On Linear, Fractional, and Submodular Optimization

Zhuan Khye Koh

Department of Mathematics
London School of Economics and Political Science

This dissertation is submitted for the degree of
Doctor of Philosophy

London, January 2023

I would like to dedicate this thesis to my loving parents.

Declaration

I certify that the thesis I have presented for examination for the PhD degree of the London School of Economics and Political Science is solely my own work, with the following exceptions.

- Chapter 2 is based on [38] "An Accelerated Newton–Dinkelbach Method and its Application to Two Variables per Inequality Systems", co-authored with Daniel Dadush, Bento Natura and László Végh. It has appeared in *Proceedings of the 29th European Symposium on Algorithms (ESA 2021)* and was accepted to *Mathematics of Operations Research*.

- Chapter 3 is based on [39] "On Circuit Diameter Bounds via Circuit Imbalances", co-authored with Daniel Dadush, Bento Natura and László Végh. It has appeared in *Proceedings of the 23rd International Conference on Integer Programming and Combinatorial Optimization (IPCO 2022).*

- Chapter 4 is based on [82] "On the Correlation Gap of Matroids", co-authored with Edin Husić, Georg Loho and László Végh. It was accepted to the *24th International Conference on Integer Programming and Combinatorial Optimization (IPCO 2023).*

- Chapter 5 is based on [96] "Beyond Value Iteration for Parity Games: Strategy Iteration with Universal Trees", co-authored with Georg Loho. It has appeared in *Proceedings of the 47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022).*

Zhuan Khye Koh
London, January 2023

Acknowledgements

First and foremost, I would like to express my gratitude to my advisor László Végh, for his constant support and guidance over the past four years. This PhD journey has not been an easy one, and I was fortunate to have his wisdom and encouragement in navigating the research landscape. I have learned a great deal from him, both as a mathematician and as a human being. Thank you Laci – it has been a privilege to be your student.

During my studies, I had the great pleasure of interacting and collaborating with other fellow researchers. I am grateful to my co-authors Daniel Dadush, Edin Husić, Georg Loho and Bento Natura on the joint work included in this thesis. I am also thankful to Neil Olver and Sorrachai Yingchareonthawornchai for the insightful discussions that we have had, and the knowledge you have shared with me.

I would like to thank Tugkan Batu, Alina Ene and Nathanaël Fijalkow for agreeing to be my examiners. Thank you for reading my thesis and expressing interest in my work.

I thank all members of the Department of Mathematics at LSE for creating a friendly and supportive environment. Special mention goes to Enfale, Kate, Ed, Sarah and Sharon for their help with various administrative matters throughout my time here.

The friends that I have made constitute an integral part of my life in London. I will fondly remember our academic discussions, hangouts and trips, not to mention the climbing, squash, gym and running sessions which provided a perfect counterbalance to the demands of research. Thank you Edin, Xinyi, Christoph, Franzi, Bento, Amedeo, Raymond, Jan, Keat, Stan, Domenico, Justin, Sahar, Khairul, Neeraj . . . . Special thanks to my lovely housemates for preserving my sanity during the lockdowns. I would also like to give a shout out to my Waterloo friends Jimmy, Alex, Sharat, Akshay, Daniel and Dean, for being there with me. I will always cherish the memories and good times we share.

This thesis was written and completed in the warm and conducive home of Nicola's. I thank her for taking care of me while I was ill, and also for the nourishing meals and moral support leading up to the final days.

I am forever grateful to my parents Lai Kheng and Lai Chun, and to my brother Zhuan Hao, for their unconditional love and support throughout the years.

I gratefully acknowledge the financial support provided by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. ScaleOpt–757481). Part of the work in this thesis was carried out while I participated in the Discrete Optimization Trimester Program at the Hausdorff Research Institute for Mathematics in Bonn during the fall of 2021.

Abstract

In this thesis, we study four fundamental problems in the theory of optimization.

1. In fractional optimization, we are interested in minimizing a ratio of two functions over some domain. A well-known technique for solving this problem is the Newton–Dinkelbach method. We propose an accelerated version of this classical method and give a new analysis using the Bregman divergence. We show how it leads to improved or simplified results in three application areas.

2. The diameter of a polyhedron is the maximum length of a shortest path between any two vertices. The circuit diameter is a relaxation of this notion, whereby shortest paths are not restricted to edges of the polyhedron. For a polyhedron in standard equality form with constraint matrix A, we prove an upper bound on the circuit diameter that is quadratic in the rank of A and logarithmic in the circuit imbalance measure of A. We also give circuit augmentation algorithms for linear programming with similar iteration complexity.

3. The correlation gap of a set function is the ratio between its multilinear and concave extensions. We present improved lower bounds on the correlation gap of a matroid rank function, parametrized by the rank and girth of the matroid. We also prove that for a weighted matroid rank function, the worst correlation gap is achieved with uniform weights. Such improved lower bounds have direct applications in submodular maximization and mechanism design.

4. The last part of this thesis concerns parity games, a problem intimately related to linear programming. A parity game is an infinite-duration game between two players on a graph. The problem of deciding the winner lies in NP and co-NP, with no known polynomial algorithm to date. Many of the fastest (quasi-polynomial) algorithms have been unified via the concept of a universal tree. We propose a strategy iteration framework which can be applied on any universal tree.

Table of Contents

List of Figures

Chapter 1

Introduction

Optimization is the selection of a best element from a set of solutions based on some criterion. Due to its extremely broad scope, it arises frequently in many areas such as engineering, operations research, economics and computer science. Mathematically, given a domain $\mathcal{D}$ and a function $f : \mathcal{D} \to \mathbb{R}$, an optimization problem can be expressed as

$$\inf_{x \in \mathcal{D}} f(x). \tag{1.1}$$

In this thesis, we study the algorithmic and geometric aspects of solving optimization problems. The results in Chapters 2 – 5 focus on three classical optimization models, as detailed next.

1.1 Three Optimization Models

1.1.1 Linear Optimization

If $\mathcal{D}$ is a polyhedron and f is a linear function, then (1.1) is called a *linear program (LP)*. For a matrix $A \in \mathbb{R}^{m\times n}$ and vectors $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, an LP in *standard equality form* is given by

$$\min\{c^\top x : Ax = b, x \geq \mathbb{0}\}.$$

Linear programming is widely used in theory and practice due to its expressibility and tractibility. Many practical problems in operations research can be formulated as LPs. Furthermore, LPs are often used as subproblems in algorithms for NP-hard problems.

LPs can be solved in polynomial time using the ellipsoid method [15] or interior point methods [158]. Another method for solving LPs is the simplex method, which is commonly used in practice due to its good empirical performance. It is parametrized by a *pivot rule*, which sets out criteria for selecting the entering or leaving variable in every iteration. However, many pivot rules have been shown to be exponential in the worst case. It is a major open problem whether there exists a polynomial pivot rule for the simplex method.

The *diameter* of a polyhedron is the diameter of its associated vertex-edge graph, i.e. the maximum length of a shortest path between any two vertices. In order to have a polynomial pivot rule for the simplex method, a necessary condition is that the diameter of a polyhedron is polynomially bounded in the number of facets. This is known as the *polynomial Hirsch conjecture*, a central question in polyhedral combinatorics. The original *Hirsch conjecture*, which stipulated that the diameter of a d-dimensional polytope (bounded polyhedron) with f facets is at most $f - d$, was famously disproven by Santos [129]. The current best upper bound on the diameter is quasi-polynomial [140], first given by Kalai and Kleitman [92].

1.1.2 Fractional Optimization

If f is a ratio of two functions, then (1.1) is called a *fractional program*. The numerator and denominator are usually interpreted as the cost and weight of a solution respectively. Hence, a fractional program is useful when one is interested in a solution with the best efficiency, i.e. a solution with minimum cost-to-weight ratio. An important special case of fractional programming is when both of these functions are linear:

$$\inf_{x\in\mathcal{D}} \frac{c^\top x}{d^\top x}. \tag{1.2}$$

It is usually assumed that $d^\top x > 0$ for all $x \in \mathcal{D}$. The domain $\mathcal{D}$ could be either a convex set or a discrete set $\mathcal{D} \subseteq \{0,1\}^m$. In the latter, (1.2) is called *linear fractional combinatorial optimization*. Classical examples include finding a a minimum cost-to-time ratio cycle and a maximum mean-weight cut in a graph.

One can equivalently reformulate (1.2) as a parametric optimization problem

$$\sup\left\{\delta : \inf_{x\in\mathcal{D}} (c-\delta d)^\top x \geq 0\right\}.$$

Assuming that (1.2) has a finite optimum, it corresponds to the unique root of the function $g(\delta) := \inf_{x\in\mathcal{D}}(c-\delta d)^\top x$, which is concave and decreasing. This perspective is suggestive of solving (1.2) using a standard root-finding algorithm like Newton's method. In this context, Newton's method is also known as *Dinkelbach's method* [48].

1.1.3 Submodular Optimization

If $\mathcal{D} = \{0,1\}^n$ and f is a submodular function, then (1.1) is called *submodular function minimization (SFM)*. For a ground set $[n] := \{1,2,\dots,n\}$, a set function $f : 2^{[n]} \to \mathbb{R}$ is *submodular* if $f(S) + f(T) \geq f(S\cup T) + f(S\cap T)$ for all $S, T \subseteq [n]$. Submodular functions can be equivalently characterized by the *diminishing marginal returns* property, i.e.

$f(S+i) - f(S) \geq f(T+i) - f(T)$ for all $S \subseteq T \subseteq [n]$ and $i \in [n] \setminus T$. This natural property makes them suitable for many applications, especially in economics and game theory where they are used to model agents' preferences. Examples of submodular functions include graph cuts, matroid rank functions and coverage functions.

SFM can be solved in polynomial time in the value oracle model [74]. However, maximizing a submodular function is NP-hard because it captures the maximum cut problem. If the submodular function f is *monotone*, i.e. $f(S) \leq f(T)$ for all $S \subseteq T \subseteq [n]$, then maximizing f subject to a cardinality constraint, i.e. $\mathcal{D} = \{x \in \{0,1\}^n : \mathbb{1}^\top x \leq k\}$, remains NP-hard as it captures the maximum coverage problem. In fact, it cannot be approximated within a factor better than $1 - 1/e$ if we are only allowed polynomially many queries to the value oracle of f [113]. On the positive side, the greedy algorithm gives a matching $(1-1/e)$-approximation [65].

A natural generalization of this problem is to maximize a monotone submodular function f subject to a matroid constraint, i.e.,

$$\max_{S \in \mathcal{J}} f(S) \tag{1.3}$$

where $\mathcal{J}$ is the family of independent sets of a matroid on ground set $[n]$. Note that the cardinality version is a special case of (1.3), as it can be captured with a uniform matroid. Interestingly, (1.3) also admits a $(1-1/e)$-approximation, achieved by the *continuous greedy algorithm* [25].

1.2 Strongly vs Weakly Polynomial

In this thesis, we differentiate between strongly polynomial and weakly polynomial algorithms. Consider a problem whose input is given by N rational numbers in binary encoding. An algorithm for this problem is *strongly polynomial*, if

(i) It only uses comparisons and elementary arithmetic operations $(+, -, \times, /)$;

(ii) The total number of such operations is bounded by $\text{poly}(N)$;

(iii) The algorithm runs in *polynomial space*, that is, the size of the numbers occurring throughout the algorithm remains polynomial in the size of the input.

For a rational number p/q with $p, q \in \mathbb{Z}$, its *size* is $\lceil \log_2(|p|+1) \rceil + \lceil \log_2(|q|+1) \rceil$, the number of bits used in its binary encoding. The size of a vector of rational numbers is the sum of the sizes of its components.

The notion of strongly polynomial is inherently related to the *real model of computation.* In this model, the input is given by real numbers, which can be stored at unit cost per number. There is also an oracle which carries out comparisons and elementary arithmetic operations (with infinite precision) at unit cost per operation. An algorithm is polynomial in this model if it satisfies (i) and (ii). Thus, an algorithm is strongly polynomial if and only if it is polynomial in the Turing model and in the real model. An algorithm which is polynomial (in the Turing model) but not strongly polynomial is said to be *weakly polynomial*.

In linear programming, the ellipsoid method and interior point methods are weakly polynomial. In other words, their running times depend on the magnitude of the numbers given in the input. On the other hand, submodular function minimization is solvable in strongly polynomial time [83, 84, 99, 42, 85]. It is a major open problem whether there exists a strongly polynomial algorithm for LP. This is listed by Fields medalist Stephen Smale as one of the most important mathematical challenges for the 21st century [137]. Special classes of LPs which are known to admit strongly polynomial algorithms include feasibility of two variables per inequality systems [106], minimum-cost flow [144], maximum generalized flow [150, 119], and discounted Markov decision processes [160, 161]. Furthermore, LPs whose constraint matrices have bounded ‘condition numbers’ can also be solved in strongly polynomial time [142, 40].

1.3 Overview of our Results

In this thesis, we contribute to the aforementioned three subareas of optimization (linear, fractional, submodular), and to the notorious problem of solving parity games.

1.3.1 Fractional Optimization: An Accelerated Newton–Dinkelbach Method

In Chapter 2, we present an accelerated, or ‘look-ahead’ version of the Newton–Dinkelbach method. Given a univariate concave function $g : \mathbb{R} \to \mathbb{R} \cup \{-\infty\}$, our goal is to compute the largest root δ^*. For simplicity, let us assume that g is differentiable and has at least one root. The standard Newton–Dinkelbach method proceeds through iterates $\delta^{(1)} > \delta^{(2)} > \cdots > \delta^{(t)}$ such that $g(\delta^{(i)}) \leq 0$, and updates $\delta^{(i+1)} = \delta^{(i)} - g(\delta^{(i)})/g'(\delta^{(i)})$. Our new variant uses a more aggressive ‘look-ahead’ technique. At each iteration, we compute $\delta = \delta^{(i)} - g(\delta^{(i)})/g'(\delta^{(i)})$, and jump ahead to $\delta' = 2\delta - \delta^{(i)}$. In case $g(\delta') \leq 0$ and $g'(\delta') < 0$, we update $\delta^{(i+1)} = \delta'$. Otherwise, we continue with the standard iterate δ.

This modification leads to an improved and at the same time simplified analysis based on the *Bregman divergence* of g

$$D_g(\delta^*, \delta^{(i)}) := g(\delta^{(i)}) + g'(\delta^{(i)})(\delta^* - \delta^{(i)}) - g(\delta^*).$$

In particular, we show that this decreases by a factor of two between two iterations. Using the Bregman divergence as a potential in conjunction with combinatorial arguments, we obtain strongly polynomial algorithms in three applications domains:

(i) For linear fractional combinatorial optimization, we show a convergence bound of $O(m \log m)$ iterations; the previous best bound was $O(m^2 \log m)$ by Wang et al. [156].

(ii) We obtain a strongly polynomial label-correcting algorithm for solving the feasibility of linear systems with two variables per inequality (2VPI). A 2VPI linear system is given by $Ax \le b$, where the matrix A has at most two nonzero entries per row. For a 2VPI system with n variables and m constraints, our algorithm runs in $O(mn)$ iterations. Every iteration takes $O(mn)$ time for general 2VPI systems, and $O(m + n \log n)$ time for the special case of deterministic Markov Decision Processes (DMDPs). This extends and strengthens a previous result by Madani [103] that showed a weakly polynomial bound for a variant of the Newton–Dinkelbach method for solving DMDPs.

(iii) We give a simplified variant of the parametric submodular function minimization result by Goemans et al. [71]. In this problem, we are given a nonnegative submodular function g on a ground set $[n]$, and a vector $a \in \mathbb{R}^n$ satisfying $\max_{i\in[n]} a_i > 0$. The goal is to compute

$$\max\left\{\delta : \min_{S\subseteq[n]} g(S) - \delta a(S) \ge 0\right\},$$

where $a(S) \coloneqq \sum_{i\in S} a_i$. This problem models the line-search problem inside a submodular polyhedron [147, 112, 71]. Goemans et al. [71] showed an $O(n^2)$ bound on the number of iterations taken by the Newton–Dinkelbach method. We prove that our method also terminates in $O(n^2)$ iterations with a simpler analysis.

1.3.2 Linear Optimization: Circuit Diameter Bounds

Motivated by the polynomial Hirsch conjecture, in Chapter 3, we study the circuit diameter of polyhedra, introduced by Borgwardt, Finhold, and Hemmecke [18] as a relaxation of the (combinatorial) diameter. Consider a polyhedron in standard equality form

$$P = \{x \in \mathbb{R}^n : Ax = b, x \ge \mathbb{0}\},$$

where $A \in \mathbb{R}^{m\times n}$, $\mathrm{rk}(A) = m$ and $b \in \mathbb{R}^m$. For the subspace $\ker(A)$, an *elementary vector* is a support-minimal nonzero vector in $\ker(A)$. A *circuit* is the support of some elementary vector. These are precisely the circuits of the linear matroid associated with A. All edge-directions of P are elementary vectors, and the set of elementary vectors $\mathcal{E}(A)$ equals the set of all possible edge-directions of P when varying $b \in \mathbb{R}^m$.

A *circuit walk* is a sequence of points $x^{(1)}, x^{(2)}, \dots, x^{(k+1)}$ in P such that for each $i \in [k]$, $x^{(i+1)} = x^{(i)} + g^{(i)}$ for some elementary vector $g^{(i)} \in \ker(A)$, and further, $x^{(i)} + (1+\varepsilon)g^{(i)} \notin P$ for all $\varepsilon > 0$, i.e., each consecutive circuit step is *maximal*. The *circuit diameter* of P is the maximum length (number of steps) of a shortest circuit walk between any two vertices of P. The circuit analogue of Hirsch conjecture [18], asserts that the circuit diameter of a

d-dimensional polyhedron with f facets is at most $f-d$. This is still open, and may even be true for unbounded polyhedra [19]. For P in standard equality form, we have $d=n-m$ and $f \leq n$. Hence, the conjectured bound is m.

We prove that the circuit diameter of P is bounded by $O(m \min\{m, n-m\} \log(m+\kappa_A))$, where

$$\kappa_A := \max_{g \in \mathcal{E}(A)} \left\{ \frac{|g_i|}{|g_j|} : i, j \in \operatorname{supp}(g) \right\}$$

is the *circuit imbalance measure* of the constraint matrix. This yields a strongly polynomial circuit diameter bound if e.g., all entries of A have polynomially bounded encoding length in n. We also consider polyhedra in capacitated form

$$P_u = \{x \in \mathbb{R}^n : Ax = b, \mathbb{0} \leq x \leq u\}$$

and show a circuit diameter bound of $O(m \min\{m, n-m\} \log(m+\kappa_A) + n \log n)$.

The proof is via a simple 'shoot towards the optimum' scheme. It exploits the well-known property that every vector x in a subspace $W \subseteq \mathbb{R}^n$ admits a *conformal circuit decomposition*. Namely, x can be decomposed into at most n elementary vectors $\sum_{i=1}^k h^{(i)}$ such that $h_j^{(i)} x_j \geq 0$ and $|h_j^{(i)}| \leq |x_j|$ for all $i \in [k]$ and $j \in [n]$. The scheme is described as follows. Given a target vertex $y \in P$ with corresponding basis $B \subseteq [n]$, i.e., $A_B y_B = b$ and $y_{[n] \setminus B} = \mathbb{0}$, let $x \in P$ our current point. Consider a conformal circuit decomposition $\sum_{i=1}^k h^{(i)}$ of $y - x \in \ker(A)$. Then, we pick an elementary vector in $\arg\max_{i \in [k]} \|h_{[n] \setminus B}^{(i)}\|_1$, and take a maximal step in that direction. This procedure is repeated until $x = y$.

The above scheme is not algorithmic, as it requires knowing the target vertex y. To this end, we complement our circuit diameter bounds with circuit augmentation algorithms. These are algorithms that converge to an optimal solution via a circuit walk. Many network optimization algorithms can be seen as special circuit augmentation algorithms, such as the Edmonds–Karp–Dinic [55] algorithm for maximum flow, and various cycle cancelling algorithms [72, 154] for minimum cost flow. Our algorithm is based on the minimum-ratio circuit canceling rule. Even though the standard minimum-ratio circuit cancelling algorithm is not finite in general [104], our variant can solve an LP in $O(mn^2 \log(n+\kappa_A))$ augmentation steps. This is achieved by occasionally canceling circuits in the support of our current point.

1.3.3 Submodular Optimization: Correlation Gap Bounds for Matroids

An important special case of (1.3) is when f is given as a sum of weighted matroid rank functions

$$f = \sum_{i=1}^m f_i. \tag{1.4}$$

This model was considered by Calinescu et al. [26], who gave a $(1-1/e)$-approximation algorithm. The crucial quantity which governs the approximation ratio of this algorithm is the correlation gap of each weighted matroid rank function f_i.

For a set function $g : \{0,1\}^n \to \mathbb{R}$, its *correlation gap* is the smallest ratio between two natural extensions of g to the unit cube $[0,1]^n$

$$\min_{x\in[0,1]^n} \frac{G(x)}{\hat{g}(x)}.$$

The function $G : [0,1]^n \to \mathbb{R}$ is called the *multilinear extension* of g. For $x \in [0,1]^n$, $G(x)$ is equal to the expected value of g when each element $i \in [n]$ is sampled independently with probability x_i

$$G(x) := \sum_{S\subseteq[n]} g(S) \prod_{i\in S} x_i \prod_{i\notin S} (1-x_i).$$

In other words, $G(x)$ is the expectation under a product distribution. On the other hand, the function $\hat{g} : [0,1]^n \to \mathbb{R}$ is called the *concave extension* of g. For $x \in [0,1]^n$, $\hat{g}(x)$ corresponds to the probability distribution with marginals x which maximizes expectation, i.e.,

$$\hat{g}(x) := \max\left\{ \sum_{S\subseteq[n]} \lambda_S g(S) : \sum_{S\subseteq[n]:i\in S} \lambda_S = x_i \ \forall i \in [n], \sum_{S\subseteq[n]} \lambda_S = 1, \lambda \geq \mathbb{0} \right\}.$$

Note that for every set $S \subseteq [n]$, we have $g(S) = G(\chi_S) = \hat{g}(\chi_S)$, where χ_S denotes the 0-1 indicator vector of S.

Calinescu et al. [26] proved that the correlation gap of a monotone submodular function is at least $1-1/e$, and this is tight for the rank function of a uniform rank-1 matroid. Yan [159] and Barman et al. [9] proved that the correlation gap of the rank function of a uniform rank-ℓ matroid is $1-e^{-\ell}\ell^\ell/\ell! \geq 1-1/e$. This bound yields a $(1-e^{-\ell}\ell^\ell/\ell!)$-approximation algorithm for (1.3) in the form (1.4), if each f_i is the rank function of a uniform rank-ℓ matroid [9]. Furthermore, such an improved bound has direct applications in mechanism design [159] and contention resolution schemes [30].

Motivated by these results, and the significance of correlation gap in algorithmic applications, we initiate a fine-grained study of the correlation gap of matroid rank functions in Chapter 4. In particular, we are interested in identifying parameters of a matroid which affect its correlation gap. A natural candidate is the rank of the matroid. However, as pointed out by Yan [159], there exist matroids with arbitrarily high rank whose correlation gap is still $1-1/e$, e.g., a partition matroid with rank-1 parts. Another potential candidate is the *girth* of the matroid – the minimum size of a dependent set. However, we show that for any $\gamma \in \mathbb{N}$, there exist matroids with girth γ whose correlation gap is arbitrarily close to $1-1/e$.

Our first result is an improved lower bound on the correlation gap, as parametrized by the rank and the girth of the matroid. Our bound is an increasing function of the girth when the rank is fixed, and a decreasing function of the rank when the girth is fixed. We also provide a complementing albeit non-tight upper bound which behaves similarly with respect to these two parameters. Note that a matroid is uniform if and only if the rank is one less than the girth. In this case, our bound coincides with $1 - e^{\ell}\ell^{\ell}/\ell!$ [159, 9].

We briefly describe the overall proof strategy. Let r be the rank function of a matroid with girth γ. The analysis starts by locating a point on which the correlation gap is realized, i.e., $x^* \in \arg\min_{x\in[0,1]^n} R(x)/\hat{r}(x)$. We show that such a point can always be found in the independent set polytope of the matroid. This is useful because the concave extension evaluates to $\hat{r}(x) = \mathbb{1}^\top x$ inside this polytope, in particular, $\hat{r}(x^*) = \mathbb{1}^\top x^*$.

To analyze the multilinear extension $R(x^*)$, we decompose the rank function into $r = g + h$, where g is the rank function of a uniform matroid of rank-$(\gamma - 1)$. The residual function h is nonnegative and monotone, but no longer submodular. Since $R(x^*) = G(x^*) + H(x^*)$ by linearity of expectation, we can lower bound $G(x^*)$ and $H(x^*)$ separately. The former can be done by simply mimicking the analysis of Yan [159] or Barman et al. [9] for uniform matroids. In contrast, the latter is based on a non-trivial extension of the probabilistic analysis by Calinescu et al. [26].

As our second result, we show that for any matroid, the smallest correlation gap of its weighted rank function is achieved under uniform weights. Consequently, our correlation gap bound for matroid rank functions applies to weighted matroid rank functions as well. The proof crucially relies on the greedy maximization property of matroids.

1.3.4 Parity Games: Strategy Iteration with Universal Trees

A *parity game* is an infinite-duration game played between two players, Even and Odd, on a sinkless directed graph $G = (V, E)$. Let $n = |V|$ and $m = |E|$. The graph G is equipped with a *priority* function $\pi : V \to [d]$ for some integer $d \leq n$. The node set V is partitioned into V_0 and V_1, such that Even owns V_0 and Odd owns V_1. The game starts when a token is placed on a node $v \in V$. In every turn, the owner of the current node gets to move the token along an outgoing arc to the next node. Since the duration of the game is infinite and G is sinkless, a play gives rise to an infinite walk $P = v_1 v_2 v_3 \cdots$ in G, where $v_i \in V$ for all $i \in \mathbb{N}$. Let $\pi(P)$ denote the maximum priority in P that occurs infinitely often, i.e.

$$\pi(P) := \lim_{k\to\infty} \left(\max_{i\geq k} \pi(v_i) \right).$$

Player Even's objective is to make $\pi(P)$ even, while Player Odd's objective is to make $\pi(P)$ odd.

Parity games are *determined* [59]. That is, for every instance and every starting node $v \in V$, either Even has a strategy which guarantees that $\pi(P)$ is even, or Odd has a strategy which guarantees that $\pi(P)$ is odd. We say that Even *wins* in the former, and Odd *wins* in the latter. Furthermore, they can guarantee this outcome using *positional strategies*, which are strategies that only depend on the current position of the token. A *positional strategy* for Even (resp. Odd) is a choice of an outgoing arc from every node owned by Even (resp. Odd). The main algorithmic problem is to decide the winner given a starting node. It lies in NP ∩ co-NP [60], with no known polynomial algorithm to date.

Besides having an intriguing complexity status, parity games play a fundamental role in logic and automata theory [60, 97]. For example, the problem of solving parity games is equivalent to the model-checking problem for modal μ-calculus [60, 21]. For these reasons, it has been a subject of intense study over the past three decades. The current fastest algorithms run in quasi-polynomial time, first given in the breakthrough result of Calude et al. [27].

Since [27], parity games have witnessed several new quasi-polynomial algorithms [61, 88, 100, 122, 11]. Although they appear distinct at first sight, the central combinatorial object underlying these approaches is a *universal tree*, as identified by Czerwiński et al. [35]. Roughly speaking, a universal tree is an ordered tree into which every ordered tree of a certain size can be embedded. It is known that universal trees of quasi-polynomial size exist [88, 43]. On the negative side, Czerwiński et al. [35] proved a quasi-polynomial lower bound on the size of a universal tree. This lower bound highlights a barrier that must be overcome by all existing approaches to attain polynomial running time, because there are worst case instances which force these algorithms to explore a large portion of the tree.

As an attempt to overcome this barrier, in Chapter 5, we propose a strategy iteration framework which can be applied on any universal tree. Strategy iteration is a well-known method for solving games on graphs. It proceeds via a sequence of positional strategies for one of the players, until the best strategy is reached. Like the simplex method, it is also parameterized by a pivot rule, which determines how the strategy is modified in every iteration.

Strategy iteration algorithms for parity games have been developed in the past [124, 152, 13, 130]. They usually perform well in practice, but tedious constructions of their (sub)exponential complexity is known [67]. Our framework yields the first quasi-polynomial strategy iteration algorithm when applied on known constructions of quasi-polynomial universal trees [88, 43]. It is at least as fast as its value iteration counterparts, while allowing one to take bigger leaps in the universal tree. Identifying a pivot rule that may provide strictly improved (and possibly even polynomial) running time is left for future research.

Our main technical contribution is an efficient method for computing the least fixed point of 1-player games. This is achieved via a careful adaptation of shortest path algorithms to the setting of ordered trees. By plugging in the universal tree of Jurdziński–Lazić [88], or the Strahler universal tree of Daviaud et al. [43], we obtain instantiations of the general framework that take time $O(mn^2 \log n \log d)$ and $O(mn^2 \log^3 n \log d)$ respectively per iteration.

1.4 Linear Programming and Parity Games

In this section, we illustrate the connection between linear programming and parity games. Following Schewe [131], we give a reduction from parity games to LP, which is polynomial in the real model but not polynomial in the Turing model. We remark that this section is not required to understand the results in the thesis, and can be skipped if the reader wishes to.

The reduction goes through another infinite-duration two-player game on graphs, called mean-payoff games (MPG). A *mean-payoff game* is played between two players, Min and Max, on a sinkless directed graph $G = (V, E)$ with arc weights $w : E \to \mathbb{Z}$. The node set V is partitioned into $V_{\min}$ and $V_{\max}$, such that Min owns $V_{\min}$ and Max owns $V_{\max}$. The game starts when a token is placed on a node $v \in V$. In every turn, the owner of the current node gets to move the token along an outgoing arc to the next node. Since the duration of the game is infinite and G is sinkless, a play gives rise to an infinite walk $P = e_1 e_2 e_3 \cdots$ where $e_i \in E$ for all $i \in \mathbb{N}$. Player Min's objective is to minimize the payoff

$$\mathrm{val}_{\min}(P) := \limsup_{k\to\infty} \sum_{i=1}^{k} \frac{w(e_i)}{k},$$

while Player Max's objective is to maximize the payoff

$$\mathrm{val}_{\max}(P) := \liminf_{k\to\infty} \sum_{i=1}^{k} \frac{w(e_i)}{k}.$$

A fundamental result [56] of mean payoff games states that for every instance and every starting node $v \in V$, there exists a unique number $\mathrm{val}(v) \in \mathbb{Q}$ such that Min has a strategy which guarantees that $\mathrm{val}_{\min}(P) \leq \mathrm{val}(v)$, and Max has a strategy which guarantees that $\mathrm{val}_{\max}(P) \geq \mathrm{val}(v)$. Moreover, they can guarantee this outcome using positional strategies. For a starting node $v \in V$, we say that Max *wins* the game if $\mathrm{val}(v) \geq 0$, and Min *wins* the game if $\mathrm{val}(v) < 0$. The main algorithmic problem is to compute the nodes from which Max wins, or equivalently, to decide the winner given a starting node. Like parity games, it also lies in NP $\cap$ co-NP [94], with no known polynomial algorithm to date.

The problem of determining the winning nodes of Max can be formulated as the following system of inequalities in $(\mathbb{R} \cup \{-\infty\})^n$:

$$\begin{aligned} y_u &\leq w(uv) + y_v && \forall uv \in E, u \in V_{\min} \\ y_u &\leq \max\{w(uv) + y_v : uv \in \delta^+(u)\} && \forall u \in V_{\max}, \end{aligned} \tag{1.5}$$

where $\delta^+(u)$ denotes the set of outgoing arcs at node u. This system has a trivial feasible solution given by $y_v = -\infty$ for all $v \in V$. More interestingly, if y is a feasible solution with maximal finite support, then $y_v > \infty$ if and only if Max wins from v. Observe that if y is feasible, then $y + \alpha\mathbb{1}$ is also feasible for any $\alpha \in \mathbb{R}$. Consequently, if (1.5) has a feasible solution with at least one finite component, then it is unbounded. The system is actually a system of linear inequalities in the tropical (max-plus) semiring $(\mathbb{R} \cup \{-\infty\}, \oplus, \odot)$, with the operations $a \oplus b = \max\{a, b\}$ and $a \odot b = a + b$. In fact, tropical polyhedra and mean-payoff games are equivalent [5].

Reducing Parity Game to MPG A parity game can be reduced to a mean-payoff game as follows [124]. For each node $v \in V$, assign weight $(-n)^{\pi(v)}$ to every outgoing arc $e \in \delta^+(v)$. Then, player Even takes the role of Max, while player Odd takes the role of Min. Note that this reduction is polynomial both in the Turing model and the real model.

Reducing MPG to LP For a sufficiently large $t > 1$, the system (1.5) can be approximated by the following LP

$$\begin{aligned} x_u &\leq t^{w(uv)} \cdot x_v && \forall uv \in E, u \in V_{\min} \\ x_u &\leq \sum_{uv \in \delta^+(u)} t^{w(uv)} \cdot x_v && \forall u \in V_{\max} \\ \mathbb{0} &\leq x \leq \mathbb{1}, \end{aligned} \tag{1.6}$$

using the variable transformation $x_v = t^{y_v}$ for all $v \in V$. Observe that the first set of inequalities in (1.5) and (1.6) are equivalent; the inaccuracy of this transformation stems from the second set of inequalities. This LP is feasible because $\mathbb{0}$ is a feasible solution. It was shown in [131] that if $t > \max\{|\delta^+(u)|^n : u \in V_{\max}\}$, then solving (1.6) with the objective function $\max \mathbb{1}^\top x$ allows us to decide the winner of an MPG. In particular, if x^* is an optimal solution, then $x^*_v > 0$ if and only if Max wins from v.

