb{R}^2 \mid -x_2 \leq 0 \wedge x_1 + Mx_2 \leq M \wedge -x_1 + Mx_2 \leq M\}.$$

Adding 1 to the first inequality, i.e., $(1,0,0)$-inflating T, will change the triangle to $T' = \operatorname{conv}((-2M,-1),(2M,-1),(0,1))$ and we can see that the distance of the point $(2M,1)$ from the original triangle T is at least M, and therefore the inflation rate of T is at least M. See Figure 4.1 for illustration.

Now we present our main result of this section: although there is no universal bound on the inflation rate holding for all polyhedra, we can bound the inflation rate of any particular polyhedron by the norm of its circuits. For that purpose, let us define $c_{\|\cdot\|}(A,B)$ for an arbitrary norm $\|\cdot\|$ as $c_{\|\cdot\|}(A,B) = \max_{\mathbf{g}\in\mathcal{C}(A,B)} \|\mathbf{g}\|$. Specially, $c_1(A,B) \coloneqq c_{\|\cdot\|_1}(A,B)$.

Theorem 4.5. *Let $P = \{\, \mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{a} \wedge B\mathbf{x} \le \mathbf{b} \,\}$ be a nonempty polyhedron, $\mathcal{C}(A,B)$ its set of circuits and $\|\cdot\|$ an arbitrary vector norm. Then P has an inflation rate at most $c_{\|\cdot\|}(A,B)$.*

We will prove the theorem with the help of the following lemma, which says that P has inflation rate $c_{\|\cdot\|}(A,B)$ provided that we only limit ourselves to inflations that change only one inequality.

Lemma 4.6. *Let $P = \{\, \mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{a} \wedge B\mathbf{x} \le \mathbf{b} \,\}$ be a nonempty polyhedron, $\mathcal{C}(A,B)$ its set of circuits and $\|\cdot\|$ a norm. Furthermore, let $\boldsymbol{\delta} = \delta\mathbf{e}^j$ where $j \in \{1,\dots,n\}$, $\delta > 0$ and $\mathbf{e}^j$ is the j-th unit vector. Then for any $\mathbf{x} \in P^{\boldsymbol{\delta}}$, there exists $\mathbf{y} \in P$ such that $\|\mathbf{x}-\mathbf{y}\| \le c_{\|\cdot\|}(A,B)\|\delta\mathbf{e}^j\|_1 = c_{\|\cdot\|}(A,B)\delta$.*

Proof. The cases with $\mathbf{x} \in P$ hold trivially. Let $\mathbf{x} = \mathbf{x}_1 \in P^{\boldsymbol{\delta}} \setminus P$. Now we will use steepest-descent augmentation in $P^{\boldsymbol{\delta}}$ that starts with $\mathbf{x}_1$, with $\mathbf{c} = B_{j*}$ as the cost vector. Clearly, $\mathbf{x} \in P \iff \mathbf{x} \in P^{\boldsymbol{\delta}} \wedge \mathbf{c}\cdot\mathbf{x} \le d_j$. Therefore, we modify the algorithm so that it stops once $\mathbf{c}\cdot\mathbf{x} = d_j$. Specially, should $\mathbf{c}(\mathbf{x}+\alpha\mathbf{g}) \le d_j$ after some step, we scale the step down so that $\mathbf{c}(\mathbf{x}+\alpha'\mathbf{g}) = d_j$, and stop.

This algorithm will always terminate and produce sequences $\mathbf{x}_1,\dots,\mathbf{x}_t$ and $\alpha_1\mathbf{g}_1,\dots,\alpha_{t-1}\mathbf{g}_{t-1}$ such that $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i\mathbf{g}_i$ and $\mathbf{x}_t \in P$. We claim that $\mathbf{x}_t$ is the sought point in P such that $\|\mathbf{x}_1 - \mathbf{x}_t\| \le c_{\|\cdot\|}(A,B)\delta$.

Observe that each augmentation step $\mathbf{x}_{i+1} \coloneqq \mathbf{x}_i + \alpha_i\mathbf{g}_i$ improves the objective by $\mathbf{c}\mathbf{x}_i - \mathbf{c}\mathbf{x}_{i+1} = \mathbf{c}\mathbf{x}_i - \mathbf{c}(\mathbf{x}_i + \alpha_i\mathbf{g}_i) = \alpha_i\mathbf{c}\mathbf{g}_i = \alpha_i B_{j*}\mathbf{g}_i \ge \alpha_i$. The inequality holds because each augmentation step strictly improves the objective, and both B_{j*} and $\mathbf{g}_i$ are integer and their dot product therefore has to be at least one. Since $\mathbf{c}\mathbf{x}_1 \le d_j + \delta$ and $\mathbf{c}\mathbf{x}_t = d_j$, it follows that $\delta \ge \mathbf{c}\mathbf{x}_1 - \mathbf{c}\mathbf{x}_t = \sum_{i=1}^{t-1}(\mathbf{c}\mathbf{x}_i - \mathbf{c}\mathbf{x}_{i+1}) \ge \sum_{i=1}^{t-1}\alpha_i$.

At the same time, we may bound the change of $\mathbf{x}$ between steps: $\|\mathbf{x}_i - \mathbf{x}_{i+1}\| = \|\mathbf{x}_i - \mathbf{x}_i - \alpha_i\mathbf{g}_i\| = \alpha_i\|\mathbf{g}_i\| \le \alpha_i c_{\|\cdot\|}(A,B)$. The last inequality follows from the bound on the norm of circuits. Summing up over all i and applying the bound from the previous paragraph yields:

$$\sum_{i=1}^{t-1}\|\mathbf{x}_i - \mathbf{x}_{i+1}\| \le \sum_{i=1}^{t-1}\alpha_i c_{\|\cdot\|}(A,B) \le \delta c_{\|\cdot\|}(A,B)$$

Now we apply the triangle inequality and obtain that

$$c_{\|\cdot\|}(A,B)\delta \ge \sum_{i=1}^{t-1}\|\mathbf{x}_i - \mathbf{x}_{i+1}\| \ge \left\|\sum_{i=1}^{t-1}\mathbf{x}_i - \mathbf{x}_{i+1}\right\| = \|\mathbf{x}_1 - \mathbf{x}_t\|,$$

which is what we wanted to prove. $\square$

With this lemma, proving Theorem 4.5 is straightforward:

Proof. We proceed by induction. Let $\boldsymbol{\delta}^{\leq i} = (\delta_1, \delta_2, \ldots, \delta_i, 0, \ldots, 0) \geq 0$. Define a sequence of polyhedra $P = P_0 \subseteq P_1 \subseteq \cdots \subseteq P_{m_B} = P^{\boldsymbol{\delta}}$, where $P_i = P^{\boldsymbol{\delta}^{\leq i}} = \{\, \mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{a} \wedge B\mathbf{x} \leq \mathbf{b} + \boldsymbol{\delta}^{\leq i} \,\}$. We will prove the following inductive claim: given $\mathbf{x} \in P_i$, there exists $\mathbf{y} \in P$ such that $\|\mathbf{x} - \mathbf{y}\| \leq c_{\|\cdot\|}(A, B) \sum_{j=1}^{i} \delta_j$. The claim trivially holds for $i = 0$, and proving it for $i = m_B$ will prove the theorem.

Let $1 \leq i \leq m_B$. Thanks to the way we defined the auxiliary polyhedra, Lemma 4.6 may be used on P_{i-1} and P_i, which gives us that for any $\mathbf{x} \in P_i$, there is $\mathbf{y} \in P_{i-1}$ such that $\|\mathbf{x} - \mathbf{y}\| \leq \delta_i$. Now we use the inductive hypothesis on P_{i-1} to obtain that for our $\mathbf{y} \in P_{i-1}$, there exists $\mathbf{z} \in P$ such that $\|\mathbf{y} - \mathbf{z}\| \leq \sum_{j=1}^{i-1} \delta_j$. Summing these two inequalities and using the triangle inequality, we obtain that $\|\mathbf{x} - \mathbf{z}\| \leq \|\mathbf{x} - \mathbf{y}\| + \|\mathbf{y} - \mathbf{z}\| \leq \sum_{j=1}^{i} \delta_j$, exactly as we wanted to show. ☐

Remark 4.7. The only place in the proof of Theorem 4.5 and Lemma 4.6 where we use the fact that $\mathcal{C}(A, B)$ is a set of circuits, is when it is used as a universal test set in the steepest-descent augmentation algorithm. We can easily replace $\mathcal{C}(A, B)$ with any set of integer vectors, provided that the steepest-descent still works with it – for example, we may as well have used the set of all edge directions. This way, we lose the other beneficial properties of the steepest-descent algorithm that depend on the test set being a circuit set, but as long as the algorithm still converges, we may use it in our proof. The precise requirement on the test set is stated formally in Definition 4.4.

As promised in Section 3.7, we now show an equivalence between the inflation rate and a Hölder Error Bound of a related function. A similar observation was made by Beck and Shtern [BS17] and probably others. For the sake of simplicity, we only show it in our MCF / LBF setting although it could be generalised for arbitrary polyhedra.

Theorem 4.8. *Let $\mathcal{G}$ be a satisfying MCF / LBF instance, let $P = \{\, \mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{a} \wedge B\mathbf{x} \leq \mathbf{b} \,\} \neq \emptyset$ be a feasible set of its exact LP formulation* (MCF-LP) / (LBF-LP)*, let $Q \supseteq P$ be a feasible set of its relaxed convex formulation* (MCF-P) / (LBF-P) *or* (MCF-P*) / (LBF-P*) *and let $\varphi : Q \to \mathbb{R}$ be the associated penalty function. If P has ℓ_1-inflation rate $H = H_1(A, \mathbf{a}, B, \mathbf{b})$, then φ is $(H^2|E|)$-HEB.*

On the other hand, assuming that Q comes from (MCF-P) / (LBF-P)*, if φ is σ-HEB, then the ℓ_2-inflation rate of P satisfies $H_2(A, \mathbf{a}, B, \mathbf{b}) \leq \sqrt{\sigma}$.*

Proof. It is clear that P is exactly the set of $\mathbf{x}^* \in Q$ such that $\varphi(\mathbf{x}^*) = 0$.

We start with the first part of the proof. Pick $\mathbf{x} \in Q$ and set $\boldsymbol{\delta} = [B\mathbf{x} - \mathbf{b}]_+ = \left[\mathbf{x}^\Sigma - \mathbf{c}\right]_+$. Then $\mathbf{x}$ lies in $P^{\boldsymbol{\delta}}$ and Definition 4.3, along with the relation between ℓ_1- and ℓ_2-norms, guarantee the existence of $\mathbf{y} \in P$ such that $H \cdot \|\boldsymbol{\delta}\|_1 \geq \|\mathbf{x} - \mathbf{y}\|_1 \geq \|\mathbf{x} - \mathbf{y}\|_2$ and thus, $\|\boldsymbol{\delta}\|_1 \geq \|\mathbf{x} - \mathbf{y}\|_2 / H$.

Now, realise that

$$\varphi(\mathbf{x}) = \sum_{e \in E} \left[\mathbf{x}^\Sigma - \mathbf{c}\right]_+^2 = \sum_{e \in E} \delta_e^2 = \|\boldsymbol{\delta}\|_2^2.$$

Using the relation between ℓ_1- and ℓ_2-norms for $\boldsymbol{\delta} \in \mathbb{R}^E$, and using that $\varphi^* =$

$\min_{\mathbf{x}' \in Q} \varphi(\mathbf{x}') = \varphi(\mathbf{y}) = 0$,

$$\begin{aligned}\varphi(\mathbf{x}) - \varphi^* = \varphi(\mathbf{x}) = \|\boldsymbol{\delta}\|_2^2 \geq \frac{\|\boldsymbol{\delta}\|_1^2}{|E|} &\geq \frac{1}{H^2 \cdot |E|} \cdot \|\mathbf{x} - \mathbf{y}\|_2^2 \\ &\geq \frac{1}{H^2 \cdot |E|} \min_{\mathbf{x}^* \in P} \|\mathbf{x} - \mathbf{x}^*\|_2^2,\end{aligned}$$

which is exactly the definition of an $(H^2|E|)$-HEB function.

For the second part, we need to show that for any $\boldsymbol{\delta}$ and $\mathbf{x} \in P^{\boldsymbol{\delta}}$, there exists $\mathbf{z} \in P$ close enough to $\mathbf{x}$. Fix $\mathbf{x}$ and $\boldsymbol{\delta}$ and define $\boldsymbol{\delta}' := \left[\mathbf{x}^{\Sigma} - \mathbf{c}\right]_+$. Necessarily $\boldsymbol{\delta}' \leq \boldsymbol{\delta}$ and $\|\boldsymbol{\delta}'\|_2 \leq \|\boldsymbol{\delta}\|_2$. We know that

$$\frac{1}{\sigma} \min_{\mathbf{x}^* \in P} \cdot \|\mathbf{x} - \mathbf{x}^*\|_2^2 . \leq \varphi(\mathbf{x}) - \varphi^*.$$

Picking $\mathbf{x}^* := \mathbf{z} \in P$ that minimises the expression and using the relationship between ℓ_1- and ℓ_2-norms yields:

$$\begin{aligned}\frac{1}{\sigma} \cdot \|\mathbf{x} - \mathbf{z}\|_2^2 . \leq \varphi(\mathbf{x}) - \varphi^* = \varphi(\mathbf{x}) &= \|\boldsymbol{\delta}'\|_2^2 \leq \|\boldsymbol{\delta}\|_2^2 \\ &\leq \|\boldsymbol{\delta}\|_1^2,\end{aligned}$$

and therefore, $\|\mathbf{x} - \mathbf{z}\|_2^2 \leq \sigma \|\boldsymbol{\delta}\|_1^2$ and $\|\mathbf{x} - \mathbf{z}\|_2 \leq \sqrt{\sigma} \|\boldsymbol{\delta}\|_1$. Since we have proven this for arbitrary $\boldsymbol{\delta}$ and $\mathbf{x}$, it follows that $H_2(A, \mathbf{a}, B, \mathbf{b}) \leq \sqrt{\sigma}$. □

Perhaps anticlimatically, it turns out that polyhedra of MCF and LBF *can* in fact have large – even exponential-sized – circuits, and we will shortly present such instances. Although these instances serve as evidence of hardness of MCF and LBF, this does not automatically mean that they have large inflation rates. The theorem may well still give us a polynomial bound if we use a different universal test set, and bounds on inflation rate may still be proven by completely different means. It is also worth noting that the MCF and LBF LPs have fairly special right-hand sides, which makes it difficult to construct an instance where the steepest-descent augmentation needs to use a large circuit at some point, or where such a circuit appears as an edge of the polyhedron. The following definition summarises the properties of a test set that would suffice for plugging it into Theorem 4.5 and obtaining a bound on the inflation rate.

Definition 4.4 (MCF-/LBF-universal test set)**.** Let $\{\, \mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{a} \wedge B\mathbf{x} \leq \mathbf{b} \,\}$ be a feasible polyhedron of an MCF instance. We say that a set S of integer vectors is an $(A, \mathbf{a}, B, \mathbf{b})$-*universal* test set if, for any choice of the cost function $\mathbf{c} = B_{j*}$, any vector $\boldsymbol{\delta} \geq \mathbf{0}$ and any $\mathbf{x} \in P^{\boldsymbol{\delta}}$ that is not optimal with respect to $\mathbf{cx}$, there exists $\varepsilon > 0$ and $\mathbf{s} \in S$ such that $\mathbf{x} + \varepsilon\mathbf{s} \in P^{\boldsymbol{\delta}}$ and the cost improves, i.e., $\mathbf{cs} < 0$. For brevity, we call these test sets *MCF-universal* test sets (with respect to the current instance). Analogically for LBF instances and LBF-universality.

Any universal test set is also an $(A, \mathbf{a}, B, \mathbf{b})$-universal test set, but not vice versa.

4.4 Examining MCF Circuits

In this section, we take a closer look at the circuits of MCF, and, indirectly, of LBF, since the circuits of both problems have a similar structure.

Recall that $\mathbf{x}$ is a circuit direction if and only if $\mathbf{x} = \alpha\mathbf{g}$ for some $\alpha \neq 0$ and a circuit $\mathbf{g}$. As a warm-up, we show that for a single-commodity flow, circuits are exactly undirected cycles:

Lemma 4.9. *Let $P = \{\, \mathbf{f} \in \mathbb{R}^E \mid A\mathbf{f} = \mathbf{a} \wedge \mathbf{f} \geq 0 \wedge \mathbf{f} \leq \mathbf{c} \,\} = \{\, \mathbf{f} \in \mathbb{R}^E \mid A\mathbf{f} = \mathbf{a} \wedge B\mathbf{f} \leq \mathbf{b} \,\}$, for some B and $\mathbf{b}$, be a (single-commodity) flow polytope. Then $\mathbf{g} \in \mathbb{R}^E$ is a circuit if and only if it is an (incidence vector of an) undirected cycle.*

Proof. We know that all circuits lie in $\ker(A)\setminus\{\mathbf{0}\}$, which, in this case, is the space of all non-zero circulations (since we can recall that the i-th row of $A\mathbf{f}$ measures the excess flow at vertex i and circulations are exactly those vectors with zero excess flow everywhere). Hence, circuit directions are exactly those circulations $\mathbf{g}$ for which $B\mathbf{g}$ is support-minimal. Since $B = (-I_E, I_E)^T$, this is equivalent with $\mathbf{g}$ being support-minimal. Thus, we want to prove that undirected weighted cycles are exactly support-minimal circulations; the transition from circuit directions to circuits follows trivially by scaling.