Due to the magnitude of the numbers in (1.6), this reduction is not polynomial in the Turing model. In the real model, recall that exponentiation is not considered an elementary arithmetic operation. One could still exponentiate $t^{w(uv)}$ with $\lceil \log(|w(uv)|) \rceil$ multiplications via repeated squaring, but this is not a polynomial quantity in the real model. Nevertheless, for MPGs that arise from parity games, the arc weights are of the form $(-n)^p$ for $p \leq d \leq n$. So, the reduction is polynomial in the real model for this class of MPGs.

Thus, if we have a strongly polynomial algorithm for LP, then the sequence of reductions above yields a polynomial algorithm for parity games in the real model. It is also conceivable that a polynomial algorithm (in the Turing model) for parity games can be inferred from the application of the strongly polynomial LP algorithm to (1.6).

Chapter 2

Fractional Optimization: An Accelerated Newton–Dinkelbach Method

2.1 Introduction

Linear fractional optimization problems are well-studied in combinatorial optimization. Given a closed domain $\mathcal{D} \subseteq \mathbb{R}^m$ and $c, d \in \mathbb{R}^m$ such that $d^\top x > 0$ for all $x \in \mathcal{D}$, the problem is

$$\inf_{x \in \mathcal{D}} \frac{c^\top x}{d^\top x} \,. \tag{2.1}$$

The domain $\mathcal{D}$ could be either a convex set or a discrete set $\mathcal{D} \subseteq \{0,1\}^m$. Classical examples include finding a minimum cost-to-time ratio cycle and a maximum mean-weight cut in a graph. One can equivalently formulate (2.1) as a parametric search problem. Let

$$f(\delta) = \inf_{x \in \mathcal{D}} (c - \delta d)^\top x \,, \tag{2.2}$$

be a concave and decreasing function. Assuming (2.1) has a finite optimum δ, it corresponds to the unique root $f(\delta) = 0$.

A natural question is to investigate how the computational complexity of solving the minimum ratio problem (2.1) may depend on the complexity of the corresponding linear optimization problem $\min c^\top x$ s.t. $x \in \mathcal{D}$. Using the reformulation (2.2), one can reduce the fractional problem to the linear problem via binary search; however, the number of iterations needed to find an exact solution may depend on the bit complexity of the input. A particularly interesting question is: assuming there exists a strongly polynomial algorithm for linear optimization over a domain $\mathcal{D}$, can we find a strongly polynomial algorithm for linear fractional optimization over the same domain?

A seminal paper by Megiddo [105] introduced the *parametric search* technique to solve linear fractional combinatorial optimization problems. He showed that if the linear optimization algorithm only uses $p(m)$ comparisons and $q(m)$ additions, then there exists an $O(p(m)(p(m)+q(m))$ algorithm for the linear fractional optimization problem. This in particular yielded the first strongly polynomial algorithm for the minimum cost-to-time ratio cycle problem. On a very high level, parametric search works by simulating the linear optimization algorithm for the parametric problem (2.2), with the parameter $\delta \in \mathbb{R}$ being indeterminate.

A natural alternative approach is to solve (2.2) using a standard root finding algorithm. Radzik [126] showed that for a discrete domain $\mathcal{D} \subseteq \{0,1\}^m$, the *discrete Newton* method—in this context, also known as *Dinkelbach's method* [48]—terminates in a strongly polynomial number of iterations. In contrast to parametric search, there are no restrictions on the possible operations in the linear optimization algorithm. In certain settings, such as the maximum ratio cut problem, the discrete Newton method outperforms parametric search; we refer to the comprehensive survey by Radzik [125] for details and comparison of the two methods.

2.1.1 Our Contributions

We introduce a new, *accelerated variant of Newton's method for univariate functions*. Let $f : \mathbb{R} \to \mathbb{R} \cup \{-\infty\}$ be a concave function. Under some mild assumptions on f, our goal is to either find the largest root, or show that no root exists. Let δ^* denote the largest root, or in case $f < 0$, let δ^* denote the largest maximizer of f. For simplicity, we now describe the method for differentiable functions. This will not hold in general: functions of the form (2.2) will be piecewise linear if $\mathcal{D}$ is finite or polyhedral. The algorithm description in Section 2.3 uses a form with supergradients (that can be choosen arbitrarily between the left and right derivatives).

The standard Newton method, also used by Radzik, proceeds through iterates $\delta^{(1)} > \delta^{(2)} > \ldots > \delta^{(t)}$ such that $f(\delta^{(i)}) \leq 0$, and updates $\delta^{(i+1)} = \delta^{(i)} - f(\delta^{(i)})/f'(\delta^{(i)})$.

Our new variant uses a more aggressive *'look-ahead'* technique. At each iteration, we compute $\delta = \delta^{(i)} - f(\delta^{(i)})/f'(\delta^{(i)})$, and jump ahead to $\delta' = 2\delta - \delta^{(i)}$. In case $f(\delta') \leq 0$ and $f'(\delta') < 0$, we update $\delta^{(i+1)} = \delta'$; otherwise, we continue with the standard iterate δ.

This modification leads to an improved and at the same time simplified analysis based on the *Bregman divergence* $D_f(\delta^*, \delta^{(i)}) = f(\delta^{(i)}) + f'(\delta^{(i)})(\delta^* - \delta^{(i)}) - f(\delta^*)$. We show that this decreases by a factor of two between any two iterations.

A salient feature of the algorithm is that it handles both feasible and infeasible outcomes in a unified framework. In the context of linear fractional optimization, this means that the assumption $d^\top x > 0$ for all $x \in \mathcal{D}$ in (2.1) can be waived. Instead, $d^\top x > 0$ is now added as a feasibility constraint to (2.1). This generalization is important when we use the algorithm to solve two variables per inequality systems.

This general result leads to improvements and simplifications of a number of algorithms using the discrete Newton method.

- For *linear fractional combinatorial optimization*, namely the setting (2.1) with $\mathcal{D} \subseteq \{0,1\}^m$, we obtain an $O(m \log m)$ bound on the number of iterations, a factor m improvement over the previous best bound $O(m^2 \log m)$ by Wang et al. [156] from 2006. We remark that Radzik's first analysis [126] yielded a bound of $O(m^4 \log^2 m)$ iterations, improved to $O(m^2 \log^2 m)$ in [125].
- Goemans et al. [71] used the discrete Newton method to obtain a strongly polynomial algorithm for parametric submodular function minimization. We give a simple new variant of this result with the same asymptotic running time, using the accelerated algorithm.
- For *two variable per inequality (2VPI)* systems, we obtain a *strongly polynomial label-correcting algorithm*. This will be discussed in more detail next.

2.1.2 Two Variables per Inequality Systems

A major open question in the theory of linear programming is whether there exists a strongly polynomial algorithm for LP. The notion of a strongly polynomial algorithm was formally introduced by Megiddo [106] in 1983 (using the term *'genuinely polynomial'*), where he gave the first such algorithm for *two variables per inequality (2VPI)* systems. These are feasibility LPs where every inequality contains at most two variables. More formally, let $\mathcal{M}_2(n,m)$ be the set of $n \times m$ matrices with at most two nonzero entries per column. A 2VPI system is of the form $A^\top y \le c$ for $A \in \mathcal{M}_2(n,m)$ and $c \in \mathbb{R}^m$.

If we further require that every inequality has at most one positive and at most one negative coefficient, it is called a *monotone two variables per inequality* (M2VPI) system. A simple and efficient reduction is known from 2VPI systems with n variables and m inequalities to M2VPI systems with $2n$ variables and $\le 2m$ inequalities [54, 79] (sketch in Section A.1).

Connection between 2VPI and parametric optimization An M2VPI system has a natural graphical interpretation: after normalization, we can assume every constraint is of the form $y_u - \gamma_e y_v \le c_e$. Such a constraint naturally maps to an arc $e = (u,v)$ with *gain factor* $\gamma_e > 0$ and cost c_e. Based on Shostak's work [136] that characterized feasibility in terms of this graph, Aspvall and Shiloach [7] gave the first weakly polynomial algorithm for M2VPI systems.

We say that a directed cycle C is *flow absorbing* if $\prod_{e\in C} \gamma_e < 1$ and *flow generating* if $\prod_{e\in C} \gamma_e > 1$. Every flow absorbing cycle C implies an upper bound for every variable y_u

incident to C; similarly, flow generating cycles imply lower bounds. The crux of Aspvall and Shiloach's algorithm is to find the tightest upper and lower bounds for each variable y_u.

Finding these bounds corresponds to solving fractional optimization problems of the form (2.1), where $\mathcal{D} \subseteq \mathbb{R}^m$ describes 'generalized flows' around cycles. The paper [7] introduced the *Grapevine* algorithm—a natural modification of the Bellman-Ford algorithm—to decide whether the optimum ratio is smaller or larger than a fixed value δ. The optimum value can be found using binary search on the parameter.

Megiddo's strongly polynomial algorithm [106] replaced the binary search framework in Aspvall and Shiloach's algorithm by extending the parametric search technique in [105]. Subsequently, Cohen and Megiddo [32] devised faster strongly polynomial algorithms for the problem. The current fastest deterministic strongly polynomial algorithm is given by Hochbaum and Naor [80], an efficient Fourier–Motzkin elimination with running time of $O(mn^2 \log m)$. Recently, Karczmarz [93] gave a randomized trade-off algorithm which runs in $\widetilde{O}(nmh + (n/h)^3)$ time and uses $\widetilde{O}(n^2/h + m)$ space for any parameter $h \in [1, n]$.

2VPI via Newton's method Since Newton's method proved to be an efficient and viable alternative to parametric search, a natural question is to see whether it can solve the parametric problems occuring in 2VPI systems. Radzik's fractional combinatorial optimization results [126, 125] are not directly applicable, since the domain $\mathcal{D}$ in this setting is a polyhedron and not a discrete set.[1] Madani [103] used a variant of the Newton–Dinkelbach method as a tool to analyze the convergence of policy iteration on *deterministic Markov Decision Processes (DMDPs)*, a special class of M2VPI systems (discussed later in more detail). He obtained a weakly polynomial convergence bound; it remained open whether such an algorithm could be strongly polynomial.

Our 2VPI algorithm We introduce a new type of strongly polynomial 2VPI algorithm by combining the accelerated Newton–Dinkelbach method with a *'variable fixing'* analysis. Variable fixing was first introduced in the seminal work of Tardos [144] on minimum-cost flows, and has been a central idea of strongly polynomial algorithms, see in particular [73, 127] for cycle cancelling minimum-cost flow algorithms, and [119, 150] for maximum generalized flows, a dual to the 2VPI problem.

We show that for every iterate $\delta^{(i)}$, there is a constraint that has been 'actively used' at $\delta^{(i)}$ but will not be used ever again after a strongly polynomial number of iterations. The analysis combines the decay in *Bregman divergence* shown in the general accelerated Newton–Dinkelbach analysis with a combinatorial *'subpath monotonicity'* property.

Our overall algorithm can be seen as an extension of Madani's DMDP algorithm. In particular, we adapt his 'unfreezing' idea: the variables y_u are admitted to the system

[1]The problem could be alternatively formulated with $\mathcal{D} \subseteq \{0,1\}^m$ but with nonlinear functions instead of $c^\top x$ and $d^\top x$.

one-by-one, and the accelerated Newton–Dinkelbach method is used to find the best 'cycle bound' attainable at the newly admitted y_u in the graph induced by the current variable set. This returns a feasible solution or reports infeasibility within $O(m)$ iterations. As every iteration takes $O(mn)$ time, our overall algorithm terminates in $O(m^2n^2)$ time. For the special setting of deterministic MDPs, the runtime per iteration improves to $O(m + n \log n)$, giving a total runtime of $O(mn(m + n \log n))$.

Even though our running time bound is worse than the state-of-the-art 2VPI algorithm [80], it is of a very different nature from all previous 2VPI algorithms. In fact, our algorithm is a *label correcting algorithm*, naturally fitting to the family of algorithms used in other combinatorial optimization problems with constraint matrices from $\mathcal{M}_2(n, m)$ such as maximum flow, shortest paths, minimum-cost flow, and generalized flow problems. We next elaborate on this connection.

Label-correcting algorithms An important special case of M2VPI systems corresponds to the shortest paths problem: given a directed graph $G = (V, E)$ with target node $t \in V$ and arc costs $c \in \mathbb{R}^E$, we associate constraints $y_u - y_v \le c_e$ for every arc $e = (u, v) \in E$ and $y_t = 0$. If the system is feasible and bounded, the pointwise maximal solution corresponds to the shortest path labels to t; an infeasible system contains a negative cost cycle. A generic label-correcting algorithm maintains distance labels y that are upper bounds on the shortest path distances to t. The labels are decreased according to violated constraints. Namely, if $y_u - y_v > c_e$, then decreasing y_u to $c_e + y_v$ gives a smaller valid distance label at u. We terminate with the shortest path labels once all constraints are satisfied. The Bellman–Ford algorithm for the shortest paths problem is a particular implementation of the generic label-correcting algorithm; we refer the reader to [3, Chapter 5] for more details.

It is a natural question if label-correcting algorithms can be extended to general M2VPI systems, where constraints are of the form $y_u - \gamma_e y_v \le c_e$ for a 'gain/loss factor' $\gamma_e > 0$ associated with each arc. A fundamental property of M2VPI systems is that, whenever bounded, a unique pointwise maximal solution exists, i.e. a feasible solution y^* such that $y \le y^*$ for every feasible solution y. A label-correcting algorithm for such a setting can be naturally defined as follows. Let us assume that the problem is bounded. The algorithm should proceed via a decreasing sequence $y^{(0)} \ge y^{(1)} \ge \ldots \ge y^{(t)}$ of labels that are all valid upper bounds on any feasible solution y to the system. The algorithm either terminates with the unique pointwise maximal solution $y^{(t)} = y^*$, or finds an infeasibility certificate.

The basic label-correcting operation is the 'arc update', decreasing y_u to $\min\{y_u, c_e + \gamma_e y_v\}$ for some arc $e = (u, v) \in E$. Such updates suffice in the shortest path setting. However, in the general setting arc operations only may not lead to finite termination. Consider a system with only two variables, y_u and y_v, and two constraints, $y_u - y_v \le 0$, and $y_v - \frac{1}{2} y_u \le -1$. The alternating sequence of arc updates converges to $(y_u^*, y_v^*) = (-2, -2)$, but does not finitely terminate. In this example, we can 'detect' the cycle formed by the two arcs, that implies the bound $y_u - \frac{1}{2}y_u \leq -1$.

Shostak's [136] result demonstrates that arc updates, together with such 'cycle updates' should be sufficient for finite termination. Our M2VPI algorithm amounts to the first strongly polynomial label-correcting algorithm for general M2VPI systems, using arc updates and cycle updates.

Deterministic Markov decision processes A well-studied special case of M2VPI systems in which $\gamma \leq \mathbb{1}$ is known as *deterministic Markov decision process* (DMDP). A *policy* corresponds to selecting an outgoing arc from every node, and the objective is to find a policy that minimizes the total discounted cost over an infinite time horizon. The pointwise maximal solution of this system corresponds to the optimal values of a policy.

The standard policy iteration, value iteration, and simplex algorithms can be all interpreted as variants of the label-correcting framework.[2] Value iteration can be seen as a generalization of the Bellman–Ford algorithm to the DMDP setting. As our previous example shows, value iteration may not be finite. One could still consider as the termination criterion the point where value iteration 'reveals' the optimal policy, i.e. updates are only performed using constraints that are tight in the optimal solution. If each discount factor γ_e is at most γ' for some $\gamma' > 0$, then it is well-known that value iteration converges at the rate $1/(1-\gamma')$. This is in fact true more generally, for nondeterministic MDPs [102]. However, if the discount factors can be arbitrarily close to 1, then Feinberg and Huang [63] showed that value iteration cannot reveal the optimal policy in strongly polynomial time even for DMDPs. Post and Ye [123] proved that simplex with the most negative reduced cost pivoting rule is strongly polynomial for DMDPs; this was later improved by Hansen et al. [77]. These papers heavily rely on the assumption $\gamma \leq \mathbb{1}$, and does not seem to extend to general M2VPI systems.

Madani's previously mentioned work [103] used a variant of the Newton–Dinkelbach method as a tool to analyze the convergence of policy iteration on deterministic MDPs, and derived a weakly polynomial runtime bound.

Chapter organization We start by giving preliminaries and introducing notation in Section 2.2. In Section 2.3, we present an accelerated Newton's method for univariate concave functions, and apply it to linear fractional combinatorial optimization and linear fractional programming. Section 2.4 contains our main application of the method to the 2VPI problem. Our results on parametric submodular function minimization are in Section 2.5. Additional results are given in Section 2.6.

[2]The value sequence may violate monotonicity in certain cases of value iteration.

2.2 Preliminaries

Let $\mathbb{R}_+$ and $\mathbb{R}_{++}$ be the nonnegative and positive reals respectively, and denote $\bar{\mathbb{R}} := \mathbb{R} \cup \{\pm\infty\}$. Given a proper concave function $f : \mathbb{R} \to \bar{\mathbb{R}}$, let $\mathrm{dom}(f) := \{x : -\infty < f(x) < \infty\}$ be the effective domain of f. For a point $x_0 \in \mathrm{dom}(f)$, denote the set of supergradients of f at x_0 as $\partial f(x_0) := \{g : f(x) \leq f(x_0) + g(x - x_0)\ \forall x \in \mathbb{R}\}$. If x_0 is in the interior of $\mathrm{dom}(f)$, then $\partial f(x_0) = [f'_-(x_0), f'_+(x_0)]$, where $f'_-(x_0)$ and $f'_+(x_0)$ are the left and right derivatives. Throughout, we use $\log(x) = \log_2(x)$ to indicate base 2 logarithm. For $x, y \in \mathbb{R}^m$, denote $x \circ y \in \mathbb{R}^m$ as the element-wise product of the two vectors.

2.3 An Accelerated Newton–Dinkelbach Method

Let $f : \mathbb{R} \to \bar{\mathbb{R}}$ be a proper concave function such that $f(\delta) \leq 0$ and $\partial f(\delta) \cap \mathbb{R}_{<0} \neq \emptyset$ for some $\delta \in \mathrm{dom}(f)$. Given a suitable starting point, as well as value and supergradient oracles of f, the Newton–Dinkelbach method either computes the largest root of f or declares that it does not have a root. In this chapter, we make the mild assumption that f has a root or attains its maximum. Consequently, the point

$$\delta^* := \max(\{\delta : f(\delta) = 0\} \cup \arg\max f(\delta))$$

is well-defined. It is the largest root of f if f has a root. Otherwise, it is the largest maximizer of f (see Figure 2.1 for examples). Therefore, the Newton–Dinkelbach method returns δ^* if f has a root, and certifies that $f(\delta^*) < 0$ otherwise.

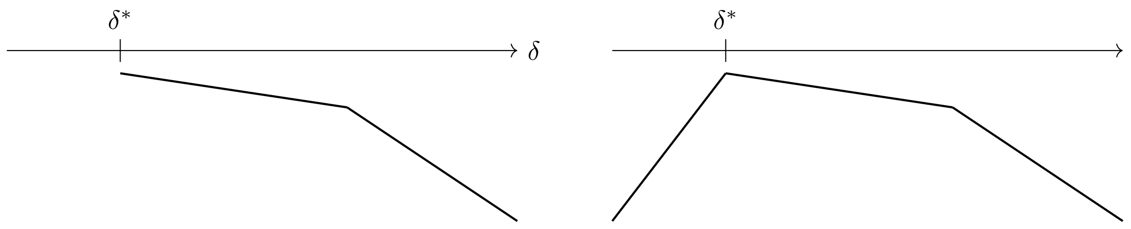

Fig. 2.1 Two examples of f with no root. In the left picture, $f(\delta) = -\infty$ for all $\delta < \delta^*$.

The algorithm takes as input an initial point $\delta^{(1)} \in \mathrm{dom}(f)$ and a supergradient $g^{(1)} \in \partial f(\delta^{(1)})$ such that $f(\delta^{(1)}) \leq 0$ and $g^{(1)} < 0$. At the start of every iteration $i \geq 1$, it maintains a point $\delta^{(i)} \in \mathrm{dom}(f)$ and a supergradient $g^{(i)} \in \partial f(\delta^{(i)})$ where $f(\delta^{(i)}) \leq 0$. If $f(\delta^{(i)}) = 0$, then it returns $\delta^{(i)}$ as the largest root of f. Otherwise, a new point $\delta := \delta^{(i)} - f(\delta^{(i)})/g^{(i)}$ is generated. Now, there are two scenarios in which the algorithm terminates and reports that f does not have a root: (1) $f(\delta) = -\infty$; (2) $f(\delta) < 0$ and $g \geq 0$ where $g \in \partial f(\delta)$ is the supergradient given by the oracle. If both scenarios do not apply, the next point and supergradient is set to $\delta^{(i+1)} := \delta$ and $g^{(i+1)} := g$ respectively. Then, a new iteration begins (see Figure 2.2 for an example).

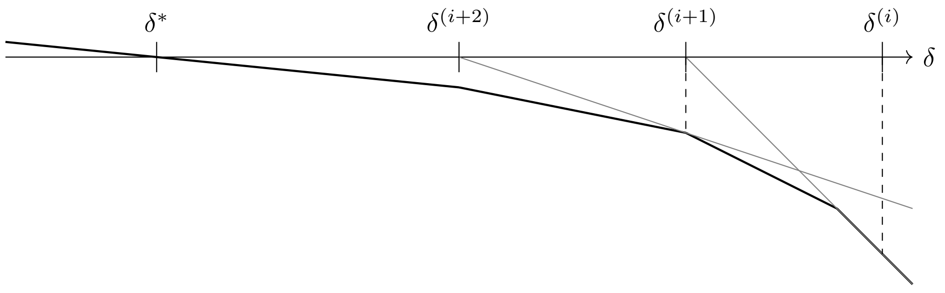

Fig. 2.2 An example run of the Newton–Dinkelbach method when f has a root.

According to this update rule, observe that $g^{(i)} < 0$ except possibly in the final iteration when $f(\delta^{(i)}) = 0$. This proves the correctness of the algorithm. Indeed, $\delta^{(i)} = \delta^*$ if $f(\delta^{(i)}) = 0$. On the other hand, if either of the aforementioned scenarios apply, then combining it with $f(\delta^{(i)}) < 0$ and $g^{(i)} < 0$ certifies that $f(\delta^*) < 0$.

The following lemma shows that $\delta^{(i)}$ is monotonically decreasing while $f(\delta^{(i)})$ is monotonically increasing. Furthermore, $g^{(i)}$ is monotonically increasing except in the final iteration where it may remain unchanged. The lemma also illustrates the useful property that $|f(\delta^{(i)})|$ or $|g^{(i)}|$ decreases geometrically. These are well-known facts and similar statements can be found in e.g. Radzik [125, Lemmas 3.1 & 3.2].

Lemma 2.1. *For every iteration $i \geq 2$, we have $\delta^* \leq \delta^{(i)} < \delta^{(i-1)}$, $f(\delta^*) \geq f(\delta^{(i)}) > f(\delta^{(i-1)})$ and $g^{(i)} \geq g^{(i-1)}$, where the last inequality holds at equality if and only if $g^{(i)} = \inf_{g \in \partial f(\delta^{(i)})} g$, $g^{(i-1)} = \sup_{g \in \partial f(\delta^{(i-1)})} g$ and $f(\delta^{(i)}) = 0$. Moreover,*

$$\frac{f(\delta^{(i)})}{f(\delta^{(i-1)})} + \frac{g^{(i)}}{g^{(i-1)}} \leq 1\,.$$

Proof. Since $f(\delta^{(i)}) \leq 0$ and $g^{(i)} < 0$, by concavity of f we have that $f(\delta) \leq f(\delta^{(i)}) + g^{(i)}(\delta - \delta^{(i)}) < f(\delta^{(i)}) \leq 0$, for all $\delta > \delta^{(i)}$. Given this, we must have $\delta^* \leq \delta^{(i)}$ since either $f(\delta^*) = 0 \geq f(\delta^{(i)})$ or $0 > f(\delta^*) = \max_{z \in \mathbb{R}} f(z) \geq f(\delta^{(i)})$. As $\delta^{(i)} = \delta^{(i-1)} - \frac{f(\delta^{(i-1)})}{g^{(i-1)}} < \delta^{(i-1)}$, since $f(\delta^{(i-1)}), g^{(i-1)} < 0$, we have $f(\delta^{(i-1)}) < f(\delta^{(i)})$. Furthermore, $g^{(i)} \geq g^{(i-1)}$ is immediate from the concavity of f.

To understand when $g^{(i)} = g^{(i-1)}$, we see by concavity that

$$g^{(i)} \geq \inf_{g \in \partial f(\delta^{(i)})} g \geq \frac{f(\delta^{(i-1)}) - f(\delta^{(i)})}{\delta^{(i-1)} - \delta^{(i)}} \geq \sup_{g \in \partial f(\delta^{(i-1)})} g \geq g^{(i-1)}.$$

To have equality throughout, we must therefore have that $g^{(i)}$ and $g^{(i-1)}$ are equal to the respective infimum and supremum. We must also have $f(\delta^{(i)}) = 0$ since

$$\frac{f(\delta^{(i-1)}) - f(\delta^{(i)})}{\delta^{(i-1)} - \delta^{(i)}} = \frac{f(\delta^{(i-1)}) - f(\delta^{(i)})}{\frac{f(\delta^{(i-1)})}{g^{(i-1)}}} = g^{(i-1)}\left(1 - \frac{f(\delta^{(i)})}{f(\delta^{(i-1)})}\right)$$

To have equality throughout, we must therefore have that $g^{(i)}$ and $g^{(i-1)}$ are equal to the respective infimum and supremum and that $f(\delta^{(i)}) = 0$.

Lastly, since f is concave

$$f(\delta^{(i-1)}) \leq f(\delta^{(i)}) + g^{(i)}(\delta^{(i-1)} - \delta^{(i)}) = f(\delta^{(i)}) + g^{(i)} \frac{f(\delta^{(i-1)})}{g^{(i-1)}}.$$

The moreover now follows by dividing both sides by $f(\delta^{(i-1)}) < 0$. □

Our analysis of the Newton–Dinkelbach method utilizes the Bregman divergence associated with f as a potential. Even though the original definition requires f to be differentiable and strictly concave, it can be naturally extended to our setting in the following way.

Definition 2.2. Given a proper concave function $f : \mathbb{R} \to \bar{\mathbb{R}}$, the *Bregman divergence associated with* f is defined as

$$D_f(\delta', \delta) := \begin{cases} f(\delta) + \sup\limits_{g \in \partial f(\delta)} g(\delta' - \delta) - f(\delta') & \text{if } \delta \neq \delta', \\ 0 & \text{otherwise.} \end{cases}$$

for all $\delta, \delta' \in \mathrm{dom}(f)$ such that $\partial f(\delta) \neq \emptyset$.

Since f is concave, the Bregman divergence is nonnegative. See Figure 2.3 for an example. The next lemma shows that $D_f(\delta^*, \delta^{(i)})$ is monotonically decreasing except in the final iteration where it may remain unchanged.

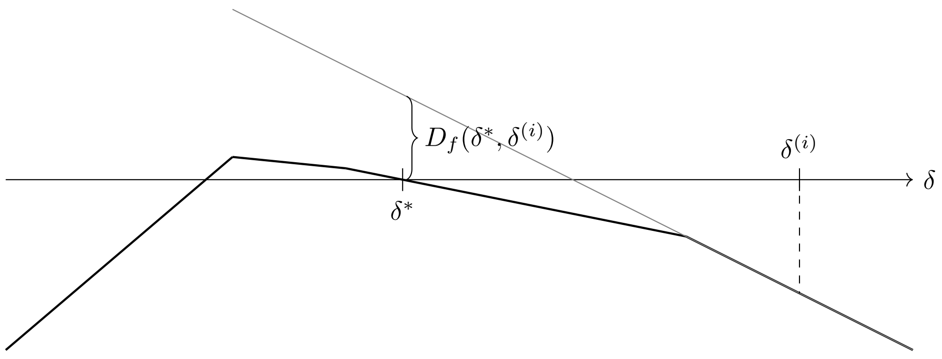

Fig. 2.3 The Bregman divergence $D_f(\delta^*, \delta^{(i)})$ of an example function f.

Lemma 2.3. *For every iteration $i \geq 2$, we have $D_f(\delta^*, \delta^{(i)}) \leq D_f(\delta^*, \delta^{(i-1)})$ which holds at equality if and only if $g^{(i-1)} = \inf_{g \in \partial f(\delta^{(i-1)})} g$ and $f(\delta^{(i)}) = 0$.*

Proof. By Lemma 2.1, we know that $\delta^* \leq \delta^{(i)} < \delta^{(i-1)}$ and $0 \geq f(\delta^{(i)}) > f(\delta^{(i-1)})$. Hence,

$$\begin{aligned}
D_f(\delta^*, \delta^{(i-1)}) &= f(\delta^{(i-1)}) + \sup_{g \in \partial f(\delta^{(i-1)})} g(\delta^* - \delta^{(i-1)}) - f(\delta^*) \\
&\geq f(\delta^{(i-1)}) + g^{(i-1)}(\delta^{(i)} - \delta^{(i-1)}) + g^{(i-1)}(\delta^* - \delta^{(i)}) - f(\delta^*) \\
&\geq f(\delta^{(i)}) + g^{(i-1)}(\delta^* - \delta^{(i)}) - f(\delta^*) \qquad \text{(by concavity of } f) \\
&\geq f(\delta^{(i)}) + \sup_{g \in \partial f(\delta^{(i)})} g(\delta^* - \delta^{(i)}) - f(\delta^*) \\
&= D_f(\delta^*, \delta^{(i)}).
\end{aligned}$$

For the equality condition, note that the first inequality holds at equality if and only if $g^{(i-1)} = \inf_{g \in \partial f(\delta^{(i-1)})} g$. The second inequality holds at equality if and only if $f(\delta^{(i)}) = 0$ because $f(\delta^{(i-1)}) + g^{(i-1)}(\delta^{(i)} - \delta^{(i-1)}) = 0$ from the definition of $\delta^{(i)}$. If $f(\delta^{(i)}) = 0$, then $\delta^{(i)} = \delta^*$, and hence the third inequality holds at equality as well. □

To accelerate this classical method, we perform an aggressive guess $\delta' := 2\delta - \delta^{(i)}$ on the next point at the end of every iteration i. Note that this is twice the usual Newton step, i.e., $\delta' = \delta^{(i)} + 2(\delta - \delta^{(i)}) < \delta$. We call this procedure *look-ahead*, which is implemented on Lines 7–10 of Algorithm 1. Let $g' \in \partial f(\delta')$ be the supergradient returned by the oracle. If $-\infty < f(\delta') < 0$ and $g' < 0$, then the next point and supergradient are set to $\delta^{(i+1)} := \delta'$ and $g^{(i+1)} := g'$ respectively as $\delta' \geq \delta^*$. In this case, we say that look-ahead is *successful* in iteration i. Otherwise, we proceed as usual by taking $\delta^{(i+1)} := \delta$ and $g^{(i+1)} := g$ (see Figure 2.4 for an example). It is easy to verify that Lemmas 2.1 and 2.3 also hold for Algorithm 1.

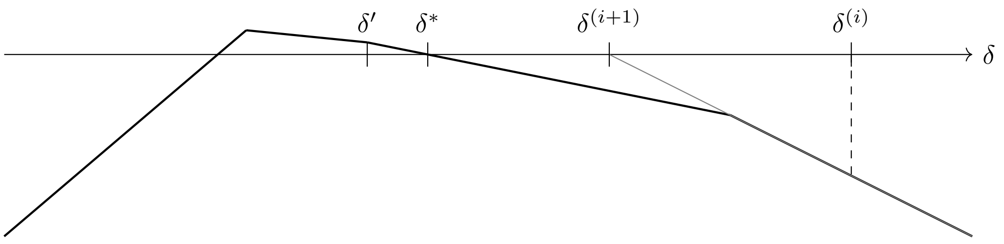

Fig. 2.4 An example run of Algorithm 1 where look-ahead failed in iteration i.

If look-ahead is successful, then we have made significant progress. Otherwise, by our choice of δ', we learn that we are not too far away from δ^*. The next lemma demonstrates the advantage of using the look-ahead Newton–Dinkelbach method. It exploits the proximity to δ^* to produce a geometric decay in the Bregman divergence of $\delta^{(i)}$ and δ^*.

Lemma 2.4. *For every iteration $i > 2$ in Algorithm 1, we have $D_f(\delta^*, \delta^{(i)}) < \frac{1}{2} D_f(\delta^*, \delta^{(i-2)})$.*

Proof. Fix an iteration $i > 2$ of Algorithm 1. Let $g_+^{(i)} = \min_{g \in \partial f(\delta^{(i)})} g$ denote the right derivative of f at $\delta^{(i)}$. From Lemma 2.1, we know that $\delta^* \leq \delta^{(i)} < \delta^{(i-1)} < \delta^{(i-2)}$, $0 \geq$

Algorithm 1: LOOK-AHEADNEWTON

```
input  : Value and supergradient oracles for a proper concave function $f$, an initial
         point $\delta^{(1)} \in \mathrm{dom}(f)$ and supergradient $g^{(1)} \in \partial f(\delta^{(1)})$ where $f(\delta^{(1)}) \leq 0$
         and $g^{(1)} < 0$.
output : The largest root of $f$ if it exists; report NO ROOT otherwise.
1  $i \leftarrow 1$
2  while $f(\delta^{(i)}) < 0$ do
3      $\delta \leftarrow \delta^{(i)} - f(\delta^{(i)})/g^{(i)}$
4      $g \in \partial f(\delta)$                                  ▷ Empty if $f(\delta) = -\infty$
5      if $f(\delta) = -\infty$ or ($f(\delta) < 0$ and $g \geq 0$) then
6          return NO ROOT
7      $\delta' \leftarrow 2\delta - \delta^{(i)}$                             ▷ Look-ahead guess
8      $g' \in \partial f(\delta')$                                ▷ Empty if $f(\delta') = -\infty$
9      if $-\infty < f(\delta') < 0$ and $g' < 0$ then          ▷ Is the guess successful?
10         $\delta \leftarrow \delta'$, $g \leftarrow g'$
11     $\delta^{(i+1)} \leftarrow \delta$, $g^{(i+1)} \leftarrow g$
12     $i \leftarrow i + 1$
13 return $\delta^{(i)}$
```

$f(\delta^*) \geq f(\delta^{(i)}) > f(\delta^{(i-1)}) > f(\delta^{(i-2)})$ and $0 > g_+^{(i)} \geq g^{(i-1)} > g^{(i-2)}$. Since $\delta^* \leq \delta^{(i)}$, we see that $D_f(\delta^*, \delta^{(i)}) = f(\delta^{(i)}) + g_+^{(i)}(\delta^* - \delta^{(i)}) - f(\delta^*)$.