The implication "undirected cycle $\implies$ support-minimal" is trivial, since one can immediately see that any nonzero $\mathbf{x}$ with smaller support than the cycle cannot be a circulation. On the other hand, let $\boldsymbol{\Delta}$ be a circulation that is not a weighted cycle. By Lemma 1.4, it can be decomposed into a conformal sum of nonzero weighted cycles, $\boldsymbol{\Delta} = \sum_{i=1}^{\ell} \alpha_i \mathbf{C}^i$, and necessarily $\ell > 1$. But then, thanks to conformality, $\alpha_i \mathbf{C}^i$ has a strictly smaller support, which contradicts the support-minimality of $\boldsymbol{\Delta}$. $\square$

The situation is more complicated in the multi-commodity case: the nonnegativity constraints stay the same, but the capacity constraints change from $\mathbf{f} \geq \mathbf{c}$ to $\mathbf{f}^{\Sigma} \geq \mathbf{c}$. (Recall that $\mathbf{f}^{\Sigma}(e)$ is the total flow through edge e.) However, the observation about $\ker(A)$ still holds, and all circuits are therefore multicirculations. Summed up, we get that circuit directions in MCF are exactly (multi-)circulations $\boldsymbol{\Delta}$ that are support-minimal with respect to both $\boldsymbol{\Delta}$ and $\boldsymbol{\Delta}^{\Sigma}$, in other words, for which the set $\operatorname{supp}(\boldsymbol{\Delta}) \cup \operatorname{supp}(\boldsymbol{\Delta}^{\Sigma}) = \{\, (e,k) \in E \times K \mid \Delta_e^k \neq 0 \,\} \cup \{\, e \in E \mid \boldsymbol{\Delta}^{\Sigma}(e) \neq 0 \,\}$ is inclusion-minimal (with respect to other circulations).

If we look at $\operatorname{supp}(\boldsymbol{\Delta}^{\Sigma})$, we may see that it is a subset of the set of all edges that participate in $\operatorname{supp}(\boldsymbol{\Delta})$, since all edges that have a nonzero net flow change must have a nonzero flow change in some commodity, but on the other hand, on some edges the flows may cancel out and the net flow is zero. This motivates the following definition.

Definition 4.5. An edge $e \in E$ is a *balancing edge* with respect to a circulation $\boldsymbol{\Delta}$ if $\Delta_e^k \neq 0$ for some k, but $\boldsymbol{\Delta}^{\Sigma}(e) = 0$. Denote the set of all balancing edges induced by $\boldsymbol{\Delta}$ as $\operatorname{bal}(\boldsymbol{\Delta})$.

We can immediately see that $\operatorname{bal}(\boldsymbol{\Delta}) = \{\, e \in E \mid \exists k :\ (e,k) \in \operatorname{supp}(\boldsymbol{\Delta}) \,\} \setminus \operatorname{supp}(\boldsymbol{\Delta}^{\Sigma})$ and therefore $\boldsymbol{\Delta}$ is a circuit direction if and only if $\operatorname{supp}(\boldsymbol{\Delta})$ is inclusion-minimal and at the same time, $\operatorname{bal}(\boldsymbol{\Delta})$ is inclusion-maximal; that is, if $\operatorname{supp}(\boldsymbol{\Gamma}) \subsetneq \operatorname{supp}(\boldsymbol{\Delta})$ for some other circulation $\boldsymbol{\Gamma}$, then $\operatorname{bal}(\boldsymbol{\Gamma}) \not\supseteq \operatorname{bal}(\boldsymbol{\Delta})$, and vice versa, if $\operatorname{bal}(\boldsymbol{\Gamma}) \supsetneq \operatorname{bal}(\boldsymbol{\Delta})$, then $\operatorname{supp}(\boldsymbol{\Gamma}) \not\subseteq \operatorname{supp}(\boldsymbol{\Delta})$. Colloquially, $\boldsymbol{\Delta}$ is a circuit direction if it balances as many edges while using as few edges as possible.

We will use this brief characterization in the next section when proving that some particular circulations are circuits.

4.5 Exponential Circuits for MCF and LBF

As promised, we now show there exist infinitely many MCF instances that have exponentially-sized circuits. Later, in Section 4.5.1, we use the same idea for LBF. Pick a parameter $p \in \mathbb{Z}, p > 1$ and construct a network as per the following scheme displayed in Figure 4.2. The network consists of p partially overlapping blocks, each consisting of 5 vertices and 7 edges, from which 4 vertices and 2 edges are shared with the adjacent blocks (except for the leftmost and rightmost block).

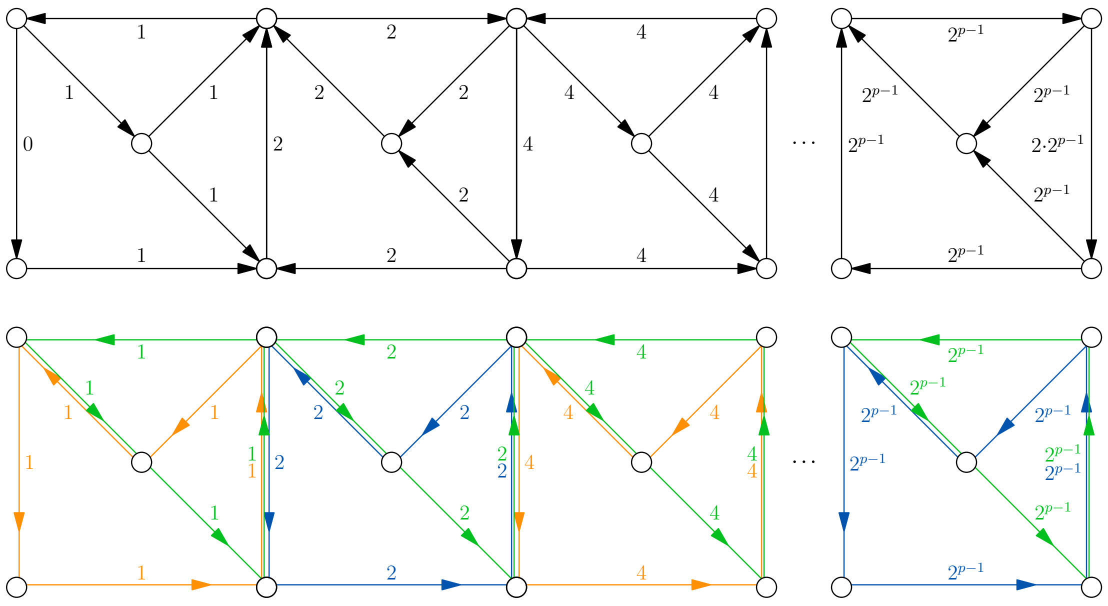

Figure 4.2: Top: a schema of an MCF instance with $3p + 2$ vertices, $6p + 1$ edges and 3 commodities (not visible). Without loss of generality, it displays the case with even p. Bottom: a circuit $\mathbf{g}$ of norm $\|\mathbf{g}\|_2 \geq 2^p$ in this network. When a flow is portrayed to go against the edge's prescribed direction, it simply means the flow is negative there.

The norm of the displayed circuit can be lower bounded as:

$$\|\mathbf{g}\|_2 = \sqrt{\sum_{i=1}^{p-1} 9 \cdot (2^i)^2} \geq \sqrt{9 \cdot 2^{2(p-1)}} = 3 \cdot 2^{p-1} \geq 2^p$$

For simplicity, we pick $p = 4$ and reproduce the circuit of this MCF instance in a more readable fashion, see Figures 4.3 and 4.4.

Now that we have described the circuit, what remains to be proven is that it *is* a circuit. We start with an auxiliary lemma.

Lemma 4.10. *Let* $\mathbf{g}$ *be the circulation depicted in Figure 4.2 and let* $\mathbf{h}$ *be a circulation with* $\text{supp}(\mathbf{h}) \cup \text{supp}(\mathbf{h}^{\Sigma}) \subseteq \text{supp}(\mathbf{g}) \cup \text{supp}(\mathbf{g}^{\Sigma})$*. Let* β*,* $1 \leq \beta \leq p$*, be a block, and* $k \in \{y, b, g\}$ *be a commodity and further assume that the commodity participates nontrivially in the block, i.e., if* $k = y$*, then* p *is odd, and if* $k = b$*, then* p *is even. Denote the seven edges of this block as* $E_\beta = \{e_1, \ldots, e_7\}$ *such that* e_1 *is the rightmost edge and the other edges are assigned in an arbitrary order. Then the following holds: for all* $e \in E_\beta$ *such that* $g_e^k = 0$*, also* $h_e^k = 0$*.*

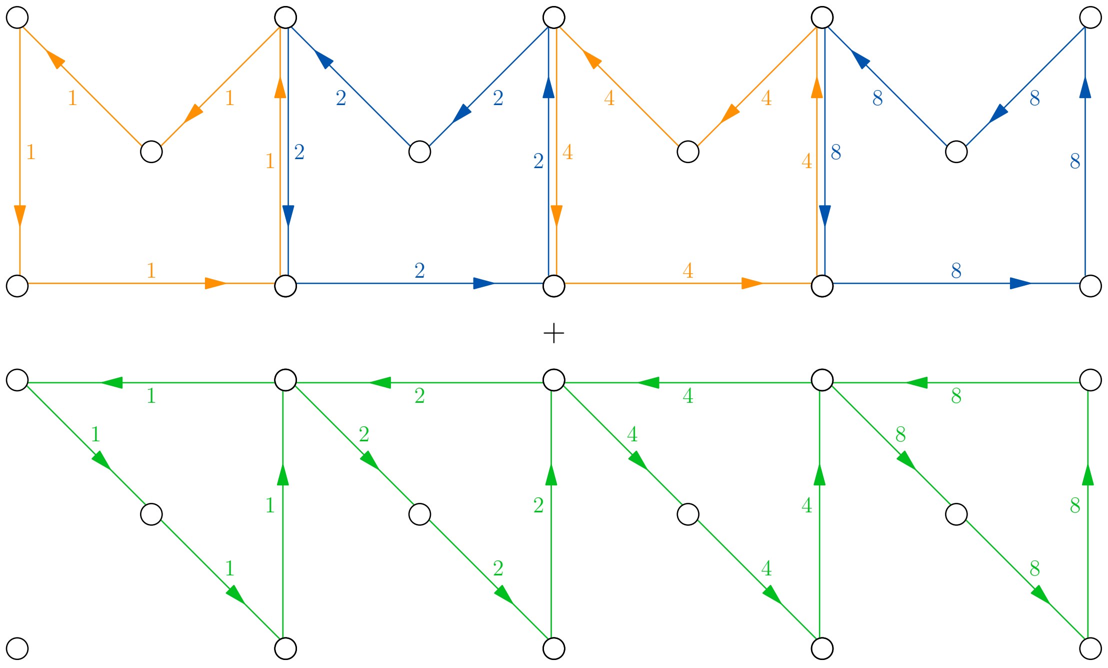

Figure 4.3: A large MCF circuit with $p = 4$. The depiction of the circuit is split into two parts for better readability.

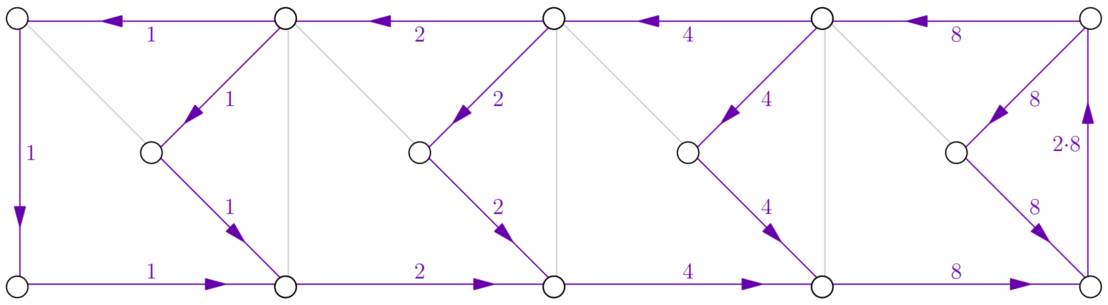

Figure 4.4: A large MCF circuit with $p = 4$, depiction of $\mathbf{g}^{\Sigma}$. Light grey edges are those where the total flow does not change at all (i.e., $\mathbf{g}^{\Sigma}(e) = 0$), while violet edges depict non-zero $\mathbf{g}^{\Sigma}(e)$, with the sign determined by the edge direction.

Furthermore, there exists some $x^k_\beta \in \mathbb{R}$ (potentially zero) such that for all the remaining edges (with $g^k_e \neq 0$), $h^k_e = x^k_\beta$.

Proof. For the sake of a clearer exposition, let us follow the proof on Figures 4.3 and 4.4. For a circulation $\mathbf{\Delta}$ and a commodity k, let us define $E(\mathbf{\Delta}, k)$ as the subset of E containing only edges e such that $\Delta^k_e \neq 0$. Clearly $E(\mathbf{h}, k) \subseteq E(\mathbf{g}, k)$. Similarly, for a block β, let $E_\beta(\mathbf{\Delta}, k) = E(\mathbf{\Delta}, k) \cap E_\beta$.

The first part is trivial, since if $g^k_e = 0$, but $h^k_e \neq 0$, then $\operatorname{supp}(\mathbf{h}) \nsubseteq \operatorname{supp}(\mathbf{g})$. For the second part, we will distinguish two cases based on the commodity k. Notice that always $g^k_{e_1} \neq 0$ due to our choice of e_1.

If $k = y$ or $k = b$, the proof is easy, since $E(\mathbf{g}, k)$ consists of vertex-disjoint cycles and the flow conservation property automatically ensures that the value of $h^k_{e_1}$ determines all the other edges in the sense that $h^k_{e_\beta} = h^k_{e_1}$ for all $e_\beta \in E_\beta(\mathbf{h}, k)$. To elaborate on that, if we look at some endpoint of e_1, we can see that its degree in $E(\mathbf{h}, k)$ is it most its degree in $E(\mathbf{g}, k)$, which is 2, and therefore the value $h^k_{e_\alpha}$

on the only other edge $e_\alpha \neq e_1$ incident with it must equal to $h^k_{e_1}$.

For $k = g$, this is more complicated, since $E(\mathbf{g}, k)$ decomposes into edge-disjoint, but not vertex-disjoint cycles and it is theoretically possible that $\mathbf{h}$ has a different flow on the topmost edge than on the other edges, and the "excess"/"deficit" is handled by the neighbouring blocks. We will proceed by induction on β. For the leftmost block $\beta = 1$, the lemma holds, since we can start with $x = h^g_{e_1}$ and then apply the flow conservation constraints in anti-clockwise order to again show that $h^g_{e_\beta} = h^g_{e_1}$ for all $e_\beta \in E_\beta(\mathbf{h}, g)$. The only vertex that can have degree ≥ 2 and therefore cannot be used for the inference, is the top-right vertex, but by the time we arrive there, we have already determined all edges in the block.

The induction step proceeds in a similar fashion. The only additional problem could occur in the top-left vertex which is of degree 4 in $E(\mathbf{g}, g)$, but due to the induction hypothesis, we know that it is balanced with respect to the previous block (i.e., the flows of the two edges from the previous block cancel out) and there only remain the two free variables in the equation representing the edges $e_a, e_b \in E_\beta$ from the current block, for which it therefore must hold that $h^g_{e_a} = h^g_{e_b}$, and we may therefore use the same chain of inferences as in the base case to get that $h^g_{e_\beta} = h^g_{e_1}$ for all $e_\beta \in E_\beta(\mathbf{h}, g)$. $\square$

Now we may prove the main lemma.

Lemma 4.11. *For any choice of p, the circulation $\mathbf{g}$ depicted in Figure 4.2 is a circuit.*

Proof. Let $\mathbf{h}$ be a circulation with $\operatorname{supp}(\mathbf{h}) \cup \operatorname{supp}(\mathbf{h}^\Sigma) \subsetneq \operatorname{supp}(\mathbf{g}) \cup \operatorname{supp}(\mathbf{g}^\Sigma)$. We will show that $\mathbf{h} = \mathbf{0}$, which proves that $\mathbf{g}$ is support-minimal among nonzero circulations.