Assume first that the look-ahead step in iteration $i-1$ was successful. We now claim that $0 < -2g_+^{(i)} \leq -g^{(i-1)}$. To see this, we have that

$$\begin{aligned}
f(\delta^{(i-1)}) &\leq f(\delta^{(i)}) + g_+^{(i)}(\delta^{(i-1)} - \delta^{(i)}) && \text{(by concavity of } f) \\
&\leq g_+^{(i)}(\delta^{(i-1)} - \delta^{(i)}) && \text{(since } f(\delta^{(i)}) \leq 0) \\
&= 2g_+^{(i)} \frac{f(\delta^{(i-1)})}{g^{(i-1)}}. && \text{(by definition of the accelerated step)}
\end{aligned}$$

The desired inequality follows by multiplying through by $-\frac{g^{(i-1)}}{f(\delta^{(i-1)})} < 0$.

Using the above inequality, we compare Bregman divergences as follows:

$$\begin{aligned}
D_f(\delta^*, \delta^{(i-1)}) &\geq f(\delta^{(i-1)}) + g^{(i-1)}(\delta^* - \delta^{(i-1)}) - f(\delta^*) && (D_f \text{ is a max over supergradients}) \\
&> g^{(i-1)}(\delta^* - \delta^{(i)}) - f(\delta^*) && (f(\delta^{(i-1)}) + g^{(i-1)}(\delta^{(i)} - \delta^{(i-1)}) > 0) \\
&\geq g^{(i-1)}(\delta^* - \delta^{(i)}) && (-f(\delta^*) \geq 0) \\
&\geq 2g_+^{(i)}(\delta^* - \delta^{(i)}) && (-g^{(i-1)} \geq -2g_+^{(i)} \text{ and } \delta^{(i)} > \delta^*) \\
&\geq 2(f(\delta^{(i)}) + g_+^{(i)}(\delta^* - \delta^{(i)}) - f(\delta^*)) && \text{(since } f(\delta^*) \geq f(\delta^{(i)})) \\
&= 2D_f(\delta^*, \delta^{(i)}). && \text{(by our choice of } g_+^{(i)})
\end{aligned}$$

The desired inequality nows follows from $D_f(\delta^*, \delta^{(i-2)}) > D_f(\delta^*, \delta^{(i-1)})$ by Lemma 2.3.

Now assume that the look-ahead step at iteration $i-1$ was unsuccessful. This implies that $2\delta^{(i)} - \delta^{(i-1)} \le \delta^* \Leftrightarrow 2(\delta^{(i)} - \delta^*) \le \delta^{(i-1)} - \delta^*$, i.e. that the look-ahead step "went past or exactly to" δ^*. We compare Bregman-divergences as follows:

$$\begin{aligned}
D_f(\delta^*, \delta^{(i-2)}) &\ge f(\delta^{(i-2)}) + g^{(i-2)}(\delta^* - \delta^{(i-2)}) - f(\delta^*) && (D_f \text{ is a max over supergradients})\\
&\ge g^{(i-2)}(\delta^* - \delta^{(i-1)}) - f(\delta^*) && (f(\delta^{(i-2)}) + g^{(i-2)}(\delta^{(i-1)} - \delta^{(i-2)}) \ge 0)\\
&\ge g^{(i-2)}(\delta^* - \delta^{(i-1)}) && (-f(\delta^*) \ge 0)\\
&> g_+^{(i)}(\delta^* - \delta^{(i-1)}) && (0 > g_+^{(i)} > g^{(i-2)} \text{ and } \delta^{(i-1)} > \delta^*)\\
&\ge 2g_+^{(i)}(\delta^* - \delta^{(i)}) && (0 > g_+^{(i)} \text{ and } \delta^{(i-1)} - \delta^* \ge 2(\delta^{(i)} - \delta^*))\\
&\ge 2(f(\delta^{(i)}) + g_+^{(i)}(\delta^* - \delta^{(i)}) - f(\delta^*)) && (\text{since } f(\delta^*) \ge f(\delta^{(i)}))\\
&= 2D_f(\delta^*, \delta^{(i)}). && (\text{by our choice of } g_+^{(i)})
\end{aligned}$$

This concludes the proof. □

Remark 2.5. Instead of taking twice the usual Newton step during look-ahead, one could consider $\delta' := \delta^{(i)} + \alpha(\delta - \delta^{(i)})$ for any $\alpha > 1$. By redoing the proof of Lemma 2.4 with this choice of δ', one gets $D_f(\delta^*, \delta^{(i-2)}) \ge \alpha D_f(\delta^*, \delta^{(i)})$ if look-ahead was successful in iteration $i-1$, and $D_f(\delta^*, \delta^{(i-2)}) \ge \frac{\alpha}{\alpha-1} D_f(\delta^*, \delta^{(i)})$ if look-ahead failed in iteration $i-1$. So, choosing $\alpha = 2$ balances the decay in Bregman divergence for both cases.

In the remaining of this section, we apply the accelerated Newton–Dinkelbach method to linear fractional combinatorial optimization and linear fractional programming. The application to parametric submodular function minimization is in Section 2.5.

2.3.1 Linear Fractional Combinatorial Optimization

The problem (2.1) with $\mathcal{D} \subseteq \{0,1\}^m$ is known as *linear fractional combinatorial optimization.* Radzik [126] showed that the Newton–Dinkelbach method applied to the function $f(\delta)$ as in (2.2) terminates in a strongly polynomial number of iterations. Recall that $f(\delta) = \min_{x \in \mathcal{D}} (c - \delta d)^\top x$. By the assumption $d^\top x > 0$ for all $x \in \mathcal{D}$, this function is concave, strictly decreasing, finite and piecewise-linear. Hence, it has a unique root. Moreover, $f(\delta) < 0$ and $\partial f(\delta) \cap \mathbb{R}_{<0} \neq \emptyset$ for sufficiently large δ. To implement the value and supergradient oracles, we assume that a linear optimization oracle over $\mathcal{D}$ is available, i.e. it returns an element in $\arg\min_{x \in \mathcal{D}} (c - \delta d)^\top x$ for any $\delta \in \mathbb{R}$.

Our result for the accelerated variant improves the state-of-the-art bound $O(m^2 \log m)$ by Wang et al. [156] on the standard Newton–Dinkelbach method. We will need the following lemma, given by Radzik and credited to Goemans in [125]. It gives a strongly polynomial bound on the length of a geometrically decreasing sequence of sums.

Lemma 2.6 ([125]). *Let $c \in \mathbb{R}^m_+$ and $x^{(1)}, x^{(2)}, \dots, x^{(k)} \in \{-1, 0, 1\}^m$. If $0 < c^\top x^{(i+1)} \leq \frac{1}{2} c^\top x^{(i)}$ for all $i < k$, then $k = O(m \log m)$.*

Theorem 2.7. *Algorithm 1 converges in $O(m \log m)$ iterations for linear fractional combinatorial optimization problems.*

Proof. Observe that Algorithm 1 terminates in a finite number of iterations because f is piecewise linear. Let $\delta^{(1)} > \delta^{(2)} > \cdots > \delta^{(k)} = \delta^*$ denote the sequence of iterates at the start of Algorithm 1. Since f is concave, we have $D_f(\delta^*, \delta^{(i)}) \geq 0$ for all $i \in [k]$. For each $i \in [k]$, pick $x^{(i)} \in \arg\min_{x \in \mathcal{D}} (c - \delta^{(i)} d)^\top x$ which maximizes $d^\top x$. This is well-defined because f is finite. Note that $-d^\top x^{(i)} = \min \partial f(\delta^{(i)})$. As $f(\delta^*) = 0$, the Bregman divergence of $\delta^{(i)}$ and δ^* can be written as

$$D_f(\delta^*, \delta^{(i)}) = f(\delta^{(i)}) + \max_{g \in \partial f(\delta^{(i)})} g(\delta^* - \delta^{(i)}) = (c - \delta^{(i)} d)^\top x^{(i)} - d^\top x^{(i)} (\delta^* - \delta^{(i)}) = (c - \delta^* d)^\top x^{(i)}\,.$$

According to Lemma 2.4, $(c - \delta^* d)^\top x^{(i)} = D_f(\delta^*, \delta^{(i)}) < \frac{1}{2} D_f(\delta^*, \delta^{(i-2)}) = \frac{1}{2}(c - \delta^* d)^\top x^{(i-2)}$ for all $3 \leq i \leq k$. By Lemma 2.3, we also know that $D_f(\delta^*, \delta^{(i)}) > 0$ for all $1 \leq i \leq k - 2$. Thus, applying Lemma 2.6 yields $k = O(m \log m)$. □

2.3.2 Linear Fractional Programming

We next consider *linear fractional programming*, an extension of (2.1) with the assumption that the domain $\mathcal{D} \subseteq \mathbb{R}^m$ is a polyhedron, but removing the condition $d^\top x > 0$ for $x \in \mathcal{D}$. For $c, d \in \mathbb{R}^m$, the problem is

$$\inf c^\top x / d^\top x \quad \text{s.t. } d^\top x > 0,\ x \in \mathcal{D}\,. \tag{F}$$

For the problem to be meaningful, we assume that $\mathcal{D} \cap \left\{ x : d^\top x > 0 \right\} \neq \emptyset$. The common form in the literature assumes $d^\top x > 0$ for all $x \in \mathcal{D}$ as in (2.1); we consider the more general setup for the purpose of solving M2VPI systems in Section 2.4. It is easy to see that any linear fractional combinatorial optimization problem on a domain $\mathcal{X} \subseteq \{0, 1\}^m$ can be cast as a linear fractional program with the polytope $\mathcal{D} = \operatorname{conv}(\mathcal{X})$ because $c^\top \bar{x} / d^\top \bar{x} \geq \min_{x \in \mathcal{X}} c^\top x / d^\top x$ for all $\bar{x} \in \mathcal{D}$ by the mediant inequality. The next theorem characterizes when (F) is unbounded.

Theorem 2.8. *If $\mathcal{D} \cap \left\{ x : d^\top x > 0 \right\} \neq \emptyset$, then the optimal value of (F) is $-\infty$ if and only if at least one of the following two conditions hold:*

1. *There exists $x \in \mathcal{D}$ such that $c^\top x < 0$ and $d^\top x = 0$;*
2. *There exists $r \in \mathbb{R}^m$ such that $c^\top r < 0$, $d^\top r = 0$ and $x + \lambda r \in \mathcal{D}$ for all $x \in \mathcal{D}, \lambda \geq 0$.*

Proof. By the Minkowski-Weyl Theorem, the polyhedron $\bar{\mathcal{D}} := \mathcal{D} \cap \{x : d^\top x \geq 0\}$ can be written as

$$\bar{\mathcal{D}} = \left\{ \sum_{i=1}^{k} \lambda_i g_i + \sum_{j=1}^{\ell} \nu_j h_j : \lambda \geq 0, \nu \geq 0, \|\lambda\|_1 = 1 \right\}$$

for some vectors $g_1, \dots, g_k$ and $h_1, \dots, h_\ell$. Note that $d^\top g_i \geq 0$ for all $i \in [k]$ and $d^\top h_j \geq 0$ for all $j \in [\ell]$. Let $x^\circ \in \mathcal{D} \cap \{x : d^\top x > 0\}$. If there exists $i \in [k]$ such that $c^\top g_i < 0$ and $d^\top g_i = 0$ or $j \in [\ell]$ such that $c^\top h_j < 0$ and $d^\top h_j = 0$, then,

$$\lim_{\lambda \nearrow 1} \frac{c^\top(\lambda g_i + (1-\lambda)x^\circ)}{d^\top(\lambda g_i + (1-\lambda)x^\circ)} = -\infty \quad \text{or} \quad \lim_{\lambda \to \infty} \frac{c^\top(x^\circ + \lambda h_j)}{d^\top(x^\circ + \lambda h_j)} = -\infty$$

as in Condition 1 or Condition 2.

Otherwise, the fractional value of any element in $\mathcal{D} \cap \{x : d^\top x > 0\}$ can be lower bounded by

$$\begin{aligned} \frac{c^\top(\sum_{i=1}^k \lambda_i g_i + \sum_{j=1}^\ell \nu_j h_j)}{d^\top(\sum_{i=1}^k \lambda_i g_i + \sum_{j=1}^\ell \nu_j h_j)} &\geq \frac{\sum_{i \in [k], d^\top g_i > 0} \lambda_i c^\top g_i + \sum_{j \in [\ell], d^\top h_j > 0} \nu_j c^\top h_j}{\sum_{i \in [k], d^\top g_i > 0} \lambda_i d^\top g_i + \sum_{j \in [\ell], d^\top h_j > 0} \nu_j d^\top h_j} \\ &\geq \min \left\{ \min_{i \in [k], d^\top g_i > 0} \frac{c^\top g_i}{d^\top g_i}, \min_{j \in [\ell], d^\top h_j > 0} \frac{c^\top h_j}{d^\top h_j} \right\}, \end{aligned}$$

where the last expression is finite by the assumption that $\mathcal{D} \cap \{x : d^\top x > 0\}$ is non-empty. □

Example 2.9. Unlike in linear programming, the optimal value may not be attained even if it is finite. Consider the instance given by $\inf(-x_1 + x_2)/(x_1 + x_2)$ subject to $x_1 + x_2 > 0$ and $-x_1 + x_2 = 1$. The numerator is equal to 1 for any feasible solution, while the denominator can be made arbitrarily large. Hence, the optimal value of this program is 0, which is not attained in the feasible region.

We use the Newton–Dinkelbach method for f as in (2.2), that is, $f(\delta) = \inf_{x \in \mathcal{D}} (c - \delta d)^\top x$. Since $\mathcal{D} \neq \emptyset$, $f(\delta) < \infty$ for all $\delta \in \mathbb{R}$. By the Minkowski–Weyl theorem, there exist finitely many points $P \subseteq \mathcal{D}$ such that $f(\delta) = \min_{x \in P} (c - \delta d)^\top x$ for all $\delta \in \operatorname{dom}(f)$. Hence, f is concave and piecewise linear. Observe that $f(\delta) > -\infty$ if and only if every ray r in the recession cone of $\mathcal{D}$ satisfies $(c - \delta d)^\top r \geq 0$. For f to be proper, we need to assume that Condition 2 in Theorem 2.8 does not hold. Moreover, we require the existence of a point $\delta' \in \operatorname{dom}(f)$ such that $f(\delta') = (c - \delta' d)^\top x' \leq 0$ for some $x' \in \mathcal{D}$ with $d^\top x' > 0$. It follows that f has a root or attains its maximum because $\operatorname{dom}(f)$ is closed. We are ready to characterize the optimal value of (F) using f.

Lemma 2.10. *Assume that there exists $\delta' \in \operatorname{dom}(f)$ such that $f(\delta') = (c - \delta' d)^\top x' \leq 0$ for some $x' \in \mathcal{D}$ with $d^\top x' > 0$. If f has a root, then the optimal value of (F) is equal to the largest root and is attained. Otherwise, the optimal value is $-\infty$.*

Proof. Recall the definition of $\delta^* = \max(\{\delta : f(\delta) = 0\} \cup \arg\max f(\delta))$. By our assumption on f, there exists $x^* \in \mathcal{D}$ such that $f(\delta^*) = (c - \delta^* d)^\top x^*$ and $d^\top x^* > 0$. If f has a root, then $f(\delta^*) = 0$. This implies that $c^\top x/d^\top x \geq \delta^* = c^\top x^*/d^\top x^*$ for all $x \in \mathcal{D}$ with $d^\top x > 0$ as desired. Next, assume that f does not have a root. Then $f(\delta^*) < 0$ and $0 \in \partial f(\delta^*)$. By convexity, there exists $\bar{x} \in \mathcal{D}$ such that $(c - \delta^* d)^\top \bar{x} = f(\delta^*) < 0$ and $d^\top \bar{x} = 0$. Then $c^\top \bar{x} < 0$, so $\bar{x}$ is a point as in Condition 1 of Theorem 2.8.

□

2.4 Monotone Two Variables per Inequality Systems

Recall that an M2VPI system can be represented as a directed multigraph $G = (V, E)$ with arc costs $c \in \mathbb{R}^m$ and gain factors $\gamma \in \mathbb{R}^m_{++}$. For a u-v walk P in G with $E(P) = (e_1, e_2, \dots, e_k)$, its *cost* and *gain factor* are defined as $c(P) := \sum_{i=1}^k \left(\prod_{j=1}^{i-1} \gamma_{e_j}\right) c_{e_i}$ and $\gamma(P) := \prod_{i=1}^k \gamma_{e_i}$ respectively. If P is a single vertex, then $c(P) := 0$ and $\gamma(P) := 1$. The walk P induces the valid inequality $y_u \leq c(P) + \gamma(P) y_v$, implied by the sequence of arcs/inequalities in $E(P)$. It is also worth considering the dual interpretation. Dual variables on arcs correspond to generalized flows: if 1 unit of flow enter the arc $e = (u, v)$ at u, then γ_e units reach v, at a shipping cost of c_e. Thus, if 1 unit of flow enter a path P, then $\gamma(P)$ units reach the end of the path, incurring a cost of $c(P)$.

Given node labels $y \in \bar{\mathbb{R}}^n$, the *y-cost* of a u-v walk P is defined as $c(P) + \gamma(P) y_v$. Note that the y-cost of a walk only depends on the label at the sink. A u-v path is called a *shortest u-v path with respect to y* if it has the smallest y-cost among all u-v walks. A *shortest path from u with respect to y* is a shortest u-v path with respect to y for some node v. Such a path does not always exist, as demonstrated in Section A.2.

If P is a u-u walk such that its intermediate nodes are distinct, then it is called a *cycle at u*. Given a u-v walk P and a v-w walk Q, we denote PQ as the u-w walk obtained by concatenating P and Q.

Definition 2.11. A cycle C is called *flow-generating* if $\gamma(C) > 1$, *unit-gain* if $\gamma(C) = 1$, and *flow-absorbing* if $\gamma(C) < 1$. We say that a unit-gain cycle C is *negative* if $c(C) < 0$.

Note that $c(C)$ depends on the starting point u of a cycle C. This ambiguity is resolved by using the term *cycle at u*. For a unit-gain cycle C, it is not hard to see that the starting point does not affect the sign of $c(C)$. Hence, the definition of a negative unit-gain cycle is sound. Observe that a flow-absorbing cycle C induces an upper bound $y_u \leq c(C)/(1 - \gamma(C))$, while a flow-generating cycle C induces a lower bound $y_u \geq -c(C)(\gamma(C) - 1)$. Let $\mathcal{C}_u^{abs}(G)$ and $\mathcal{C}_u^{gen}(G)$ denote the set of flow-absorbing cycles and flow-generating cycles at u in G respectively.

Definition 2.12. Given a flow-generating cycle C at u, a flow-absorbing cycle D at v, and a u-v path P, the walk CPD is called a *bicycle*. We say that the bicycle is *negative* if

$$c(P) + \gamma(P) \frac{c(D)}{1 - \gamma(D)} < \frac{-c(C)}{\gamma(C) - 1}\,.$$

Using these two structures, Shostak characterized the feasibility of M2VPI systems.

Theorem 2.13 ([136]). *An M2VPI system (G, c, γ) is infeasible if and only if G contains a negative unit-gain cycle or a negative bicycle.*

2.4.1 A Linear Fractional Programming Formulation

Our goal is to compute the pointwise maximal solution $y^{\max} \in \bar{\mathbb{R}}^n$ to an M2VPI system if it is feasible, where $y_u^{\max} := \infty$ if and only if the variable y_u is unbounded from above. It is well known how to convert $y^{\max}$ into a finite feasible solution — we refer to Section A.3 for details. In order to apply Algorithm 1, we first need to reformulate the problem as a linear fractional program. Now, every coordinate $y_u^{\max}$ can be expressed as the following primal-dual pair of linear programs, where $\nabla x_v := \sum_{e\in\delta^+(v)} x_e - \sum_{e\in\delta^-(v)} \gamma_e x_e$ denotes the net flow at a node v.

$$\begin{aligned} \min\ & c^\top x & & \text{(P}_u\text{)} \\ \text{s.t.}\ & \nabla x_u = 1 & & \\ & \nabla x_v = 0 & \forall v \in V \setminus u & \\ & x \geq \mathbb{0} & & \end{aligned} \qquad \begin{aligned} \max\ & y_u & & \text{(D}_u\text{)} \\ \text{s.t.}\ & y_v - \gamma_e y_w \leq c_e & \forall e = (v,w) \in E & \end{aligned}$$

The primal LP (P_u) is a minimum-cost generalized flow problem with a supply of 1 at node u. It asks for the cheapest way to destroy one unit of flow at u. Observe that it is feasible if and only if u can reach a flow-absorbing cycle in G. If it is feasible, then it is unbounded if and only if there exists a negative unit-gain cycle or a negative bicycle in G. It can be reformulated as the following linear fractional program

$$\inf \frac{c^\top x}{1 - \sum_{e\in\delta^-(u)} \gamma_e x_e} \quad \text{s.t. } 1 - \sum_{e\in\delta^-(u)} \gamma_e x_e > 0,\ x \in \mathcal{D}\,. \tag{F$_u$}$$

with the polyhedron

$$\mathcal{D} := \left\{ x \in \mathbb{R}^m_+ : x(\delta^+(u)) = 1, \nabla x_v = 0\ \forall v \in V \setminus u \right\}.$$

Indeed, if x is a feasible solution to (P_u), then $x/x(\delta^+(u))$ is a feasible solution to (F_u) with the same objective value. This is because $1 - \sum_{e\in\delta^-(u)} \gamma_e x_e / x(\delta^+(u)) = 1/x(\delta^+(u))$. Conversely, if x is a feasible solution to (F_u), then $x/(1 - \sum_{e\in\delta^-(u)} \gamma_e x_e)$ is a feasible solution to (P_u) with the same objective value. Even though the denominator is an affine function of x, it can be made linear to conform with (F) by working with the polyhedron $\{(x, 1) : x \in \mathcal{D}\}$.

Our goal is to solve (F_u) using Algorithm 1. For a fixed $\delta \in \mathbb{R}$, the value of the parametric function $f(\delta)$ can be written as the following pair of primal and dual LPs respectively

$$\begin{aligned}
&\min\ c^\top x + \delta \sum_{e\in\delta^-(u)} \gamma_e x_e - \delta \\
&\text{s.t.}\ \ x \in \mathcal{D}
\end{aligned}
\qquad\qquad
\begin{aligned}
&\max\ y_u - \delta \\
&\text{s.t.}\quad y_v - \gamma_e \delta \le c_e \qquad \forall e = (v,u) \in \delta^-(u) \\
&\qquad\ \ y_v - \gamma_e y_w \le c_e \qquad \forall e = (v,w) \notin \delta^-(u).
\end{aligned}$$

We refer to them as the *primal (resp. dual) LP for* $f(\delta)$, and their corresponding feasible/optimal solution as a *feasible/optimal primal (resp. dual) solution to* $f(\delta)$.

Due to the specific structure of this linear fractional program, a suitable initial point for the Newton–Dinkelbach method can be obtained from any feasible solution to (F_u). This is a consequence of the unboundedness test given by the following lemma.

Lemma 2.14. *Let x be a feasible solution to (F_u) and $\bar\delta := c^\top x/(1-\sum_{e\in\delta^-(u)}\gamma_e x_e)$. If either $f(\bar\delta) = -\infty$ or $f(\bar\delta) = c^\top \bar x - \bar\delta(1-\sum_{e\in\delta^-(u)}\gamma_e \bar x_e) < 0$ for some $\bar x \in \mathcal{D}$ with $1-\sum_{e\in\delta^-(u)}\gamma_e \bar x_e \le 0$, then the optimal value of (F_u) is $-\infty$.*

Proof. First, assume that $f(\bar\delta) > -\infty$. Let $\lambda := (1-\sum_{e\in\delta^-(u)}\gamma_e x_e)/\sum_{e\in\delta^-(u)}\gamma_e(\bar x_e - x_e)$. Note that $\lambda \in (0,1]$. Consider the convex combination $\hat x := \lambda \bar x + (1-\lambda)x \in \mathcal{D}$. Then, $c^\top \hat x < 0$ and $1-\sum_{e\in\delta^-(u)}\gamma_e \hat x_e = 0$. Hence, the optimal value of (F_u) is unbounded by Condition 1 of Theorem 2.8. Next, assume that $f(\bar\delta) = -\infty$. There exists a ray r in the recession cone of $\mathcal{D}$ such that $c^\top r - \bar\delta\sum_{e\in\delta^-(u)}\gamma_e r_e < 0$. Note that $r \ge 0$. If $r(\delta^-(u)) = 0$, then r satisfies Condition 2 of Theorem 2.8. So, the optimal value is unbounded. Otherwise, for a sufficiently large $\alpha > 0$, we have $c^\top(x+\alpha r) + \bar\delta(1-\sum_{e\in\delta^-(u)}\gamma_e(x_e+\alpha r_e)) < 0$ and $1-\sum_{e\in\delta^-(u)}\gamma_e(x_e+\alpha r_e) < 0$. Then, taking an appropriate convex combination of $x+\alpha r$ and x like before produces a point in $\mathcal{D}$ which satisfies Condition 1 of Theorem 2.8. □

In order to characterize the finiteness of $f(\delta)$, we introduce the following notion of a negative flow-generating cycle.

Definition 2.15. For a fixed $\delta \in \mathbb{R}$ and $u \in V$, a flow-generating cycle C is said to be (δ,u)*-negative* if there exists a path P from a node $v \in V(C)$ to node u such that

$$c(C) + (\gamma(C)-1)(c(P)+\gamma(P)\delta) < 0$$

where C is treated as a v-v walk in $c(C)$.

Lemma 2.16. *For any $\delta \in \mathbb{R}$, $f(\delta) = -\infty$ if and only if $\mathcal{D} \neq \emptyset$ and there exists a negative unit-gain cycle, a negative bicycle, or a (δ,u)-negative flow-generating cycle in $G \setminus \delta^+(u)$.*

Proof. The primal LP for $f(\delta)$ is unbounded if and only if $\mathcal{D} \neq \emptyset$ and there exists an extreme ray r in the recession cone of $\mathcal{D}$ such that $c^\top r + \delta\sum_{e\in\delta^-(u)}\gamma_e r_e < 0$. Note that the recession cone of $\mathcal{D}$ is $\{x \in \mathbb{R}^m_+ : x(\delta^+(u)) = 0, \nabla x_v = 0\ \forall v \neq u\}$. By the generalized flow decomposition theorem, r belongs to one of the following three fundamental flows in $G\setminus\delta^+(u)$:

(1) a unit-gain cycle, (2) a bicycle, (3) a flow-generating cycle C and a path P from C to u. In the first two cases, $r_e = 0$ for all $e \in \delta^-(u)$. Thus, the unit-gain cycle or bicycle is negative. In the last case, we have $c(C) + (\gamma(C) - 1)(c(P) + \gamma(P)\delta) = c^\top r + \delta \sum_{e\in\delta^-(u)} \gamma_e r_e$. $\square$

It turns out that if we have an optimal dual solution y to $f(\delta)$ for some $\delta \in \mathbb{R}$, then we can compute an optimal dual solution to $f(\delta')$ for any $\delta' < \delta$. A suitable subroutine for this task is the so called Grapevine algorithm (Algorithm 2), developed by Aspvall and Shiloach [7].

Algorithm 2: Grapevine

```
input  : A directed multigraph $G = (V, E)$ with arc costs $c \in \mathbb{R}^m$ and gain factors
         $\gamma \in \mathbb{R}^m_{++}$, node labels $y \in \bar{\mathbb{R}}^n$, and a node $u \in V$.
output : Node labels $y \in \bar{\mathbb{R}}^n$ and a walk $P$ of length at most $n$ starting from $u$.

1  for $i = 1$ to $n$ do
2      foreach $v \in V$ do
3          $y'_v \leftarrow \min(y_v, \min_{vw\in\delta^+(v)} c_{vw} + \gamma_{vw} y_w)$
4          if $y'_v < y_v$ then
5              pred$(v, i) \leftarrow \arg\min_{vw\in\delta^+(v)} c_{vw} + \gamma_{vw} y_w$      ▷ Break ties arbitrarily
6          else
7              pred$(v, i) \leftarrow \emptyset$
8      $y \leftarrow y'$
9  Let $P$ be the walk obtained by tracing from pred$(u, n)$
10 return $(y, P)$
```

Given initial node labels $y \in \bar{\mathbb{R}}^n$ and a specified node u, Grapevine runs for n iterations. We say that an arc $e = (v, w)$ is *violated with respect to* y if $y_v > c_e + \gamma_e y_w$. In an iteration $i \in [n]$, the algorithm records the most violated arc with respect to y in $\delta^+(v)$ as pred(v, i), for each node $v \in V$ (ties are broken arbitrarily). Note that pred$(v, i) = \emptyset$ if there are no violated arcs in $\delta^+(v)$. Then, each y_v is decreased by the amount of violation in the corresponding recorded arc. After n iterations, the algorithm traces a walk P from u by following the recorded arcs in reverse chronological order. During the trace, if pred$(v, i) = \emptyset$ for some $v \in V$ and $i > 1$, then pred$(v, i - 1)$ is read. Finally, the updated node labels y and the walk P are returned. Clearly, the running time of Grapevine is $O(mn)$.

Given an optimal dual solution $y \in \mathbb{R}^n$ to $f(\delta)$ and $\delta' < \delta$, the dual LP for $f(\delta')$ can be solved using Grapevine as follows. Define the directed graph $G_u := (V \cup \{u'\}, E_u)$ where $E_u := (E \setminus \delta^-(u)) \cup \{vu' : vu \in \delta^-(u)\}$. The graph G_u is obtained from G by splitting u into two nodes u, u' and reassigning the incoming arcs of u to u'. These arcs inherit the same costs and gain factors from their counterparts in G. Let $\bar{y} \in \mathbb{R}^{n+1}$ be node labels in G_u defined by $\bar{y}_{u'} := \delta'$ and $\bar{y}_v := y_v$ for all $v \neq u'$. Then, we run Grapevine on G_u with input node labels $\bar{y}$ and node u. Note that $\bar{y}_{u'}$ remains unchanged throughout the algorithm. The next lemma verifies the correctness of this method.

Lemma 2.17. *Given an optimal dual solution $y \in \mathbb{R}^n$ to $f(\delta)$ and $\delta' < \delta$, define $\bar{y} \in \mathbb{R}^{n+1}$ as $\bar{y}_{u'} := \delta'$ and $\bar{y}_v := y_v$ for all $v \in V$. Let $(\bar{z}, P)$ be the node labels and walk returned by* Grapevine$(G_u, \bar{y}, u)$. *If $\bar{z}_V$ is not feasible to the dual LP for $f(\delta')$, then $f(\delta') = -\infty$. Otherwise, $\bar{z}_V$ is a dual optimal solution to $f(\delta')$ and P is a shortest path from u with respect to $\bar{y}$ in G_u.*

Proof. Since $f(\delta) = y_u - \delta$ is finite, we have $\mathcal{D} \neq \emptyset$. First, assume that $\bar{z}_V$ is not feasible to the dual LP for $f(\delta')$. Then, there exists a violated arc in G_u with respect to $\bar{z}$. Let w be the head of this arc and let R be the walk obtained by tracing $\text{pred}(w, n)$ in reverse chronological order. Then, R ends at u' because y is dual feasible to $f(\delta)$. Since R has n edges, decompose it into $R = QCP'$ where Q is a w-v walk, C is a nontrivial cycle at v, and P' is a v-u' path for some node v. Then, we have $c(CP') + \gamma(CP')\delta' < c(P') + \gamma(P')\delta' \leq \bar{y}_v$. Due to Lemma 2.16, it suffices to show that $\gamma(C) > 1$, as this would imply that C is a (δ', u)-negative flow-generating cycle in G. Suppose otherwise for a contradiction. Since y is dual feasible to $f(\delta)$ and $u' \notin V(C)$, we have $\bar{y}_v \leq c(C) + \gamma(C)\bar{y}_v$. If $\gamma(C) = 1$, then we obtain $0 \leq c(C) < 0$ from the previous two inequalities. Otherwise, we get the following contradiction

$$\bar{y}_v \leq \frac{c(C)}{1 - \gamma(C)} < c(P') + \gamma(P')\delta' \leq \bar{y}_v.$$

Next, assume that $\bar{z}_V$ is a dual feasible solution to $f(\delta')$. Then, P is a u-t path for some node t. This is because if P is not simple, repeating the argument from the previous paragraph proves that the dual LP for $f(\delta')$ is infeasible. Note that $\bar{y}_t = \bar{z}_t$. Moreover, $\bar{z}_v \leq c_{vw} + \gamma_{vw}\bar{z}_w$ for all $vw \in E_u$, with equality on $E(P)$. Let $c^{\bar{z}} \in \mathbb{R}^m_+$ be the reduced cost defined by $c^{\bar{z}}_{vw} := c_{vw} + \gamma_{vw}\bar{z}_w - \bar{z}_v$ for all $vw \in E_u$. Since for every u-t walk P' we have

$$c(P) + \gamma(P)\bar{z}_t - \bar{z}_u = c^{\bar{z}}(P) = 0 \leq c^{\bar{z}}(P') = c(P') + \gamma(P')\bar{z}_t - \bar{z}_u,$$

it follows that P is a shortest u-t path with respect to $\bar{y}$.

It is left to show that $\bar{z}_V$ is a dual optimal solution to $f(\delta')$. Let z^* be an optimal dual solution to $f(\delta')$. Note that $z^*_u \leq y_u$ because $\delta' < \delta$. For the purpose of contradiction, suppose that $\bar{z}_u < z^*_u$. Since $\bar{z}_u < \bar{y}_u$, the path P ends at u' because y is dual feasible to $f(\delta)$. Thus, $\bar{z}_u = c(P) + \gamma(P)\delta'$. However, P also implies the valid inequality $z^*_u \leq c(P) + \gamma(P)\delta'$, which is a contradiction. □

If $\bar{z}_V$ is an optimal dual solution to $f(\delta')$, a supergradient in $\partial f(\delta')$ can be inferred from the returned path P. We say that an arc $e = (v, w)$ is *tight with respect to* $\bar{z}$ if $\bar{z}_v = c_e + \gamma_e\bar{z}_w$. By complementary slackness, every optimal primal solution to $f(\delta')$ is supported on the subgraph of G_u induced by tight arcs with respect to $\bar{z}$. In particular, any u-u' path or any path from u to a flow-absorbing cycle in this subgraph constitutes a basic optimal primal solution to $f(\delta')$. As P is also a path in this subgraph, we have $\gamma(P) - 1 \in \partial f(\delta')$ if P ends at u'. Otherwise, u can reach a flow-absorbing cycle in this subgraph because $\delta' < \delta$. In this case, $-1 \in \partial f(\delta')$.

2.4.2 A Strongly Polynomial Label-Correcting Algorithm

Using Algorithm 1, we develop a strongly polynomial label-correcting algorithm for solving an M2VPI system (G, c, γ). The main idea is to start with a subsystem for which (D_u) is trivial, and progressively solve (D_u) for larger and larger subsystems. Throughout the algorithm, we maintain node labels $y \in \bar{\mathbb{R}}^n$ which form valid upper bounds on each variable. They are initialized to ∞ at every node. We also maintain a subgraph of G, which initially is $G^{(0)} := (V, \emptyset)$.

Algorithm 3: Label-correcting algorithm for M2VPI systems

```
input  : An M2VPI system $(G, c, \gamma)$.
output : The pointwise maximal solution $y^{\max}$ or the string INFEASIBLE.
1  Initialize graph $G^{(0)} \leftarrow (V, \emptyset)$ and counter $k \leftarrow 0$
2  Initialize node labels $y \in \bar{\mathbb{R}}^n$ as $y_v \leftarrow \infty\ \forall v \in V$
3  foreach $u \in V$ do
4      $k \leftarrow k + 1$
5      $G^{(k)} \leftarrow G^{(k-1)} \cup \delta^+(u)$
6      $y_u \leftarrow \min_{uv \in \delta^+(u)} c_{uv} + \gamma_{uv} y_v$
7      if $y_u = \infty$ and $\mathcal{C}_u^{abs}(G^{(k)}) \neq \emptyset$ then
8          $y_u \leftarrow c(C)/(1 - \gamma(C))$ for any $C \in \mathcal{C}_u^{abs}(G^{(k)})$
9      if $y_u < \infty$ then
10         Define node labels $\bar{y} \in \bar{\mathbb{R}}^{n+1}$ as $\bar{y}_{u'} \leftarrow y_u$ and $\bar{y}_v \leftarrow y_v\ \forall v \in V$
11         $(\bar{y}, P) \leftarrow$ GRAPEVINE($G_u^{(k)}$, $\bar{y}$, $u$)
12         if $\exists$ a violated arc w.r.t. $\bar{y}$ in $G_u^{(k)}$ or ($|E(P)| > 0$ and $\gamma(P) \geq 1$) then
13             return INFEASIBLE
14         $\bar{y}_{u'} \leftarrow$ LOOK-AHEADNEWTON(GRAPEVINE($G_u^{(k)}, \cdot, u$), $\bar{y}_{u'}$, $\gamma(P) - 1$)
15         if $\bar{y}_{u'} =$ NO ROOT then
16             return INFEASIBLE
17         $y \leftarrow \bar{y}_V$
18 return $y$
```

The algorithm (Algorithm 3) is divided into n *phases*. At the start of phase $k \in [n]$, a new node $u \in V$ is selected and all of its outgoing arcs in G are added to $G^{(k-1)}$, resulting in a larger subgraph $G^{(k)}$. Since $y_u = \infty$ at this point, we update it to the smallest upper bound implied by its outgoing arcs and the labels of its outneighbours. If y_u is still infinity, then we know that $\delta^+(u) = \emptyset$ or $y_v = \infty$ for all $v \in N^+(u)$. In this case, we find a flow-absorbing cycle at u in $G^{(k)}$ using the multiplicative Bellman–Ford algorithm, by treating the gain factors as arc costs. If there is none, then we proceed to the next phase immediately as y_u is unbounded from above in the subsystem $(G^{(k)}, c, \gamma)$. This is because u cannot reach a flow-absorbing cycle in $G^{(k)}$ by induction. We would like to point out that this does not necessarily imply that the full system (G, c, γ) is feasible (see Section A.3 for details). On the other hand, if Bellman–Ford returns a flow-absorbing cycle, then y_u is set to the upper bound implied by the cycle. Then, we apply Algorithm 1 to solve (D_u) for the subsystem $(G^{(k)}, c, \gamma)$.

The value and supergradient oracle for the parametric function $f(\delta)$ is Grapevine. Let $G_u^{(k)}$ be the modified graph and $\bar{y} \in \bar{\mathbb{R}}^{n+1}$ be the node labels as defined in the previous subsection. In order to provide Algorithm 1 with a suitable initial point and supergradient, we run Grapevine on $G_u^{(k)}$ with input node labels $\bar{y}$. It updates $\bar{y}$ and returns a walk P from u. If $\bar{y}_V$ is not feasible to the dual LP for $f(\bar{y}_{u'})$ or P is a non-trivial walk with $\gamma(P) \geq 1$, then we declare infeasibility. Otherwise, we run Algorithm 1 with the initial point $\bar{y}_{u'}$ and supergradient $\gamma(P) - 1$. We remark that Grapevine continues to update $\bar{y}$ throughout the execution of Algorithm 1.

Theorem 2.18. *If Algorithm 3 returns $y \in \bar{\mathbb{R}}^n$, then $y = y^{\max}$ if the M2VPI system is feasible. Otherwise, the system is infeasible.*

Proof. It suffices to prove the theorem for the subsystem $(G^{(k)}, c, \gamma)$ encountered in each phase k. We proceed by induction on k. For the base case $k = 0$, the system $(G^{(0)}, c, \gamma)$ is trivially feasible as it does not have any constraints. Hence, $y^{\max} = (\infty, \infty, \dots, \infty) = y$, where the second equality is due to our initialization. For the inductive step, assume that the theorem is true for some $0 \leq k < n$ and consider the system $(G^{(k+1)}, c, \gamma)$. If Algorithm 3 terminated in phase k, then $(G^{(k+1)}, c, \gamma)$ is infeasible by the inductive hypothesis. So, let $y \in \bar{\mathbb{R}}^n$ be the node labels maintained by the algorithm during Line 9 of phase $k+1$. We have $y_u = \infty$ if and only if $\mathcal{C}_u^{abs}(G^{(k+1)}) = \emptyset$ and $y_v = \infty$ for all $v \in N^+(u)$. For each $v \neq u$, we also have $y_v = \infty$ if and only if v cannot reach a flow-absorbing cycle in $G^{(k)}$. So, if $y_u = \infty$, then u cannot reach a flow-absorbing cycle in $G^{(k+1)}$. By the inductive hypothesis, $y = y^{\max}$ if the system $(G^{(k+1)}, c, \gamma)$ is feasible.

Next, assume that $y_u < \infty$. Without loss of generality, we may assume that every node v with $y_v = \infty$ can reach u in $G^{(k+1)}$. Let $W := \{v \in V : y_v = \infty\}$. Note that the cut W does not have any outgoing edges in $G^{(k+1)}$. If there exists a negative unit-gain cycle in $G^{(k+1)}[W]$, then it contains a violated arc with respect to any finite labels. In this case, the algorithm correctly detects infeasibility. Otherwise, by Lemma 2.16, $f(\delta') > -\infty$ for a sufficiently high $\delta' \in \mathbb{R}$ because there are no flow-absorbing cycles in $G^{(k+1)}[W]$. Pick $\delta' > y_u$ big enough such that an optimal dual solution $y' \in \mathbb{R}^n$ to $f(\delta')$ satisfies $y'_v = y_v$ for all $v \in V \setminus W$. Among all such optimal dual solutions, choose y' as the pointwise maximal one. Then, every vertex $v \in W$ has a tight path to u in $G^{(k+1)}$. Now, let $\bar{y}' \in \mathbb{R}^{n+1}$ be node labels defined by $\bar{y}'_{u'} := y_u$ and $\bar{y}'_v := y'_v$ for all $v \in V$. It is easy to see that running GRAPEVINE on $G_u^{(k+1)}$ with input node labels $\bar{y}$ and $\bar{y}'$ yields the same behaviour. Let $(\bar{z}, P)$ be the node labels and walk returned by GRAPEVINE.

Let $x \in \mathbb{R}_+^{E(G^{(k+1)})}$ be a feasible solution to (F_u) such that $y_u = c^\top x/(1 - \sum_{e\in\delta^-(u)} \gamma_e x_e)$. Clearly, such an x exists if $y_u = c(C)/1-\gamma(C)$ for some flow-absorbing cycle $C \in \mathcal{C}_u^{abs}(G^{(k+1)})$. Otherwise, if $y_u = c_{uv} + \gamma_{uv} y_v$ for some $uv \in \delta^+(u)$, then $y_v = c(Q) + \gamma(Q)(c(C)/1 - \gamma(C))$ where Q is a path leading to a flow-absorbing cycle C in $G^{(k)}[V \setminus W]$. This is because $y_{V\setminus W}$ is the pointwise maximal solution to the feasible subsystem $(G^{(k)}[V \setminus W], c, \gamma)$ by the inductive hypothesis. Hence, x can be chosen as the fundamental flow from u to the cycle C via the path $Q + uv$.

Now, according to Lemma 2.17, if $\bar{z}_V$ is not feasible to the dual LP for $f(y_u)$, then $f(y_u) = -\infty$. By Lemma 2.14, the optimal value of (F_u) is $-\infty$. On the other hand, if $\bar{z}_V$ is a feasible solution to the dual LP for $f(y_u)$, then it is also optimal. Moreover, P is a shortest path from u with respect to $\bar{y}'$ in $G_u^{(k+1)}$. If $|E(P)| > 0$ and $\gamma(P) \geq 1$, then the path ends at u' because $\bar{y}'$ is dual feasible to $f(\delta')$. Let $\bar{x}$ be the fundamental u-u' flow on P. By complementary slackness, $\bar{x}$ is an optimal primal solution to $f(y_u) < 0$ and $1 - \sum_{e\in\delta^-(u)} \gamma_e \bar{x}_e = 1 - \gamma(P) \leq 0$. Applying Lemma 2.14 again yields unboundedness of (F_u). In both cases, as (P_u) is feasible, $(G^{(k+1)}, c, \gamma)$ is infeasible.

If the above cases do not apply, then $\bar{z}_u$ and $\gamma(P) - 1$ constitute a suitable initial point and supergradient for Algorithm 1 respectively. Note that the node labels $\bar{y}$ are updated to $\bar{z} \in \mathbb{R}^{n+1}$. Throughout the execution of Algorithm 1, it is easy to see that $\bar{y}_V$ remains an upper bound on every feasible solution to the system $(G^{(k+1)}, c, \gamma)$. If phase $k+1$ terminates with node labels $y := \bar{y}_V$, then y_u is the largest root of f. By Lemma 2.10, y_u is the optimal value of (F_u). Since y is an optimal solution to (D_u), we obtain $y = y^{\max}$ as desired. On the other hand, if phase $k+1$ terminates with `INFEASIBLE`, then f does not have a root. By Lemma 2.10, the optimal value of (F_u) is $-\infty$. As (P_u) is feasible, this implies that $(G^{(k+1)}, c, \gamma)$ is infeasible. $\square$

We would like to point out that Algorithm 3 may return node labels $y \in \bar{\mathbb{R}}^n$ even if the M2VPI system is infeasible. This happens when y contains ∞ entries. It is well-known how to ascertain the system's feasibility status in this case (see Section A.3 for details).

To bound the running time of Algorithm 3, it suffices to bound the running time of Algorithm 1 in every phase. Our strategy is to analyze the sequence of paths whose gain factors determine the right derivative of f at each iterate of Algorithm 1. The next property is crucial in our arc elimination argument.

Definition 2.19. Let $\mathcal{P} = (P^{(1)}, P^{(2)}, \ldots, P^{(\ell)})$ be a sequence of paths from u. We say that $\mathcal{P}$ satisfies *subpath monotonicity at* u if for every pair $P^{(i)}, P^{(j)}$ where $i < j$ and for every shared node $v \neq u$, we have $\gamma(P_{uv}^{(i)}) \leq \gamma(P_{uv}^{(j)})$.

Lemma 2.20. *Let $\delta^{(1)} > \delta^{(2)} > \cdots > \delta^{(\ell)}$ be a decreasing sequence of iterates. For each $\delta^{(i)} \in \mathbb{R}$, let $P^{(i)}$ be a u-u' path in G_u on which a unit flow is an optimal primal solution to $f(\delta^{(i)})$. Then, the sequence $(P^{(1)}, P^{(2)}, \dots, P^{(\ell)})$ satisfies subpath monotonicity at u.*

Proof. For each $i \in [\ell]$, let $y^{(i)} \in \mathbb{R}^n$ be an optimal dual solution to $f(\delta^{(i)})$. Let $\bar{y}^{(i)} \in \mathbb{R}^{n+1}$ be the node labels in G_u defined by $\bar{y}^{(i)}_{u'} := \delta^{(i)}$ and $\bar{y}^{(i)}_v := y^{(i)}_v$ for all $v \neq u'$. By complementary slackness, every edge in $P^{(i)}$ is tight with respect to $\bar{y}^{(i)}$. Hence, $P^{(i)}$ is a shortest u-u' path in G_u with respect to $\bar{y}^{(i)}$. Now, pick a pair of paths $P^{(i)}$ and $P^{(j)}$ such that $i < j$ and they share a node $v \neq u$. Then, the subpaths $P^{(i)}_{uv}$ and $P^{(j)}_{uv}$ are also shortest u-v paths in G_u with respect to $\bar{y}^{(i)}$ and $\bar{y}^{(j)}$ respectively. Observe that $\bar{y}^{(i)}_v > \bar{y}^{(j)}_v$ because $\bar{y}^{(i)}_{u'} = \delta^{(i)} > \delta^{(j)} = \bar{y}^{(j)}_{u'}$. Define the function $\psi : [\bar{y}^{(j)}_v, \bar{y}^{(i)}_v] \to \bar{\mathbb{R}}$ as

$$\psi(\alpha) := \inf \{c(P) + \gamma(P)\alpha : P \text{ is a } u\text{-}v \text{ walk in } G_u\}.$$

Clearly, it is increasing and concave. It is also finite because $\psi(\bar{y}^{(i)}_v) = c(P^{(i)}_{uv}) + \gamma(P^{(i)}_{uv})\bar{y}^{(i)}_v$ and $\psi(\bar{y}^{(j)}_v) = c(P^{(j)}_{uv}) + \gamma(P^{(j)}_{uv})\bar{y}^{(j)}_v$. Subpath monotonicity then follows from concavity of ψ. □

Theorem 2.21. *In each phase k of Algorithm 3, Algorithm 1 terminates in $O(|E(G^{(k)})|)$ iterations.*

Proof. Fix a phase $k \in [n]$ and denote $m_k := |E(G^{(k)})|$. Let $\bar{\mathcal{Y}} = (\bar{y}^{(1)}, \bar{y}^{(2)}, \dots, \bar{y}^{(\ell)})$ be the sequence of node labels at the start of every iteration of Algorithm 1 in phase k. Note that $\bar{y}^{(i)} \geq \bar{y}^{(i+1)}$ and $\bar{y}^{(i)}_{u'} > \bar{y}^{(i+1)}_{u'}$ for all $i < \ell$. Let $f : \mathbb{R} \to \bar{\mathbb{R}}$ be the parametric function associated with the linear fractional program (F_u) for the subsystem $(G^{(k)}, c, \gamma)$. We may assume that $\ell \geq 1$, which in turn implies that $f(y^{(1)}_{u'})$ is finite by Lemma 2.14. By Lemma 2.16, there are no negative unit-gain cycles or bicycles in $G^{(k)} \setminus \delta^+(u)$. It follows that all negative unit-gain cycles and negative bicycles in $G^{(k)}$ contain u. Hence, there exists a smallest $\varepsilon \geq 0$ such that the subsystem $(G^{(k)}, \hat{c}, \gamma)$ is feasible, where $\hat{c} \in \mathbb{R}^{m_k}$ are modified arc costs defined by $\hat{c}_e := c_e + \varepsilon$ if $e \in \delta^+(u)$ and $\hat{c}_e := c_e$ otherwise.

For each $i > 1$, every basic optimal primal solution to $f(\bar{y}^{(i)}_{u'})$ is a path flow from u to u' in $G^{(k)}_u$. This is because u cannot reach a flow-absorbing cycle in the subgraph of $G^{(k)}_u$ induced by tight arcs with respect to $\bar{y}^{(i)}_u$. Indeed, such a cycle would impose an upper bound of $\bar{y}^{(i)}_u$ on the variable y_u. As $\bar{y}^{(i-1)}_u > \bar{y}^{(i)}_u$, this contradicts the feasibility of $\bar{y}^{(i-1)}_V$ to the dual LP for $f(\bar{y}^{(i-1)}_{u'})$. For each $i > 1$, let $P^{(i)}$ be a u-u' path with the smallest gain factor in the subgraph of $G^{(k)}_u$ induced by tight arcs with respect to $\bar{y}^{(i)}$. Note that $P^{(i)}$ is well-defined due to the same reason as above. Then, $\gamma(P^{(i)}) - 1 = \min \partial f(\bar{y}^{(i)}_{u'})$. Denote this sequence of u-u' paths as $\mathcal{P} := (P^{(2)}, P^{(3)}, \dots, P^{(\ell)})$.

Without loss of generality, we may assume that $\bar{y}^{(i)}$ is finite for all $i \geq 1$. Since every vertex can reach a flow-absorbing cycle in $G^{(k)}$, there exists a pointwise maximal solution $y^* \in \mathbb{R}^n$ to the modified system $(G^{(k)}, \hat{c}, \gamma)$. Define the reduced cost $c^* \in \mathbb{R}^{m_k}_+$ as $c^*_{vw} := \hat{c}_{vw} + \gamma_{vw} y^*_w - y^*_v$

for all $vw \in E(G^{(k)})$. Since $f(y_u^*) = -\varepsilon$, we obtain

$$\begin{aligned} c^*(P^{(i)}) &= c(P^{(i)}) - (1-\gamma(P^{(i)}))y_u^* + \varepsilon \\ &= f(\bar{y}_{u'}^{(i)}) - (1-\gamma(P^{(i)}))(y_u^* - \bar{y}_u^{(i)}) - f(y_u^*) \\ &= D_f(y_u^*, \bar{y}_{u'}^{(i)}) \le \frac{1}{2}D_f(y_u^*, \bar{y}_{u'}^{(i-2)}) = \frac{1}{2}c^*(P^{(i-2)}) \end{aligned}$$

for all $i > 3$, where the inequality is due to Lemma 2.4.

Consider the vector $x \in \mathbb{R}_+^{m_k}$ defined by

$$x_{vw} := \begin{cases} \max_{i\in[\ell]} \left\{\gamma(P_{uv}^{(i)}) : vw \in E(P^{(i)})\right\} & \text{if } vw \in \cup_{i=1}^{\ell} E(P^{(i)}), \\ 0 & \text{otherwise.} \end{cases}$$

By Lemma 2.20, the sequence $\mathcal{P}$ satisfies subpath monotonicity at u. Hence, x_{vw} is equal to the gain factor of the u-v subpath of the last path in $\mathcal{P}$ that contains vw. Let $0 \le c_1^*x_1 \le c_2^*x_2 \le \cdots \le c_{m_k}^*x_{m_k}$ be the elements of $c^* \circ x$ in nondecreasing order. Let $e_1, e_2, \dots, e_{m_k}$ denote the arcs in $G^{(k)}$ according to this order, and define $d_i := \sum_{j=1}^{i} c_j^*x_j$ for every $i \in [m_k]$. Then, $c^*(P^{(i)}) \in [d_1, d_{m_k}]$ for all $i \in [\ell]$ because $c^*(P^{(\ell)}) \ge d_1$ and $c^*(P^{(1)}) \le d_{m_k}$. To prove that $\ell = O(m_k)$, it suffices to show that every interval $(d_i, d_{i+1}]$ contains the cost of at most two paths from $\mathcal{P}$.

Pick $j < m_k$. Among all the paths in $\mathcal{P}$ whose costs lie in $(d_j, d_{j+1}]$, let $P^{(i)}$ be the most expensive one. If $d_j \ge d_{j+1}/2$, then

$$c^*(P^{(i+2)}) \le \frac{1}{2}c^*(P^{(i)}) \le \frac{1}{2}d_{j+1} \le d_j.$$

On the other hand, if $d_j < d_{j+1}/2$, then

$$c^*(P^{(i+2)}) \le \frac{1}{2}c^*(P^{(i)}) \le \frac{1}{2}d_{j+1} = d_{j+1} - \frac{1}{2}d_{j+1} = c_{j+1}^*x_{j+1} + d_j - \frac{1}{2}d_{j+1} < c_{j+1}^*x_{j+1}.$$

By subpath monotonicity, the paths from $P^{(i+2)}$ onwards do not contain an arc from the set $\{e_{j+1}, e_{j+2}, \dots, e_{m_k}\}$. Therefore, their costs are at most d_j each. □

The runtime of every iteration of Algorithm 1 is dominated by Grapevine. Thus, following the discussion in Section A.3, we obtain the following result.

Corollary 2.22. *Algorithm 3 solves the feasibility of M2VPI linear systems in $O(m^2n^2)$ time.*

One might wonder if Algorithm 3 is still strongly polynomial if we replace the look-ahead Newton–Dinkelbach method on Line 14 with the standard version. In Section 2.6, we show that this is indeed the case, though with a slower convergence.

2.4.3 Deterministic Markov Decision Processes

In this subsection, we replace Grapevine with a variant of Dijkstra's algorithm (Algorithm 4) in order to speed up Algorithm 3 for solving a special class of 2VPI linear programs, known as deterministic Markov decision processes (DMDPs). This idea was briefly mentioned by Madani in [103]; we will supply the details. Recall that an instance of DMDP is described by a directed multigraph $G = (V, E)$ with arc costs $c \in \mathbb{R}^m$ and *discount* factors $\gamma \in (0, 1]^m$. The goal is to select an outgoing arc from every node so as to minimize the total discounted cost over an infinite time horizon. It can be formulated as the following pair of primal and dual LPs.

$$\begin{aligned} \min\ & c^\top x & & \text{(P)} \\ \text{s.t.}\ & \nabla x_v = 1 & \forall v \in V & \\ & x \geq \mathbb{0} & & \end{aligned} \qquad \begin{aligned} \max\ & \mathbb{1}^\top y & & \text{(D)} \\ \text{s.t.}\ & y_v - \gamma_e y_w \leq c_e & \forall e = (v, w) \in E & \end{aligned}$$

Since the discount factor of every cycle is at most 1, there are no bicycles in G. Consequently, by Theorem 2.13, the linear program (D) is infeasible if and only if there is a negative unit-gain cycle in G. This condition can be easily checked by running a negative cycle detection algorithm on the subgraph induced by arcs with discount factor 1.

Algorithm 4 is slightly modified from the standard Dijkstra's algorithm [47] to handle our notion of shortest paths that depends on node labels. As part of the input, it requires a target node t with out-degree zero, node labels $y \in \mathbb{R}^n$ which induce nonnegative reduced costs, and a parameter $\alpha < y_t$. As output, it returns a shortest path tree T to t when y_t is decreased to α. It also returns node labels $z \in \mathbb{R}^n$ which certify the optimality of T, i.e. z induces nonnegative reduced costs with zero reduced costs on T, and $z_t = \alpha$.

An *iteration* of Algorithm 4 refers to a repetition of the while loop. In the pseudocode, observe that $\bar{c}_e \geq 0$ for all $e \in E \setminus \delta^-(u)$.

Lemma 2.23. *Algorithm 4 is correct.*

Proof. We proceed by induction on the number of elapsed iterations k. Let z be the node labels at the end of iteration k. For each $i \leq k$, let v_i be the node added to S in iteration i. Note that z_S remains unchanged in future iterations. We first show that $z_{v_2} \leq z_{v_3} \leq \cdots \leq z_{v_k} < z_{v_1} = 0$. The base case $k = 1$ is true due to our initialization, while the base case $k = 2$ is true because $v_2 \in R$. For the inductive step, suppose that the claim is true for some $k \geq 2$. Let $v_{k+1} = \arg\min_{v \in R} \{z_v\}$ and $v_j = \mathrm{pred}(v_{k+1})$ for some $j \leq k$. We know that $z_{v_{k+1}} < 0$ because $v_{k+1} \in R$. If $j < k$, then $z_{v_{k+1}} \geq z_{v_k}$, as otherwise v_k would not have been chosen to enter S in iteration k. If $j = k$, using the fact that $\gamma_{v_{k+1}v_k} \leq 1$ and $\bar{c}_{v_{k+1}v_k} \geq 0$, we obtain

$$z_{v_{k+1}} = \bar{c}_{v_{k+1}v_k} + \gamma_{v_{k+1}v_k} z_{v_k} \geq z_{v_k}.$$

Algorithm 4: Recompute shortest paths to t

```
input  : A directed multigraph $G = (V, E)$ with arc costs $c \in \mathbb{R}^E$ and discount
         factors $\gamma \in (0,1]^E$, a target node $t \in V$ where $\delta^+(t) = \emptyset$, node labels $y \in \mathbb{R}^V$
         such that $c_{vw} + \gamma_{vw} y_w - y_v \geq 0$ for every $vw \in E$, and a parameter $\alpha < y_t$
output : An in-tree $T$ rooted at $t$ and node labels $z \in \mathbb{R}^V$ such that $z \leq y$, $z_u = \alpha$
         and $c_{vw} + \gamma_{vw} z_w - z_v \geq 0$ for every $vw \in E$, with equality on every arc of $T$.

1  $y_u \leftarrow \alpha$
2  Define reduced cost $\bar{c} \in \mathbb{R}^E$ by $\bar{c}_{vw} \leftarrow c_{vw} + \gamma_{vw} y_w - y_v$ for all $vw \in E$
3  Initialize node labels $z \in \mathbb{R}^V$ by $z_v \leftarrow 0$ for all $v \in V$
4  Initialize sets $R \leftarrow \{t\}$ and $S \leftarrow \emptyset$
5  while $R \neq \emptyset$ do
6      $w \leftarrow \arg\min_{v \in R} \{z_v\}$
7      $R \leftarrow R \setminus \{w\}$
8      $S \leftarrow S \cup \{w\}$
9      foreach $vw \in E$ where $v \notin S$ do
10         if $z_v > \bar{c}_{vw} + \gamma_{vw} z_w$ then
11             $z_v \leftarrow \bar{c}_{vw} + \gamma_{vw} z_w$
12             pred$(v) \leftarrow vw$
13             $R \leftarrow R \cup \{v\}$
14 Let $T$ be the in-tree defined by pred()
15 $z \leftarrow y + z$
16 return $(z, T)$
```

It is left to show that $\bar{c}_{vw} + \gamma_{vw} z_w - z_v \geq 0$ for all $vw \in E(G[S])$. The base case $k = 1$ is trivially true. For the inductive step, suppose that the statement is true for some $k \geq 1$. We know that $z_{v_{k+1}} \leq \bar{c}_{v_{k+1}v} + \gamma_{v_{k+1}v} z_v$ for every outgoing arc $v_{k+1}v \in E(G[S])$. For every incoming arc $vv_{k+1} \in E(G[S])$, using the fact that $\gamma_{vv_{k+1}} \leq 1$ and $\bar{c}_{vv_{k+1}} \geq 0$, we get

$$z_v \leq \bar{c}_{vv_{k+1}} + \gamma_{vv_{k+1}} z_v \leq \bar{c}_{vv_{k+1}} + \gamma_{vv_{k+1}} z_{v_{k+1}},$$

where the second inequality follows from $z_v \leq z_{v_{k+1}}$. □

In every phase k of Algorithm 3, Algorithm 4 now replaces Grapevine as the new value and supergradient oracle of f. Given an optimal dual solution y to $f(\alpha)$ for some $\alpha \in \mathbb{R}$, Algorithm 4 is used to compute an optimal dual solution to $f(\alpha')$ for any $\alpha' < \alpha$. In particular, we run it on the modified graph $G_u^{(k)}$ with input node labels $\bar{y}$ defined by $\bar{y}_{u'} := \alpha$ and $\bar{y}_v := y_v$ for all $v \neq u'$, target node $t = u'$, and parameter $\alpha' < \alpha$. Note that u' has out-degree zero in $G_u^{(k)}$ by construction. Let $(\bar{z}, T)$ be the node labels and tree returned by Algorithm 4, where $\bar{z}_V$ is an optimal dual solution to $f(\alpha')$. A supergradient at $f(\alpha')$ can be inferred from the output via complementary slackness. Specifically, if $u \in V(T)$, then $\gamma(P) - 1 \in \partial f(\alpha')$ where P is the unique u-u' path in T. Otherwise, u can reach a flow-absorbing cycle in the tight subgraph with respect to $\bar{z}$, so $-1 \in \partial f(\alpha')$.

An efficient implementation of Dijkstra's algorithm using Fibonacci heaps was given by Fredman and Tarjan [66]. It can also be applied to our setting, with the same running time of $O(m + n \log n)$. Consequently, we obtain a faster running time of Algorithm 3 for DMDPs.

Corollary 2.24. *Algorithm 3 solves deterministic MDPs in $O(mn(m + n \log n))$ time.*

2.5 Parametric Submodular Function Minimization

Let V be a set with n elements and define $2^V := \{S : S \subseteq V\}$ to be the set of all subsets of V. A function $h : 2^V \to \mathbb{R}$ is submodular if

$$h(S) + h(T) \geq h(S \cap T) + h(S \cup T) \quad \forall S, T \subseteq V\,.$$

Given a non-negative submodular function $h : 2^V \to \mathbb{R}_+$ and a vector $a \in \mathbb{R}^V$ satisfying $\max_{i \in V} a_i > 0$, we examine the problem of computing

$$\delta^* := \max\{\delta : \min_{S \subseteq V} h(S) - \delta a(S) \geq 0\}, \tag{2.3}$$

where $a(S) := \sum_{i \in S} a_i$. As the input model, we assume access to an evaluation oracle for h, which allows us to query $h(S)$ for any set $S \subseteq V$. The above problem models the line-search problem inside a submodular polyhedron and has been studied in [71, 112, 147].

To connect to the root finding problem studied in previous sections, for $\delta \in \mathbb{R}$, we define

$$f(\delta) := \min_{S \subseteq V} h_\delta(S) := \min_{S \subseteq V} h(S) - \delta a(S).$$

Since f is the minimum of 2^n affine functions, f is a piecewise linear concave function. Noting that f is continuous, problem (2.3) can be equivalently restated as that of computing the largest root of f, i.e., the largest $\delta^* \in \mathbb{R}$ such that $f(\delta^*) = 0$. The assumption that h is non-negative ensures that $f(0) \geq 0$, and the assumption that $\max_{i \in V} a_i > 0$ ensures that δ^* exists and $\delta^* \geq 0$ (see the initialization section below). Given the root finding representation, we may apply the Newton–Dinkelbach method on f to compute δ^*. This approach was taken by Goemans, Gupta and Jaillet [71], who were motivated to give a more efficient alternative to the parametric search based algorithm of Nagano [112]. Their main result is as follows:

Theorem 2.25 ([71])**.** *The Newton-Dinkelbach method requires at most $n^2 + O(n \log^2 n)$ iterations to solve (2.3).*

The goal of this section is to give a simplified potential function based proof of the above theorem using the *accelerated* Newton–Dinkelbach method (Algorithm 1), where we will give a slightly weaker $2n^2 + 2n + 4$ bound on the iteration count. Our analysis uses the same combinatorial ring family analysis as in [71], but the Bregman divergence enables considerable simplifications.

2.5.1 Implementing the Accelerated Newton–Dinkelbach

We explain how to implement and initialize the accelerated Newton–Dinkelbach method in the present context. To begin, Algorithm 1 requires access to the supergradients of f. For $\delta \in \mathbb{R}$, it is easy to verify that

$$S \in \text{argmin}\{h_\delta(T) : T \subseteq V\} \Rightarrow -a(S) \in \partial f(\delta).$$

Therefore, computing supergradients of f can be reduced to computing minimizers of the submodular functions $h_\delta(S) := h(S) - \delta a(S)$, $\delta \in \mathbb{R}$. Submodular function minimization (SFM) is a classic problem in combinatorial optimization and has been extensively studied from the viewpoint of strongly polynomial algorithms [42, 83, 84, 99, 85]. The fastest strongly polynomial running time is due to Jiang [85] who gave an algorithm for SFM using $O(n^3)$ calls to the evaluation oracle.