Due to Lemma 4.10, we know that for each block β and commodity k, the behaviour of $\mathbf{h}^k$ on β can be fully described by some $x \in \mathbb{R}$. Therefore, we may represent $\mathbf{h}$ as a $p \times 2$ vector $\mathbf{x} \in \mathbb{R}^{p \times \{g, y/b\}}$, where x^g_β represents the x^k_β from Lemma 4.10 for $\beta = \beta$ and $k = g$ and $x^{y/b}_\beta$ represents the x for $\beta = \beta$ and k equal to whichever of y and b participates in block β.

As $\operatorname{supp}(\mathbf{h}^\Sigma) \subseteq \operatorname{supp}(\mathbf{g}^\Sigma)$, $\mathbf{h}$ needs to preserve all the balanced edges of $\mathbf{g}$. We can therefore write the equations by observing the pictures: we get that $x^g_i = x^{y/b}_i$ and $x^g_i + x^{y/b}_i = x^{y/b}_{i+1}$ for all $1 \leq i \leq p$, the only exception being the last block where the latter equation does not appear.

Now, we distinguish two cases, as we know that $\operatorname{supp}(\mathbf{h}) \cup \operatorname{supp}(\mathbf{h}^\Sigma) \subsetneq \operatorname{supp}(\mathbf{g}) \cup \operatorname{supp}(\mathbf{g}^\Sigma)$. First, we assume that $\operatorname{supp}(\mathbf{h}) \subsetneq \operatorname{supp}(\mathbf{g})$. Then for some e and k, $g^k_e \neq 0$ and $h^k_e = 0$, and due to how we defined $\mathbf{x}$, the corresponding x^k_i must be zero. But then one of the balancing equations shown in the previous paragraph implies that $x^g_i = x^{y/b}_i = 0$, and from another one, we get that $x^{y/b}_{i+1} = x^g_i + x^{y/b}_i = 0$. Similarly, $x^g_{i-1} + x^{y/b}_{i-1} = x^{y/b}_i = 0$, and we can see that $x^g_{i-1} + x^{y/b}_{i-1} = 2 \cdot x^{y/b}_{i-1} = 0$. Hence, the zeros propagate in both directions, and we can carry on with both $i := i - 1$ and $i := i + 1$. After reaching both 1 and p, we get that $\mathbf{x} = \mathbf{0}$ and therefore $\mathbf{h} = \mathbf{0}$, exactly as we wanted to show.

In the other case, $\operatorname{supp}(\mathbf{h}) = \operatorname{supp}(\mathbf{g})$. But then $\operatorname{supp}(\mathbf{h}^\Sigma) \subsetneq \operatorname{supp}(\mathbf{g}^\Sigma)$, which is an immediate contradiction, because we may see from the figures that there is no way of balancing any additional edge compared with $\mathbf{g}$ without shrinking supp($\mathbf{g}$), since all the other edges have only one commodity flowing through them. □

Corollary 4.12. There exist infinitely many MCF instances with $|E| = 6p+1$ and $K = 3$ that contain circuits of norm $\geq 2^p$.

Remark 4.13. It is possible to construct a flow $\mathbf{f}$ in the network displayed in Figure 4.2 such that $\varphi(\mathbf{f}) = 1$ and $\varphi(\mathbf{f} + \mathbf{g}) = 0$ where $\mathbf{g}$ is the exponential circuit, potentially by introducing new commodities and picking sources and sinks appropriately. However, in all examples we have produced so far, there always exists some small circuit $\mathbf{h}$ such that $\varphi(\mathbf{f} + \mathbf{h})$ also equals 0, and such examples therefore do not serve as a proof of large inflation rates.

4.5.1 Adaptation for LBF

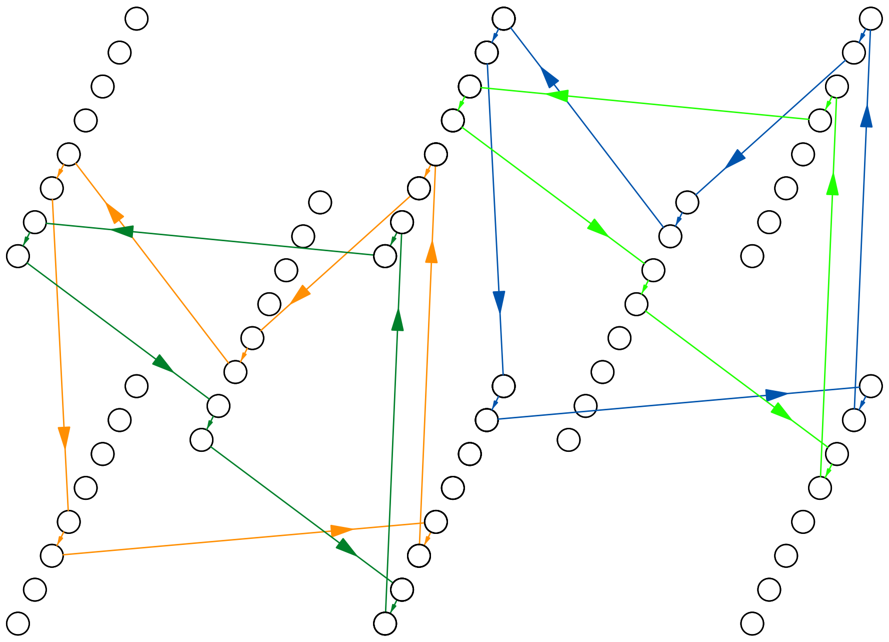

Figure 4.5: A schema of two blocks of the exponential MCF circuit adapted for LBF. The whole network has eight layers. All cycles belong to the same (and only) commodity; the colours are used for better visibility.

We now briefly sketch the adaptation of the exponential circuit construction for LBF. The overall structure is very similar to the construction from Figure 4.2, however a crucial difference is that there is only one commodity and edges instead have multiple copies in different layers. We describe the procedure for translating the presented MCF circuit to a LBF circuit. See Figure 4.5 for an illustration.

For simplicity, we first modify the MCF instance by splitting the commodity g into two, ga and gb. The flow commodity g in the MCF circuit gets reassigned to ga and gb as follows: we split it into edge-disjoint cycles and assign the cycles alternatively to ga and gb. Note that this modified circulation is also a circuit: the outcomes of Lemma 4.10 are even easier to prove since all four commodities now consist of vertex-disjoint cycles, and the proof Lemma 4.11 applies almost identically.

The idea is to use this instance and circuit to construct an exponential circuit in LBF. The LBF network will have $L = 8$ layers (i.e., the path length restriction will be 7). The k-th commodity for ($k \in \{1, 2, 3, 4\} \cong \{ga, y, gb, b\}$) will be restricted exclusively to layers $2k - 2$ and $2k - 1$ (recall that in LBF we index layers from zero). Let $\mathbf{g}$ denote the original MCF circuit. We start with a zero circulation $\mathbf{h}$ and then, for commodity k and edge uv in turn, we set $\mathbf{h}_{u^{2k-2}v^{2k-1}} \coloneqq \mathbf{h}_{v^{2k-1}v^{2k-2}} \coloneqq g_e^k$.

This construction maps different commodities to different layers that do not interact. It also produces a circulation that consists of vertex-disjoint weighted cycles, which makes it easy to prove an analogy of Lemma 4.10. The fact that a circulation produced this way is a circuit then follows from an adaptation of Lemma 4.11, since the balancing equations that arise are identical to those of MCF. Generally, the situation in LBF is not entirely analogous to MCF due to loop edges, but those do not pose a problem here because of the vertex-disjoint structure.

5. Most Helpful Cycles

In this chapter, we propose an iterative algorithm for MCF and provide some guarantees on its running time, although the question of $\mathcal{O}(\log 1/\varepsilon)$-style convergence will remain unanswered. However, practical experiments suggest it runs in time polynomial in input size and $\log(1/\varepsilon)$ on instances from the standard testset [Ope13], where ε is the desired error in penalty.

The algorithm is based on a modified version of Weintraub's algorithm [BT89] for the Minimum-Cost Maximum Flow problem (MCMF), which we will introduce in the following section. We describe it in a fair amount of detail, since its understanding is crucial for the understanding of our algorithm. Later, in Section 5.2, we will describe our adaptation for MCF.

The main idea of both algorithms is to start with an initial flow of suboptimal cost and iteratively improve it by a adding a weighted cycle to it. However, since finding the best cycle would be $\mathcal{NP}$-hard, we will instead find a *collection* of vertex-disjoint weighted cycles that improves the solution by at least as much as the best single cycle. In the case of the MCMF algorithm, and conditionally, in our case, we will then be able to prove that this approach leads to fast convergence.

The chapter is organised as follows: in Section 5.1, we describe Weintraub's algorithm for MCMF, and demonstrate on it many concepts useful later. Our contributions start in Section 5.2 where we describe our adaptation of Weintraub's algorithm for MCF. In Section 5.3, we analyse this algorithm.

5.1 Minimum-Cost Maximum Flow

Recall that the Minimum-Cost Maximum Flow problem is a generalisation of the Maximum Flow problem in which we are given a vector $\mathbf{w} \in \mathbb{R}^E$ of costs per one flow unit transported along each edge, and want to find a flow $\mathbf{f}$ that primarily maximises $|\mathbf{f}|$ and secondarily minimises $\mathbf{w} \cdot \mathbf{f}$.

MCMF has been widely studied and many weakly and strongly polynomial algorithms have been developed; for an overview, see [AMO88]. The algorithm we describe was originally proposed by Weintraub [Wei74] and its modification and subsequent analysis was described in [BT89].

Weintraub's algorithm repeatedly solves a problem called Most Helpful Cycles. The task is to find an optimal vertex-disjoint collection of undirected cycles with respect to some edge costs. Actually, there are two vectors of edge costs, $\mathbf{x}^+$ and $\mathbf{x}^-$, where x_e^+ is the cost of picking e in its normal orientation and x_e^- is the cost of picking e in its opposite orientation. Formally, the problem is:

Most Helpful Cycles

Input: Graph $G = (V, E)$, edge costs $\mathbf{x}^+, \mathbf{x}^- \in \mathbb{R}^E$

Find: An incidence vector $\mathbf{D} \in \{-1, 0, 1\}^E$ of a vertex-disjoint collection of undirected cycles minimising the *cost function:*

$$\operatorname{cost}(\mathbf{D}) = \sum_{e \in E} \begin{cases} x_e^+ & \text{if } D_e = 1, \\ x_e^- & \text{if } D_e = -1, \\ 0 & \text{otherwise.} \end{cases}$$

We will discuss the Most Helpful Cycles problem shortly. For now, we only need to know that it is solvable in polynomial time.

The outline of the modified Weintraub's algorithm is presented in Algorithm 1.

Algorithm 1: Most helpful cycles for MCMF (Helpful-MCMF)

```
Input: A flow network, weights $\mathbf{w} \in \mathbb{R}^E$, allowed absolute error $\varepsilon > 0$
Output: Maximum flow $\mathbf{f}^Q$ whose cost is within $\varepsilon$ of the optimal cost
/* calculate the upper bound on the difference between the initial and
   optimal cost:                                                          */
$C \leftarrow \sum_{e \in E} |w_e| \cdot c_e$
$Q \leftarrow \log(C/\varepsilon) \cdot 2|E|$      /* number of iterations, see Theorem 5.5 */
$\mathbf{f}^0 \leftarrow$ an arbitrary maximum satisfying flow
for $i = 0, \ldots, Q-1$ do
5   for $\lambda_p = \lambda_1, \ldots, \lambda_q$ do /* try several step sizes $\lambda_p$, see Remark 5.1 */
        define $\mathbf{x}^+, \mathbf{x}^- \in \mathbb{R}^E$ the cost of increasing/decreasing the flow on
        each edge by $\lambda_p$:
        $x_e^+ = \begin{cases} \lambda_p \cdot w_e & \text{if } f_e^i + \lambda_p \le c_e \\ +\infty & \text{otherwise} \end{cases} \quad x_e^- = \begin{cases} -\lambda_p \cdot w_e & \text{if } f_e^i - \lambda_p \ge 0 \\ +\infty & \text{otherwise} \end{cases}$
        $\mathbf{D}^p \leftarrow$ FindMostHelpfulCycles($G$, $\mathbf{x}^+$, $\mathbf{x}^-$)
        /* due to properties of $\mathbf{x}^+$ and $\mathbf{x}^-$, $\mathbf{D}^p$ is a disjoint collection of
           undirected cycles such that $\mathbf{f}^i + \lambda_p \mathbf{D}^p$ is satisfying and $\mathbf{D}^p$
           minimises $\mathbf{w} \cdot \lambda_p \mathbf{D}^p$, i.e., it maximises the cost improvement */
    end
    /* find the $\lambda_p$ and the cycle collection that decreases the total cost by
       the most and add it to the flow:                                       */
9   $\lambda, \mathbf{D} \leftarrow \arg\min_{\lambda_p, \mathbf{D}_p} \{ \mathbf{w} \cdot \lambda_p \mathbf{D}^p \}$
    if $\lambda \mathbf{D} = \mathbf{0}$ then
        $\mathbf{f}^i$ is already optimal, return it
    end
13  $\mathbf{f}^{i+1} \leftarrow \mathbf{f}^i + \lambda \mathbf{D}$
end
```

Remark 5.1. In Algorithm 1, we intentionally omit the choice of candidate step sizes $\lambda_1, \ldots, \lambda_q$. We present two options:

1. *Exhaustive step sizes.* We try all λ_p from the set $U_i = \{ f_e^i \mid e \in E \} \cup \{ c_e - f_e^i \mid e \in E \}$. As we will later see, this is sufficient to always find the globally best cycle collection.

2. *Power-of-two step sizes.* Let u be the maximum element from U_i. We try all powers of 2 smaller than u, starting from 1.[1] This way we will try $\Theta(\log u)$ different step sizes instead of $\mathcal{O}(|E|)$.

We will analyse both approaches when discussing the runtime of the algorithm.

[1]Here we assume that $\mathbf{c} \in \mathbb{N}^E$ and therefore there is an integer optimal solution. Otherwise we would also have to try all negative powers of two greater than some $\delta > 0$.

5.1.1 Finding Most Helpful Cycles

We will now describe the procedure FINDMOSTHELPFULCYCLES($G, \mathbf{x}^+, \mathbf{x}^-$) for solving the MOST HELPFUL CYCLES problem. As mentioned before, insisting that the returned incidence vector $\mathbf{D}$ has to be just a single undirected cycle makes the problem $\mathcal{NP}$-hard. For example, by setting $\mathbf{x}^+ = -\mathbf{1}$ and $\mathbf{x}^- = +\boldsymbol{\infty}$ and checking if the minimum cost of $\mathbf{D}$ is $-|E|$, we obtain the (directed) Hamiltonian cycle problem.

On the other hand, the MOST HELPFUL CYCLES problem is solvable in polynomial time by a reduction to the ASSIGNMENT PROBLEM, which we now briefly define; see also Ramshaw and Tarjan [RT12] for a survey.

A *weighted perfect complete bipartite graph* is a graph $G = (V, E, \mathbf{w})$ such that $V = A \cup B$, $|A| = |B|$, $E = A \times B$, $\mathbf{w} \in \mathbb{R}^E$. A *matching* in G is a subset of edges $M \subseteq E$ such that no two edges of M share a common vertex. A matching M is *perfect*, if $|M| = |A| = |B|$, i.e., each vertex $v \in V$ is incident to some (and exactly one) edge in M. The *weight* of a matching M is defined as $w(M) = \sum_{e \in M} w_e$. The ASSIGNMENT PROBLEM is defined as:

ASSIGNMENT PROBLEM

Input:	Weighted perfect complete bipartite graph G
Find:	Perfect matching M of minimum weight

Lemma 5.2. *MOST HELPFUL CYCLES is solvable in polynomial time by reduction to ASSIGNMENT PROBLEM.*

Proof. The idea is that we let each vertex choose its successor on the cycle, with an additional option with cost 0 of not belonging to any cycle. For that, we create a bipartite graph $H = (V_{\text{out}} \cup V_{\text{in}}, V_{\text{out}} \times V_{\text{in}}, \mathbf{x})$ with V_{out} and V_{in} being two disjoint copies of the original vertex set V and $\mathbf{x}$ defined as

$$x_{u_{out}v_{in}} = \begin{cases} 0 & \text{if } u = v, \\ \min(x^+_{uv}, x^-_{vu}) & \text{otherwise,} \end{cases}$$

where x^+_{uv} and x^-_{vu} are defined to be $+\infty$ if the respective edges do not exist. Let M be the perfect matching solving the ASSIGNMENT PROBLEM and create the resulting incidence vector $\mathbf{D}$ as follows: set $\mathbf{D} = 0$ for all entries with the following exception:

$$\text{for each } u_{out}v_{in} \in M, \quad \begin{cases} D_{uv} := 1 & \text{if } x^+_{uv} \leq x^-_{vu}, \\ D_{vu} := -1 & \text{if } x^+_{uv} > x^-_{vu}. \end{cases}$$

That is, we just translate the choice of M into which edge and in which direction to pick in the original graph.