In what follows, we assume access to an SFM oracle, that we will call on the submodular functions h_δ, for $\delta \in \mathbb{R}$. Each iteration of Algorithm 1 requires two calls to a supergradient oracle, one for the standard step and one for the look-ahead step, and hence can be implemented using two calls to the SFM oracle. Gupta, Goemans and Jaillet [71] were directly concerned with the number of calls to an SFM oracle, which is exactly equal to the number of iterations of standard Newton–Dinkelbach (it requires only one SFM call per iteration instead of two). As mentioned above, we will prove a $2n^2 + 2n + 4$ bound on the iteration count for accelerated Newton–Dinkelbach, which will recover the bound on the number of SFM calls of [71] up to a factor 4. Since accelerated Newton–Dinkelbach is always as fast as the standard method (it goes at least as far in each iteration), the iteration bound in Theorem 2.25 in fact applies to the accelerated method as well.

We now explain how to initialize the method. For this purpose, Algorithm 1 requires $\delta^{(1)} \in \mathbb{R}$ and $g^{(1)} \in \partial f(\delta^{(1)})$ such that $f(\delta^{(1)}) \leq 0$ and $g^{(1)} < 0$. We proceed as in [71] and let $\delta^{(1)} := \min\{h(\{i\})/a_i : i \in V, a_i > 0\} \geq 0$, which is well-defined by assumption on a. We compute $f(\delta^{(1)})$ by the SFM oracle. Note that

$$f(\delta^{(1)}) = \min_{S \subseteq V} h_\delta(S) \leq \min_{i \in V, a_i > 0} h(\{i\}) - \delta^{(1)} a_i = 0.$$

If $f(\delta^{(1)}) = 0$, we return $\delta^{(1)}$, as we are already done. Otherwise if $f(\delta^{(1)}) < 0$, set $g^{(1)} = -a(S^{(1)})$, where $S^{(1)} \in \text{argmin}_{S \subseteq V} h_{\delta^{(1)}}(S)$ as returned by the oracle. From here, note that

$$0 > f(\delta^{(1)}) = h_{\delta^{(1)}}(S^{(1)}) = h(S^{(1)}) - \delta^{(1)} a(S^{(1)}) = h(S^{(1)}) + g^{(1)}\delta^{(1)} \geq g^{(1)}\delta^{(1)},$$

where the last inequality follows by non-negativity of h. Since $\delta^{(1)} \geq 0$, the above implies that $\delta^{(1)} > 0$ and $g^{(1)} < 0$. We may therefore initialize Algorithm 1 with $\delta^{(1)}$ and $g^{(1)}$.

Assuming $f(\delta^{(1)}) < 0$, the largest root δ^* of f is guaranteed to exist in the interval $[0, \delta^{(1)})$. This follows since f is continuous, $f(0) = \min_{S \subseteq V} h(S) \geq 0$ (by non-negativity of h) and $f(\delta^{(1)}) < 0$. Note that there does not exist a root larger than $\delta^{(1)}$ due to the concavity of f. So, Algorithm 1 on input $f, \delta^{(1)}, g^{(1)}$ is guaranteed to output the desired largest root δ^* in a finite number of iterations (recalling that f is piecewise affine with 2^n pieces). In the next subsection, we prove a $2n^2 + 2n + 4$ bound on the number of iterations.

2.5.2 Proof of the $2n^2 + 2n + 4$ Iteration Bound

Let $\delta^{(1)} > \cdots > \delta^{(\ell)} = \delta^*$ denote the iterates of Algorithm 1 on input f and $\delta^{(1)} > 0, g^{(1)} < 0$ as above. For each $i \in [\ell]$, let $S^{(i)}$ be any set satisfying

$$S^{(i)} \in \operatorname{argmax}\{a(S) : S \in \operatorname{argmin}_{T \subseteq V} h_{\delta^{(i)}}(T)\}.$$

It is not hard to verify that $S^{(i)}$, $i \in [\ell]$, is a minimizer of $h_{\delta^{(i)}}$ inducing the right derivative of f at $\delta^{(i)}$. Precisely, $-a(S^{(i)}) = \inf_{g \in \partial f(\delta^{(i)})} g$, $\forall i \in [\ell]$. We note that the sets $S^{(i)}$, $i \in [\ell]$, need not be the sets outputted by the SFM oracle, and are only required for the analysis of the algorithm.

Our goal is to prove that $\ell \leq 2n^2 + 2n + 4$. For this purpose, we rely on the key idea of [71], which is to extract an increasing sequence of *ring-families* from the sets $S^{(i)}$, $i \in [\ell]$.

A *ring family* $\mathcal{R} \subseteq 2^V$ is a subsystem of sets that is closed under unions and intersections, precisely $A, B \in \mathcal{R} \Rightarrow A \cap B, A \cup B \in \mathcal{R}$. Given $\mathcal{T} \subseteq 2^V$, we let $\mathcal{R}(\mathcal{T})$ denote the smallest ring-family containing $\mathcal{T}$. We will use the following lemma of [71] which bounds the length of an increasing sequence of ring-families:

Lemma 2.26 ([71, Theorem 2]). *Let $\emptyset \neq \mathcal{R}_1 \subsetneq \mathcal{R}_2 \subsetneq \cdots \subsetneq \mathcal{R}_k \subseteq 2^V$, where $|V| = n$. Then $k \leq \binom{n+1}{2} + 1$.*

The proof of the above lemma is based on the Birkhoff representation of a ring family. Precisely, for any ring-family $\mathcal{R} \subseteq 2^V$, with $\emptyset, V \in \mathcal{R}$, there exists a directed graph G on V, such that the sets $S \in \mathcal{R}$ are exactly the subsets of vertices of G having no out-neighbors. The main idea for the bound is that the digraph representation of $\mathcal{R}_i, i \in [k]$, must lose edges as i increases. The next statement is a slightly adapted version of [71, Theorem 5] that is sufficient for our purposes. It shows that a sequence of sets with geometrically increasing h values forms an increasing sequence of ring families. We include a proof for completeness.

Lemma 2.27. *Let $h : 2^V \to \mathbb{R}_+$ be a non-negative submodular function. Consider a sequence of distinct sets $T_1, T_2, \dots, T_q \subseteq V$ such that $h(T_{i+1}) > 4h(T_i)$ for $i \in [q-1]$. Then $T_{i+1} \notin \mathcal{R}(\{T_1, \dots, T_i\})$ for all $i \in [q-1]$.*

Proof. Let $\mathcal{R}_i := \mathcal{R}(\{T_1, \dots, T_i\})$, $\forall i \in [q]$. We claim that $\max_{S \in \mathcal{R}_i} h(S) \le 2h(T_i)$, $\forall i \in [q]$. This proves $T_{i+1} \notin \mathcal{R}_i$, for $i \in [q-1]$, since $h(T_{i+1}) > 4h(T_i) \ge 2h(T_i) \ge \max_{S \in \mathcal{R}_i} h(S)$, noting that the second inequality uses that h is non-negative.

We now prove the claim by induction on $i \in [q]$. The base case $i = 1$ is trivial since $\mathcal{R}_1 = \{T_1\}$. We now assume that $\max_{S \in \mathcal{R}_i} h(S) \le 2h(T_i)$, for $1 \le i \le q-1$, and prove the corresponding bound for $\mathcal{R}_{i+1}$. Recalling that $\mathcal{R}_{i+1}$ is the ring-family generated by $\mathcal{R}_i$ and T_{i+1}, it is easy to verify that the set system

$$\mathcal{R}_i \cup \{T_{i+1}\} \cup \{S \cup T_{i+1} : S \in \mathcal{R}_i\} \cup \{S \cap T_{i+1} : S \in \mathcal{R}_i\} \cup \{S_1 \cup (S_2 \cap T_{i+1}) : S_1, S_2 \in \mathcal{R}_i\}$$

is a ring-family and hence is equal to $\mathcal{R}_{i+1}$. It therefore suffices to upper bound $h(X)$ for a set X of the above type. For $X \in \mathcal{R}_i$ or $X = T_{i+1}$, the bound is by assumption. For $X = S_1 \cup (S_2 \cap T_{i+1})$, $S_1, S_2 \in R_{i+1}$, we prove the bound as follows:

$$\begin{aligned}
h(S_1 \cup (S_2 \cap T_{i+1})) &\le h(S_1) + h(S_2 \cap T_{i+1}) - h(S_1 \cap S_2 \cap T_{i+1}) && \text{(by submodularity of } h\text{)} \\
&\le h(S_1) + h(S_2) + h(T_{i+1}) - h(S_2 \cup T_{i+1}) - h(S_1 \cap S_2 \cap T_{i+1}) \\
&\le h(S_1) + h(S_2) + h(T_{i+1}) && \text{(by non-negativity of } h\text{)} \\
&\le 4h(T_i) + h(T_{i+1}) && \text{(by the induction hypothesis)} \\
&\le 2h(T_{i+1}). && \text{(since } 4h(T_i) < h(T_{i+1})\text{)}
\end{aligned}$$

For $X = S \cup T_{i+1}$ or $X = S \cap T_{i+1}$, $S \in \mathcal{R}_i$, similarly to the above, one has

$$h(X) \le h(S) + h(T_{i+1}) \le 2h(T_i) + h(T_{i+1}) \le \frac{3}{2} h(T_{i+1}),$$

as required. □

We now use the Bregman-divergence analysis to show that for the function h_{δ^*}, the sequence of sets $T_i = S^{(\ell - 4(i-1))}$, $1 \le i \le \lfloor \frac{\ell+3}{4} \rfloor$ satisifes the conditions of this lemma. Combined with Lemma 2.26, we get that the number of iterations satisfies

$$\lfloor (\ell+3)/4 \rfloor \le \binom{n+1}{2} + 1 \Rightarrow \ell \le 2n^2 + 2n + 4, \text{ as needed.}$$

Lemma 2.28. *Let us define*

$$T_i := S^{(\ell - 4(i-1))}, \quad i \in [q] \quad \textit{for } q := \left\lfloor \frac{\ell+3}{4} \right\rfloor .$$

Then, the function h_{δ^*} and the sequence of sets $T_1, T_2, \ldots, T_q$ satisfy the conditions in Lemma 2.27.

Proof. The function h_{δ^*} is clearly submodular, and its minimum is 0 since $0 = f(\delta^*) = \min_{S \subseteq V} h_{\delta^*}(S) = h_{\delta^*}(S^{(\ell)}) = h_{\delta^*}(T_1)$. In particular, h_{δ^*} is non-negative. It is left to show $h_{\delta^*}(T_{i+1}) > 4h_{\delta^*}(T_i)$ for $i \in [q-1]$. For each $\delta^{(i)}$, $i \in [\ell]$, we see that

$$\begin{aligned} D_f(\delta^*, \delta^{(i)}) &= f(\delta^{(i)}) + \sup_{g \in \partial f(\delta^{(i)})} g(\delta^* - \delta^{(i)}) - f(\delta^*) \\ &= h_{\delta^{(i)}}(S^{(i)}) - a(S^{(i)})(\delta^* - \delta^{(i)}) \qquad \text{(by our choice of } S^{(i)} \text{ and } f(\delta^*) = 0) \\ &= h(S^{(i)}) - \delta^{(i)} a(S^{(i)}) - a(S^{(i)})(\delta^* - \delta^{(i)}) = h_{\delta^*}(S^{(i)}). \end{aligned}$$

By Lemma 2.4 and the above, we get for $3 \le i \le \ell$ that

$$D_f(\delta^*, \delta^{(i)}) < \frac{1}{2} D_f(\delta^*, \delta^{(i-2)}) \Leftrightarrow h_{\delta^*}(S^{(i)}) < \frac{1}{2} h_{\delta^*}(S^{(i-2)}). \tag{2.4}$$

Then, $h_{\delta^*}(T_{i+1}) > 4h_{\delta^*}(T_i)$ for $i \in [q-1]$ follows by the definition of the T_i sets. □

2.6 2VPI Analysis without Acceleration

In this section, we analyze the convergence of Algorithm 3 when the look-ahead Newton–Dinkelbach method is replaced with the standard version. Interestingly, we also obtain a strongly polynomial runtime in this case, albeit slower than the accelerated version by a factor of $O(\log n)$. To achieve the desired runtime, we slightly strengthen Lemma 2.6, whose proof remains largely the same.

Lemma 2.29. *Let $c \in \mathbb{R}^m_+$ and $x^{(1)}, x^{(2)}, \ldots, x^{(k)} \in \mathbb{Z}^m$ such that $\left\| x^{(i)} \right\|_1 \le n$ for all $i \in [k]$. If*

$$0 < c^\top x^{(i+1)} \le \frac{1}{2} c^\top x^{(i)}$$

for all $i < k$, then $k = O(m \log n)$.

Proof of Lemma 2.29. Consider the polyhedron $P \subseteq \mathbb{R}^m$ defined by the following constraints:

$$\begin{aligned} (x^{(i)} - 2x^{(i+1)})^\top z &\ge 0 \qquad \forall i < k \\ (x^{(k)})^\top z &= 1 \\ z &\ge \mathbb{0}. \end{aligned}$$

Let $A \in \mathbb{R}^{(k+m) \times m}$ and $b \in \mathbb{R}^{k+m}$ denote the coefficient matrix and right-hand side vector of this system. The polyhedron P is nonempty because it contains the vector $c/(x^{(k)})^\top c$. Moreover, since P does not contain a line, it has an extreme point. So there exists a vector

$c' \in P$ such that $A'c' = b'$ for some nonsingular submatrix $A' \in \mathbb{R}^{m \times m}$ of the matrix A and a subvector $b' \in \mathbb{R}^m$ of the vector b. Cramer's rule says that for each $i \in [m]$,

$$c'_i = \frac{\det A'_i}{\det A'}$$

where the matrix A'_i is obtained from matrix A' by replacing the i-th column with vector b'. The 1-norm of the rows of A'_i is bounded by $3n$ and so by Hadamard's inequality $|\det(A'_i)| \leq (3n)^m$.

As the matrix A' is nonsingular, we also have $|\det A'| \geq 1$, which implies that $c'_i \leq (3n)^m$ for all $i \in [m]$. Finally, using the constraints which define the polyhedron P, we obtain

$$1 = (x^{(k)})^\top c' \leq \frac{(x^{(1)})^\top c'}{2^{k-1}} \leq \frac{n(3n)^m}{2^{k-1}}.$$

So, $k \leq \log(3^m n^{m+1}) + 1 = O(m \log n)$ as desired. □

Fix a phase $k \in [n]$ and denote $m_k = |E(G^{(k)})|$. It is helpful to classify the iterations of the Newton–Dinkelbach method based on the magnitude by which the supergradient changes. Recall that the supergradient at the start of iteration $i > 1$ is given by $\gamma(P^{(i)}) - 1$, where $P^{(i)}$ is the u-u' path returned by GRAPEVINE in the previous iteration.

Definition 2.30. For every $i > 1$, we say that iteration i is *good* if $1 - \gamma(P^{(i)}) \leq \frac{1}{2}(1 - \gamma(P^{(i-1)}))$. Otherwise, we say that it is *bad*.

The next lemma gives a strongly polynomial bound on the number of good iterations.

Lemma 2.31. *In each phase $k \in [n]$, the number of good iterations is $O(m_k \log k)$.*

Proof. Let $\mathcal{P}$ be a sequence of u-u' paths in $G_u^{(k)}$ at the start of every iteration of the Newton–Dinkelbach method. Let $\mathcal{P}^* = (P^{(1)}, P^{(2)}, \dots, P^{(t)})$ be the subsequence of $\mathcal{P}$ restricted to good iterations. We claim that $\gamma(P^{(i+1)}) \geq \sqrt{\gamma(P^{(i)})}$ for all $i < t$. We use the simple inequality that $(1 - x)/2 \leq 1 - \sqrt{x}$ for all $x \in \mathbb{R}_+$; one can derive this by rearranging $(\sqrt{x} - 1)^2/2 \geq 0$. This gives

$$1 - \gamma(P^{(i+1)}) \leq \frac{1}{2}\left(1 - \gamma(P^{(i)})\right) \leq 1 - \sqrt{\gamma(P^{(i)})},$$

which proves the claim. Next, enumerate the arcs of each path by $P^{(i)} = (e_1^{(i)}, e_2^{(i)}, \dots, e_{\ell_i}^{(i)})$. By taking logarithms, the claim can be equivalently stated as

$$\sum_{j=1}^{\ell_{i+1}} \log \gamma_{e_j^{(i+1)}} \geq \frac{1}{2} \sum_{j=1}^{\ell_i} \log \gamma_{e_j^{(i)}}.$$

Note that both sides of the expression above are negative because $\gamma(P^{(i)}) < 1$ for all $i \in [t]$. Let $c \in \mathbb{R}_+^{m_k}$ be the vector defined by $c_e = |\log \gamma_e|$ for all $e \in E(G^{(k)})$. In addition, for every

$i \in [t]$, define the vector $x^{(i)} \in \mathbb{Z}^m$ as

$$x_e^{(i)} = -\operatorname{sgn}(\log \gamma_e) \left| \left\{ j \in [\ell_i] : e_j^{(i)} = e \right\} \right| .$$

Then, we obtain

$$0 < c^\top x^{(i+1)} = \sum_{j=1}^{\ell_{i+1}} -\log \gamma_{e_j^{(i+1)}} \leq \frac{1}{2} \sum_{j=1}^{\ell_i} -\log \gamma_{e_j^{(i)}} = \frac{1}{2} c^\top x^{(i)}.$$

for all $i < t$. Since $\|x^{(i)}\|_1 \leq k$ for all $i \in [t]$, we conclude that $t = O(m_k \log k)$ by Lemma 2.29. □

It is left to bound the number of bad iterations. We approach this by arguing that in a strongly polynomial number of bad iterations, an arc will no longer appear in future paths produced by the Newton–Dinkelbach method.

Lemma 2.32. *In each phase $k \in [n]$, the number of bad iterations is $O(m_k \log k)$.*

Proof. Let $\bar{\mathcal{Y}} = (\bar{y}^{(1)}, \bar{y}^{(2)}, \dots, \bar{y}^{(\ell)})$ and $\mathcal{P} = (P^{(1)}, P^{(2)}, \dots, P^{(\ell)})$ be a sequence of node labels and u-u' paths in $G_u^{(k)}$ respectively at the start of every iteration of the Newton–Dinkelbach method. Without loss of generality, we may assume that $\bar{y}^{(i)}$ is finite for all $i \in [\ell]$. For each $i \in [\ell]$, define $y^{(i)} \in \mathbb{R}^n$ as $y_u^{(i)} := \bar{y}_{u'}^{(i)}$ and $y_v^{(i)} := \bar{y}_v^{(i)}$ for all $v \notin \{u, u'\}$. Now, pick an iteration $j \in [\ell]$ such that more than $\log(2n)$ bad iterations have elapsed. Consider the reduced cost $c' \in \mathbb{R}^{m_k}$ given by $c'_{vw} := c_{vw} + \gamma_{vw} y_w^{(j)} - y_v^{(j)}$ for all $vw \in E(G^{(k)})$. Note that $c'_{vw} \geq 0$ for all $v \neq u$.

According to Lemma 2.17, each $P^{(i)}$ is a shortest u-u' path with respect to $\bar{y}^{(i)}$. By complementary slackness, the unit flow on $P^{(i)}$ is an optimal primal solution to $f(\bar{y}_{u'}^{(i)})$. Since $\bar{y}_{u'}^{(i)} > \bar{y}_{u'}^{(i+1)}$ for all $i < \ell$, the sequence $\mathcal{P}$ satisfies subpath monotonicity at u by Lemma 2.20. Define the vector $x \in \mathbb{R}_+^m$ as

$$x_{vw} := \begin{cases} \max_{i \in [\ell]} \left\{ \gamma(P_{uv}^{(i)}) : vw \in E(P^{(i)}) \right\} & \text{if } vw \in \cup_{i=1}^{\ell} E(P^{(i)}), \\ 0 & \text{otherwise.} \end{cases}$$

Observe that x_{vw} is the gain factor of the u-v subpath of the last path in $\mathcal{P}$ which contains vw, due to subpath monotonicity.

Claim 2.33. *We have $-f(\bar{y}_{u'}^{(j)}) < \|c' \circ x\|_\infty$.*

Proof. For every $i \in [\ell]$, we have

$$f(\bar{y}_{u'}^{(i)}) = c(P^{(i)}) - \bar{y}_{u'}^{(i)}(1 - \gamma(P^{(i)})) = c'(P^{(i)}) - (\bar{y}_{u'}^{(i)} - \bar{y}_{u'}^{(j)})(1 - \gamma(P^{(i)})).$$

By applying the definition of $\bar{y}^{(i)}_{u'}$, we can upper bound its negation by

$$-f(\bar{y}^{(i)}_{u'}) = -c'(P^{(i)}) + \frac{1-\gamma(P^{(i)})}{1-\gamma(P^{(i-1)})}c'(P^{(i-1)}) \leq \left|c'(P^{(i)})\right| + \left|c'(P^{(i-1)})\right| \leq 2k \left\|c' \circ x\right\|_\infty .$$

Lemma 2.1 tells us that $-f(\bar{y}^{(i)}_{u'})$ is nonnegative and monotonically decreasing. Moreover, it decreases geometrically by a factor of $1/2$ during bad iterations. Hence, by our choice of j, we obtain

$$-f(\bar{y}^{(j)}_{u'}) < \left(\frac{1}{2}\right)^{\log(2n)} \cdot 2k \left\|c' \circ x\right\|_\infty = \left\|c' \circ x\right\|_\infty . \qquad \square$$

Let $d \in \mathbb{R}^{m_k}$ be the arc costs defined by

$$d_{vw} = \begin{cases} c'_{vw} & \text{if } v \neq u, \\ c'_{vw} - f(\bar{y}^{(j)}_{u'}) & \text{if } v = u. \end{cases}$$

Since $f(\bar{y}^{(j)}_{u'}) = \bar{y}^{(j)}_{u} - \bar{y}^{(j)}_{u'}$, observe that $d \geq 0$ because $\bar{y}^{(j)}_{V}$ is feasible to the dual LP for $f(\bar{y}^{(j)}_{u'})$.

Claim 2.34. *We have* $\left\|d \circ x\right\|_\infty \geq \left\|c' \circ x\right\|_\infty$.

Proof. Let $e^* = \arg\max_{e \in E(G^{(k)})} |c'_e x_e|$. The claim is trivial unless $e^* \in \cup_{i=1}^{\ell} E(P^{(i)})$ and the tail of e^* is u. Since $f(\bar{y}^{(j)}_{u'}) \leq 0$ and $d_{e^*} = c'_{e^*} - f(\bar{y}^{(j)}_{u'})$, it suffices to show that $c'_{e^*} \geq 0$. For the purpose of contradiction, suppose that $c'_{e^*} < 0$. Since $d_{e^*} \geq 0$, this implies that $|c'_{e^*}| \leq -f(\bar{y}^{(j)}_{u'}) < \left\|c' \circ x\right\|_\infty$ using Claim 2.33. By the definition of x, $x_{e^*} = 1$ because e^* is the first arc of any path in $\mathcal{P}$ which uses it. However, this implies that

$$|c'_{e^*}| = |c'_{e^*} x_{e^*}| = \left\|c' \circ x\right\|_\infty ,$$

which is a contradiction. $\square$

Consider the arc $e^* := \arg\max_{e \in E} |d_e x_e|$. We claim that e^* does not appear in subsequent paths in $\mathcal{P}$ after iteration j. For the purpose of contradiction, suppose that there exists an iteration $i > j$ such that $e^* \in E(P^{(i)})$. Pick the iteration i such that $P^{(i)}$ is the last path in $\mathcal{P}$ which contains e^*. Since the iterates $\bar{y}^{(\cdot)}_{u'}$ are monotonically decreasing, we have

$$0 > \bar{y}^{(i+1)}_{u'} - \bar{y}^{(j)}_{u'} = \frac{c(P^{(i)})}{1-\gamma(P^{(i)})} - \bar{y}^{(j)}_{u'} = \frac{c'(P^{(i)})}{1-\gamma(P^{(i)})} = \frac{d(P^{(i)}) - f(\bar{y}^{(j)}_{u'})}{1-\gamma(P^{(i)})}$$

This implies that $d(P^{(i)}) < f(\bar{y}^{(j)}_{u'}) < \left\|c' \circ x\right\|_\infty$. However, it contradicts

$$d(P^{(i)}) \geq d_{e^*} x_{e^*} = \left\|d \circ x\right\|_\infty \geq \left\|c' \circ x\right\|_\infty ,$$

where the first inequality is due to our choice of i and the nonnegativity of d, while the second inequality is due to Claim 2.34. Repeating the argument above for m times yields the desired bound on the number of bad iterations. □

The runtime of every iteration of the Newton–Dinkelbach method is dominated by Grapevine. Thus, following the discussion in Section A.3, we obtain the following result.

Corollary 2.35. *If we replace Algorithm 1 with the Newton–Dinkelbach method in Algorithm 3, then it solves the feasibility of M2VPI linear systems in $O(m^2n^2 \log n)$ time.*

Chapter 3

Linear Optimization: Circuit Diameter Bounds

3.1 Introduction

The *combinatorial diameter* of a polyhedron P is the diameter of the vertex-edge graph associated with P. Hirsch's famous conjecture from 1957 asserted that the combinatorial diameter of a d-dimensional polytope (bounded polyhedron) with f facets is at most $f - d$. This was disproved by Santos in 2012 [129]. The *polynomial Hirsch conjecture*, i.e., finding a poly(f) bound on the combinatorial diameter remains a central question in the theory of linear programming.

The first quasipolynomial bound was given by Kalai and Kleitman [91, 92], see [140] for the best current bound and an overview of the literature. Dyer and Frieze [53] proved the polynomial Hirsch conjecture for totally unimodular (TU) matrices. For a system $\{x \in \mathbb{R}^d : Mx \le b\}$ with integer constraint matrix M, polynomial diameter bounds were given in terms of the maximum subdeterminant Δ_M [16, 23, 57, 36]. These arguments can be strengthened to using a parametrization by a 'discrete curvature measure' $\delta_M \ge 1/(d\Delta_M^2)$. The best such bound was given by Dadush and Hähnle [36] as $O(d^3 \log(d/\delta_M)/\delta_M)$, using a shadow vertex simplex algorithm.

As a natural relaxation of the combinatorial diameter, Borgwardt, Finhold, and Hemmecke [18] initiated the study of *circuit diameters*. Consider a polyhedron in standard equality form

$$P = \{\, x \in \mathbb{R}^n : Ax = b, x \ge \mathbb{0} \,\} \tag{P}$$

for $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$; we assume $\mathrm{rk}(A) = m$. For the linear space $W = \ker(A) \subseteq \mathbb{R}^n$, $g \in W$ is an *elementary vector* if g is a support-minimal nonzero vector in W, that is, no $h \in W \setminus \{\mathbb{0}\}$ exists such that $\mathrm{supp}(h) \subsetneq \mathrm{supp}(g)$. A *circuit* in W is the support of some elementary vector; these are precisely the circuits of the associated linear matroid $\mathcal{M}(A)$.

We let $\mathcal{E}(W) = \mathcal{E}(A) \subseteq W$ and $\mathcal{C}(W) = \mathcal{C}(A) \subseteq 2^n$ denote the set of elementary vectors and circuits in the space $W = \ker(A)$, respectively. All edge directions of P are elementary vectors, and the set of elementary vectors $\mathcal{E}(A)$ equals the set of all possible edge directions of P in the form (P) for varying $b \in \mathbb{R}^m$ [139].

A *circuit walk* is a sequence of points $x^{(1)}, x^{(2)}, \dots, x^{(k+1)}$ in P such that for each $i = 1, \dots, k$, $x^{(i+1)} = x^{(i)} + g^{(i)}$ for some $g^{(i)} \in \mathcal{E}(A)$, and further, $x^{(i)} + (1+\varepsilon)g^{(i)} \notin P$ for any $\varepsilon > 0$, i.e., each consecutive circuit step is *maximal*. The *circuit diameter* of P is the maximum length (number of steps) of a shortest circuit walk between any two vertices $x, y \in P$. Note that, in contrast to walks in the vertex-edge graph, circuit walks are non-reversible and the minimum length from x to y may be different from the one from y to x; this is due to the maximality requirement. The circuit-analogue of Hirsch conjecture, formulated in [18], asserts that the circuit diameter of d-dimensional polyhedron with f facets is at most $f - d$; this may be true even for unbounded polyhedra, see [19]. For P in the form (P), $d = n - m$ and the number of facets is at most n; hence, the conjectured bound is m.

Circuit diameter bounds have been shown for some combinatorial polytopes such as dual transportation polyhedra [18], matching, travelling salesman, and fractional stable set polytopes [90]. The paper [17] introduced several other variants of circuit diameter, and explored the relation between them. We note that [90] considers circuits for LPs given in the general form $\{x \in \mathbb{R}^n : Ax = b, Bx \le d\}$. In Section 3.7, we show that this setting can be reduced to the form (P).

Circuit augmentation algorithms Circuit diameter bounds are inherently related to *circuit augmentation algorithms*. This is a general algorithmic scheme to solve an LP

$$\min \ \langle c, x \rangle \quad \text{s.t.} \quad Ax = b\,,\ x \ge \mathbb{0}\,. \tag{LP}$$

The algorithm proceeds through a sequence of feasible solutions $x^{(t)}$. An initial feasible $x^{(0)}$ is required in the input. For $t = 0, 1, \dots,$ the current $x^{(t)}$ is updated to $x^{(t+1)} = x^{(t)} + \alpha g$ for some $g \in \mathcal{E}(A)$ such that $\langle c, g \rangle \le 0$, and $\alpha > 0$ such that $x^{(t)} + \alpha g$ is feasible. The elementary vector g is an *augmenting direction* if $\langle c, g \rangle < 0$ and such an $\alpha > 0$ exists; by LP duality, $x^{(t)}$ is optimal if and only if no augmenting direction exists. The augmentation is *maximal* if $x^{(t)} + \alpha' g$ is infeasible for any $\alpha' > \alpha$; α is called the maximal stepsize for $x^{(t)}$ and g. Clearly, an upper bound on the number of steps of a circuit augmentation algorithm with maximal augmentations for arbitrary cost c and starting point $x^{(0)}$ yields an upper bound on the circuit diameter.

Simplex is a circuit augmentation algorithm that is restricted to using special elementary vectors corresponding to edges of the polyhedron. Many network optimization algorithms can be seen as special circuit augmentation algorithms. Bland [14] introduced a circuit augmentation algorithm for LP, that generalizes the Edmonds–Karp–Dinic maximum flow algorithm and its analysis, see also [98, Proposition 3.1]. Circuit augmentation algorithms were revisited by De Loera, Hemmecke, and Lee in 2015 [45], analyzing different augmentation rules and also extending them to integer programming. De Loera, Kafer, and Sanità [46] studied the convergence of these rules on 0/1-polytopes, as well as the computational complexity of performing them. We refer the reader to [45] and [46] for a more detailed overview of the background and history of circuit augmentations.

The circuit imbalance measure For a linear space $W = \ker(A) \subseteq \mathbb{R}^n$, the *circuit imbalance* $\kappa_W = \kappa_A$ is defined as the maximum of $|g_j/g_i|$ over all elementary vectors $g \in \mathcal{E}(W)$, $i, j \in \operatorname{supp}(g)$. It can be shown that $\kappa_W = 1$ if and only if W is a unimodular space, i.e., the kernel of a totally unimodular matrix. This parameter and related variants have been used implicitly or explicitly in many areas of linear programming and discrete optimization, see [58] for a recent survey. It is closely related to the Dikin–Stewart–Todd condition number $\bar{\chi}_W$ that plays a key role in layered-least-squares interior point methods introduced by Vavasis and Ye [149]. An LP of the form (LP) for $A \in \mathbb{R}^{m\times n}$ can be solved in time $\operatorname{poly}(n, m, \log \kappa_A)$, which is strongly polynomial if $\kappa_A \le 2^{\operatorname{poly}(n)}$; see [37, 41] for recent developments and references.

Imbalance and diameter The combinatorial diameter bound $O(d^3 \log(d/\delta_M)/\delta_M)$ from [36] mentioned above translates to a bound $O((n-m)^3 m\kappa_A \log(\kappa_A + n))$ for the system in the form (P), see [58]. For circuit diameters, the Goldberg-Tarjan minimum-mean cycle cancelling algorithm for minimum-cost flows [72] naturally extends to a circuit augmentation algorithm for general LPs using the *steepest-descent* rule. This yields a circuit diameter bound $O(n^2 m\kappa_A \log(\kappa_A + n))$ [58], see also [70]. However, note that these bounds may be exponential in the bit-complexity of the input.

3.1.1 Our Contributions

Our first main contribution improves the κ_A dependence to a $\log \kappa_A$ dependence for circuit diameter bounds.

Theorem 3.1. *The circuit diameter of a system in the form* (P) *with constraint matrix* $A \in \mathbb{R}^{m\times n}$ *is* $O(m \min\{m, n-m\} \log(m + \kappa_A))$.

The proof in Section 3.3 is via a simple 'shoot towards the optimum' scheme. We need the well-known concept of *conformal circuit decompositions*. We say that $x, y \in \mathbb{R}^n$ are *sign-compatible* if $x_i y_i \ge 0$ for all $i \in [n]$. We write $x \sqsubseteq y$ if they are sign-compatible and further $|x_i| \le |y_i|$ for all $i \in [n]$. It follows from Carathéodory's theorem and Minkowski–Weyl theorem that for any linear space $W \subseteq \mathbb{R}^n$ and $x \in W$, there exists a decomposition $x = \sum_{j=1}^k h^{(j)}$ such that $h^{(j)} \in \mathcal{E}(W)$, $h^{(j)} \sqsubseteq x$ for all $j \in [k]$ and $k \le n$. This is called a *conformal circuit decomposition* of x.

Let $B \subseteq [n]$ be a feasible basis and $N = [n] \setminus B$, i.e., $x^* = (A_B^{-1}b, \mathbb{0}_N) \geq \mathbb{0}_n$ is a basic feasible solution. This is the unique optimal solution to (LP) for the cost function $c = (\mathbb{0}_B, \mathbb{1}_N)$. Let $x^{(0)} \in P$ be an arbitrary vertex. We may assume that $n \leq 2m$, by restricting to the union of the support of x^* and $x^{(0)}$, and setting all other variables to 0. For the current iterate $x^{(t)}$, let us consider a conformal circuit decomposition $x^* - x^{(t)} = \sum_{j=1}^{k} h^{(j)}$. Note that the existence of such a decomposition *does not* yield a circuit diameter bound of n, due to the maximality requirement in the definition of circuit walks. For each $j \in [k]$, $x^{(t)} + h^{(j)} \in P$, but there might be a larger augmentation $x^{(t)} + \alpha h^{(j)} \in P$ for some $\alpha > 1$.

Still, one can use this decomposition to construct a circuit walk. Let us pick the most improving circuit from the decomposition, i.e., the one maximizing $-\left\langle c, h^{(j)} \right\rangle = \|h_N^{(j)}\|_1$, and obtain $x^{(t+1)} = x^{(t)} + \alpha^{(t)} h^{(j)}$ for the maximum stepsize $\alpha^{(t)} \geq 1$. The proof of Theorem 3.1 is based on analyzing this procedure. The first key observation is that $\left\langle c, x^{(t)} \right\rangle = \|x_N^{(t)}\|_1$ decreases geometrically. Then, we look at the set of indices $L_t = \{i \in [n] : x_i^* > n\kappa_A \|x_N^{(t)}\|_1\}$ and $R_t = \{i \in [n] : x_i^{(t)} \leq (n-m)x_i^*\}$, and show that indices may never leave these sets once they enter. Moreover, a new index is added to either set every $O(m \log(m + \kappa_A))$ iterations. In Section 3.4, we extend this bound to the setting with upper bounds on the variables.

Theorem 3.2. *The circuit diameter of a system in the form $Ax = b$, $\mathbb{0} \leq x \leq u$ with constraint matrix $A \in \mathbb{R}^{m \times n}$ is $O(m \min\{m, n-m\} \log(m + \kappa_A) + (n-m)\log n)$.*

There is a straightforward reduction from the capacitated form to (P) by adding n slack variables; however, this would give an $O(n^2 \log(n + \kappa_A))$ bound. For the stronger bound, we use a preprocessing that involves cancelling circuits in the support of the current solution; this eliminates all but $O(m)$ of the capacity bounds in $O(n \log n)$ iterations, independently of κ_A.

For rational input, $\log(\kappa_A) = O(L_A)$ where L_A denotes the total encoding length of A [37]. Hence, our result yields an $O(m \min\{m, n-m\} L_A + n \log n)$ diameter bound on $Ax = b$, $\mathbb{0} \leq x \leq u$. This can be compared with the bounds $O(nL_{A,b})$ using deepest descent augmentation steps in [45, 46], where $L_{A,b}$ is the encoding length of (A, b). (Such a bound holds for every augmentation rule that decreases the optimality gap geometrically, including the minimum-ratio circuit rule discussed below.) Note that our bound is independent of b. Furthermore, it is also applicable to systems given by irrational inputs, in which case arguments based on subdeterminants and bit-complexity cannot be used.

In light of these results, the next important step towards the polynomial Hirsch conjecture might be to show a poly$(n, \log \kappa_A)$ bound on the combinatorial diameter of (P). Note that—in contrast with the circuit diameter—not even a poly$(n, L_{A,b})$ bound is known. In this context, the best known general bound is $O((n-m)^3 m\kappa_A \log(\kappa_A + n))$ implied by [36].

Circuit augmentation algorithms The diameter bounds in Theorems 3.1 and 3.2 rely on knowing the optimal solution x^*; thus, they do not provide efficient LP algorithms. We next present circuit augmentation algorithms with $\mathrm{poly}(n, m, \log \kappa_A)$ bounds on the number of iterations. Such algorithms require subroutines for finding augmenting circuits. In many cases, such subroutines are LPs themselves. However, they may be of a simpler form, and might be easier to solve in practice. Borgwardt and Viss [20] exhibit an implementation of a steepest-descent circuit augmentation algorithm with encouraging computational results.

We assume that a subroutine Ratio-Circuit(A, c, w) is available; this implements the well-known minimum-ratio circuit rule. It takes as input a matrix $A \in \mathbb{R}^{m\times n}$, $c \in \mathbb{R}^n$, $w \in (\mathbb{R}_+ \cup \{\infty\})^n$, and returns a basic optimal solution to the system

$$\min\ \langle c, z\rangle \quad \text{s.t.} \quad Az = \mathbb{0}\,,\ \langle w, z^-\rangle \le 1\,, \tag{3.1}$$

where $(z^-)_i := \max\{0, -z_i\}$ for $i \in [n]$. We use the convention $w_i z_i = 0$ if $w_i = \infty$ and $z_i = 0$. This system can be equivalently written as an LP using auxiliary variables. If bounded, a basic optimal solution is an elementary vector $z \in \mathcal{E}(A)$ that minimizes $\langle c, z\rangle \,/\, \langle w, z^-\rangle$.

Given $x \in P$, we use weights $w_i = 1/x_i$ (with $w_i = \infty$ if $x_i = 0$). For minimum-cost flow problems, this rule was proposed by Wallacher [154]; such a cycle can be found in strongly polynomial time for flows. The main advantage of this rule is that the optimality gap decreases by a factor $1 - 1/n$ in every iteration. This rule, along with the same convergence property, can be naturally extended to linear programming [104], and has found several combinatorial applications, e.g., [155, 157], and has also been used in the context of integer programming [134].

On the negative side, Wallacher's algorithm is *not* strongly polynomial: it does not terminate finitely for minimum-cost flows, as shown in [104]. In contrast, our algorithms achieve a strongly polynomial running time whenever $\kappa_A \le 2^{\mathrm{poly}(n)}$. An important modification is the occasional use of a second type of circuit augmentation step Support-Circuit that removes circuits in the support of the current (non-basic) iterate $x^{(t)}$ (see Subroutine 3.2.1); this can be implemented using simple linear algebra. Our first result addresses the feasibility setting:

Theorem 3.3. *Consider an LP of the form (LP) with cost function $c = (\mathbb{0}_{[n]\setminus N}, \mathbb{1}_N)$ for some $N \subseteq [n]$. There exists a circuit augmentation algorithm that either finds a solution x such that $x_N = \mathbb{0}$ or a dual certificate that no such solution exists, using $O(mn \log(n + \kappa_A))$ Ratio-Circuit and $(m + 1)n$ Support-Circuit augmentation steps.*

Such problems typically arise in Phase I of the Simplex method when we add auxiliary variables in order to find a feasible solution. The algorithm is presented in Section 3.5. The analysis extends that of Theorem 3.1, tracking large coordinates $x_i^{(t)}$. Our second result considers general optimization:

Theorem 3.4. *Consider an LP of the form (LP). There exists a circuit augmentation algorithm that finds an optimal solution or concludes unboundedness using $O(mn^2 \log(n+\kappa_A))$ Ratio-Circuit and $(m + 1)n^2$ Support-Circuit augmentation steps.*

The proof is given in Section 3.6. The main subroutine identifies a new index $i \in [n]$ such that $x_i^{(t)} = 0$ in the current iteration and $x_i^* = 0$ in an optimal solution; we henceforth fix this variable to 0. To derive this conclusion, at the end of each phase the current iterate $x^{(t)}$ will be optimal to (LP) with a slightly modified cost function $\tilde{c}$; the conclusion follows using a proximity argument (Theorem 3.14). The overall algorithm repeats this subroutine n times. The subroutine is reminiscent of the feasibility algorithm (Theorem 3.3) with the following main difference: whenever we identify a new 'large' coordinate, we slightly perturb the cost function.

Comparison to black-box LP approaches An important milestone towards strongly polynomial linear programming was Tardos's 1986 paper [143] on solving (LP) in time $\mathrm{poly}(n, m, \log \Delta_A)$, where Δ_A is the maximum subdeterminant of A. Her algorithm makes $O(nm)$ calls to a weakly polynomial LP solver for instances with small integer capacities and costs, and uses proximity arguments to gradually learn the support of an optimal solution. This approach was extended to the real model of computation for an $\mathrm{poly}(n, m, \log \kappa_A)$ bound [41]. The latter result uses proximity arguments with circuit imbalances κ_A, and eliminates all dependence on bit-complexity.

The proximity tool Theorem 3.14 derives from [41], and our circuit augmentation algorithms are inspired by the feasibility and optimization algorithms in this paper. However, using circuit augmentation oracles instead of an approximate LP oracle changes the setup. Our arguments become simpler since we proceed through a sequence of feasible solutions, whereas much effort in [41] is needed to deal with infeasibility of the solutions returned by the approximate solver. On the other hand, we need to be more careful as all steps must be implemented using circuit augmentations in the original system, in contrast to the higher degree of freedom in [41] where we can make approximate solver calls to arbitrary modified versions of the input LP.

Chapter organization The rest of the chapter is organized as follows. We first provide the necessary preliminaries in Section 3.2. In Section 3.3, we upper bound the circuit diameter of (P). In Section 3.4, this bound is extended to the setting with upper bounds on the variables. Then, we develop circuit-augmentation algorithms for solving (LP). In particular, Section 3.5 contains the algorithm for finding a feasible solution, whereas Section 3.6 contains the algorithm for solving (LP) given an initial feasible solution. Section 3.7 shows how circuits in LPs of more general forms can be reduced to ours.

3.2 Preliminaries

We use the conventions $\infty \cdot 0 = 0$ and $1/0 = \infty$. For $\alpha \in \mathbb{R}$, we denote $\alpha^+ = \max\{0, \alpha\}$ and $\alpha^- = \max\{0, -\alpha\}$. For a vector $z \in \mathbb{R}^n$ we define $z^+, z^- \in \mathbb{R}^n$ as $(z^+)_i = (z_i)^+$,

$(z^-)_i = (z_i)^-$ for $i \in [n]$. For $z \in \mathbb{R}^n$ we let $\mathrm{supp}(z) = \{i \in [n] : z_i \neq 0\}$ denote its support, and $1/z \in (\mathbb{R} \cup \{\infty\})^n$ denote the vector $(1/z_i)_{i\in[n]}$. We use $\|\cdot\|_p$ to denote the ℓ_p-norm; we simply write $\|\cdot\|$ for $\|\cdot\|_2$. For $A \in \mathbb{R}^{m\times n}$ and $S \subseteq [n]$, we let $A_S \in \mathbb{R}^{m\times|S|}$ denote the submatrix corresponding to columns S. We denote $\mathrm{rk}(S) := \mathrm{rk}(A_S)$, i.e., the rank of the set S in the linear matroid associated with A. The *closure* of S is defined as $\mathrm{cl}(S) := \{i \in [n] : \mathrm{rk}(S \cup \{i\}) = \mathrm{rk}(S)\}$.

For $A \in \mathbb{R}^{m\times n}$, let $W = \ker(A)$. Recall that $\mathcal{C}(W) = \mathcal{C}(A)$ and $\mathcal{E}(W) = \mathcal{E}(A)$ are the set of circuits and elementary vectors in W respectively. The *circuit imbalance measure* of W is defined as

$$\kappa_W = \kappa_A = \max_{g\in\mathcal{E}(W)} \left\{ \frac{|g_i|}{|g_j|} : i,j \in \mathrm{supp}(g) \right\}.$$

According to the next lemma, this is also equal to the imbalance of the dual space $\mathrm{Im}(A^\top)$:

Lemma 3.5 ([37]). *For a linear space $\{\mathbb{0}\} \neq W \subseteq \mathbb{R}^n$, we have $\kappa_W = \kappa_{W^\perp}$.*

For P as in (P), $x \in P$ and an elementary vector $g \in \mathcal{E}(A)$, we let $\mathrm{aug}_P(x,g) := x + \alpha g$ where $\alpha = \arg\max\{\bar\alpha : x + \bar\alpha g \in P\}$.

Definition 3.6. [44] We say that $x, y \in \mathbb{R}^n$ are *sign-compatible* if $x_i y_i \geq 0$ for all $i \in [n]$. We write $x \sqsubseteq y$ if they are sign-compatible and further $|x_i| \leq |y_i|$ for all $i \in [n]$. For a linear space $W \subseteq \mathbb{R}^n$ and $x \in W$, a *conformal circuit decomposition* of x is a set of elementary vectors $h^{(1)}, h^{(2)}, \ldots, h^{(k)}$ in W such that $x = \sum_{j=1}^k h^{(j)}$, $k \leq n$, and $h^{(j)} \sqsubseteq x$ for all $j \in [k]$.

The following lemma shows that every vector in a linear space has a conformal circuit decomposition. It is a simple corollary of the Minkowski–Weyl and Carathéodory theorems.

Lemma 3.7. *For a linear space $W \subseteq \mathbb{R}^n$, every $x \in W$ has a conformal circuit decomposition $x = \sum_{j=1}^k h^{(j)}$ such that $k \leq \min\{\dim(W), |\mathrm{supp}(x)|\}$.*

3.2.1 Circuit Oracles

In Sections 3.4, 3.5, and 3.6, we use a simple circuit finding subroutine Support-Circuit(A,c,x,S) that will be used to identify circuits in the support of a solution x. This can be implemented easily using Gaussian elimination. Note that the constraint $\langle c, z\rangle \leq 0$ is superficial as $-z$ is also an elementary vector for every elementary vector z.

Subroutine 3.2.1. Support-Circuit(A,c,x,S)

For a matrix $A \in \mathbb{R}^{m\times n}$, vectors $c, x \in \mathbb{R}^n$ and $S \subseteq [n]$, the output is an elementary vector $z \in \mathcal{E}(A)$ with $\mathrm{supp}(z) \subseteq \mathrm{supp}(x)$, $\mathrm{supp}(z) \cap S \neq \emptyset$ with $\langle c, z\rangle \leq 0$, or concludes that no such elementary vector exists.

The circuit augmentation algorithms in Sections 3.5 and 3.6 will use the subroutine Ratio-Circuit(A,c,w).

Subroutine 3.2.2. RATIO-CIRCUIT(A, c, w)

The input is a matrix $A \in \mathbb{R}^{m\times n}$, $c \in \mathbb{R}^n$, $w \in (\mathbb{R}_+ \cup \{\infty\})^n$, and returns a basic optimal solution to the system:

$$\min\ \langle c, z\rangle \quad \text{s.t.} \quad Az = \mathbb{0}\,,\ \langle w, z^-\rangle \le 1\,, \tag{3.2}$$

and a basic optimal solution (y, s) to the following dual program:

$$\max\ -\lambda \quad \text{s.t.} \quad s = c + A^\top y \quad \mathbb{0} \le s \le \lambda w \tag{3.3}$$

Note that (3.2) can be reformulated as an LP using additional variables, and its dual LP can be equivalently written as (3.3). If (3.2) is bounded, then a basic optimal solution is an elementary vector $z \in \mathcal{E}(A)$ that minimizes $\langle c, z\rangle / \langle w, z^-\rangle$. Moreover, observe that every feasible solution to (3.3) is also feasible to the dual of (LP). The following lemma is well-known, see e.g., [104, Lemma 2.2].

Lemma 3.8. *Let* OPT *denote the optimum value of* (LP)*. Given a feasible solution x to* (LP)*, let g be the elementary vector returned by* RATIO-CIRCUIT$(A, c, 1/x)$*, and $x' = \text{aug}_P(x, g)$. Then, $\alpha \ge 1$ for the augmentation stepsize, and*

$$\langle c, x'\rangle - \text{OPT} \le \left(1 - \frac{1}{|\operatorname{supp}(x)|}\right)(\langle c, x\rangle - \text{OPT})\ .$$

Proof. The stepsize bound $\alpha \ge 1$ follows since $\langle 1/x, g^-\rangle \le 1$; thus, $x + g \in P$. Let x^* be an optimal solution to (LP), and let $z = (x^* - x)/|\operatorname{supp}(x)|$. Then, z is feasible to (3.2) for $w = 1/x$. Therefore,

$$\alpha\,\langle c, g\rangle \le \langle c, g\rangle \le \langle c, z\rangle = \frac{\text{OPT} - \langle c, x\rangle}{|\operatorname{supp}(x)|}\ ,$$

implying the second claim. □

Remark 3.9. It is worth noting that Lemma 3.8 shows that applying RATIO-CIRCUIT to vectors x with small support gives better convergence guarantees. Algorithms 7 and 8 for feasibility and optimization in Sections 3.5 and 3.6 apply RATIO-CIRCUIT to vectors x which have large support $|\operatorname{supp}(x)| = \Theta(n)$ in general. These algorithms could be reformulated in that one first runs SUPPORT-CIRCUIT to reduce the size of the support to size $O(m)$ and only then runs RATIO-CIRCUIT. The guarantees of Lemma 3.8 now imply that to reduce the optimality gap by a constant factor we would replace $O(n)$ calls to RATIO-CIRCUIT with only $O(m)$ calls. On the other hand, this comes at the cost of n additional calls to SUPPORT-CIRCUIT for every call to RATIO-CIRCUIT.

3.2.2 Proximity Results

The imbalance measure κ_A is mainly used for proving norm bounds that can be interpreted as special forms of Hoffman-proximity results. We formulate such statements that will be needed for our analyses. These can be derived from more general results in [41]; see also [58]. The references also explain the background and similar results in previous literature, in particular, to proximity bounds via Δ_A in e.g., [143] and [34]. For completeness, we include the proofs.

Lemma 3.10. *For $A \in \mathbb{R}^{m\times n}$, let $N \subseteq [n]$ such that $A_{[n]\setminus N}$ has full column rank. Then, for any $z \in \ker(A)$, we have $\|z\|_\infty \le \kappa_A \|z_N\|_1$.*

Proof. Let $h^{(1)}, \dots, h^{(k)}$ be a conformal circuit decomposition of z. Conformality implies that $\|z\|_\infty \le \sum_{t=1}^k \|h^{(t)}\|_\infty$. For each $h^{(t)}$, we have $\operatorname{supp}(h^{(t)}) \cap N \neq \emptyset$ because $A_{[n]\setminus N}$ has full column rank. Hence, $\|h^{(t)}\|_\infty \le \kappa_A |h_j^{(t)}|$ for some $j \in N$. By conformality again, we obtain $\sum_{t=1}^k \|h^{(t)}\|_\infty \le \kappa_A \|z_N\|_1$ as desired. □

The next technical lemma will be key in our arguments. See Corollay 3.13 below for a simple implication.

Lemma 3.11. *Let $A \in \mathbb{R}^{m\times n}$ and $x \in \mathbb{R}^n$. Let $L \subseteq \operatorname{supp}(x)$ and $S \subseteq [n] \setminus L$. If there is no circuit $C \subseteq \operatorname{supp}(x)$ such that $C \cap S \neq \emptyset$, then*

$$\|x_S\|_\infty \le \kappa_A \min_{z \in \ker(A)+x} \|z_{[n]\setminus \operatorname{cl}(L)}\|_1 .$$

Proof. First, observe that $x_{S\cap \operatorname{cl}(L)} = \mathbb{0}$ due to our assumption. Indeed, any $i \in S \cap \operatorname{cl}(L)$ with $x_i \neq 0$ gives rise to a circuit in $L \cup \{i\} \subseteq \operatorname{supp}(x)$. It follows that $\|x_S\|_\infty = \|x_{S\setminus \operatorname{cl}(L)}\|_\infty$; let $j \in S \setminus \operatorname{cl}(L)$ such that $|x_j| = \|x_S\|_\infty$. Let $z \in \ker(A) + x$ be a minimizer of the RHS in the statement. We may assume that $|x_j| > |z_j|$, as otherwise we are done because $\kappa_A \ge 1$.

Let $h^{(1)}, \dots, h^{(k)}$ be a conformal circuit decomposition of $z - x \in \ker(A)$. Among these elementary vectors, consider the set $R := \{t \in [k] : h_j^{(t)} \neq 0\}$.

Claim 3.12. *For each $t \in R$, there exists an index $i_t \in \operatorname{supp}(h^{(t)}) \setminus \operatorname{cl}(L)$ such that $x_{i_t} = 0$ and $z_{i_t} \neq 0$.*

Proof. For the purpose of contradiction, suppose that $\operatorname{supp}(h^{(t)}) \setminus \operatorname{cl}(L) \subseteq \operatorname{supp}(x)$. For every $i \in \operatorname{cl}(L)$, we can write $A_i = A_L y^{(i)}$ for some $y^{(i)} \in \mathbb{R}^L$. By applying this to the coordinates in $\operatorname{cl}(L) \setminus L$, we can transform $h^{(t)}$ into a vector $h \in \ker(A)$ such that $h_{\operatorname{cl}(L)\setminus L} = \mathbb{0}$ and $h_{[n]\setminus \operatorname{cl}(L)} = h^{(t)}_{[n]\setminus \operatorname{cl}(L)}$. However, we now have $\operatorname{supp}(h) \subseteq \operatorname{supp}(x)$ because $L \subseteq \operatorname{supp}(x)$. Moreover, $\operatorname{supp}(h) \cap S \neq \emptyset$ because $j \in S \setminus \operatorname{cl}(L)$. Then, $\operatorname{supp}(h)$ must contain a circuit C with $C \cap S \neq \emptyset$, contradicting the assumption of the lemma. □

By the claim above, we get $|h_j^{(t)}| \leq \kappa_A |z_{i_t}|$ for all $t \in R$. Note that $i_t \neq j$ for all $t \in R$ because we assumed that $|x_j| > 0$. By the conformality of the decomposition, we obtain

$$|x_j - z_j| = \sum_{t \in R} |h_j^{(t)}| \leq \kappa_A \|z_{[n] \setminus (\mathrm{cl}(L) \cup \{j\})}\|_1 .$$

This gives us

$$\|x_S\|_\infty = |x_j| \leq |z_j| + |x_j - z_j| \leq \kappa_A \|z_{[n] \setminus \mathrm{cl}(L)}\|_1$$

as desired. □

For $L = \emptyset$ and $S = [n]$, we obtain the following corollary.

Corollary 3.13. *Let x be a basic (but not necessarily feasible) solution to* (LP)*. Then, for any z where $Az = b$, we have $\|x\|_\infty \leq \kappa_A \|z\|_1$.*

The following proximity theorem will be key to derive $x_i^* = 0$ for certain variables in our optimization algorithm; see [41] and [58, Theorem 6.5]. For $\tilde{c} \in \mathbb{R}^n$, we use $\mathrm{LP}(\tilde{c})$ to denote (LP) with cost vector $\tilde{c}$, and $\mathrm{OPT}(\tilde{c})$ as the optimal value of $\mathrm{LP}(\tilde{c})$.

Theorem 3.14. *Let $c, c' \in \mathbb{R}^n$ be two cost vectors, such that both $\mathrm{LP}(c)$ and $\mathrm{LP}(c')$ have finite optimum values. Let s' be a dual optimal solution to $\mathrm{LP}(c')$. For all indices $j \in [n]$ such that*

$$s'_j > (m+1)\kappa_A \|c - c'\|_\infty ,$$

it follows that $x_j^* = 0$ for every optimal solution x^* to $\mathrm{LP}(c)$.

Proof. We may assume that $c \neq c'$, as otherwise we are done by complementary slackness. Let x' be an optimal solution to $\mathrm{LP}(c')$. By complementary slackness, $s'_j x'_j = 0$, and therefore $x'_j = 0$. For the purpose of contradiction, suppose that there exists an optimal solution x^* to $\mathrm{LP}(c)$ such that $x_j^* > 0$. Let $h^{(1)}, \dots, h^{(k)}$ be a conformal circuit decomposition of $x^* - x'$. Then, $h_j^{(t)} > 0$ for some $t \in [k]$, and therefore $\|h^{(t)}\|_1 \leq (m+1)\|h^{(t)}\|_\infty \leq (m+1)\kappa_A h_j^{(t)}$ Observe that for any $i \in [n]$ where $h_i^{(t)} < 0$, we have $s'_i = 0$ because $x'_i > x_i^* \geq 0$. Hence,

$$\begin{aligned}\left\langle c, h^{(t)} \right\rangle &= \left\langle c - c', h^{(t)} \right\rangle + \left\langle c', h^{(t)} \right\rangle \geq -\|c - c'\|_\infty \|h^{(t)}\|_1 + \left\langle s', h^{(t)} \right\rangle \\ &\geq -(m+1)\kappa_A \|c - c'\|_\infty \, h_j^{(t)} + s'_j h_j^{(t)} > 0 .\end{aligned}$$

The first inequality here used Hölder's inequality and that $\left\langle c', h^{(t)} \right\rangle = \left\langle s', h^{(t)} \right\rangle$ since $c' - s'$ and $h^{(t)}$ are in orthogonal spaces. Since $x^* - h^{(t)}$ is feasible to $\mathrm{LP}(c)$, this contradicts the optimality of x^*. □

The following lemma provides an upper bound on the norm of the perturbation $c - c'$ for which the existences of an index j as in Theorem 3.14 is guaranteed.

Lemma 3.15. *Let $c, c' \in \mathbb{R}^n$ be two cost vectors, and let s' be an optimal dual solution to* $\mathrm{LP}(c')$. *If $c \in \ker(A)$, $\|c\|_2 = 1$ and $\|c - c'\|_\infty < 1/(\sqrt{n}(m+2)\kappa_A)$ for some $\kappa_A \geq 1$, then there exists an index $j \in [n]$ such that*

$$s'_j > \frac{m+1}{\sqrt{n}(m+2)}.$$

Proof. Let $r = c - c'$. Note that $s' + r \in \mathrm{Im}(A^\top) + c$. Then,

$$\|s'\|_\infty + \|r\|_\infty \geq \|s' + r\|_\infty \geq \frac{1}{\sqrt{n}}\|s' + r\|_2 \geq \frac{1}{\sqrt{n}}\|c\|_2 = \frac{1}{\sqrt{n}},$$

where the last inequality is due to $s' + r - c$ and c being orthogonal. This gives us

$$\|s'\|_\infty \geq \frac{1}{\sqrt{n}} - \|r\|_\infty > \frac{(m+2)\kappa_A - 1}{\sqrt{n}(m+2)\kappa_A} \geq \frac{m+1}{\sqrt{n}(m+2)}$$

as desired because $\kappa_A \geq 1$. □

3.2.3 Estimating Circuit Imbalances

The circuit augmentation algorithms in Sections 3.5 and 3.6 explicitly use the circuit imbalance measure κ_A. However, this is NP-hard to approximate within a factor $2^{O(n)}$, see [148, 37]. We circumvent this problem using a standard guessing procedure, see e.g., [149, 37]. Instead of κ_A, we use an estimate $\hat{\kappa}$, initialized as $\hat{\kappa} = n$. Running the algorithm with this estimate either finds the desired feasible or optimal solution (which one can verify), or fails. In case of failure, we conclude that $\hat{\kappa} < \kappa_A$, and replace $\hat{\kappa}$ by $\hat{\kappa}^2$. Since the running time of the algorithms is linear in $\log(n + \hat{\kappa})$, the running time of all runs will be dominated by the last run, giving the desired bound. For simplicity, the algorithm descriptions use the explicit value κ_A.

3.3 The Circuit Diameter Bound

In this section, we show Theorem 3.1, namely the bound $O(m \min\{m, n-m\} \log(m + \kappa_A))$ on the circuit diameter of a polyhedron in standard form (P). As outlined in the Introduction, let $B \subseteq [n]$ be a feasible basis and $N = [n] \setminus B$ such that $x^* = (A_B^{-1}b, \mathbb{0}_N)$ is a basic solution to (LP). We can assume $n \leq 2m$: the union of the supports of the starting vertex $x^{(0)}$ and the target vertex x^* is at most $2m$; we can fix all other variables to 0. Defining $\tilde{n} := |\mathrm{supp}(x^*) \cup \mathrm{supp}(x^{(0)})| \leq 2m$ and restricting A to these columns, we show a circuit diameter bound $O(\tilde{n}(\tilde{n} - m)\log(m + \kappa_A))$. This implies Theorem 3.1 for general n. In the rest of this section, we use n instead of $\tilde{n}$, but assume $n \leq 2m$. The simple 'shoot towards the optimum' procedure is shown in Algorithm 5.

```
Algorithm 5: DIAMETER-BOUND
    Input  : Polyhedron in standard form (P), basis $B \subseteq [n]$ with its corresponding
             vertex $x^* = (A_B^{-1}b, \mathbb{0}_N)$, and initial vertex $x^{(0)}$.
    Output : Length of a circuit walk from $x^{(0)}$ to $x^*$.
1   $t \leftarrow 0$
2   while $x^{(t)} \neq x^*$ do
3       Let $h^{(1)}, h^{(2)}, \dots, h^{(k)}$ be a conformal circuit decomposition of $x^* - x^{(t)}$
4       $g^{(t)} \leftarrow h^{(j)}$ for any $j \in \arg\max_{i\in[k]} \|h_N^{(i)}\|_1$
5       $x^{(t+1)} \leftarrow \mathrm{aug}_P(x^{(t)}, g^{(t)})$; $t \leftarrow t+1$
6   return $t$
```

A priori, even finite termination is not clear. The first key lemma shows that $\|x_N^{(t)}\|_1$ decreases geometrically, and bounds the relative error to x^*.

Lemma 3.16. *For every iteration $t \geq 0$ in Algorithm 5, we have $\|x_N^{(t+1)}\|_1 \leq (1-\frac{1}{n-m})\|x_N^{(t)}\|_1$ and for all $i \in [n]$ we have $|x_i^{(t+1)} - x_i^{(t)}| \leq (n-m)|x_i^* - x_i^{(t)}|$.*

Proof. Let $h^{(1)}, \dots, h^{(k)}$ with $k \leq n-m$ be the conformal circuit decomposition of $x^* - x^{(t)}$ used in Algorithm 5. Let $\alpha^{(t)}$ be such that $x^{(t+1)} = x^{(t)} + \alpha^{(t)} g^{(t)}$. Clearly, $\alpha^{(t)} \geq 1$ since $x^{(t)} + g^{(t)} \in P$.

Note that $h_N^{(i)} \leq \mathbb{0}_N$ for $i \in [k]$ as $x_N^* = \mathbb{0}_N$ and $x^{(t)} \geq \mathbb{0}$. Then

$$\begin{aligned} \|g_N^{(t)}\|_1 &= \max_{i\in[k]} \|h_N^{(i)}\|_1 \geq \frac{1}{k}\sum_{i\in[k]} \|h_N^{(i)}\|_1 = \frac{1}{k}\|x_N^{(t)}\|_1 \quad \text{and so} \\ \|x_N^{(t+1)}\|_1 &= \|\,\mathrm{aug}_P\,(x^{(t)}, g^{(t)})_N\|_1 \leq \|x_N^{(t)} + g_N^{(t)}\|_1 \\ &= \|x_N^{(t)}\|_1 - \|g_N^{(t)}\|_1 \leq (1-\frac{1}{k})\|x_N^{(t)}\|_1\,, \end{aligned} \tag{3.4}$$

where the last equality uses conformality of the decomposition. Further, using that $0 \leq x_N^{(t+1)} \leq x_N^{(t)}$, we see that

$$\alpha^{(t)} = \frac{\|x_N^{(t+1)} - x_N^{(t)}\|_1}{\|g_N^{(t)}\|_1} \leq \frac{\|x_N^{(t)}\|_1}{\|g_N^{(t)}\|_1} \leq k,$$

and so for all i we have $|x_i^{(t+1)} - x_i^{(t)}| = \alpha^{(t)}|g_i^{(t)}| \leq k|g_i^{(t)}| \leq k|x_i^* - x_i^{(t)}|$. □

We analyze the sets

$$L_t := \{i \in [n] : \, x_i^* > n\kappa_A\|x_N^{(t)}\|_1\}\,, \quad T_t := [n] \setminus L_t\,, \quad R_t := \{i \in [n] : x_i^{(t)} \leq (n-m)x_i^*\}\,.$$

Lemma 3.17. *For every iteration $t \geq 0$, we have $L_t \subseteq L_{t+1} \subseteq B$ and $R_t \subseteq R_{t+1}$.*

Proof. Clearly, $L_t \subseteq L_{t+1}$ as $\|x_N^{(t)}\|_1$ is monotonically decreasing by Lemma 3.16, and $L_t \subseteq B$ as $x_N^* = \mathbb{0}_N$. Next, let $j \in R_t$. If $x_j^{(t)} \geq x_j^*$, then $x_j^{(t+1)} \leq x_j^{(t)}$ by conformality. If $x_j^{(t)} < x_j^*$, then $x_j^{(t+1)} \leq x_j^{(t)} + (n-m)(x_j^* - x_j^{(t)}) \leq (n-m)x_j^*$ by Lemma 3.16. In both cases, we conclude that $j \in R_{t+1}$. □

Our goal is to show that either R_t or L_t is extended within $O((n-m)\log(n+\kappa_A))$ iterations. First, note that by the maximality of the augmentation, there is a variable $i \in \operatorname{supp}(x^{(t)}) \setminus \operatorname{supp}(x^{(t+1)})$ in each iteration. Clearly, $i \in R_{t+1}$. First, we show that if $\|x_{T_t}^{(t)} - x_{T_t}^*\|_\infty$ is sufficiently large, then $i \notin R_t$ for all such coordinates.