The properties of matchings in H allow each vertex in G to have exactly one successor and predecessor (including itself), which means perfect matchings in H have one-to-one correspondence to undirected cycle collections in G.[2] Additionally, the costs are chosen in such a way that each perfect matching of H has the same cost as the corresponding cycle collection in G. □

[2]Technically, this is true only if G does not contain two edges of opposite orientation, i.e., $uv \in E \implies vu \notin E$, but if it does, we always want to pick the cheaper of the two edges anyway.

5.1.2 Convergence Analysis

Let $\mathbf{f}^*$ be some fixed optimal solution, i.e., a solution minimising $\mathbf{w} \cdot \mathbf{f}$. Then, for the purposes of the section, we define the optimality gap as a function $\mathrm{GAP}(\mathbf{f}) := \mathbf{w} \cdot \mathbf{f} - \mathbf{w} \cdot \mathbf{f}^* = \mathbf{w} \cdot (\mathbf{f} - \mathbf{f}^*) \geq 0$.

We will now show that in each step, the optimality gap of the current solution gets reduced by a multiplicative factor $c \in (0, 1)$ that is linear in $1/|E|$, which means that after Q iterations, the optimality gap reduces to $(1 - c)^Q$ of the initial gap. We start by analysing the improvement possible by picking an appropriate single cycle, and later show that finding a most helpful collection of disjoint cycles yields at least as much improvement.

Lemma 5.3. *Let $\mathbf{f}^i$ be a flow at the start of the i-th iteration of Algorithm 1 and let $\mathrm{GAP}(\mathbf{f})$ and $\mathbf{f}^*$ be the optimality gap and a fixed optimal solution, respectively. Then there exists a weighted cycle $\alpha\mathbf{C}$ such that $\mathbf{f}^i + \alpha\mathbf{C}$ is satisfying and*

$$\mathrm{GAP}(\mathbf{f}^i + \alpha\mathbf{C}) \leq (1 - 1/m) \cdot \mathrm{GAP}(\mathbf{f}^i),$$

where $m = |E|$.

Proof. Set $\mathbf{\Delta} = \mathbf{f}^* - \mathbf{f}^i$. From Section 1.7, we know that $\mathbf{\Delta}$ is a circulation, and that it can be expressed as a conformal sum of $\ell \leq m = |E|$ weighted cycles:

$$\mathbf{\Delta} = \sum_{i=1}^{\ell} \alpha_i \mathbf{C}^i.$$

Since the dot product and therefore also our cost function $\mathbf{w} \cdot \mathbf{f}$ is linear, it follows that

$$\mathrm{GAP}(\mathbf{f}^i) = \mathbf{w} \cdot (\mathbf{f}^i - \mathbf{f}^*) = -\mathbf{w} \cdot \mathbf{\Delta} = -\sum_{i=1}^{\ell} \mathbf{w} \cdot \alpha_i \mathbf{C}^i.$$

Let $\alpha\mathbf{C}$ be the weighted cycle with the largest improvement, i.e., the most negative value of $\mathbf{w} \cdot \alpha\mathbf{C}$, among the cycles in the sum. We immediately see that $\mathbf{w} \cdot \alpha\mathbf{C} \leq 1/\ell \cdot \mathbf{w} \cdot \mathbf{\Delta}$, since the smallest of ℓ values must certainly be below their average. Thus, $\mathrm{GAP}(\mathbf{f}^i + \alpha\mathbf{C}) = \mathbf{w}(\mathbf{f}^i + \alpha\mathbf{C} - \mathbf{f}^*) = \mathbf{w} \cdot (\mathbf{f}^i - \mathbf{f}^*) + \mathbf{w} \cdot \alpha\mathbf{C} \leq \mathbf{w} \cdot (\mathbf{f}^i - \mathbf{f}^*) + 1/\ell \cdot \mathbf{w} \cdot \mathbf{\Delta} = \mathbf{w} \cdot (\mathbf{f}^i - \mathbf{f}^*) + 1/\ell \cdot \mathbf{w} \cdot (\mathbf{f}^* - \mathbf{f}^i) = (1 - 1/\ell) \cdot \mathrm{GAP}(\mathbf{f}^i) \leq (1 - 1/m) \cdot \mathrm{GAP}(\mathbf{f}^i)$. □

The algorithm obviously does not find $\alpha\mathbf{C}$ since it does not know $\mathbf{\Delta}$, but we will show that the weighted cycle collection found by the algorithm decreases the optimality gap enough:

Lemma 5.4. *Let $\mathbf{f}^i$, $\mathbf{f}^{i+1}$ be flows at the start of iteration i and $i+1$ of Algorithm 1 and let $\mathrm{GAP}(\mathbf{f})$ and $\mathbf{f}^*$ be the optimality gap and an optimal solution, respectively. Then*

- $\mathrm{GAP}(\mathbf{f}^{i+1}) \leq (1 - \frac{1}{m}) \cdot \mathrm{GAP}(\mathbf{f}^i)$ *if we use the exhaustive step size rule (Remark 5.1),*
- $\mathrm{GAP}(\mathbf{f}^{i+1}) \leq (1 - \frac{1}{2m}) \cdot \mathrm{GAP}(\mathbf{f}^i)$ *if we choose the power-of-two step size rule.*

Proof. We focus on the former case first.

Let $\mathbf{\Delta} = \mathbf{f}^* - \mathbf{f}^i$ and let $\alpha\mathbf{C}$ be the most-improving cycle from the canonical cycle decomposition of $\mathbf{\Delta}$. We already know from the previous lemma that $\text{GAP}(\mathbf{f}^i + \alpha\mathbf{C}) \leq (1 - 1/m) \cdot \text{GAP}(\mathbf{f}^i)$. We fix a satisfying weighted cycle $\gamma\mathbf{B}$ with $\gamma > 0$ that minimises $\mathbf{w} \cdot \gamma\mathbf{B}$ among all satisfying weighted cycles; clearly $\mathbf{w} \cdot \gamma\mathbf{B} \leq \mathbf{w} \cdot \alpha\mathbf{C} < 0$ and $\mathbf{w} \cdot \mathbf{B} < 0$. We will show that $\gamma \in U_i$ (i.e., $\gamma = f_e^i$ or $\gamma = c_e - f_e^i$ for some $e \in E$), which means that γ has been considered as a step size during the current iteration. We proceed by contradiction: if this is not the case, then adding $\gamma\mathbf{B}$ to $\mathbf{f}^i$ does not bring any new edge to 0 or its capacity, and we can therefore increase γ to some $\gamma' > \gamma$. But then $\mathbf{w} \cdot \gamma'\mathbf{B} = \gamma'(\mathbf{w} \cdot \mathbf{B}) < \gamma(\mathbf{w} \cdot \mathbf{B}) = \mathbf{w} \cdot \gamma\mathbf{B}$, which is a contradiction.

Therefore, $\lambda_p = \gamma$ for some p on Line 5 of Algorithm 1. Then clearly $\mathbf{w} \cdot \lambda_p\mathbf{D}^p \leq \mathbf{w} \cdot \gamma\mathbf{B}$, since $\gamma\mathbf{B} = \lambda_p\mathbf{B}$ is a disjoint collection of cycles (albeit consisting only of a single cycle) with step size λ_p, and $\lambda_p\mathbf{D}^p$ is the best such collection. This proves the first case of the lemma, since the most helpful cycle collection $\lambda\mathbf{D}$ found by the algorithm (Line 9) decreases the optimality gap by $-\mathbf{w} \cdot \lambda\mathbf{D} \geq -\mathbf{w} \cdot \lambda_p\mathbf{D}^p \geq -\mathbf{w} \cdot \gamma\mathbf{B} \geq -\mathbf{w} \cdot \alpha\mathbf{C} \geq -1/m \cdot \mathbf{w} \cdot \mathbf{\Delta} = 1/m \cdot \text{GAP}(\mathbf{f}^i)$.

For the second case, take the best cycle collection $\lambda\mathbf{D}$ from the previous paragraph and observe that there exists some power-of-two step size $\lambda^* \in [\lambda/2, \lambda]$ tried by the algorithm. Since $\lambda^*\mathbf{D}$ is a satisfying cycle collection for this step size (because $\lambda^*\mathbf{D} \sqsubseteq \lambda\mathbf{D}$), the best cycle collection must improve the optimality gap at least by $-\mathbf{w} \cdot \lambda^*\mathbf{D} \geq -\mathbf{w} \cdot (\lambda/2)\mathbf{D} \geq -1/2 \cdot \mathbf{w} \cdot \lambda\mathbf{D}$. By the argument from the previous paragraph, the optimality gap decreases by $-\mathbf{w} \cdot \lambda^*\mathbf{D} \geq \frac{1}{2m} \cdot \text{GAP}(\mathbf{f}^i)$. □

Theorem 5.5. *The modified Weintraub's algorithm solves MCMF in time polynomial in $|V|$, $|E|$ and $\log(C/\varepsilon)$, and gives a solution with optimality gap $\leq \varepsilon$, where $C \leq \sum_{e \in E} |w_e| \cdot c_e$ is some upper bound on the initial optimality gap of $\mathbf{f}^0$.*

Proof. Each iteration runs in time polynomial in $|V|$, $|E|$ and $\log(C)$, and the optimality gaps of the solutions satisfy $\text{GAP}(\mathbf{f}^0) \leq C$, $\text{GAP}(\mathbf{f}^{i+1}) \leq (1 - \frac{1}{2|E|}) \cdot \text{GAP}(\mathbf{f}^i)$. According to Lemma 1.11, after $\log(C/\varepsilon)2|E|$ iterations, the optimality gap drops below ε. □

5.2 Most Helpful Cycles for MCF

We now present our adaptation of Weintraub's algorithm for MCF (from now on referred to as Helpful-MCF), namely for the (MCF-P) formulation where the capacity constraints have been replaced with a quadratic penalty function. We are focusing only on MCF and not LBF, since the latter problem has so far resisted our attempts at adopting the same approach for it – although we would not be surprised if this idea eventually turned out to be applicable there as well.

The quadratic penalty presents an obvious complication because many arguments in the previous section relied on the cost function being simply a scalar product $\mathbf{w} \cdot \mathbf{f}$. Consequently, we will not be able to straightforwardly prove fast convergence, although we will still show weaker properties, such as the existence of an improving cycle.

The changes in Helpful-MCF as compared to Helpful-MCMF are as follows:

1. The upper bounds C and Q, on, respectively, the optimality gap and the number of iterations, no longer apply. We postpone their determination until after convergence analysis, to Section 5.3.6. Now we informally replace them in the algorithm by "sufficiently large numbers".

2. It is no longer true that trying all step sizes $\lambda_p \in U_i$ (i.e., those of the form f_e^k and $c_e - f_e^k$) is enough to find the optimal cycle collection. Furthermore, the optimal solution no longer has to be integral. Therefore, we will use a modified power-of-two step size rule, which considers $\lambda_p \in \{\lambda_{\min} = 2^\ell, 2^{\ell+1}, \ldots, 2^{r-1}, 2^r = \lambda_{\max}\}$ with $\ell, r \in \mathbb{Z}$. We discuss the correct values of $\lambda_{\min}$ and $\lambda_{\max}$ later, also in Section 5.3.6.

3. The body of the main loop in Algorithm 1 (lines 5 to 13) is now performed K times, once for each commodity $k \in \{1, \ldots, K\}$. Each time, we leave other commodities fixed and try to find the most helpful cycles by changing only the flow in the current commodity. At the end of each iteration (after trying all commodities), we choose the most improving cycle collection among all λ and k.

4. The cost function changes from $\mathbf{w} \cdot \mathbf{f}$ to $\varphi(\mathbf{f}) = \sum_{e \in E} \varphi_e(\mathbf{f})$.[3] Also, the capacities can now be exceeded. This entails changes in the definition of $\mathbf{x}^+$ and $\mathbf{x}^-$ (which now depends not only on λ_p, but also on k):

$$x_e^+ = \varphi_e(\mathbf{f} + \lambda_p \mathbf{e}^{k,e}) - \varphi_e(\mathbf{f}),$$
$$x_e^- = \begin{cases} \varphi_e(\mathbf{f} - \lambda_p \mathbf{e}^{k,e}) - \varphi_e(\mathbf{f}) & \text{if } \mathbf{f} - \lambda_p \mathbf{e}^{k,e} \geq 0, \\ +\infty & \text{otherwise,} \end{cases}$$

   where $\mathbf{e}^{k,e} \in \mathbb{R}^{E \times K}$ is the (k,e)-th unit vector, i.e., a vector that is 0 everywhere except for the k-th commodity on the edge e, where it is 1. In other words, for each $e \in E$, we calculate the difference in penalty if we respectively increase or decrease the flow along edge e in the k-th commodity by λ_p.

All the other details, as well as the fact that we use the FindMostHelpfulCycles subprocedure, remain unchanged.

Remark 5.6. Note that keeping FindMostHelpfulCycles intact is only possible because of how φ is defined. Firstly, edges do not interfere with each other, meaning that the total change in φ is equal to the sum of changes of φ_e over all $e \in E$, i.e., the function φ is *separable* with respect to E. Secondly, for each edge, we only ever change the flow in one commodity.

This is why the same algorithm cannot be used verbatim for LBF. There, the edges still do not interact with each other, but it is no longer the case that at most one Δ_e^k may be changed on each edge e, since the graph is one big layered network, and not k independent ones. In such cases, $\mathbf{x}^+$ and $\mathbf{x}^-$ no longer accurately reflect the costs: for a circulation $\boldsymbol{\Delta}$ that has $\Delta_e^i, \Delta_e^j \geq 0$ for $i \neq j$ and $\Delta_e^* = 0$ elsewhere, the final cost change for edge e is $\varphi_e(\mathbf{f} + \Delta_e^i + \Delta_e^j) - \varphi_e(\mathbf{f})$, which cannot generally be expressed in terms of $\varphi_e(\mathbf{f})$, $\varphi_e(\mathbf{f} + \Delta_e^i)$ and $\varphi_e(\mathbf{f} + \Delta_e^j)$, precisely because φ_e is not a linear function.

[3]Recall that φ_e is defined in Section 2.2.2 as $\varphi_e(\mathbf{f}) = \left[\mathbf{f}^\Sigma(e) - c_e\right]_+^2$.

5.3 Analysing Helpful-MCF

Although the analysis is much more complicated than in the case of Helpful-MCMF due to a more complex penalty function, we will show that Helpful-MCF still finds the best possible cycle in each iteration (or rather a constant approximation thereof). The question is, however, whether this is enough. In the case of MCMF, we have shown that finding the best cycle decreases the optimality gap by a multiplicative factor of $\Omega(1/m)$ in each iteration. It is a priori unclear whether any such bounds hold here.

To illustrate the added complexity that non-linearity brings: in MCMF, when we look at the cycle decomposition of the circulation that leads to the optimum, it does not matter in which order we add those cycles to the current flow, since due to linearity, a given cycle always improves the penalty function by the same amount regardless of to which flow it is applied. In MCF, on the other hand, the order in which we apply the cycles matters – in fact, applying the cycles from the optimal decomposition one-by-one may, under some circumstances, even lead to the penalty first *increasing* and only then decreasing. Therefore, it is not even immediately clear that there must exist an *improving* cycle.

The organisation of this section is as follows: in Section 5.3.1, we show that Helpful-MCF always finds a cycle collection that is, up to a multiplicative constant, best possible. In Section 5.3.2, we show that there always exists an improving cycle, and subsequently provide bounds on its improvement in Section 5.3.3. In Section 5.3.4, we prove a $\mathcal{O}(1/\varepsilon)$-style convergence of Helpful-MCF. Then, in Section 5.3.5 we call on our knowledge of inflation rate and prove a $\mathcal{O}(H_{\mathrm{MCF}} \cdot \log(1/\varepsilon))$-style convergence of Helpful-MCF, where H_{MCF} is the inflation rate of the MCF polyhedron. Finally, in Section 5.3.6, we fill in the last remaining technical details of Helpful-MCF, which follow from our analysis.

5.3.1 Finding the Best Cycle Collection

We will reuse the arguments from Lemma 5.4 to show that whenever there exists a cycle that decreases the penalty function by ε, Helpful-MCF finds a cycle collection that decreases the penalty function by at least $\varepsilon/2$.

In this and following sections, we will heavily use the notation from Section 2.2.3.