Lemma 3.18. *If $\|x_{T_t}^{(t)} - x_{T_t}^*\|_\infty > 2mn^2\kappa_A^2 \left\|x_{T_t}^*\right\|_\infty$, then $R_t \subsetneq R_{t+1}$.*

Proof. Let $i \in \operatorname{supp}(x^{(t)}) \setminus \operatorname{supp}(x^{(t+1)})$; such a variable exists by the maximality of the augmentation. Lemma 3.10 for $x^{(t+1)} - x^* \in \ker(A)$ implies that

$$x_i^* \leq \|x^{(t+1)} - x^*\|_\infty \leq \kappa_A \|x_N^{(t+1)} - x_N^*\|_1 = \kappa_A \|x_N^{(t+1)}\|_1 < \kappa_A \|x_N^{(t)}\|_1, \tag{3.5}$$

and so $i \notin L_t$. Noting that $x^{(t+1)} - x^{(t)} = \alpha^{(t)} g^{(t)}$ is an elementary vector and $x_i^{(t+1)} = 0$, it follows that

$$\left\|x_N^{(t)} - x_N^{(t+1)}\right\|_1 \leq (m\kappa_A + 1)x_i^{(t)} \leq 2m\kappa_A x_i^{(t)}\,. \tag{3.6}$$

On the other hand, let $h^{(1)}, \ldots, h^{(k)}$ with $k \leq n-m$ be the conformal circuit decomposition of $x^* - x^{(t)}$ used in iteration t in Algorithm 5. Let $j \in T_t$ such that $|x_j^{(t)} - x_j^*| = \|x_{T_t}^{(t)} - x_{T_t}^*\|_\infty$. There exists $\tilde{h}$ in this decomposition such that $|\tilde{h}_j| \geq \frac{1}{k}|x_j^{(t)} - x_j^*|$. Since A_B has full column rank, we have $\operatorname{supp}(\tilde{h}) \cap N \neq \emptyset$ and so

$$\|\tilde{h}_N\|_1 \geq \frac{|\tilde{h}_j|}{\kappa_A} \geq \frac{|x_j^{(t)} - x_j^*|}{k\kappa_A}\,. \tag{3.7}$$

From (3.6), (3.7) and noting that $\|\tilde{h}_N\|_1 \leq \|g_N^{(t)}\|_1 \leq \|x_N^{(t)} - x_N^{(t+1)}\|_1$ we get

$$x_i^{(t)} \geq \frac{\|x_N^{(t)} - x_N^{(t+1)}\|_1}{2m\kappa_A} \geq \frac{\|\tilde{h}_N\|_1}{2m\kappa_A} \geq \frac{\|x_{T_t}^{(t)} - x_{T_t}^*\|_\infty}{2mk\kappa_A^2}\,. \tag{3.8}$$

In particular, if as in the assumption of the lemma $\|x_{T_t}^{(t)} - x_{T_t}^*\|_\infty > 2mn^2\kappa_A^2\|x_{T_t}^*\|_\infty$, then $x_i^{(t)} > n\|x_{T_t}^*\|_\infty \geq nx_i^*$. We conclude that $i \notin R_t$ and $i \in R_{t+1}$ as $x_i^{(t+1)} = 0$. □

We are ready to give the convergence bound. Above we have shown that a large $\|x_{T_t}^{(t)} - x_{T_t}^*\|_\infty$ guarantees the extension of R_t. Using the geometric decay of $\|x_N^{(t)}\|$ (Lemma 3.16), we show that whenever this distance is small, L_t must be extended.

Proof of Theorem 3.1. In light of Lemma 3.17, it suffices to show that either L_t or R_t is extended in every $O((n-m)\log(n+\kappa_A))$ iterations; recall the assumption $n \le 2m$. By Lemma 3.18, if $\|x^{(t)}_{T_t} - x^*_{T_t}\|_\infty > 2mn^2\kappa_A^2 \left\|x^*_{T_t}\right\|_\infty$, then $R_t \subsetneq R_{t+1}$ is extended.

Otherwise, $\|x^{(t)}_{T_t} - x^*_{T_t}\|_\infty \le 2mn^2\kappa_A^2 \left\|x^*_{T_t}\right\|_\infty$, that is, $\|x^{(t)}_{T_t}\|_\infty \le (2mn^2\kappa_A^2+1)\left\|x^*_{T_t}\right\|_\infty$. Assuming $\|x^{(t)}_N\|_1 > 0$, by Lemma 3.16, there is an iteration $r = t + O((n-m)\log(n+\kappa_A)) = t + O(\min\{n-m, m\}\log(n+\kappa_A))$ such that $n^2\kappa_A(2mn^2\kappa_A^2+1)\|x^{(r)}_N\|_1 < \|x^{(t)}_N\|_1$. In particular,

$$(2mn^2\kappa_A^2+1)\|x^*_{T_t}\|_\infty \ge \|x^{(t)}_{T_t}\|_\infty \ge \|x^{(t)}_N\|_\infty \ge \frac{1}{n}\|x^{(t)}_N\|_1 > n\kappa_A(2mn^2\kappa_A^2+1)\|x^{(r)}_N\|_1. \quad (3.9)$$

Therefore $\|x^*_{T_t}\|_\infty > n\kappa_A\|x^{(r)}_N\|_1$ and so $L_t \subsetneq L_r$. □

3.4 Circuit Diameter Bound for the Capacitated Case

In this section we consider diameter bounds for systems of the form

$$P_u = \{x \in \mathbb{R}^n : Ax = b, \mathbb{0} \le x \le u\}. \quad \text{(Cap-P)}$$

The theory in Section 3.3 carries over to P_u at the cost of turning m into n via the standard reformulation

$$\widetilde{P}_u = \left\{(x,y) \in \mathbb{R}^{n+n} : \begin{bmatrix} A & 0 \\ I & I \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ u \end{bmatrix}, x, y \ge 0\right\}, \quad P_u = \{x : (x,y) \in \widetilde{P}_u\}. \quad (3.10)$$

Corollary 3.19. *The circuit diameter of a system in the form (P) with constraint matrix $A \in \mathbb{R}^{m\times n}$ is $O(n^2\log(n+\kappa_A))$.*

Proof. Follows straightforward from Theorem 3.1 together with the reformulation (3.10). It is easy to check that κ_A of the constraint matrix of (3.10) coincides with κ_A, and that there is a one-to-one mapping between the circuits and maximal circuit augmentations of the two systems. □

Intuitively, the polyhedron should not become more complex; related theory in [22] also shows how two-sided bounds can be incorporated in a linear program without significantly changing the complexity of solving the program.

We prove Theorem 3.2 via the following new procedure. A basic feasible point $x^* \in P_u$ is characterised by a partition $B \cup L \cup H = [n]$ where A_B is a basis (has full column rank), $x^*_L = \mathbb{0}_L$ and $x^*_H = u_H$. In $O(n\log n)$ iterations, we fix all but $2m$ variables to the same bound as in x^*; for the remaining system with $2m$ variables, we can use the standard reformulation.

Proof of Theorem 3.2. We show that Algorithm 6 has the claimed number of iterations. First, note that $\langle c, x^*\rangle = -|H|$ is the optimum value. Initially, $\left\langle c, x^{(0)}\right\rangle = -\sum_{i\in H}\frac{x^{(0)}}{u_i} + \sum_{i\in L}\frac{x^{(0)}}{u_i} \le$

```
Algorithm 6: CAPACITATED-DIAMETER-BOUND
   Input  : Polyhedron in the form (Cap-P), partition $B \cup L \cup H = [n]$ with its
            corresponding vertex $x^* = (A_B^{-1} b, \mathbb{0}_L, u_H)$, and initial vertex $x^{(0)}$.
   Output : Length of a circuit walk from $x^{(0)}$ to $x^*$.
 1 Set the cost $c \in \mathbb{R}^n$ as $c_i = 0$ if $i \in B$, $c_i = 1/u_i$ if $i \in L$, and $c_i = -1/u_i$ if $i \in H$
 2 $t \leftarrow 0$
 3 $S_0 \leftarrow \{i \in L \cup H : x_i^{(0)} \neq x_i^*\}$
 4 while $|S_t| \le m$ do
 5     if $\langle c, x^{(t)} \rangle \ge -|H| + 1$ then
 6         Let $h^{(1)}, h^{(2)}, \dots, h^{(k)}$ be a conformal circuit decomposition of $x^* - x^{(t)}$
 7         $g^{(t)} \leftarrow h^{(j)}$ for any $j \in \arg\min_{i \in [k]} \langle c, h^{(i)} \rangle$
 8     else
 9         $g^{(t)} \leftarrow$ SUPPORT-CIRTCUIT$(A, c, x^{(t)}, S_t)$
10     $x^{(t+1)} \leftarrow \mathrm{aug}_P(x^{(t)}, g^{(t)})$
11     $S_{t+1} \leftarrow \{i \in L \cup H : x_i^{(t+1)} \neq x_i^*\}$; $t \leftarrow t + 1$
12 Run Algorithm 5 on $\widetilde{A} := \begin{bmatrix} A_{B \cup S_t} & 0 \\ I & I \end{bmatrix}$ and $\widetilde{b} = \begin{bmatrix} b \\ u \end{bmatrix}$ to get $t' \in \mathbb{Z}_+$
13 return $t + t'$
```

n. Similar to Lemma 3.16, due to our choice of $g^{(t)}$ from the conformal circuit decomposition, we have $\left\langle c, x^{(t+1)} \right\rangle + |H| \le (1 - \frac{1}{n-m})(\left\langle c, x^{(t)} \right\rangle + |H|)$. In particular, $O((n-m)\log n)$ iterations suffice to find an iterate t such that $\langle c, x^{(t)} \rangle < -|H| + 1$.

Note that the calls to SUPPORT-CIRCUIT do not increase $\left\langle c, x^{(t)} \right\rangle$, so from now we will never make use of the conformal circuit decomposition again. A call to SUPPORT-CIRCUIT will set at least one variable $i \in \mathrm{supp}(g^{(t)})$ to either 0 or u_i. We claim that either $x_i^{(t+1)} = 0$ for some $i \in L$, or $x_i^{(t+1)} = u_i$ for some $i \in H$, that is, we set a variable to the 'correct' boundary. To see this, note that if $x_i^{(t+1)}$ hits the wrong boundary, then the gap between $\left\langle c, x^{(t+1)} \right\rangle$ and $-|H|$ must be at least 1, a clear contradiction to $\left\langle c, x^{(t+1)} \right\rangle < -|H| + 1$.

Thus, after at most n calls to SUPPORT-CIRCUIT, we get $|S_t| \le m$, at which point we call Algorithm 5 with $\le 2m$ variables, so the diameter bound of Theorem 3.1 applies. □

3.5 A Circuit Augmentation Algorithm for Feasibility

In this section we prove Theorem 3.3: given a system (LP) with cost $c = (\mathbb{0}_{[n]\setminus N}, \mathbb{1}_N)$ for some $N \subseteq [n]$, find a solution x with $x_N = \mathbb{0}$, or show that no such solution exists.

As an application, assume we are looking for a feasible solution to the program (P). We can construct an auxiliary linear program, that has trivial feasible solutions and whose optimal solutions correspond to feasible solutions of the original program (P). This is in the same tune as Phase I of the Simplex method.

$$\min\ \langle \mathbb{1}_n, z\rangle \quad \text{s.t.} \quad Ay - Az = b\,,\ y, z \geq 0\,. \tag{Aux-LP}$$

For the constraint matrix $\widetilde{A} = \begin{bmatrix} A & -A \end{bmatrix}$ it is easy to see that $\kappa_{\widetilde{A}} = \kappa_A$ and that any solution $Ax = b$ can be converted into a feasible solution to (Aux-LP) via $(y, z) = (x^+, x^-)$. Hence, if the subroutines Support-Circuit and Ratio-Circuit are available for (Aux-LP), then we can find a feasible solution to (P) in $O(mn \log(n + \kappa_A))$ augmentation steps.

Our algorithm is presented in Algorithm 7. We maintain a set $\mathcal{L}_t \subseteq [n] \setminus N$, initialized as $\emptyset$. Whenever $x_i^{(t)} \geq 4mn\kappa_A \|x_N^{(t)}\|_1$ for the current iterate $x^{(t)}$, we add i to $\mathcal{L}_t$. The key part of the analysis is to show that $\mathrm{rk}(\mathcal{L}_t)$ increases in every $O(n \log(n + \kappa_A))$ iterations.

Whenever $\mathrm{rk}(\mathcal{L}_t)$ increases, we run a sequence of at most n Support-Circuit$(A, c, x^{(t)}, N)$ iterations. This is repeated as long as there exists a circuit in $\mathrm{supp}(x^{(t)})$ intersecting N. Afterwards, we run a sequence of Ratio-Circuit iterations until $\mathrm{rk}(\mathcal{L}_t)$ increases again.

Algorithm 7: Feasibility-Algorithm

```
Input  : Linear program in standard form (LP) with cost $c = (\mathbb{0}_{[n]\setminus N}, \mathbb{1}_N)$ for some
         $N \subseteq [n]$, and initial feasible solution $x^{(0)}$.
Output : A solution $x$ with $x_N = \mathbb{0}$, or a dual solution $y$ with $\langle b, y\rangle > 0$.
1  $t \leftarrow 0$ ; $\mathcal{L}_{t-1} \leftarrow \emptyset$
2  while $x_N^{(t)} \neq \mathbb{0}$ do
3      $\mathcal{L}_t \leftarrow \mathcal{L}_{t-1} \cup \{i \in [n] : x_i^{(t)} \geq 4mn\kappa_A \|x_N^{(t)}\|_1\}$
4      if $t = 0$ or $\mathrm{rk}(\mathcal{L}_t) > \mathrm{rk}(\mathcal{L}_{t-1})$ then
5          while $\exists$ a circuit in $\mathrm{supp}(x^{(t)})$ intersecting $N$ do
6              $g^{(t)} \leftarrow$ Support-Circuit$(A, c, x^{(t)}, N)$
7              $x^{(t+1)} \leftarrow \mathrm{aug}_P(x^{(t)}, g^{(t)})$; $t \leftarrow t+1$ ; $\mathcal{L}_t \leftarrow \mathcal{L}_{t-1}$
8      $(g^{(t)}, y^{(t)}, s^{(t)}) \leftarrow$ Ratio-Circuit$(A, c, 1/x^{(t)})$
9      if $\left\langle b, y^{(t)} \right\rangle > 0$ then
10         Terminate with infeasibility certificate ;
11     $x^{(t+1)} \leftarrow \mathrm{aug}_P(x^{(t)}, g^{(t)})$; $t \leftarrow t+1$
12 return $x^{(t)}$
```

Proof of Theorem 3.3. Clearly, Algorithm 7 performs at most $(m+1)n$ Support-Circuit iterations. So, it is left to show that $\mathrm{rk}(\mathcal{L}_t)$ increases after a sequence of $O(n \log(n + \kappa_A))$ Ratio-Circuit iterations. Let us first analyze what happens at Ratio-Circuit iterations.

Claim 3.20. *If Ratio-Circuit is used in iteration t, then either $\|x_N^{(t+1)}\|_1 \leq \left(1 - \frac{1}{n}\right) \|x_N^{(t)}\|_1$, or the algorithm terminates with a dual certificate.*

Proof. The oracle returns $g^{(t)}$ that is optimal to (3.2) and $(y^{(t)}, s^{(t)})$ with optimum value $-\lambda$. Recall that we use weights $w_i = 1/x_i^{(t)}$. If $\left\langle b, y^{(t)} \right\rangle > 0$, the algorithm terminates. Otherwise, note that

$$\left\langle c, x^{(t)} \right\rangle = \left\langle b, y^{(t)} \right\rangle + \left\langle s^{(t)}, x^{(t)} \right\rangle \le \lambda \left\langle w, x^{(t)} \right\rangle \le n\lambda\,,$$

implying $\lambda \ge \left\langle c, x^{(t)} \right\rangle / n$, and therefore $\left\langle c, g^{(t)} \right\rangle = -\lambda \le -\left\langle c, x^{(t)} \right\rangle / n$. This implies the claim, noting that

$$\|x_N^{(t+1)}\|_1 = \left\langle c, x^{(t+1)} \right\rangle \le \left\langle c, x^{(t)} \right\rangle + \left\langle c, g^{(t)} \right\rangle \le \left(1 - \frac{1}{n}\right) \|x_N^{(t)}\|_1\,. \qquad \square$$

Observe that during a Ratio-Circuit iteration t, if a coordinate $j \in [n]$ satisfies $x_j^{(t)} \ge 2n\kappa_A \|x_N^{(t)}\|_1$, then

$$\begin{aligned} \frac{x_j^{(t+1)}}{\|x_N^{(t+1)}\|_1} &\ge \frac{x_j^{(t)} - \kappa_A \|x_N^{(t+1)} - x_N^{(t)}\|_1}{(1-\frac{1}{n})\|x_N^{(t)}\|_1} \\ &\ge \frac{x_j^{(t)} - 2\kappa_A \|x_N^{(t)}\|_1}{(1-\frac{1}{n})\|x_N^{(t)}\|_1} \ge \frac{(1-\frac{1}{n}) x_j^{(t)}}{(1-\frac{1}{n})\|x_N^{(t)}\|_1} = \frac{x_j^{(t)}}{\|x_N^{(t)}\|_1}\,. \end{aligned} \tag{3.11}$$

Claim 3.21. *For every iteration t and every coordinate $j \in \mathcal{L}_t$, we have $x_j^{(t)} \ge 2mn\kappa_A \|x_N^{(t)}\|_1$.*

Proof. We proceed by induction on the number of iterations $t \ge 0$. The base case $t = 0$ is clearly true. To verify the claim for iteration $t+1$, consider any index $j \in \mathcal{L}_t$, and let $r \le t$ be the iteration when j was added to $\mathcal{L}_r$; the claim clearly holds at iteration r. We analyse the ratio $x_j^{(t')}/\|x_N^{(t')}\|_1$ for iterations $t' = r, \dots, t+1$. At every iteration that performs Ratio-Circuit, if $x_j^{(t')}/\|x_N^{(t')}\|_1 \ge 2n\kappa_A$, then the ratio does not decrease according to (3.11).

Let us analyze the situation when Support-Circuit is called. We have $g_i^{(t)} < 0$ for some $i \in N$ because $\mathrm{supp}(g^{(t)}) \cap N \neq \emptyset$ and $\left\langle c, g^{(t)} \right\rangle \le 0$. Hence,

$$\|x^{(t+1)} - x^{(t)}\|_\infty \le \kappa_A |x_i^{(t+1)} - x_i^{(t)}| \le \kappa_A x_i^{(t)} \le \kappa_A \|x_N^{(t)}\|_1.$$

There are at most $m+1$ sequences of at most n Support-Circuit augmentations throughout the algorithm. By the above argument, each augmentation may decrease $x_j^{(t')}$ by at most $\kappa_A \|x_N^{(t')}\|_1$. Thus, the total decrease in $x_j^{(t')}/\|x_N^{(t')}\|_1$ throughout the algorithm is upper bounded by $(m+1)n\kappa_A \le 2mn\kappa_A$. Since the starting value was $\ge 4mn\kappa_A$, it follows that this ratio does not drop below $2mn\kappa_A$. $\square$

In the first iteration and whenever $\mathrm{rk}(\mathcal{L}_t)$ increases, we perform a (possibly empty) sequence of support circuit cancellations. Let us consider an iteration t right after we are done with the support circuit cancellations. Thus, there is no circuit in $\mathrm{supp}(x^{(t)})$ intersecting N. We show that $\mathrm{rk}(\mathcal{L}_t)$ increases within $O(n \log(n + \kappa_A))$ consecutive calls to Ratio-Circuit; this completes the proof.

Within $O(n\log(n+\kappa_A))$ consecutive Ratio-Circuit augmentations, we reach an iterate $r = t + O(n\log(n+\kappa_A))$ such that $\|x_N^{(r)}\|_1 \le (4mn^3\kappa_A^2)^{-1}\|x_N^{(t)}\|_1$. Since $\mathcal{L}_t \subseteq \operatorname{supp}(x^{(t)})$, $N \subseteq [n] \setminus \mathcal{L}_t$, and there is no circuit in $\operatorname{supp}(x^{(t)})$ intersecting N, applying Lemma 3.11 with $x = x^{(t)}$ and $z = x^{(r)}$ yields

$$\|x^{(r)}_{[n]\setminus \mathrm{cl}(\mathcal{L}_t)}\|_\infty \ge \frac{\|x^{(r)}_{[n]\setminus \mathrm{cl}(\mathcal{L}_t)}\|_1}{n} \ge \frac{\|x_N^{(t)}\|_\infty}{n\kappa_A} \ge \frac{\|x_N^{(t)}\|_1}{n^2\kappa_A} \ge 4mn\kappa_A\|x_N^{(r)}\|_1\,,$$

showing that some $j \in [n] \setminus \mathrm{cl}(\mathcal{L}_t)$ must be included in $\mathcal{L}_r$. □

3.6 A Circuit Augmentation Algorithm for Optimization

In this section, we give a circuit-augmentation algorithm for solving (LP), assuming an initial feasible solution $x^{(0)}$ is provided. In every iteration t, the algorithm maintains a feasible solution $x^{(t)}$ to (LP), initialized with $x^{(0)}$. The goal is to augment $x^{(t)}$ using the subroutines Support-Circuit and Ratio-Circuit until the emergence of a set $\emptyset \neq N \subseteq [n]$ which satisfies $x_N^{(t)} = x_N^* = \mathbb{0}$ for every optimal solution x^* to (LP). When this happens, we have reached a lower dimensional face of the polyhedron that contains the optimal face. Hence, we can fix $x_N^{(t')} = \mathbb{0}$ in all subsequent iterations $t' \ge t$. In particular, we repeat the same procedure on a smaller LP with constraint matrix $A_{[n]\setminus N}$, RHS vector b, and cost $c_{[n]\setminus N}$, initialized with the feasible solution $x^{(t)}_{[n]\setminus N}$. Note that a circuit walk of this smaller LP is also a circuit walk of the original LP. This gives the overall circuit-augmentation algorithm.

In what follows, we focus on the aforementioned variable fixing procedure (Algorithm 8), since the main algorithm just calls it at most n times. An instance of (LP) is given by $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$.

We fix parameters

$$\delta := \frac{1}{2n^{3/2}(m+2)\kappa_A}\,, \qquad \Gamma := \frac{4(m+2)\sqrt{n}\kappa_A^2 T}{\delta}, \qquad T := O(n\log(n+\kappa_A)).$$

Throughout the procedure, A and b will be fixed, but we will sometimes modify the cost function c. Recall that for any $\tilde{c} \in \mathbb{R}^n$, we use $\mathrm{LP}(\tilde{c})$ to denote the problem with cost vector $\tilde{c}$, and the optimum value is $\mathrm{OPT}(\tilde{c})$. We will often use the fact that if $\tilde{s} = \mathrm{Im}(A^\top) + \tilde{c}$ then the systems $\mathrm{LP}(\tilde{s})$ and $\mathrm{LP}(\tilde{c})$ are equivalent.

Let us start with a high level overview before presenting the algorithm. The inference that $x_N^{(t)} = x_N^* = \mathbb{0}$ for every optimal x^* will be made using Theorem 3.14. To apply this, our goal is to find a cost function c' and an optimal dual solution s' to $\mathrm{LP}(c')$ such that the set of indices $N := \{\, j : s'_j > (m+1)\kappa_A\|c - c'\|_\infty \,\}$ is nonempty.

Without loss of generality, we may assume that $\|c\| = 1$. Let us start from any basic feasible solution $x^{(0)}$ and basic feasible dual solution $s^{(0)}$; we can obtain one from a call to Ratio-Circuit. Within $O(n\log(n+\kappa_A))$ Ratio-Circuit augmentations, we arrive at a pair of primal and dual feasible solutions $(x,s)=(x^{(t)},s^{(t)})$ such that $\langle x,s\rangle \le \varepsilon \coloneqq \left\langle x^{(0)},s^{(0)}\right\rangle/\mathrm{poly}(n,\kappa_A)$.

Suppose that for every $i\in\mathrm{supp}(x)$, s_i is small, say $s_i<\delta$. Let $\tilde{c}_i \coloneqq s_i$ if $i\notin\mathrm{supp}(x)$ and $\tilde{c}_i\coloneqq 0$ if $i\in\mathrm{supp}(x)$. Then, x is optimal to $\mathrm{LP}(\tilde{c})$ and $\|\tilde{c}-s\|_\infty<\delta$. Since $c-s\in\mathrm{Im}(A^\top)$, the vector $c'\coloneqq c-s+\tilde{c}$ satisfies $\|c-c'\|_\infty<\delta$, and $\mathrm{LP}(c')$ and $\mathrm{LP}(\tilde{c})$ are equivalent. Thus, x and $\tilde{c}$ are primal and dual optimal solutions to $\mathrm{LP}(c')$. Then, Theorem 3.14 is applicable for the costs c,c' and the dual optimal solution $\tilde{c}$; however, we also need to guarantee that $N\neq\emptyset$. Following Tardos [145, 143], this can be ensured if we pre-process by projecting the cost vector c to $\ker(A)$; this guarantees that $\|s\|$—and thus $\|\tilde{c}\|$—must be sufficently large.

The above property may however not hold for (x,s): for certain coordinates we could have $x_i>0$ and $s_i\ge\delta$. In this case, we start the second *phase* of the algorithm. Since $x_is_i\le\langle x,s\rangle\le\varepsilon$, this implies $x_i\le\varepsilon/\delta$. Let S be the set of all such violating indices. Since $\|x_S\|$ is sufficiently small, one can show that the set of 'large' indices $\mathcal{L}=\{i\in[n]:x_i\ge\Gamma\|x_S\|_1\}$ is nonempty. We proceed by defining a new cost function $\tilde{c}_i\coloneqq s_i$ if $i\in S$ and $\tilde{c}_i\coloneqq 0$ if $i\notin S$. We perform Support-Circuit iterations as long as there exist circuits in $\mathrm{supp}(x)$ intersecting $\mathrm{supp}(\tilde{c})$, and then perform further $O(n\log(n+\kappa_A))$ ratio cycle iterations for the cost function $\tilde{c}$. If we now arrive at an iterate $(x,s)=(x^{(t')},s^{(t')})$ such that $s_i<\delta$ for every $i\in\mathrm{supp}(x)$, then we truncate s as before to an optimal dual solution to $\mathrm{LP}(c'')$ for some vector c'' where $\|c-c''\|_\infty<2\delta$. After that, Theorem 3.14 is applicable for the costs c,c'' and said optimal dual solution. Otherwise, we continue with additional phases.

The algorithm formalizes the above idea, with some technical modifications. The algorithm comprises at most $m+1$ phases; the main potential is that the rank of the large index set $\mathcal{L}$ increases in every phase. We show that if an index i was added to $\mathcal{L}$ in any phase, it must have $s_i<\delta$ at the beginning of every later phase. Thus, these indices cannot be violating anymore.

We now turn to a more formal description of Algorithm 8. We start by orthogonally projecting the input cost vector c to $\ker(A)$. This does not change the optimal face of (LP). If $c=\mathbb{0}$, then we terminate and return the current feasible solution $x^{(0)}$ as it is optimal. Otherwise, we scale the cost to $\|c\|_2=1$, and use Ratio-Circuit to obtain a basic feasible solution $\tilde{s}^{(-1)}$ to the dual of $\mathrm{LP}(c)$.

The rest of Algorithm 8 consists of repeated *phases*, ending when $\left\langle \tilde{s}^{(t-1)},x^{(t)}\right\rangle=0$. In an iteration t, let $S_t=\{i\in[n]:\tilde{s}_i^{(t-1)}\ge\delta\}$ be the set of coordinates with large dual slack. The algorithm keeps track of the following set

$$\mathcal{L}_t\coloneqq\mathcal{L}_{t-1}\cup\left\{i\in[n]:x_i^{(t)}\ge\Gamma\|x_{S_t}^{(t)}\|_1\right\}.$$

Algorithm 8: VARIABLE-FIXING

```
Input   : Linear program in standard form (LP), and initial feasible solution $x^{(0)}$.
Output  : Either an optimal solution to (LP), or a feasible solution $x$ and
          $\emptyset \neq N \subseteq [n]$ such that $x_N = x^*_N = \mathbb{0}$ for every optimal solution $x^*$ to (LP).
1  $t \leftarrow 0$; $k \leftarrow 0$; $\mathcal{L}_{-1} \leftarrow \emptyset$
2  $c \leftarrow \Pi_{\ker(A)}(c)$
3  if $c = \mathbb{0}$ then
4      return $x^{(0)}$
5  $c \leftarrow c/\|c\|_2$
6  $(\cdot, \cdot, \tilde{s}^{(-1)}) \leftarrow$ RATIO-CIRCUIT$(A, c, \mathbb{1})$        ▷ Or any bfs to the dual of LP$(c)$
7  while $\left\langle \tilde{s}^{(t-1)}, x^{(t)} \right\rangle > 0$ do
8      $S_t \leftarrow \{i \in [n] : \tilde{s}_i^{(t-1)} \geq \delta\}$
9      $\mathcal{L}_t \leftarrow \mathcal{L}_{t-1} \cup \{i \in [n] : x_i^{(t)} \geq \Gamma\|x_{S_t}^{(t)}\|_1\}$
10     if $t = 0$ or rk$(\mathcal{L}_t) >$ rk$(\mathcal{L}_{t-1})$ then
11         $k \leftarrow k + 1$                                                ▷ New phase
12         Set modified cost $\tilde{c}^{(k)} \in \mathbb{R}^n_+$ as $\tilde{c}_i^{(k)} \leftarrow \tilde{s}_i^{(t-1)}$ if $i \in S_t$, and $\tilde{c}_i^{(k)} \leftarrow 0$ otherwise
13         while $\exists$ a circuit in supp$(x^{(t)})$ intersecting supp$(\tilde{c}^{(k)})$ do
14             $g^{(t)} \leftarrow$ SUPPORT-CIRCUIT$(A, \tilde{c}^{(k)}, x^{(t)}, \text{supp}(\tilde{c}^{(k)}))$
15             $x^{(t+1)} \leftarrow \text{aug}_P(x^{(t)}, g^{(t)})$
16             $\mathcal{L}_{t+1} \leftarrow \mathcal{L}_t$; $t \leftarrow t + 1$
17     $(g^{(t)}, y^{(t)}, s^{(t)}) \leftarrow$ RATIO-CIRCUIT$(A, \tilde{c}^{(k)}, 1/x^{(t)})$
18     $x^{(t+1)} \leftarrow \text{aug}_P(x^{(t)}, g^{(t)})$
19     $\tilde{s}^{(t)} \leftarrow \arg\min_{s \in \{\tilde{c}^{(k)}, s^{(t)}\}} \left\langle s, x^{(t+1)} \right\rangle$; $t \leftarrow t + 1$
20 $N \leftarrow \{i \in [n] : \tilde{s}_i^{(t-1)} > \kappa_A(m + 1)n\delta\}$
21 return $(x^{(t)}, N)$
```

These are the variables that were once large with respect to $\|x_{S'_t}^{(t')}\|_1$ in iteration $t' \leq t$. Note that $|\mathcal{L}_t|$ is monotone nondecreasing.

The first phase starts at $t = 0$, and we enter a new phase k whenever $\text{rk}(\mathcal{L}_t) > \text{rk}(\mathcal{L}_{t-1})$. The iteration t is called the *first iteration* in phase k. At the start of the phase, we define a new *modified cost* $\tilde{c}^{(k)}$ from the dual slack $\tilde{s}^{(t-1)}$ by truncating entries less than δ to 0. This cost vector will be used until the end of the phase. Then, we call SUPPORT-CIRCUIT$(A, \tilde{c}^{(k)}, x^{(t)}, \text{supp}(\tilde{c}^{(k)}))$ to eliminate circuits in $\text{supp}(x^{(t)})$ intersecting $\text{supp}(\tilde{c}^{(k)})$. Note that there are at most n such calls because each call sets a primal variable $x_i^{(t)}$ to zero.

In the remaining part of the phase, we augment $x^{(t)}$ using RATIO-CIRCUIT$(A, \tilde{c}^{(k)}, 1/x^{(t)})$ until $\text{rk}(\mathcal{L}_t)$ increases, triggering a new phase. In every iteration, RATIO-CIRCUIT$(A, \tilde{c}^{(k)}, 1/x^{(t)})$ returns a minimum cost-to-weight ratio circuit $g^{(t)}$, where the choice of weights $1/x^{(t)}$ follows Wallacher [154]. It also returns a basic feasible solution $(y^{(t)}, s^{(t)})$ to the dual LP$(\tilde{c}^{(k)})$. After augmenting $x^{(t)}$ to $x^{(t+1)}$ using $g^{(t)}$, we update the dual slack as

$$\tilde{s}^{(t)} := \operatorname*{arg\,min}_{s \in \{\tilde{c}^{(k)}, s^{(t)}\}} \left\langle s, x^{(t+1)} \right\rangle .$$

This finishes the description of a phase.

Since $\operatorname{rk}(A) = m$, clearly there are at most $m+1$ phases. Let k and t be the final phase and iteration of Algorithm 8 respectively. As $\left\langle \tilde{s}^{(t-1)}, x^{(t)} \right\rangle = 0$, and $x^{(t)}, \tilde{s}^{(t-1)}$ are primal-dual feasible solutions to $\mathrm{LP}(\tilde{c}^{(k)})$, they are also optimal. Now, it is not hard to see that $\tilde{c}^{(k)} \in \operatorname{Im}(A^\top) + c - r$ for some $\mathbb{0} \le r \le (m+1)\delta$ (Claim 3.25). Hence, $\tilde{s}^{(t-1)}$ is also an optimal solution to the dual of $\mathrm{LP}(c-r)$. The last step of the algorithm consists of identifying the set N of coordinates with large dual slack $\tilde{s}_i^{(t-1)}$. Then, applying Theorem 3.14 for $c' = c - r$ allows us to conclude that they can be fixed to zero.

In order to prove Theorem 3.4, we need to show that $N \neq \emptyset$. Moreover, we need to show that there are at most T iterations of Ratio-Circuit per phase. First, we show that the objective value is monotone nonincreasing.