Lemma 5.7. *Let $\mathbf{f}^i$, $\mathbf{f}^{i+1}$ be flows at the start of iteration i and $i+1$ of Helpful-MCF and let $\alpha\mathbf{C}$ be any weighted one-commodity cycle feasible with respect to $\mathbf{f}^i$. Furthermore, assume that $\lambda_{\min} \le \alpha \le 2\lambda_{\max}$. Then,*

$$\varphi(\mathbf{f}^i) - \varphi(\mathbf{f}^{i+1}) \ge 1/2 \cdot \mathrm{IMP}_{\mathbf{f}^i}(\alpha\mathbf{C}),$$

That is, the weighted disjoint collection of cycles found by the algorithm is a 2-approximation of the best possible weighted cycle, assuming correct values of $\lambda_{\min}$ and $\lambda_{\max}$.

Proof. Let k^* be the commodity of cycle $\alpha\mathbf{C}$. Pick the step size λ_p considered by the algorithm such that $\lambda_p \le \alpha$ and λ_p is maximum possible. By our assumptions, $\alpha/2 \le \lambda_p \le \alpha$, and thus $\lambda_p = t\alpha$ for some $t \in [1/2, 1]$.

The most important observation is that taking $\lambda_p\mathbf{C}$ instead of $\alpha\mathbf{C}$ improves the penalty by at least half the amount of $\alpha\mathbf{C}$, i.e., $\mathrm{IMP}_{\mathbf{f}}(\lambda_p\mathbf{C}) \geq 1/2 \cdot \mathrm{IMP}_{\mathbf{f}}(\alpha\mathbf{C})$. This follows directly from the concavity of $\mathrm{IMP}_{\mathbf{f}}$:

$$\begin{aligned}\mathrm{IMP}_{\mathbf{f}}(\lambda_p\mathbf{C}) &= \mathrm{IMP}_{\mathbf{f}}\Big((1-t)\cdot\mathbf{0} + t\cdot\alpha\mathbf{C}\Big) \geq (1-t)\cdot\mathrm{IMP}_{\mathbf{f}}(\mathbf{0}) + t\cdot\mathrm{IMP}_{\mathbf{f}}(\alpha\mathbf{C}) \\ &= t\mathrm{IMP}_{\mathbf{f}}(\alpha\mathbf{C}) \geq \frac{1}{2}\cdot\mathrm{IMP}_{\mathbf{f}}(\alpha\mathbf{C}).\end{aligned}$$

Identically to the proof for MCMF, we can see that for each given λ and k, Helpful-MCF finds an optimal cycle collection $\lambda\mathbf{D}$ in the k-th commodity, i.e., one such that $\mathrm{IMP}_{\mathbf{f}^i}(\lambda\mathbf{D})$ is maximum possible – specially, it is greater than any $\mathrm{IMP}_{\mathbf{f}^i}(\lambda\mathbf{C})$ of a weighted cycle $\lambda\mathbf{C}$. The argumentation follows from the correctness of the FindMostHelpfulCycles subprocedure and from the correctness of our choice of $\mathbf{x}^+$ and $\mathbf{x}^-$. Hence, for $\lambda = \lambda_p$ and $k = k^*$, Helpful-MCF finds a solution $\lambda_p\mathbf{D}$ with $\mathrm{IMP}_{\mathbf{f}}(\lambda_p\mathbf{D}) \geq \mathrm{IMP}_{\mathbf{f}}(\lambda_p\mathbf{C}) \geq 1/2\cdot\mathrm{IMP}_{\mathbf{f}}(\alpha\mathbf{C})$ which proves the lemma. □

Thus, we have shown that the algorithm always finds a cycle collection that is, up to a multiplicative constant, best possible. The remaining question that we try to answer for the rest of Section 5.3 is, how good that actually is.

5.3.2 Improving Cycle

We start off by showing that there always exists an improving cycle, a fact that seemed almost trivial for MCMF. Recall from Section 2.2.3 that (for a fixed flow $\mathbf{f}$ and a feasible circulation $\mathbf{\Delta}$) the function $h_{\mathbf{f},\mathbf{\Delta}} : [0,1] \to \mathbb{R}$, $h_{\mathbf{f},\mathbf{\Delta}}(\gamma) := -\mathrm{IMP}_{\mathbf{f}}(\mathbf{f} + \gamma\mathbf{\Delta})$ is continous and convex, its derivative $h'_{\mathbf{f},\mathbf{\Delta}}$ is continous and nondecreasing and that a circulation is an improving direction if and only if $h'_{\mathbf{f},\mathbf{\Delta}}(0) < 0$. This motivates the following definition: define the *acceleration* of $\mathbf{\Delta}$ (with respect to $\mathbf{f}$) as $w_{\mathbf{f}}(\mathbf{\Delta}) = h'_{\mathbf{f},\mathbf{\Delta}}(0)$. Clearly, $\mathbf{\Delta}$ is an improving direction if and only if its acceleration is negative. Additionally, the more negative the acceleration is, the more improvement we get by adding infinitesimal multiples of $\mathbf{\Delta}$ to $\mathbf{f}$. For an illustration, see Section 5.3.2.

Lemma 5.8. *Given a flow* $\mathbf{f}$ *and a feasible and improving circulation* $\mathbf{\Delta}$*, there exists a feasible and improving weighted cycle* $\alpha\mathbf{C} \sqsubseteq \mathbf{\Delta}$*.*

Proof. We may express $h_{\mathbf{f},\mathbf{\Delta}}$ in terms of φ:

$$\begin{aligned}h_{\mathbf{f},\mathbf{\Delta}}(\gamma) &= \varphi(\gamma\mathbf{\Delta} + \mathbf{f}) - \varphi(\mathbf{f}) \\ &= \sum_{e\in E}\Big[(\gamma\mathbf{\Delta} + \mathbf{f})^{\Sigma}(e) - c_e\Big]_+^2 - \varphi(\mathbf{f}) \\ &= \sum_{e\in E}\Big[\gamma\cdot\mathbf{\Delta}^{\Sigma}(e) + \mathbf{f}^{\Sigma}(e) - c_e\Big]_+^2 - \varphi(\mathbf{f}),\end{aligned}$$

where $\mathbf{f}^{\Sigma}(e) - c_e$ and $\varphi(\mathbf{f})$ are constant for a fixed $\mathbf{f}$ and $\mathbf{\Delta}^{\Sigma}(e)$ is constant for a fixed $\mathbf{\Delta}$. Taking the first derivative yields

$$h'_{\mathbf{f},\mathbf{\Delta}}(\gamma) = 2\cdot\sum_{e\in E}\mathbf{\Delta}^{\Sigma}(e)\cdot\Big[\gamma\cdot\mathbf{\Delta}^{\Sigma}(e) + \mathbf{f}^{\Sigma}(e) - c_e\Big]_+,$$

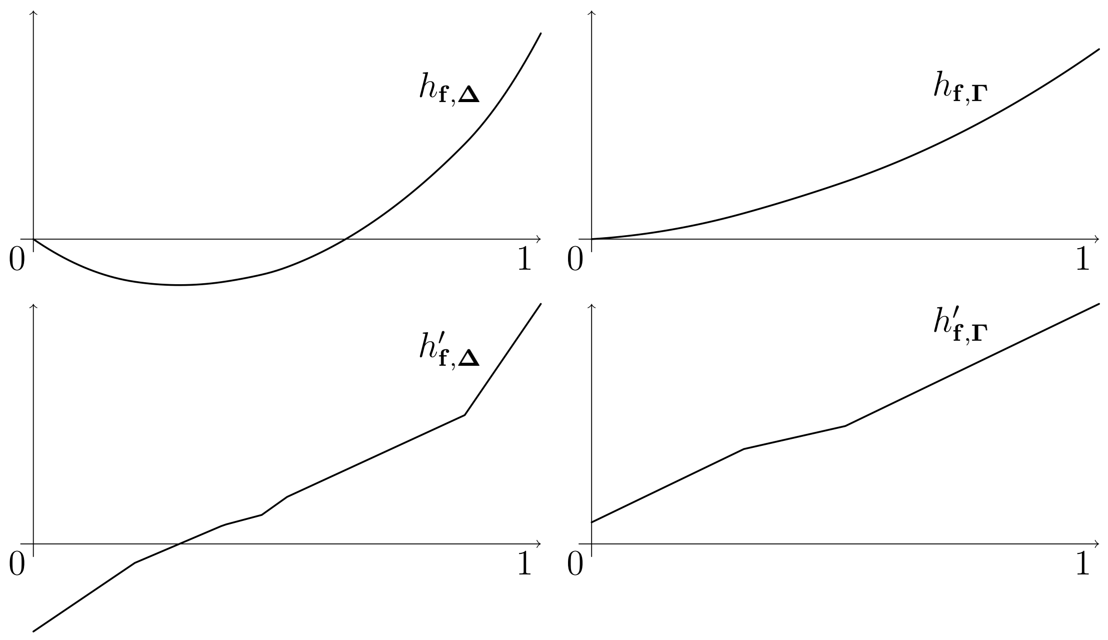

Figure 5.1: Illustration of h for the improving ($h_{\mathbf{f},\mathbf{\Delta}}$) and non-improving ($h_{\mathbf{f},\mathbf{\Gamma}}$) case.

and therefore:

$$w_{\mathbf{f}}(\mathbf{\Delta}) = h'_{\mathbf{f},\mathbf{\Delta}}(0) = 2 \cdot \sum_{e \in E} \mathbf{\Delta}^{\Sigma}(e) \cdot \left[\mathbf{f}^{\Sigma}(e) - c_e\right]_+ .$$

We see that $w_{\mathbf{f}}$ is linear: whenever $\mathbf{\Gamma} + \mathbf{\Psi} = \mathbf{\Delta}$ and all three circulations are feasible with respect to $\mathbf{f}$, it also holds that $w_{\mathbf{f}}(\mathbf{\Gamma}) + w_{\mathbf{f}}(\mathbf{\Psi}) = w_{\mathbf{f}}(\mathbf{\Delta})$. This follows from the fact that $\mathbf{\Gamma}^{\Sigma}(e) + \mathbf{\Psi}^{\Sigma}(e) = \mathbf{\Delta}^{\Sigma}(e)$.

From Remark 2.2 we know that $\mathbf{\Delta}$ can be decomposed into a conformal sum of weighted cycles, $\mathbf{\Delta} = \sum_{i=0}^{\ell} \alpha_i \mathbf{C}_i$. Due to conformality, $\alpha_i \mathbf{C}_i \geq \mathbf{0}$ is feasible for all i, as is $\varepsilon \mathbf{C}_i$ for $0 \leq \varepsilon \leq \alpha_i$.

Applying linearity of $w_{\mathbf{f}}$, we get $w_{\mathbf{f}}(\mathbf{\Delta}) = \sum_{i=0}^{\ell} w_{\mathbf{f}}(\alpha_i \mathbf{C}_i)$. Since $\mathbf{\Delta}$ is improving, $w_{\mathbf{f}}(\mathbf{\Delta})$ must be negative. But then so must be the right-hand side, meaning that some $w_{\mathbf{f}}(\alpha_i \mathbf{C}_i)$ also has to be negative. Then by the properties of $h_{\mathbf{f},\mathbf{\Delta}}$, $\alpha_i \mathbf{C}_i$ is an improving direction. Taking $\varepsilon \alpha_i \mathbf{C}_i$ for a sufficiently small ε finishes the proof. □

In fact, we get an even stronger property which we will use soon:

Lemma 5.9. *Given a flow* $\mathbf{f}$ *and a feasible and improving circulation* $\mathbf{\Delta}$*, there exists a feasible and improving weighted cycle* $\alpha \mathbf{C} \sqsubseteq \mathbf{\Delta}$ *such that* $w_{\mathbf{f}}(\alpha \mathbf{C}) \leq w_{\mathbf{f}}(\mathbf{\Delta})/(|E| \cdot K)$.

Proof. We know that $w_{\mathbf{f}}(\mathbf{\Delta}) = \sum_{i=0}^{\ell} w_{\mathbf{f}}(\alpha_i \mathbf{C}_i)$ and that $\ell \leq |E| \cdot K$. We may therefore take the cycle with the smallest $w_{\mathbf{f}}(\alpha_i \mathbf{C}_i)$, and the inequality follows from the argument that the smallest of ℓ numbers has to be less than their average, which we already used in the proof of Lemma 5.3. □

5.3.3 Bounding Cycle Improvement

An obvious next step is to try to use these bounds to establish some penalty improvement guarantees. Our ultimate goal is to have a theorem akin to Lemma 5.3, showing that there is a cycle which improves the penalty by some large-enough fraction of the optimality gap. We start by proving a useful upper and lower bounds from which other bounds can be generated.

Lemma 5.10. *For a flow* $\mathbf{f}$ *and a feasible circulation* $\boldsymbol{\Delta}$ *with* $w_{\mathbf{f}}(\boldsymbol{\Delta}) < 0$*, it holds that:*

$$-w_{\mathbf{f}}(\boldsymbol{\Delta}) \geq \mathrm{IMP}_{\mathbf{f}}(\boldsymbol{\Delta}).$$

Additionally, there exists some $\varepsilon \in (0, 1]$ *such that:*

$$\mathrm{IMP}_{\mathbf{f}}(\varepsilon\boldsymbol{\Delta}) \geq \frac{w_{\mathbf{f}}(\boldsymbol{\Delta})^2}{4\|\boldsymbol{\Delta}\|_1^2}.$$

Proof. Let us prove the top inequality first. Using the definition of $\mathrm{IMP}_{\mathbf{f}}$ and $h_{\mathbf{f},\boldsymbol{\Delta}}$, the fact that $h_{\mathbf{f},\boldsymbol{\Delta}}(0) = 0$ and that $\int_a^b f'(x)\,\mathrm{d}x = f(b) - f(a)$, we have

$$\mathrm{IMP}_{\mathbf{f}}(\boldsymbol{\Delta}) = -h_{\mathbf{f},\boldsymbol{\Delta}}(1) = h_{\mathbf{f},\boldsymbol{\Delta}}(0) - h_{\mathbf{f},\boldsymbol{\Delta}}(1) = -\int_0^1 h'_{\mathbf{f},\boldsymbol{\Delta}}(\gamma)\,\mathrm{d}\gamma.$$

Now, since $h'_{\mathbf{f},\boldsymbol{\Delta}}(0) = w_{\mathbf{f}}(\boldsymbol{\Delta})$ and $h'_{\mathbf{f},\boldsymbol{\Delta}}$ is nondecreasing, it means that $h'_{\mathbf{f},\boldsymbol{\Delta}}(\gamma) \geq w_{\mathbf{f}}(\boldsymbol{\Delta})\ \forall\gamma \in [0, 1]$. Therefore, the negated integral can be bounded from above by the area of the box $[0, 1] \times [w_{\mathbf{f}}(\boldsymbol{\Delta}), 0]$ and thus $\mathrm{IMP}_{\mathbf{f}}(\boldsymbol{\Delta}) \leq 1 \cdot (-w_{\mathbf{f}}(\boldsymbol{\Delta}))$, which is what we wanted to prove.

The idea behind the second part is to bound the steepness of $h'_{\mathbf{f},\boldsymbol{\Delta}}$, and, subsequently, bound the area between $h'_{\mathbf{f},\boldsymbol{\Delta}}$ and the x-axis in the part of the graph where $h'_{\mathbf{f},\boldsymbol{\Delta}}$ is negative. That way, even if $h'_{\mathbf{f},\boldsymbol{\Delta}}$ reaches zero as fast as possible, it will still delineate at least this much area (and therefore imply at least this much improvement).

Recall that

$$h'_{\mathbf{f},\boldsymbol{\Delta}}(\gamma) = 2 \cdot \sum_{e\in E} \boldsymbol{\Delta}^{\Sigma}(e) \cdot \left[\gamma \cdot \boldsymbol{\Delta}^{\Sigma}(e) + \mathbf{f}^{\Sigma}(e) - c_e\right]_+,$$

and thus,

$$h''_{\mathbf{f},\boldsymbol{\Delta}}(\gamma) = 2 \cdot \sum_{e\in E} \boldsymbol{\Delta}^{\Sigma}(e)^2 \cdot \mathbb{1}_{x>0}(\gamma \cdot \boldsymbol{\Delta}^{\Sigma}(e) + \mathbf{f}^{\Sigma}(e) - c_e),$$

where $\mathbb{1}_{x>0}(x) = 0$ for $x \leq 0$ and $\mathbb{1}_{x>0}(x) = 1$ for $x > 0$. $h''_{\mathbf{f},\boldsymbol{\Delta}}(\gamma)$ may generally be undefined in finitely many points, but those do not affect the reasoning in any way since their measure is zero and $h'_{\mathbf{f},\boldsymbol{\Delta}}$ is continuous (and therefore cannot have any sudden jumps, which is the only thing in which this could give us trouble). An important observation is that $\mathbb{1}_{x>0}(x) \leq 1$ and therefore,

$$\begin{aligned} h''_{\mathbf{f},\boldsymbol{\Delta}}(\gamma) &\leq 2 \cdot \sum_{e\in E} \boldsymbol{\Delta}^{\Sigma}(e)^2 \cdot 1 = 2 \cdot \sum_{e\in E} \left(\sum_{k=1}^{K} \Delta_e^k\right)^2 \\ &\leq 2 \cdot \sum_{e\in E} \left(\sum_{k=1}^{K} \left|\Delta_e^k\right|\right)^2 \leq 2 \cdot \left(\sum_{e\in E}\sum_{k=1}^{K} \left|\Delta_e^k\right|\right)^2 = 2 \cdot \|\boldsymbol{\Delta}\|_1^2, \end{aligned}$$

where we use, in turn, the properties of $\mathbb{1}_{x>0}(x)$, expansion, the triangle inequality for $|\cdot|$, the fact that $\sum_i a_i^2 \leq (\sum_i a_i)^2$ for $a_i \geq 0$ and the definition of $\|\cdot\|_1$.

This bound on the steepness of $h'_{\mathbf{f},\mathbf{\Delta}}$ means that $h'_{\mathbf{f},\mathbf{\Delta}}(\gamma) \leq h'_{\mathbf{f},\mathbf{\Delta}}(0) + 2\|\mathbf{\Delta}\|_1^2\gamma = w_{\mathbf{f}}(\mathbf{\Delta}) + 2\|\mathbf{\Delta}\|_1^2\gamma$. Specially, we get that

$$h'_{\mathbf{f},\mathbf{\Delta}}\left(-\frac{w_{\mathbf{f}}(\mathbf{\Delta})}{2\|\mathbf{\Delta}\|_1^2}\right) \leq 0.$$

Let $\varepsilon := -w_{\mathbf{f}}(\mathbf{\Delta})/(2\|\mathbf{\Delta}\|_1^2)$. This means that the total area below the x-axis can be bounded from below by the area of the triangle with vertices in $(0,0)$, $(0, w_{\mathbf{f}}(\mathbf{\Delta}))$ and $(\varepsilon, 0)$. Therefore,

$$\begin{aligned} \mathrm{IMP}_{\mathbf{f}}(\varepsilon\mathbf{\Delta}) &= -h_{\mathbf{f},\varepsilon\mathbf{\Delta}}(1) = \varphi(\mathbf{f} + \varepsilon\mathbf{\Delta}) - \varphi(\mathbf{f}) = -h_{\mathbf{f},\mathbf{\Delta}}(\varepsilon) \\ &\geq \textit{triangle area} = \left|\frac{1}{2} \cdot w_{\mathbf{f}}(\mathbf{\Delta}) \cdot \left(-\frac{w_{\mathbf{f}}(\mathbf{\Delta})}{2\|\mathbf{\Delta}\|_1^2}\right)\right| = \frac{w_{\mathbf{f}}(\mathbf{\Delta})^2}{4\|\mathbf{\Delta}\|_1^2}, \end{aligned}$$

which is precisely what we wanted to show. □

Remark 5.11. This bound is fairly crude and most likely not tight in the general case. It is, however, good enough to let us establish polynomial convergence (in some sense), although it is to blame for introducing relatively big exponents to the derived bounds.

Lemma 5.12. *Let* $\mathbf{f}$ *and* $\mathbf{\Delta}$ *satisfy the conditions from the previous lemma. Additionally, let it for all* $e \in E$ *hold that* $\mathbf{f}_e \leq \mathbf{d}$ *and* $(\mathbf{f} + \mathbf{\Delta})_e \leq \mathbf{d}$*, where* $\mathbf{d}$ *is the demand vector. Then there exists some* $\varepsilon \in (0,1]$ *such that:*

$$\mathrm{IMP}_{\mathbf{f}}(\varepsilon\mathbf{\Delta}) \geq \frac{w_{\mathbf{f}}(\mathbf{\Delta})^2}{4|E| \cdot \|\mathbf{d}\|_1^2}.$$

Proof. The proof is identical to that of the previous lemma, we just use a different bound on the steepness of $h'_{\mathbf{f},\mathbf{\Delta}}$. We already know that

$$h''_{\mathbf{f},\mathbf{\Delta}}(\gamma) \leq 2 \cdot \sum_{e \in E} \mathbf{\Delta}^{\Sigma}(e)^2.$$

The additional condition ensures that $|\mathbf{\Delta}_e^k| \leq d_k$ and therefore $\mathbf{\Delta}^{\Sigma}(e) \leq \sum_{k=1}^{K} |\mathbf{\Delta}_e^k| \leq \|\mathbf{d}\|_1$. From this it follows that $h''_{\mathbf{f},\mathbf{\Delta}}(\gamma) \leq 2|E| \cdot \|\mathbf{d}\|_1^2$. We may then repeat the rest of the proof with $2|E| \cdot \|\mathbf{d}\|_1^2$ in place of $2\|\mathbf{\Delta}\|_1^2$. □

Remark 5.13. This condition on $\mathbf{f}$, $\mathbf{\Delta}$ and $\mathbf{d}$ is easily achieved in HELPFUL-MCF (and indeed any "stateless" iterative algorithm). After each iteration, we may, in $\mathcal{O}(|E| \cdot K)$ time, find the canonical decomposition of the current $\mathbf{f}^i$ into weighted paths and positive cycles, and then drop all the cycles. This way, the flow on each edge may only decrease, and the penalty may thus only improve. In each commodity k, the values of all the paths sum up to d_k, meaning that the flow on each edge cannot exceed d_k either.

5.3.4 Achieving $\mathcal{O}(1/\varepsilon)$-like Convergence

The previous bound may be used almost directly to drive the convergence rate.

Lemma 5.14. *Let* $\mathbf{f} \in \mathbb{R}^{E\times K}$ *be a flow and* $\mathbf{\Delta}^*$ *a circulation such that* $\mathbf{f} + \mathbf{\Delta}^*$ *is feasible and optimal. Then there exists some feasible cycle* $\alpha\mathbf{C} \sqsubseteq \mathbf{\Delta}^*$ *such that:*

$$\mathrm{IMP}_{\mathbf{f}}(\alpha\mathbf{C}) \geq \frac{\mathrm{GAP}(\mathbf{f})^2}{4 \cdot |E \times K|^2 \cdot \|\mathbf{\Delta}^*\|_1^2}.$$

Moreover, if $(\mathbf{f} + \mathbf{\Delta}^*)_e, \mathbf{f}_e \leq \mathbf{d}$ *for all* e *(as is the case in* Helpful*-MCF), then:*

$$\mathrm{IMP}_{\mathbf{f}}(\alpha\mathbf{C}) \geq \frac{\mathrm{GAP}(\mathbf{f})^2}{4|E|^3|K|^2\|\mathbf{d}\|_1^2}.$$

Proof. We may assume that $\mathbf{f}$ is not optimal, otherwise $\alpha\mathbf{C} := \mathbf{0}$ trivially satisfies the conditions. Lemma 5.9 guarantees the existence of a cycle $\alpha'\mathbf{C} \sqsubseteq \mathbf{\Delta}^*$ such that $\frac{1}{|E|\cdot K} \cdot w_{\mathbf{f}}(\mathbf{\Delta}^*) \geq w_{\mathbf{f}}(\alpha'\mathbf{C}) \geq w_{\mathbf{f}}(\mathbf{\Delta}^*)$ (the latter inequality follows from the optimality of $\mathbf{\Delta}^*$). Additionally, $-w_{\mathbf{f}}(\mathbf{\Delta}^*) \geq \mathrm{IMP}_{\mathbf{f}}(\mathbf{\Delta}^*) = \mathrm{GAP}(\mathbf{f})$, which we get, respectively, from Lemma 5.10 and $\mathbf{f} + \mathbf{\Delta}^*$ being optimal. Also observe that $\mathbf{x} \sqsubseteq \mathbf{y}$ implies $\|\mathbf{x}\|_1 \leq \|\mathbf{y}\|_1$. Using Lemma 5.10 on $\alpha'\mathbf{C}$ gives us $\alpha \leq \alpha'$ such that:

$$\mathrm{IMP}_{\mathbf{f}}(\alpha\mathbf{C}) \geq \frac{w_{\mathbf{f}}(\alpha'\mathbf{C})^2}{4\|\alpha'\mathbf{C}\|_1^2}$$

Now we use, respectively, Lemma 5.9 (remember that both $w_{\mathbf{f}}$ are negative), the fact that $\alpha'\mathbf{C} \sqsubseteq \mathbf{\Delta}^*$, and twice the relationship between $w_{\mathbf{f}}$, $\mathrm{IMP}_{\mathbf{f}}$ and GAP.

$$\begin{aligned}\mathrm{IMP}_{\mathbf{f}}(\alpha\mathbf{C}) &\geq \frac{w_{\mathbf{f}}(\mathbf{\Delta}^*)^2}{4|E\times K|^2\|\alpha'\mathbf{C}\|_1^2} \geq \frac{w_{\mathbf{f}}(\mathbf{\Delta}^*)^2}{4|E\times K|^2\|\mathbf{\Delta}^*\|_1^2} \\ &\geq \frac{\mathrm{IMP}_{\mathbf{f}}(\mathbf{\Delta}^*)^2}{4|E\times K|^2\|\mathbf{\Delta}^*\|_1^2} = \frac{\mathrm{GAP}(\mathbf{f})^2}{4|E\times K|^2\|\mathbf{\Delta}^*\|_1^2}\end{aligned}$$

To prove the second part of the proof, we may redo it with Lemma 5.12 instead of Lemma 5.10, which gives us the factor of $|E| \cdot \|\mathbf{d}\|_1^2$ instead of $\|\mathbf{\Delta}^*\|_1^2$. □

Lemma 5.15. *Given a multiflow network* $\mathcal{G} = (G, \mathcal{K})$ *with* $G = (V, E, \mathbf{c})$ *and demands* $\mathbf{d} \in R^K$*, let* $\mathbf{f}^Q$ *be the flow obtained by performing* $Q \geq 1$ *iterations of the* Helpful*-MCF algorithm. Then*

$$\mathrm{GAP}(\mathbf{f}^Q) \leq \frac{16|E|^3|K|^2\|\mathbf{d}\|_1^2}{Q}.$$

Proof. Let $\alpha\mathbf{C}$ be the most-improving feasible cycle. Assuming correct bounds $\lambda_{\min}$, $\lambda_{\max}$ on the step size tried by Helpful-MCF, Lemma 5.7 guarantees us that it will find a cycle collection $\lambda\mathbf{D}$ with $\mathrm{IMP}_{\mathbf{f}^i}(\lambda\mathbf{D}) \geq 1/2 \cdot \mathrm{IMP}_{\mathbf{f}^i}(\alpha\mathbf{C})$. Plugging this into the bound from Lemma 5.14 (and using the relationship between GAP and $\mathrm{IMP}_{\mathbf{f}}$) yields:

$$\mathrm{IMP}_{\mathbf{f}^i}(\lambda\mathbf{D}) \geq \frac{\mathrm{GAP}(\mathbf{f}^i)^2}{8|E|^3|K|^2\|\mathbf{d}\|_1^2},$$

and thus,

$$\begin{aligned}\mathrm{GAP}(\mathbf{f}^{i+1}) = \mathrm{GAP}(\mathbf{f}^i) - \mathrm{IMP}_{\mathbf{f}^i}(\lambda\mathbf{D}) &\leq \mathrm{GAP}(\mathbf{f}^i) - \frac{\mathrm{GAP}(\mathbf{f}^i)^2}{8|E|^3|K|^2\|\mathbf{d}\|_1^2}, \\ &= \mathrm{GAP}(\mathbf{f}^i) \cdot \left(1 - \frac{\mathrm{GAP}(\mathbf{f}^i)}{8|E|^3|K|^2\|\mathbf{d}\|_1^2}\right).\end{aligned}$$

The rest then follows automatically from Lemma 1.10. □

Theorem 5.16. *Given a multiflow network $\mathcal{G} = (G, \mathcal{K})$ with $G = (V, E, \mathbf{c})$ and demands $\mathbf{d} \in \mathbb{R}^K$, Helpful-MCF finds a solution with total penalty within ε of the optimal penalty in time polynomial in input size, $\|\mathbf{d}\|_1$ and $1/\varepsilon$.*

Proof. Setting $Q = 16|E|^3|K|^2\|\mathbf{d}\|_1^2/\varepsilon$ and plugging it into Lemma 5.15 yields $\mathrm{GAP}(\mathbf{f}^Q) \leq \varepsilon$. Therefore, the number of iterations is polynomial, and each iteration takes polynomial time, assuming “reasonable” $\lambda_{\max}$ and $\lambda_{\min}$. $\square$

5.3.5 Inflation to the Rescue

As it turns out, using the inflation rate is enough to obtain an algorithm with $\mathcal{O}(\log(1/\varepsilon))$-style convergence, although the rate of convergence depends on the inflation rate. We start by improving the bound from Lemma 5.14.

Lemma 5.17. *Let $\mathcal{G}$ be a MCF instance, $P = \{\, \mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{a} \wedge B\mathbf{x} \leq \mathbf{b} \,\}$ the feasible polytope of its* (MCF-LP) *formulation, $H_{\mathrm{MCF}} = H_1(A, \mathbf{a}, B, \mathbf{b})$ the (ℓ_1-)inflation rate of P and let $\mathbf{f} \in \mathbb{R}^{E \times K}$ be a flow. Furthermore assume that the instance is satisfying, i.e., it has a solution with zero penalty. Then there exists a weighted cycle $\alpha\mathbf{C}$ such that*

$$\mathrm{IMP}_{\mathbf{f}}(\alpha\mathbf{C}) \geq \frac{\mathrm{GAP}(\mathbf{f})}{4|E|^3K^2H_{\mathrm{MCF}}^2}.$$

Proof. We may assume that $\mathbf{f}$ is not optimal, otherwise $\mathbf{0}$ satisfies the conditions. For a moment, we switch from our current interpretation (MCF-P) without capacity constraints back to the “stricter” one, (MCF-LP). The point $\mathbf{f}$ does not belong to the (MCF-LP) polyhedron, as it has nonzero penalty and therefore violates some constraints. In the language of polyhedra, if $P = \{\, \mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{a} \wedge B\mathbf{x} \leq \mathbf{b} \,\}$ is the (MCF-LP) polyhedron, then $\mathbf{f} \notin \{\, \mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{a} \wedge B\mathbf{x} \leq \mathbf{b} \,\}$, but there exists a nonnegative vector $\boldsymbol{\delta} = [B\mathbf{f} - \mathbf{b}]_+ \in \mathbb{R}^E$ – and, to account for the nonnegativity constraints which are kept intact, its zero-padded extension $\boldsymbol{\delta}'$ into $\mathbb{R}^{E \times (K+1)}$ – such that $\mathbf{f} \in P^{\boldsymbol{\delta}'} = \{\, \mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{a} \wedge B\mathbf{x} \leq \mathbf{b} + \boldsymbol{\delta}' \,\}$.

We may observe that $P^{\boldsymbol{\delta}'}$ is precisely the $\boldsymbol{\delta}'$-inflation of P. By the definition of the inflation rate, there exists some satisfying flow $\mathbf{f}^* \in P$ such that $\|\mathbf{f}^* - \mathbf{f}\|_1 \leq H_{\mathrm{MCF}} \cdot \|\boldsymbol{\delta}'\|_1 = H_{\mathrm{MCF}} \cdot \|\boldsymbol{\delta}\|_1$. Stated differently, there exists a circulation $\boldsymbol{\Delta}^*$ such that $\mathbf{f} + \boldsymbol{\Delta}^* \in P$ and $\|\boldsymbol{\Delta}^*\|_1 \leq H_{\mathrm{MCF}} \cdot \|\boldsymbol{\delta}\|_1$. Let us fix this circulation.

By Lemma 5.14, there exists a feasible cycle $\alpha\mathbf{C} \sqsubseteq \boldsymbol{\Delta}^*$ such that

$$\mathrm{IMP}_{\mathbf{f}}(\alpha\mathbf{C}) \geq \frac{\mathrm{GAP}(\mathbf{f})^2}{4 \cdot |E \times K|^2 \cdot \|\boldsymbol{\Delta}^*\|_1^2}.$$

Our strategy is to express $\mathrm{GAP}(\mathbf{f})$ in terms of $\|\boldsymbol{\delta}\|_1$, so that we can use the relationship between $\|\boldsymbol{\delta}\|_1$ and $\|\boldsymbol{\Delta}^*\|_1$. Realise that $\delta_e = \mathbf{f}^{\Sigma}(e) - c_e$. Then:

$$\mathrm{GAP}(\mathbf{f}) = \varphi(\mathbf{f}) - \varphi(\mathbf{f} + \boldsymbol{\Delta}^*) = \varphi(\mathbf{f}) = \sum_{e \in E} \left[\mathbf{f}^{\Sigma}(e) - c_e\right]_+^2 = \sum_{e \in E} \delta_e^2 = \|\boldsymbol{\delta}\|_2^2$$

Using the relation between ℓ_1- and ℓ_2-norms in R^E,

$$\|\boldsymbol{\delta}\|_2^2 \geq \|\boldsymbol{\delta}\|_1^2/|E|.$$

Using $\|\boldsymbol{\Delta}^*\|_1 \leq H_{\text{MCF}} \cdot \|\boldsymbol{\delta}\|_1$ yields:

$$\frac{\|\boldsymbol{\delta}\|_1}{\|\boldsymbol{\Delta}^*\|_1} \geq \frac{1}{H_{\text{MCF}}}.$$

Putting this all together:

$$\frac{\text{GAP}(\mathbf{f})}{4|E|^2K^2\|\boldsymbol{\Delta}^*\|_1^2} \geq \frac{\|\boldsymbol{\delta}\|_2^2}{4|E|^2K^2\|\boldsymbol{\Delta}^*\|_1^2} \geq \frac{\|\boldsymbol{\delta}\|_1^2}{4|E|^3K^2\|\boldsymbol{\Delta}^*\|_1^2} \geq \frac{1}{4|E|^3K^2H_{\text{MCF}}^2}.$$

Plugging this into the original inequality gives us the needed bound. □

Lemma 5.18. *Given a multiflow network $\mathcal{G} = (G, \mathcal{K})$ with $G = (V, E, \mathbf{c})$ and demands $\mathbf{d} \in \mathbb{R}^K$, let $\mathbf{f}^Q$ be the flow obtained by performing $Q \geq 0$ iterations of the* Helpful*-MCF algorithm. Furthermore assume that $\mathcal{G}$ has a solution with zero penalty. Then*

$$\text{GAP}(\mathbf{f}^Q) \leq |E| \cdot \|\mathbf{d}\|_1^2 \cdot \left(1 - \frac{1}{8|E|^3K^2H_{\text{MCF}}^2}\right)^Q.$$

Proof. The initial flow $\mathbf{f} = \mathbf{f}^0$ surely satisfies $e \in E : \mathbf{f}_e \leq \mathbf{d}$ and therefore

$$\text{GAP}(\mathbf{f}) = \varphi(\mathbf{f}) = \sum_{e\in E} \left[\mathbf{f}^\Sigma(e) - c_e\right]_+^2 \leq \sum_{e\in E} \mathbf{f}^\Sigma(e)^2 \leq \sum_{e\in E} \|\mathbf{d}\|_1^2 = |E| \cdot \|\mathbf{d}\|_1^2.$$

For $Q \geq 1$, we repeat the proof of Lemma 5.15 by assuming the algorithm has correct bounds $\lambda_{\min}$ and $\lambda_{\max}$ on the step size and invoking Lemma 5.7 about 2-approximation. Hence, the algorithm finds a cycle collection $\lambda\mathbf{D}$ such that $\text{IMP}_{\mathbf{f}^i}(\lambda\mathbf{D}) \geq 1/2 \cdot \text{IMP}_{\mathbf{f}^i}(\alpha\mathbf{C})$, where $\alpha\mathbf{C}$ is the most-improving weighted cycle. Plugging this into the bound from the previous lemma yields:

$$\text{IMP}_{\mathbf{f}^i}(\lambda\mathbf{D}) \geq \frac{\text{GAP}(\mathbf{f})}{8|E|^3K^2H_{\text{MCF}}^2}$$

and thus,

$$\begin{aligned}\text{GAP}(\mathbf{f}^{i+1}) = \text{GAP}(\mathbf{f}^i) - \text{IMP}_{\mathbf{f}^i}(\lambda\mathbf{D}) &\leq \text{GAP}(\mathbf{f}^i) - \frac{\text{GAP}(\mathbf{f})}{8|E|^3K^2H_{\text{MCF}}^2} \\ &= \text{GAP}(\mathbf{f}^i) \cdot \left(1 - \frac{1}{8|E|^3K^2H_{\text{MCF}}^2}\right).\end{aligned}$$

The rest is just applying induction on Q. □

Theorem 5.19. *Let $\mathcal{G} = (G, \mathcal{K})$ be a multiflow network with $G = (V, E, \mathbf{c})$ and demands $\mathbf{d} \in \mathbb{R}^K$. Assuming $\mathcal{G}$ has a solution with zero penalty,* Helpful*-MCF finds a solution with penalty at most ε in time polynomial in input size, H_{MCF} and $\log(\|\mathbf{d}\|_1/\varepsilon)$.*

Proof. By applying Lemma 1.11 on the previous lemma, we get that we obtain $\varphi(\mathbf{f}^Q) = \text{GAP}(\mathbf{f}^Q) \leq \varepsilon$ after

$$Q = 2\log(|E| \cdot \|\mathbf{d}\|_1/\varepsilon) \cdot 8|E|^3K^2H_{\text{MCF}}^2$$

iterations. □

Corollary 5.20. Let $\mathcal{G} = (G, \mathcal{K})$ be a multiflow network with $G = (V, E, \mathbf{c})$ and demands $\mathbf{d} \in \mathbb{R}^K$. Assuming that H_{MCF} is polynomial in input size and $\mathcal{G}$ has a solution with zero penalty, Helpful-MCF finds a solution with penalty at most ε in time polynomial in input size and $\log(\|\mathbf{d}\|_1/\varepsilon)$.

5.3.6 Filling in the Gaps in Helpful-MCF

There are two remaining details: how to set $\lambda_{\min}$ and $\lambda_{\max}$, and how to deal with instances that do not have a satisfying solution.

The latter question has an easy, but impractical answer. The algorithm simply assumes that a satisfying solution exists and that $\mathrm{GAP}(\mathbf{f}) = \varphi(\mathbf{f})$, and maintains an upper bound on $\mathrm{GAP}(\mathbf{f})$, as per the previous lemma, while continously checking that $\varphi(\mathbf{f})$ still satisfies it. As long as the upper bound holds, the algorithm makes good enough progress whether or not the instance is satisfying. If the upper bound is violated, this necessarily means that in fact $\mathrm{GAP}(\mathbf{f}) < \varphi(\mathbf{f})$ and the instance is unsatisfying. In that case, the algorithm stops.

A major disadvantage is that although the algorithm may generally be much faster than what is guaranteed by our bounds (and we are almost certain of that because of their crudeness), this approach will most probably lead to a self-fulfilling prophecy in the unsatisfying case, that is, the bounds will be tight. Additionally, unless one uses the weaker $\mathcal{O}(1/\varepsilon)$-style bound, one needs to have a reasonable bound on H_{MCF}, but we are currently unaware of even a subexponential bound.

What about $\lambda_{\min}$ and $\lambda_{\max}$? The latter can be set to $\max_{k=1}^{K} d_k$ (or to the nearest greater power of two thereof). For the former, we need to provide a lower bound on $|\alpha|$ of the most-improving weighed cycle $\alpha\mathbf{C} \sqsubseteq \mathbf{\Delta}^*$ that appears in Lemmas 5.14 and 5.17. Thanks to Remark 5.13, we may use the second part of Lemma 5.14, which guarantees that

$$\mathrm{IMP}_{\mathbf{f}}(\alpha\mathbf{C}) \geq \frac{\mathrm{GAP}(\mathbf{f})^2}{4|E|^3|K|^2\|\mathbf{d}\|_1^2}.$$

At the same time, from Lemma 5.10, we know that $-w_{\mathbf{f}}(\alpha\mathbf{C}) \geq \mathrm{IMP}_{\mathbf{f}}(\alpha\mathbf{C})$ and by the definition of $w_{\mathbf{f}}$,

$$\begin{aligned}
-w_{\mathbf{f}}(\alpha\mathbf{C}) &= -\alpha w_{\mathbf{f}}(\mathbf{C}) = -\alpha h'_{\mathbf{f},\mathbf{C}}(0) = -2\alpha \cdot \sum_{e\in E} \mathbf{C}^{\Sigma}(e) \cdot \Big[\mathbf{f}^{\Sigma}(e) - c_e\Big]_+ \\
&\leq -2\alpha \cdot \sum_{e\in E} (-1) \cdot \Big[\mathbf{f}^{\Sigma}(e) - c_e\Big]_+ = 2\alpha \cdot \Big\|\Big[\mathbf{f}^{\Sigma} - \mathbf{c}\Big]_+\Big\|_1 \\
&\leq 2\alpha \cdot \Big\|\mathbf{f}^{\Sigma}\Big\|_1 = 2\alpha \cdot \|\mathbf{f}\|_1 \leq 2\alpha|E| \cdot \|\mathbf{d}\|_1 .
\end{aligned}$$

Chained together, we get

$$2\alpha|E| \cdot \|\mathbf{d}\|_1 \geq -w_{\mathbf{f}}(\alpha\mathbf{C}) \geq \mathrm{IMP}_{\mathbf{f}}(\alpha\mathbf{C}) \geq \frac{\mathrm{GAP}(\mathbf{f})^2}{4|E|^3|K|^2\|\mathbf{d}\|_1^2},$$

and therefore,

$$\alpha \geq \frac{\mathrm{GAP}(\mathbf{f})^2}{8|E|^4|K|^2\|\mathbf{d}\|_1^3} \geq \frac{\varepsilon^2}{8|E|^4|K|^2\|\mathbf{d}\|_1^3},$$

where ε is the final approximation error. This is the value that we may set $\lambda_{\min}$ to. Although it may seem large at first, remember that we are only trying $\log(\lambda_{\max}/\lambda_{\min})$ different step sizes in each iteration, and

$$\log(\lambda_{\max}/\lambda_{\min}) \leq \log\left(\frac{8|E|^4|K|^2\|\mathbf{d}\|_1^4}{\varepsilon^2}\right) = \mathcal{O}\left(\log\left(|E| \cdot |K| \cdot \|\mathbf{d}\|_1/\varepsilon\right)\right),$$

and hence the total time complexity is not impacted in any significant way.

Conclusion

In this work, we studied multicommodity and length-bounded flows, and although these two problems do not seem very related at the first sight, they share not only a similar LP formulation, but also a very similar structure and open questions. Both problems seem in some sense inherently more difficult than the maximum flow problem and both have so far resisted attempts both at creating exact polynomial-time combinatorial algorithms for them, and at proving that such algorithms cannot exist – for whatever definition of "combinatorial".

We have tried to bridge this gap, and even though we have not succeeded fully, we have contributed several new pieces to the puzzle. We have proposed two new algorithms: an adaptation of Away-Steps FW for MCF and LBF and the Helpful-MCF algorithm for MCF. Both of them run in time polynomial in the input size, $1/\varepsilon$ and the demands. While Helpful-MCF is quite undoubtedly combinatorial, the boundaries are more blurred in the case of Away-Steps FW – the algorithm itself is simple, but its analysis is fairly complex. Whatever the reader's feeling about this may be, our ultimate motivation is to look for fast algorithms that do not use the deep machinery of LP, and in this regard, both algorithms have brought us further along this path.

One of our main contributions is examining the conditions under which both algorithms are "practically exact" – with only $\log(1/\varepsilon)$ dependence on the approximation error[4] – while simultaneously uncovering the common bottleneck of both approaches. These results are summarised in Figure 5.2.

The central definition is the inflation rate. Surprisingly, this property summed up as "a flow that only oversteps the capacities a little must be close to some flow that satisfies the capacities" is enough to guarantee fast convergence of both algorithms – even though they do not seem very related at first glance. Another surprising fact is that although small inflation rate seems almost trivial for single-commodity maximum flow, it is far from that once one introduces more commodities or layers.

A similar situation occurs with circuits: while single-commodity circuits are simply all undirected cycles, the situation gets more complicated with multiple commodities / layers, and as we have shown, there exist circuits of exponential size that send 1 unit of flow over some edge and $2^{\Theta(|E| \times K)}$ units over some other edge. This is in a sense both fortunate and unfortunate. It is fortunate because it gives us a possible clue as to why MCF and LBF seem so much harder than other popular flow problems, and potentially opens doors to other hardness results. At the same time, it is unfortunate, since one of our results is that (for a general LP problem) small circuits imply small inflation rate, which, in our case, would imply fast convergence.

Luckily, the proof does not rely on all properties of circuits – it works with any universal test set, and even with any MCF-/LBF-universal test set. Lastly,

[4]With a bit of LP theory, algorithms that converge this quickly actually *are*, in a sense, exact, since for solutions with error $\varepsilon = \Theta(1/n^n)$ (so that $\log(1/\varepsilon) = \Theta(n \log n)$), the optimal solution can be obtained by rounding the current solution to the nearest vertex of the polyhedron – although the known procedures for finding this vertex perhaps would not be considered combinatorial.

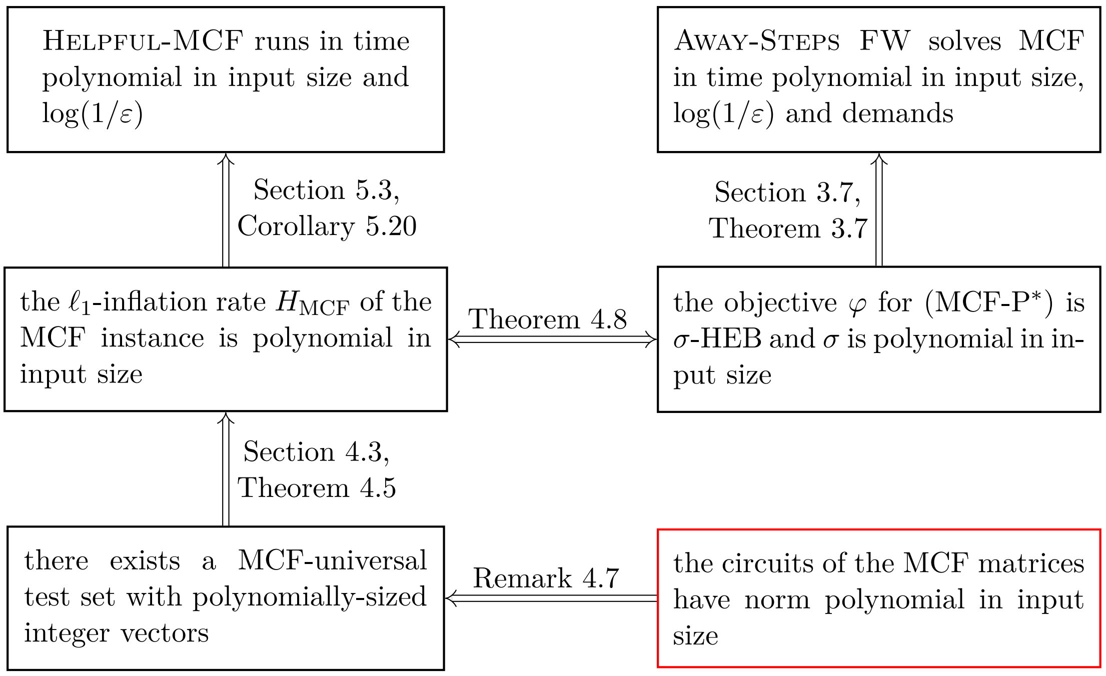
HELPFUL-MCF runs in time polynomial in input size and $\log(1/\varepsilon)$
AWAY-STEPS FW solves MCF in time polynomial in input size, $\log(1/\varepsilon)$ and demands
Section 5.3, Corollary 5.20
Section 3.7, Theorem 3.7
the ℓ_1-inflation rate H_{MCF} of the MCF instance is polynomial in input size
Theorem 4.8
the objective φ for (MCF-P*) is σ-HEB and σ is polynomial in input size
Section 4.3, Theorem 4.5
there exists a MCF-universal test set with polynomially-sized integer vectors
Remark 4.7
the circuits of the MCF matrices have norm polynomial in input size

Figure 5.2: The implication diagram from Figure 1 revisited. The statement in red has been proven false, all other statements remain undecided. The diagram for LBF is practically the same, except that there is no HELPFUL-LBF algorithm.

it is not the only path for showing small inflation rate or fast convergence, unless, of course, the opposite implication is proven.

Further Work

Alternative MCF-/LBF-universal test sets are an area we deem worthy of exploring. A prominent example is the set of all edge directions that can ever appear (for some choice of constraints' right-hand side that forms a valid MCF / LBF instance), which may generally be much smaller (and contain smaller vectors) than the circuit set.

This is a possible reason why we have not so far been successful in constructing a network and a large circuit that appears as an edge in the corresponding polyhedron, which is another topic worth exploring. A similar endeavour is trying to construct an instance where the optimal solution has large fractionality, i.e., where it is described by a rational vector with at least one large denominator. We are convinced that using a large circuit to construct a polytope with the corresponding constraint matrices and with a vertex of high fractionality is possible, but it gets more interesting in our case where not all right-hand sides form a valid MCF / LBF instance.

An analogous question can be asked about the inflation rate: do large circuits already imply large inflation rate? Again, it seems fairly feasible to construct a polyhedron with large inflation rate once we have a large circuit and unlimited freedom in choosing the right-hand side, but the question gets much more interesting if we only limit ourselves to MCF / LBF instances.

Lastly, we are interested in the similarities and differences between MCF and

LBF. Although they share almost all results, this may not be the case with perhaps the most important one: there is HELPFUL-MCF for MCF, but so far no HELPFUL-LBF for LBF. An important question to answer is whether there always exists an improving cycle that does not use more than one copy of each edge in the layered network, and generally, whether there is a way of adapting HELPFUL-MCF for LBF.

Bibliography

[AK71] Jiří Adámek and Václav Koubek. "Remarks on flows in network with short paths". In: *Commentationes Mathematicae Universitatis Carolinae* 12.4 (1971), pp. 661–667.

[AMO88] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network Flows*. Massachusetts Institute of Technology, Operations Research Center, 1988.

[Alt18] Kateřina Altmanová. "Toky cestami omezené délky". BA thesis. Univerzita Karlova, Matematicko-fyzikální fakulta, 2018.

[AKV20] Kateřina Altmanová, Petr Kolman, and Jan Voborník. "On Polynomial-Time Combinatorial Algorithms for Maximum L-Bounded Flow". In: *J. Graph Algorithms Appl.* 24.3 (2020), pp. 303–322. DOI: 10.7155/jgaa.00534. URL: https://doi.org/10.7155/jgaa.00534.

[Ass78] Arjang A Assad. "Multicommodity network flows—a survey". In: *Networks* 8.1 (1978), pp. 37–91.

[Bai04] Georg Baier. *Flows with path restrictions*. Göttingen: Cuvillier, 2004. ISBN: 9783898739641.

[Bai+10] Georg Baier et al.. "Length-bounded cuts and flows". In: *ACM Transactions on Algorithms (TALG)* 7.1 (2010), pp. 1–27.

[BT89] Francisco Barahona and Éva Tardos. "Note on Weintraub's minimum-cost circulation algorithm". In: *SIAM Journal on Computing* 18.3 (1989), pp. 579–583.

[BS17] Amir Beck and Shimrit Shtern. "Linearly convergent away-step conditional gradient for non-strongly convex functions". In: *Math. Program.* 164.1-2 (2017), pp. 1–27. DOI: 10.1007/s10107-016-1069-4. URL: https://doi.org/10.1007/s10107-016-1069-4.

[BDF16] S Borgwardt, JA De Loera, and E Finhold. "Edges versus circuits: A hierarchy of diameters in polyhedra". In: *Advances in Geometry* 16.4 (2016), pp. 511–530.

[BSY18] Steffen Borgwardt, Tamon Stephen, and Timothy Yusun. "On the Circuit Diameter Conjecture". In: *Discret. Comput. Geom.* 60.3 (2018), pp. 558–587. DOI: 10.1007/s00454-018-9995-y. URL: https://doi.org/10.1007/s00454-018-9995-y.

[BV19] Steffen Borgwardt and Charles Viss. "A polyhedral model for enumeration and optimization over the set of circuits". In: *Discrete Applied Mathematics* (2019).

[BV20] Steffen Borgwardt and Charles Viss. "An implementation of steepest-descent augmentation for linear programs". In: *Operations Research Letters* 48.3 (2020), pp. 323–328.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Mar. 2004. ISBN: 9780521833783.

[BPZ19] Gábor Braun, Sebastian Pokutta, and Daniel Zink. “Lazifying Conditional Gradient Algorithms”. In: *J. Mach. Learn. Res.* 20 (2019), 71:1–71:42. URL: http://jmlr.org/papers/v20/18-114.html.

[CD74] H Chen and CG Dewald. “A generalized chain labelling algorithm for solving multicommodity flow problems”. In: *Computers & Operations Research* 1.3-4 (1974), pp. 437–465.

[CK20] Eden Chlamtáč and Petr Kolman. “How to Cut a Ball Without Separating: Improved Approximations for Length Bounded Cut”. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*. Ed. by Jaroslaw Byrka and Raghu Meka. Vol. 176. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 41:1–41:17. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2020.41. URL: https://doi.org/10.4230/LIPIcs.APPROX/RANDOM.2020.41.

[CST70] JE Cremeans, RA Smith, and GR Tyndall. “Optimal multicommodity network flows with resource allocation”. In: *Naval Research Logistics Quarterly* 17.3 (1970), pp. 269–279.

[Dad+20] Daniel Dadush et al.. “A scaling-invariant algorithm for linear programming whose running time depends only on the constraint matrix”. In: *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*. Ed. by Konstantin Makarychev et al.. ACM, 2020, pp. 761–774. DOI: 10.1145/3357713.3384326. URL: https://doi.org/10.1145/3357713.3384326.

[DW60] George B Dantzig and Philip Wolfe. “Decomposition principle for linear programs”. In: *Operations research* 8.1 (1960), pp. 101–111.

[DK18] Pavel Dvořák and Dušan Knop. “Parameterized Complexity of Length-bounded Cuts and Multicuts”. In: *Algorithmica* 80.12 (2018), pp. 3597–3617. DOI: 10.1007/s00453-018-0408-7. URL: https://doi.org/10.1007/s00453-018-0408-7.

[EIS76] S Even, A Itai, and A Shamir. “On the Complexity of Timetable and Multicommodity Flow Problems”. In: *SIAM Journal on Computing* 5.4 (1976), pp. 691–703.

[Fle99] Lisa Fleischer. “Approximating fractional multicommodity flow independent of the number of commodities”. In: *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)* (1999), pp. 24–31.

[FF56] LR Ford and DR Fulkerson. “Maximal Flow Through a Network”. In: *Canadian Journal of Mathematics* 8 (1956), pp. 399–404.

[FF58] Lester Randolph Ford Jr and Delbert R Fulkerson. “A suggested computation for maximal multi-commodity network flows”. In: *Management Science* 5.1 (1958), pp. 97–101.

[FW+56] Marguerite Frank, Philip Wolfe, et al.. “An algorithm for quadratic programming”. In: *Naval Research Logistics Quarterly* 3.1-2 (1956), pp. 95–110.

[GK07] Naveen Garg and Jochen Könemann. "Faster and simpler algorithms for multicommodity flow and other fractional packing problems". In: *SIAM Journal on Computing* 37.2 (2007), pp. 630–652.

[GDL15] Jean Bertrand Gauthier, Jacques Desrosiers, and Marco E Lübbecke. "About the minimum mean cycle-canceling algorithm". In: *Discrete Applied Mathematics* 196 (2015), pp. 115–134.

[GG74] Arthur M Geoffrion and Glenn W Graves. "Multicommodity distribution system design by Benders decomposition". In: *Management science* 20.5 (1974), pp. 822–844.

[Gol92] Andrew V Goldberg. "A natural randomization strategy for multicommodity flow and related algorithms". In: *Information Processing Letters* 42.5 (1992), pp. 249–256.

[GPT91] Andrew V Goldberg, Serge A Plotkin, and Éva Tardos. "Combinatorial algorithms for the generalized circulation problem". In: *Mathematics of operations research* 16.2 (1991), pp. 351–381.

[GT90] Andrew V Goldberg and Robert E Tarjan. "Finding minimum-cost circulations by successive approximation". In: *Mathematics of Operations Research* 15.3 (1990), pp. 430–466.

[GM76] Glenn W Graves and Richard D McBride. "The factorization approach to large-scale linear programming". In: *Mathematical Programming* 10.1 (1976), pp. 91–110.

[GLS12] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization.* Vol. 2. Springer Science & Business Media, 2012.

[GM86] Jacques Guélat and Patrice Marcotte. "Some comments on Wolfe's 'away step'". In: *Mathematical Programming* 35.1 (1986), pp. 110–119.

[GHR95] Osman Güler, Alan J Hoffman, and Uriel G Rothblum. "Approximations to solutions to systems of linear inequalities". In: *SIAM Journal on Matrix Analysis and Applications* 16.2 (1995), pp. 688–696.

[Gup21] Swati Gupta. Private communication. May 2021.

[Hof52] Alan J Hoffman. "On Approximate Solutions of Systems of Linear Inequalities". In: *Journal of Research of the National Bureau of Standards* 49.4 (1952).

[Ita78] Alon Itai. "Two-commodity flow". In: *Journal of the ACM (JACM)* 25.4 (1978), pp. 596–611.

[IPS82a] Alon Itai, Yehoshua Perl, and Yossi Shiloach. "The complexity of finding maximum disjoint paths with length constraints". In: *Networks* 12.3 (1982), pp. 277–286.

[IPS82b] Alon Itai, Yehoshua Perl, and Yossi Shiloach. "The complexity of finding maximum disjoint paths with length constraints". In: *Networks* 12.3 (1982), pp. 277–286. DOI: `10.1002/net.3230120306`. URL: `https://doi.org/10.1002/net.3230120306`.

[JLR20] Klaus Jansen, Alexandra Lassota, and Lars Rohwedder. "Near-Linear Time Algorithm for n-Fold ILPs via Color Coding". In: *SIAM J. Discret. Math.* 34.4 (2020), pp. 2282–2299. DOI: 10.1137/19M1303873. URL: https://doi.org/10.1137/19M1303873.

[Kar84] Narendra Karmarkar. "A new polynomial-time algorithm for linear programming". In: *Proceedings of the sixteenth annual ACM symposium on Theory of computing.* 1984, pp. 302–311.

[Ken78] Jeff L Kennington. "A survey of linear cost multicommodity network flows". In: *Operations Research* 26.2 (1978), pp. 209–236.

[KdP18] Thomas Kerdreux, Alexandre d'Aspremont, and Sebastian Pokutta. *Restarting Frank-Wolfe: Faster Rates Under Hölderian Error Bounds.* 2018. arXiv: 1810.02429 [math.OC].

[Kha80] Leonid G Khachiyan. "Polynomial algorithms in linear programming". In: *USSR Computational Mathematics and Mathematical Physics* 20.1 (1980), pp. 53–72.

[Kle+94] Philip Klein et al.. "Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts". In: *SIAM Journal on Computing* 23.3 (1994), pp. 466–487.

[KS06] Petr Kolman and Christian Scheideler. "Improved bounds for the unsplittable flow problem". In: *J. Algorithms* 61.1 (2006), pp. 20–44.

[KŘ81] Václav Koubek and Antonín Říha. "The maximum k-flow in a network". In: *International Symposium on Mathematical Foundations of Computer Science.* Springer. 1981, pp. 389–397.

[KLO18] Martin Koutecký, Asaf Levin, and Shmuel Onn. "A Parameterized Strongly Polynomial Algorithm for Block Structured Integer Programs". In: *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic.* Ed. by Ioannis Chatzigiannakis et al.. Vol. 107. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 85:1–85:14. DOI: 10.4230/LIPIcs.ICALP.2018.85. URL: https://doi.org/10.4230/LIPIcs.ICALP.2018.85.

[LJ15] Simon Lacoste-Julien and Martin Jaggi. "On the Global Linear Convergence of Frank-Wolfe Optimization Variants". In: *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada.* Ed. by Corinna Cortes et al.. 2015, pp. 496–504. URL: https://proceedings.neurips.cc/paper/2015/hash/c058f544c737782deacefa532d9add4c-Abstract.html.

[Lei+95] Tom Leighton et al.. "Fast approximation algorithms for multicommodity flow problems". In: *Journal of Computer and System Sciences* 50.2 (1995), pp. 228–243.

[Liu19] Pengfei Liu. "A Combinatorial Algorithm for the Multi-commodity Flow Problem". In: *CoRR* abs/1904.09397 (2019). arXiv: 1904.09397. URL: http://arxiv.org/abs/1904.09397.

[Mąd10] Aleksander Mądry. “Faster approximation schemes for fractional multicommodity flow problems via dynamic graph algorithms”. In: *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*. Ed. by Leonard J. Schulman. ACM, 2010, pp. 121–130. DOI: 10.1145/1806689.1806708. URL: https://doi.org/10.1145/1806689.1806708.

[Man81] Olvi L Mangasarian. *A Condition Number for Linear Inequalities and Linear Programs.*. Tech. rep.. Wisconsin Univ-Madison Mathematics Research Center, 1981.

[Mar17] Martin Mareš. *Průvodce labyrintem algoritmů*. CZ.NIC, 2017. ISBN: 9788088168195.

[Meg84] N. Megiddo. “Linear Programming in Linear Time When the Dimension is Fixed”. In: *Journal of Association for Computing Machinery* 31.1 (Jan. 1984), pp. 114–127.

[Mil16] Jacob Andrew Miller. *Transportation networks and matroids: algorithms through circuits and polyhedrality.* University of California, Davis, 2016.

[NN94] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming.* SIAM, 1994.

[OV20] Neil Olver and László A. Végh. “A Simpler and Faster Strongly Polynomial Algorithm for Generalized Flow Maximization”. In: *J. ACM* 67.2 (2020), 10:1–10:26. DOI: 10.1145/3383454. URL: https://doi.org/10.1145/3383454.

[Onn10] Shmuel Onn. *Nonlinear Discrete Optimization.* Zurich Lectures in Advanced Mathematics. European Mathematical Society Publishing House, 2010. ISBN: 978-3-03719-593-2. DOI: 10.4171/093.

[Ope13] Operations Research Group, University of Pisa. *Multicommodity Problems.* 2013. URL: http://groups.di.unipi.it/optimize/Data/MMCF.html (visited on 06/22/2020).

[PR19] Javier Peña and Daniel Rodríguez. “Polytope Conditioning and Linear Convergence of the Frank-Wolfe Algorithm”. In: *Math. Oper. Res.* 44.1 (2019), pp. 1–18. DOI: 10.1287/moor.2017.0910. URL: https://doi.org/10.1287/moor.2017.0910.

[Pok18a] Sebastian Pokutta. *Cheat Sheet: Hölder Error Bounds for Conditional Gradients.* 2018. URL: https://www.pokutta.com/blog/research/2018/12/07/cheatsheet-smooth-idealized.html (visited on 07/21/2020).

[Pok18b] Sebastian Pokutta. *Cheat Sheet: Hölder Error Bounds for Conditional Gradients.* 2018. URL: http://www.pokutta.com/blog/research/2018/11/12/heb-conv.html (visited on 07/21/2020).

[RT12] Lyle Ramshaw and Robert E Tarjan. “On minimum-cost assignments in unbalanced bipartite graphs”. In: *HP Labs, Palo Alto, CA, USA, Tech. Rep. HPL-2012-40R1* (2012).

[Sch91] Rina Schneur. “Scaling algorithms for multicommodity flow problems and network flow problems with side constraits”. PhD thesis. Massachusetts Institute of Technology, 1991.

[SO98] Rina R Schneur and James B Orlin. “A scaling algorithm for multicommodity flow problems”. In: *Operations Research* 46.2 (1998), pp. 231–246.

[Sch99] Alexander Schrijver. *Theory of linear and integer programming.* Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1999. ISBN: 978-0-471-98232-6.

[SM90] Farhad Shahrokhi and David W Matula. “The maximum concurrent flow problem”. In: *Journal of the ACM (JACM)* 37.2 (1990), pp. 318–334.

[SIM00] Maiko Shigeno, Satoru Iwata, and S Thomas McCormick. “Relaxed most negative cycle and most positive cut canceling algorithms for minimum cost flow”. In: *Mathematics of Operations Research* 25.1 (2000), pp. 76–104.

[Ste12] Clifford Stein. *Multicommodity Flow.* Apr. 2012. URL: `http://www.columbia.edu/~cs2035/courses/ieor6614.S16/multi.pdf` (visited on 12/14/2020).

[Tar86] Éva Tardos. “A strongly polynomial algorithm to solve combinatorial linear programs”. In: *Operations Research* 34.2 (1986), pp. 250–256.

[VO21] Vijay Vazirani and James Orlin. Private communication. Mar. 2021.

[Vég17] László A. Végh. “A Strongly Polynomial Algorithm for Generalized Flow Maximization”. In: *Math. Oper. Res.* 42.1 (2017), pp. 179–211. DOI: `10.1287/moor.2016.0800`. URL: `https://doi.org/10.1287/moor.2016.0800`.

[Vob16] Jan Voborník. “Algoritmy pro L-omezené toky”. MA thesis. Univerzita Karlova, Matematicko-fyzikální fakulta, 2016.

[Way02] Kevin D Wayne. “A polynomial combinatorial algorithm for generalized minimum cost flow”. In: *Mathematics of Operations Research* 27.3 (2002), pp. 445–459.

[Wei74] Andres Weintraub. “A primal algorithm to solve network flow problems with convex costs”. In: *Management Science* 21.1 (1974), pp. 87–97.

[Wol70] Philip Wolfe. “Convergence theory in nonlinear programming”. In: *Integer and nonlinear programming* (1970), pp. 1–36.

[You95] Neal E Young. “Randomized rounding without solving the linear program”. In: *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms.* Vol. 76. SIAM. 1995, p. 170.