Claim 3.22. *For any two iterations $r \ge t$ in phases $\ell \ge k \ge 1$ respectively, we have*

$$\left\langle \tilde{c}^{(\ell)}, x^{(r)} \right\rangle \le \left\langle \tilde{c}^{(k)}, x^{(t)} \right\rangle .$$

Proof. We proceed by strong induction on $\ell - k$. For the base case $\ell - k = 0$, iterations r and t occur in the same phase. So, the objective value is nonincreasing from the definition of Support Circuit and Ratio-Circuit. Now, suppose that the statement holds if $\ell - k \le d$ for some $d \ge 0$, and consider the case $\ell - k = d + 1$. Let q be the first iteration in phase $k+1$; note that $r \ge q > t$. Then, we have

$$\left\langle \tilde{c}^{(\ell)}, x^{(r)} \right\rangle \le \left\langle \tilde{c}^{(k+1)}, x^{(q)} \right\rangle \le \left\langle \tilde{s}^{(q-1)}, x^{(q)} \right\rangle \le \left\langle \tilde{c}^{(k)}, x^{(q)} \right\rangle \le \left\langle \tilde{c}^{(k)}, x^{(t)} \right\rangle .$$

The first inequality uses the inductive hypothesis. In the second inequality, we use that $\tilde{c}^{(k+1)}$ is obtained from $\tilde{s}^{(q-1)}$ by setting some nonnegative coordinates to 0. The third inequality is by the definition of $\tilde{s}^{(q-1)}$. The final inequality is by monotonicity within the same phase. □

The following claim gives a sufficient condition for Algorithm 8 to terminate.

Claim 3.23. *Let t be an iteration in phase $k \ge 1$. If* Ratio-Circuit *returns an elementary vector $g^{(t)}$ such that $\left\langle \tilde{c}^{(k)}, g^{(t)} \right\rangle = 0$, then $\left\langle \tilde{s}^{(t)}, x^{(t+1)} \right\rangle = 0$.*

Proof. Recall that the weights w in Ratio-Circuit are chosen as $w = 1/x^{(t)}$. The constraint $s^{(t)} \le \lambda w$ in (3.3) and strong duality $\lambda = -\left\langle \tilde{c}^{(k)}, g^{(t)} \right\rangle$, therefore imply that $s_i^{(t)} x_i^{(t)} \le \lambda = -\left\langle \tilde{c}^{(k)}, g^{(t)} \right\rangle$ and in particular $\left\langle s^{(t)}, x^{(t)} \right\rangle \le -n \left\langle \tilde{c}^{(k)}, g^{(t)} \right\rangle = 0$. Since $\tilde{s}^{(t)}, x^{(t+1)} \ge \mathbb{0}$, we have

$$0 \le \left\langle \tilde{s}^{(t)}, x^{(t+1)} \right\rangle \le \left\langle s^{(t)}, x^{(t+1)} \right\rangle \le \left\langle s^{(t)}, x^{(t)} \right\rangle = 0.$$ □

Corollary 3.24. *Let t be an iteration in phase $k \geq 1$. If $\left\langle \tilde{c}^{(k)}, x^{(t)} \right\rangle = 0$, then Algorithm 8 terminates in iteration t or $t+1$.*

Proof. Suppose that the algorithm does not terminate in iteration t. Then, RATIO-CIRCUIT is called in iteration t because there is no circuit in $\mathrm{supp}(x^{(t)})$ which intersects $\tilde{c}^{(k)}$. Since $x^{(t)}$ is an optimal solution to $\mathrm{LP}(\tilde{c}^{(k)})$, RATIO-CIRCUIT returns an elementary vector $g^{(t)}$ such that $\left\langle \tilde{c}^{(k)}, g^{(t)} \right\rangle = 0$. By Claim 3.23, the algorithm terminates in the next iteration. $\square$

The next two claims provide some basic properties of the modified cost $\tilde{c}^{(k)}$.

Claim 3.25. *For every phase $k \geq 1$, we have $\tilde{c}^{(k)} \in \mathrm{Im}(A^\top) + c - r$ for some $\mathbb{0} \leq r \leq k\delta\mathbb{1}$.*

Proof. Let us define $\tilde{c}^{(0)} := c$; we proceed by induction on $k \geq 0$. The base case $k = 0$ is trivial. Now, suppose that the statement holds for some $k \geq 0$, and consider the case $k+1$. Let t be the iteration in which $\tilde{c}^{(k+1)}$ is set, i.e., $\tilde{c}_i^{(k+1)} = \tilde{s}_i^{(t-1)}$ if $i \in S_t$, and $\tilde{c}_i^{(k+1)} = 0$ otherwise. Note that $\tilde{s}^{(t-1)} \in \{\tilde{c}^{(k)}, s^{(t-1)}\}$. Since both of them are feasible to the dual of $\mathrm{LP}(\tilde{c}^{(k)})$, we have $\tilde{s}^{(t-1)} \in \mathrm{Im}(A^\top) + \tilde{c}^{(k)}$. By the inductive hypothesis, $\tilde{c}^{(k)} \in \mathrm{Im}(A^\top) + c - r$ for some $\mathbb{0} \leq r \leq k\delta\mathbb{1}$. Hence, from the definition of $\tilde{c}^{(k+1)}$, we have $\tilde{c}^{(k+1)} \in \mathrm{Im}(A^\top) + c - r - q$ for some $\mathbb{0} \leq q \leq \delta\mathbb{1}$ as required. $\square$

Claim 3.26. *In every phase $k \geq 1$, we have $\|\tilde{c}^{(k)}\|_\infty \leq 2\sqrt{n}\kappa_A$.*

Proof. Let us define $\tilde{c}^{(0)} := c$; we proceed by induction on $k \geq 0$. The base case $k = 0$ is easy because $\|c\|_\infty \leq \|c\|_2 = 1$. Now, suppose that the statement holds for some $k \geq 0$, and consider the case $k+1$. Let t be the iteration in which $\tilde{c}^{(k+1)}$ is set. If $\tilde{s}^{(t-1)} = \tilde{c}^{(k)}$, then $\tilde{c}^{(k+1)} = \tilde{c}^{(k)}$ and we are done by the inductive hypothesis. Otherwise, $\tilde{s}^{(t-1)} = s^{(t-1)}$. We know that $s^{(t-1)}$ is a basic feasible solution to the dual of $\mathrm{LP}(\tilde{c}^{(k)})$. We also know that $c - r \in \mathrm{Im}(A^\top) + \tilde{c}^k$ for some $\mathbb{0} \leq r \leq k\delta\mathbb{1}$ by Claim 3.25. Let B be a matrix such that $\ker(B) = \mathrm{Im}(A^\top)$. Then, applying Corollary 3.13 to the system $Bs^{(t-1)} = B\tilde{c}^{(k)}$ yields

$$\begin{aligned}\|s^{(t-1)}\|_\infty \leq \kappa_B\|c - r\|_1 = \kappa_A\|c - r\|_1 \leq \kappa_A(\|c\|_1 + \|r\|_1)\\ \leq \kappa_A\big(\sqrt{n} + nk\delta\big) \leq \kappa_A\big(\sqrt{n} + n(m+1)\delta\big) \leq 2\sqrt{n}\kappa_A.\end{aligned}$$

The equality is by Lemma 3.5, the 3rd inequality is due to $\|c\|_2 = 1$, while the 4th inequality follows from the fact that there are at most $m+1$ phases. $\square$

We next show a primal proximity lemma that holds for iterates throughout the algorithm.

Lemma 3.27. *For iterations $t' \geq t$ we have*

$$\|x^{(t')} - x^{(t)}\|_\infty \leq (t' - t)\frac{2\sqrt{n}\kappa_A^2}{\delta}\|x_{S_t}^{(t)}\|_1\,. \tag{3.12}$$

Proof. Consider any iteration $r \geq t$, and assume that Algorithm 8 does not terminate in iteration $r+1$. Let ℓ and k be the phase in which iteration r and t occurred respectively. The elementary vector $g^{(r)}$ is returned by either SUPPORT-CIRCUIT or RATIO-CIRCUIT. We claim that $g_i^{(r)} < 0$ for some $i \in \operatorname{supp}(\tilde{c}^{(\ell)})$. In the former, this trivially follows from the definition of SUPPORT-CIRCUIT. In the latter, we know that $\left\langle \tilde{c}^{(\ell)}, g^{(r)} \right\rangle < 0$ by Claim 3.23 because Algorithm 8 does not terminate in iteration $r+1$. It follows that $g_i^{(r)} < 0$ for some $i \in \operatorname{supp}(\tilde{c}^{(\ell)})$. Then,

$$\|x^{(r+1)} - x^{(r)}\|_\infty \leq \kappa_A |x_i^{(r)}| \leq \frac{\kappa_A}{\delta} \left\langle \tilde{c}^{(\ell)}, x^{(r)} \right\rangle \leq \frac{\kappa_A}{\delta} \left\langle \tilde{c}^{(k)}, x^{(t)} \right\rangle \leq \frac{2\sqrt{n}\kappa_A^2}{\delta} \|x_{S_t}^{(t)}\|_1.$$

The second inequality uses that all nonzero coordinates of $\tilde{c}^{(\ell)}$ are $\geq \delta$. The third inequality is by Claim 3.22, whereas the fourth inequality is by Claim 3.26. The result follows by summing over all iterations in $\{t, \ldots, t'-1\}$. □

We next show that once a variable enters $\mathcal{L}_t$, it is lower bounded by $\operatorname{poly}(n, \kappa_A)\|x_{S_t}^{(t)}\|_1$ in the next $\Theta(mT)$ iterations.

Claim 3.28. *Let t be the iteration in which coordinate j is added to $\mathcal{L}_t$. Let $t' \geq t$ be a non-final iteration such that $t' - t \leq 2(m+1)T$. We have*

$$x_j^{(t')} \geq \frac{3\sqrt{n}\kappa_A}{\delta} \|x_{S_t}^{(t)}\|_1.$$

Proof. By definition, we have that $x_j^{(t)} \geq \Gamma \|x_{S_t}^{(t)}\|_1$. With Lemma 3.27 we get

$$x_j^{(t')} \geq x_j^{(t)} - \|x^{(t')} - x^{(t)}\|_\infty \geq \left(\Gamma - \frac{4(m+1)\sqrt{n}\kappa_A^2 T}{\delta}\right) \|x_{S_t}^{(t)}\|_1 \geq \frac{3\sqrt{n}\kappa_A}{\delta} \|x_{S_t}^{(t)}\|_1 . \qquad \square$$

Consequently, the set $\mathcal{L}_t$ is disjoint from the support of the modified cost $\tilde{c}^{(k)}$ in the next $\Theta(mT)$ iterations.

Corollary 3.29. *Let t be an iteration in phase $k \geq 1$ such that $t \leq 2(m+1)T$. If t is not the final iteration, then*

$$\mathcal{L}_t \cap \operatorname{supp}(\tilde{c}^{(k)}) = \emptyset .$$

Proof. For the purpose of contradiction, suppose that there exists an index $i \in \mathcal{L}_t \cap \operatorname{supp}(\tilde{c}^{(k)})$ in some iteration $t \leq 2(m+1)T$ in phase k. Let $r \leq t$ be the iteration in which i was added to $\mathcal{L}_r$. Let j and k be the phase which contains iteration r and t respectively. Since r is the first iteration in phase j, we have $S_r = \operatorname{supp}(\tilde{c}^{(j)})$. Hence, we have $r < t$, as otherwise it would contradict the definition of $\mathcal{L}_t$. Moreover, we have $\|x_{S_r}^{(r)}\|_1 > 0$ by Corollary 3.24, as otherwise the algorithm would have terminated in iteration $r+1 \leq t$. However, we get the following contradiction

$$3\sqrt{n}\kappa_A\|x^{(r)}_{S_r}\|_1 \le \delta x_i^{(t)} \le \left\langle \tilde{c}^{(k)}, x^{(t)} \right\rangle \le \left\langle \tilde{c}^{(j)}, x^{(r)} \right\rangle \le 2\sqrt{n}\kappa_A\|x^{(r)}_{S_r}\|_1.$$

The 1st inequality is by Claim 3.28, the 3rd inequality is by Claim 3.22, while the 4th inequality is by Claim 3.26. □

The next claim shows that Ratio-Circuit geometrically decreases the norm $\|x^{(t)}_{S_t}\|_1$.

Claim 3.30. *Let t be the first* Ratio-Circuit *iteration in phase $k \ge 1$. After $p \in \mathbb{N}$ consecutive* Ratio-Circuit *iterations in phase k,*

$$\|x^{(t+p)}_{S_{t+p}}\|_1 \le \frac{2n^{1.5}\kappa_A}{\delta}\left(1-\frac{1}{n}\right)^{p-1}\|x^{(t)}_{\mathrm{supp}(\tilde{c}^{(k)})}\|_1,$$

Proof.

$$\begin{aligned}
\|x^{(t+p)}_{S_{t+p}}\|_1 &\le \frac{1}{\delta}\left\langle \tilde{s}^{(t+p-1)}, x^{(t+p)} \right\rangle \\
&\le \frac{1}{\delta}\left\langle s^{(t+p-1)}, x^{(t+p)} \right\rangle && \text{(from the definition of } \tilde{s}^{(t+p-1)}) \\
&\le \frac{1}{\delta}\left\langle s^{(t+p-1)}, x^{(t+p-1)} \right\rangle && \text{(as } \left\langle s^{(t+p-1)}, g^{(t+p-1)} \right\rangle = \left\langle \tilde{c}^{(k)}, g^{(t+p-1)} \right\rangle \le 0) \\
&\le -\frac{n}{\delta}\left\langle \tilde{c}^{(k)}, g^{(t+p-1)} \right\rangle && \text{(due to the constraints in (3.3))} \\
&\le \frac{n}{\delta}\left(\left\langle \tilde{c}^{(k)}, x^{(t+p-1)} \right\rangle - \mathrm{OPT}(\tilde{c}^{(k)})\right) && \text{(by step size } \alpha \ge 1 \text{ in Lemma 3.8)} \\
&\le \frac{n}{\delta}\left(1-\frac{1}{n}\right)^{p-1}\left(\left\langle \tilde{c}^{(k)}, x^{(t)} \right\rangle - \mathrm{OPT}(\tilde{c}^{(k)})\right) && \text{(by geometric decay in Lemma 3.8)} \\
&\le \frac{n}{\delta}\left(1-\frac{1}{n}\right)^{p-1}\left\langle \tilde{c}^{(k)}, x^{(t)} \right\rangle && \text{(because } \tilde{c}^{(k)} \ge \mathbb{0}) \\
&\le \frac{2n^{1.5}\kappa_A}{\delta}\left(1-\frac{1}{n}\right)^{p-1}\|x^{(t)}_{\mathrm{supp}(\tilde{c}^{(k)})}\|_1. && \text{(by Claim 3.26)}
\end{aligned}$$

□

Before proving the main result of this section, we recall Lemma 3.15 which guarantees the existence of a coordinate with large dual slack. It explains why we chose to work with a projected and normalized cost vector in Algorithm 8.

Proof of Theorem 3.4. We first prove the correctness of Algorithm 8. Suppose that the algorithm terminates in iteration t. Since $\left\langle \tilde{s}^{(t-1)}, x^{(t)} \right\rangle = 0$ and $x^{(t)}, \tilde{s}^{(t-1)}$ are primal-dual feasible solutions to $\mathrm{LP}(\tilde{c}^{(k)})$, they are also optimal. By Claim 3.25, we know that $\tilde{c}^{(k)} \in \mathrm{Im}(A^\top) + c - r$ for some $\|r\|_\infty \le (m+1)\delta$. Hence, $\tilde{s}^{(t-1)}$ is also an optimal dual solution to $\mathrm{LP}(c')$ where $c' := c - r$. Since $c \in \ker(A)$, $\|c\|_2 = 1$ and $\|c - c'\|_\infty \le (m+1)\delta < 1/(\sqrt{n}(m+2)\kappa_A)$, by Lemma 3.15, there exists an index $j \in [n]$ such that $s_j^{(t)} > (m+1)\kappa_A n\delta \ge$

$(m+1)\kappa_A\|c-c'\|_\infty$. Thus, the algorithm returns $N \neq \emptyset$. Moreover, for all $j \in N$, Theorem 3.14 allows us to conclude that $x_j^{(t)} = x_j^* = 0$ for every optimal solution x^* to LP(c).

Next, we prove the running time of Algorithm 8. Clearly, there are at most $m+1$ phases. In every phase, there are at most n SUPPORT-CIRCUIT iterations because each call sets a primal variable to 0. It is left to show that there are at most T RATIO-CIRCUIT iterations in every phase.

Fix a phase $k \geq 1$ and assume that every phase $\ell < k$ consists of at most T many iterations. Further, let t be the first iteration in phase k. Note that $S_t = \operatorname{supp}(\tilde{c}^{(k)})$. Without loss of generality, we may assume that $\|x_{S_t}^{(t)}\|_1 > 0$. Otherwise, the algorithm would have terminated in iteration $t+1$ by Corollary 3.24. Note that this assumption also implies that $\|x_{S_{t'}}^{(t')}\|_1 > 0$ for all $t' < t$. This is because if $\|x_{S_{t'}}^{(t')}\|_1 = 0$ for some $t' < t$, then $\mathcal{L}_{t'} = [n]$ and $\operatorname{rk}(\mathcal{L}_{t'}) = m$, which contradicts $\operatorname{rk}(\mathcal{L}_{t'}) < \operatorname{rk}(\mathcal{L}_t)$.

Let $r \geq t$ be the first RATIO-CIRCUIT iteration in phase k. Since r is not the final iteration, we have $x_{\mathcal{L}_r}^{(r)} > \mathbb{0}$ due to Claim 3.28 and our assumptions. We split the remaining analysis into the following 2 cases:

Case 1: $\operatorname{rk}(\mathcal{L}_r) = m$. In this case, we have $x_{\operatorname{supp}(\tilde{c}^{(k)})}^{(r)} = \mathbb{0}$. Indeed, if $x_i^{(r)} > 0$ for some $i \in \operatorname{supp}(\tilde{c}^{(k)})$, then $\mathcal{L}_r \cup \{i\} \subseteq \operatorname{supp}(x^{(r)})$ contains a circuit. So, SUPPORT-CIRCUIT would have been called in iteration r, which is a contradiction. By Corollary 3.24, the algorithm terminates in the next iteration.

Case 2: $\operatorname{rk}(\mathcal{L}_r) < m$. In this case, we show that $\operatorname{rk}(\mathcal{L}_r)$ increases in at most T iterations of RATIO-CIRCUIT. We already know that $\mathcal{L}_r \subseteq \operatorname{supp}(x^{(r)})$. Moreover, there is no circuit in $\operatorname{supp}(x^{(r)})$ which intersects $\operatorname{supp}(\tilde{c}^{(k)})$. Since $\operatorname{supp}(\tilde{c}^{(k)}) \subseteq [n] \setminus \mathcal{L}_r$ by Corollary 3.29, we can apply Lemma 3.11 to get

$$\|x_{[n]\setminus\operatorname{cl}(\mathcal{L}_r)}^{(r+T)}\|_\infty \geq \frac{\|x_{[n]\setminus\operatorname{cl}(\mathcal{L}_r)}^{(r+T)}\|_1}{n} \geq \frac{\|x_{\operatorname{supp}(\tilde{c}^{(k)})}^{(r)}\|_\infty}{n\kappa_A} \geq \frac{\|x_{\operatorname{supp}(\tilde{c}^{(k)})}^{(r)}\|_1}{n^2\kappa_A} \geq \Gamma\|x_{S_{r+T}}^{(r+T)}\|_1 ,$$

where the last inequality follows from Claim 3.30. Thus, there exists an index $i \in [n] \setminus \operatorname{cl}(\mathcal{L}_r)$ which is added to $\mathcal{L}_{r+T}$, showing that $\operatorname{rk}(\mathcal{L}_{r+T}) > \operatorname{rk}(\mathcal{L}_r)$ as required.

Since the main circuit-augmentation algorithm consists of applying Algorithm 8 at most n times, we obtain the desired runtime.

□

3.7 Circuits in General Form

In [90], circuits are defined for polyhedra in the general form

$$P = \{x \in \mathbb{R}^n : Ax = b,\, Bx \leq d\}\,, \tag{3.13}$$

where $A \in \mathbb{R}^{m_A \times n}$, $B \in \mathbb{R}^{m_B \times n}$, $b \in \mathbb{R}^{m_A}$, $c \in \mathbb{R}^{m_B}$. For this setup, they define $g \in \mathbb{R}^n$ to be an elementary vector if

(i) $g \in \ker(A)$, and

(ii) Bg is support minimal in the collection $\{By : y \in \ker(A), y \neq 0\}$.

In [90], they use the term 'circuit' also for elementary vectors.

Let us assume that

$$\operatorname{rk}\begin{pmatrix} A \\ B \end{pmatrix} = n\,. \tag{3.14}$$

This assumption is needed to ensure that P is pointed; otherwise, there exists a vector $z \in \mathbb{R}^n$, $z \neq 0$ such that $Az = 0$, $Bz = 0$. Thus, the lineality space of P is nonempty. Note that the circuit diameter is defined as the maximum length of a circuit walk between two vertices; this implicitly assumes that vertices exists and therefore the lineality space is empty.

Under this assumption, we show that circuits in the above definition are a special case of our definition in the Introduction, and explain how our results in the standard form are applicable. Consider the matrix and vector

$$M := \begin{pmatrix} A & 0 \\ B & I_{m_B} \end{pmatrix}, \; q := \begin{pmatrix} b \\ d \end{pmatrix},$$

and let $\bar{W} := \ker(M) \subseteq \mathbb{R}^{n+m_B}$. Let J denote the set of the last m_B indices, and $W := \pi_J(\bar{W})$ denote the coordinate projection to J. The assumption (3.14) guarantees that for each $s \in W$, there is a unique $(x, s) \in \bar{W}$; further, $x \neq 0$ if and only if $s \neq 0$.

Note that P is the projection of the polyhedron

$$\bar{P} = \{(x, s) \in \mathbb{R}^n \times \mathbb{R}^{m_B} : M(x, s) = q\,, s \geq 0\}$$

to the x variables. Let $Q := \pi_J(P) \subseteq \mathbb{R}^{m_B}$ denote the projection of $\bar{P}$ to the s variables. It is easy to verify the following statements.

Lemma 3.31. *If (3.14) holds, then there is an invertible affine one-to-one mapping ψ between Q and P, defined by*

$$M(\psi(s), s) = q\,.$$

Further, $g \in \mathbb{R}^n$ is an elementary vector as in (i),(ii) *above if and only if there exists $h \in \mathbb{R}^{m_B}$ such that $(g, h) \in \bar{W}$, $h \neq 0$ and h is support minimal.*

Given such a pair $(g, h) \in \bar{W}$ of elementary vectors, let $s \in Q$ and let $s' := \operatorname{aug}_Q(s, h)$ denote the result of the circuit augmentation starting from s. Then, $\psi(s') = \operatorname{aug}_P(\psi(s), g)$.

Consequently, the elementary vectors according to the definition in [90] are in one-to-one mapping to elementary vectors in the subspace W as used in this chapter. By the last part of the statement, analyzing circuit walks on P reduces to analyzing circuit walks of Q that is given in the subspace form $Q = \{s \in \mathbb{R}^{m_B} : s \in W + r, s \geq 0\}$.

Chapter 4

Submodular Optimization: Correlation Gap Bounds for Matroids

4.1 Introduction

A continuous function $h\colon [0,1]^E \to \mathbb{R}_+$ is an *extension* of a set function $f\colon 2^E \to \mathbb{R}_+$ if for every $x \in [0,1]^E$, $h(x) = \mathbb{E}_\lambda[f(S)]$ where λ is a probability distribution over 2^E with marginals x, i.e. $\sum_{S:i\in S} \lambda_S = x_i$. Note that this in particular implies $f(X) = h(\chi_X)$ for every $X \subseteq E$, where χ_X denotes the 0-1 indicator vector of X.

Two natural extensions are the following. The first one corresponds to sampling each $i \in E$ independently with probability x_i, i.e., $\lambda_S = \prod_{i\in S} x_i \prod_{i\notin S}(1-x_i)$. Thus,

$$F(x) = \sum_{S\subseteq E} f(S) \prod_{i\in S} x_i \prod_{i\notin S}(1-x_i) \ . \tag{4.1}$$

This is known as the *multilinear extension* in the context of submodular optimization, see [26]. The second extension corresponds to the probability distribution with maximum expectation:

$$\hat{f}(x) = \max\left\{ \sum_{S\subseteq E} \lambda_S f(S) : \sum_{S\subseteq E: i\in S} \lambda_S = x_i \ \forall i \in E \,,\ \sum_{S\subseteq E} \lambda_S = 1 \,,\ \lambda \geq 0 \right\} \ . \tag{4.2}$$

Equivalently, $\hat{f}(x)$ is the upper part of the convex hull of the graph of f; we call it the *concave extension* following the terminology of discrete convex analysis [109].

Agrawal, Ding, Saberi and Ye [2] introduced the *correlation gap* as the worst case ratio

$$\mathcal{CG}(f) := \min_{x\in[0,1]^E} \frac{F(x)}{\hat{f}(x)} \ . \tag{4.3}$$

It captures the maximum gain achievable by allowing correlations as opposed to independently sampling the variables. This ratio plays a fundamental role in stochastic optimization [2, 115], mechanism design [12, 78, 159], prophet inequalities [29, 51, 128], and a variety of submodular optimization problems [6, 30].

The focus of this chapter is on monotone nondecreasing submodular functions. A function $f\colon 2^E \to \mathbb{R}_+$ is submodular if

$$f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y) \quad \forall X, Y \subseteq E \ .$$

Throughout the chapter, we assume the submodular function also satisfies $f(\emptyset) = 0$.

The lower bound $\mathcal{CG}(f) \geq 1 - 1/e$ holds for every submodular function [26]. In fact, the extreme case $1 - 1/e$ is already taken in the limit by the rank function of a rank-1 uniform matroid if $|E| \to \infty$.

We study the correlation gap of matroid rank functions and weighted matroid rank functions, which are monotone submodular; see Section 4.2 for the definitions. First, we show that among all weighted rank functions of a matroid, the worst case correlation gap is realized by its (unweighted) rank function.

Theorem 4.1. *Fix a matroid $\mathcal{M} = (E, \mathcal{I})$, and consider the weighted matroid rank functions for every choice of nonnegative weights on E. The worst case correlation gap is realized by the uniform-1 weighting.*

For matroid rank functions, we give a lower bound on the correlation gap as a function of the rank and girth of the matroid. Recall that the *girth* is the minimum length of a circuit in the matroid, or equivalently, the smallest size of a dependent set.

Theorem 4.2. *Let $\mathcal{M} = (E, \mathcal{I})$ be a loopless matroid with rank function r, rank $r(E) = \rho$, and girth γ. Then,*

$$\mathcal{CG}(r) \geq 1 - \frac{1}{e} + \frac{e^{-\rho}}{\rho}\left(\sum_{i=0}^{\gamma-2}(\gamma - 1 - i)\left[\binom{\rho}{i}(e-1)^i - \frac{\rho^i}{i!}\right]\right) > 1 - \frac{1}{e} \, .$$

We refer the reader to Figure 4.1 to understand the behaviour of the expression in Theorem 4.2. For any fixed girth γ, it is monotone decreasing in ρ (Lemma 4.38). On the other hand, for any fixed rank ρ, it is monotone increasing in γ.

In Section 4.1.4, we explain the motivation for studying the girth and rank as relevant parameters, and highlight the key proof ideas. Other than for uniform matroids, we are not aware of any previous work on giving better than $(1 - 1/e)$ bounds on the correlation gaps of specific matroids. We remark that for the purpose of determining $\mathcal{CG}(r)$, we may assume that the matroid $\mathcal{M}$ is *connected*, that is, it cannot be written as a direct sum of at least two nonempty matroids. If $\mathcal{M} = \mathcal{M}_1 \oplus \cdots \oplus \mathcal{M}_k$, let r_i be the rank function of $\mathcal{M}_i$ for all

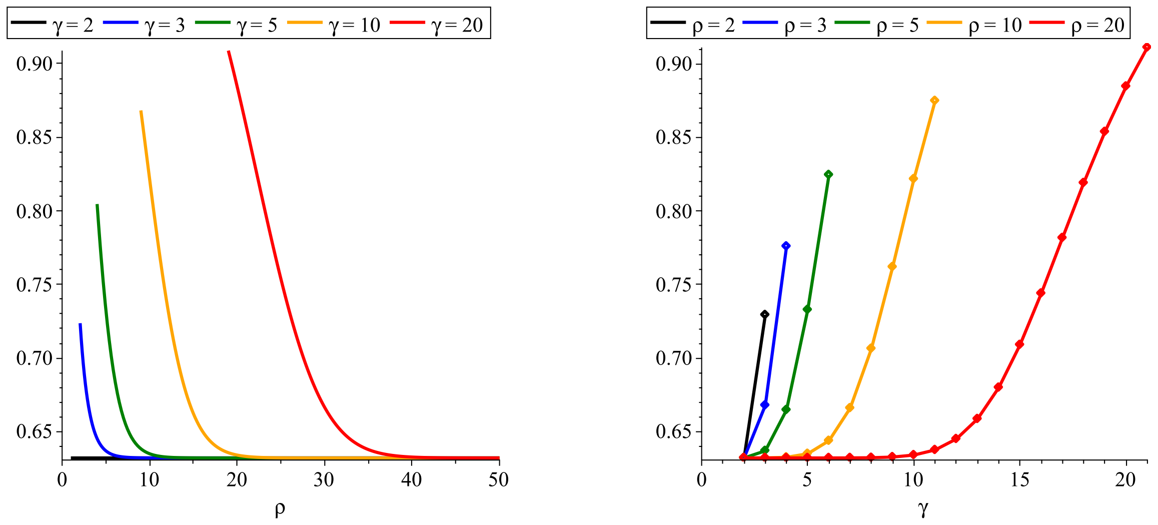

Fig. 4.1 Our correlation gap bound as a function of the rank ρ, and as a function of the girth γ separately.

$i \in [k]$. Then, the concave and multilinear extensions of r can be written as $\hat{r} = \sum_{i=1}^{k} \hat{r}_i$ and $R = \sum_{i=1}^{k} R_i$ respectively. Hence, we have $\mathcal{CG}(r) \geq \min_{i\in[k]} \mathcal{CG}(r_i)$ by the mediant inequality. As the reverse inequality holds trivially, it follows that $\mathcal{CG}(r) = \min_{i\in[k]} \mathcal{CG}(r_i)$.

Proposition 4.3. *Let $\mathcal{M}$ be a matroid with rank function r. If $\mathcal{M} = \mathcal{M}_1 \oplus \cdots \oplus \mathcal{M}_k$ where each $\mathcal{M}_i$ is a matroid with rank function r_i, then $\mathcal{CG}(r) = \min_{i\in[k]} \mathcal{CG}(r_i)$.*

As a corollary, the correlation gap of a partition matroid is equal to the smallest correlation gap of its parts (uniform matroids).

In what follows we give an overview of a number of areas in submodular optimization and mechanism design where the correlation gap is a critical parameter. For example, Theorem 4.1 enables to focus only on matroid rank functions as worst-case instances; this finds a direct application e.g., in the context of sequential posted price mechanisms. Together with Proposition 4.4 below, Theorem 4.2 leads to improved bounds in submodular maximization, extending and strengthening previous results. In all these contexts, this leads to improvements on the general $(1 - 1/e)$ bound for particular cases. We expect that our work will lead to further investigations to derive improved correlation gap bounds for important classes of submodular functions.

4.1.1 Monotone Submodular Maximization

Let $f\colon 2^E \to \mathbb{R}_+$ be a monotone submodular function, and let $(E, \mathcal{J})$ be a matroid with independent sets $\mathcal{J}$. We consider the problem of maximizing f subject to a matroid constraint,

$$\max_{X\in\mathcal{J}} f(X) \ . \tag{4.4}$$

For uniform matroids (i.e., cardinality constraints), a classical result by Fisher, Nemhauser, and Wolsey [114] showed a $(1-1/e)$-approximation guarantee for the greedy algorithm. Moreover, this factor cannot be improved if we are only allowed polynomially many calls to the value oracle of f, see Nemhauser and Wolsey [113].

The factor $(1-1/e)$ being the natural target for (4.4), Calinescu, Chekuri, Pál, and Vondrák [26] obtained it for the special case when f is a sum of weighted matroid rank functions. The journal version of the same paper [25], see also [153], shows a $(1-1/e)$-approximation for arbitrary nondecreasing submodular functions, achieving the best possible general guarantee for (4.4).

These algorithms proceed in two steps. Let

$$\mathcal{P}(r) := \left\{x \in \mathbb{R}_+^E : x(S) \le r(S) \quad \forall S \subseteq E\right\} \tag{4.5}$$

denote the independent set polytope, where r is the rank function of the matroid $(E, \mathcal{J})$. When clear from the context, we use the shorthand $\mathcal{P}$.

In the first step, the goal is to find a $(1-1/e)$-approximation algorithm for the relaxation

$$\max_{x\in\mathcal{P}} F(x) \ . \tag{4.6}$$

Let x^* be the solution obtained in the first step. In the second step, they use *pipage rounding* to find an integer solution $X \in \mathcal{J}$ with $f(X) \ge F(x^*)$.

Thus, approximation loss only happens in the first step. To solve this non-concave maximization problem, [26] introduced another relaxation $\tilde{f}(x)$ such that $F(x) \le \tilde{f}(x) \le \hat{f}(x)$ for all $x \in [0,1]^E$, and showed that $\max_{x\in\mathcal{P}} \tilde{f}(x)$ can be formulated as an LP. The $(1-1/e)$-approximation to (4.6) is obtained by solving this LP optimally. Subsequently, Shioura [135] showed that when f is a sum of $M^\natural$-concave functions, the analogous convex program can also be solved optimally. $M^\natural$-concave functions form a special class of submodular functions, and are a central concept in discrete convex analysis, see Murota's monograph [108]. They are also known as *gross substitutes functions* and play an important role in mathematical economics [76, 95, 101, 116]. In particular, every weighted matroid rank function is $M^\natural$-concave.

The $(1-1/e)$-approximation for arbitrary monotone submodular f in [25, 153] uses a different approach: instead of using another relaxation, they perform a *continuous greedy algorithm* directly on $F(x)$. Improved approximations were subsequently given for submodular functions with bounded curvature; we discuss these results in Section 4.1.5.

Bounding via correlations gaps We focus on submodular functions that are given as a sum of $M^\natural$-concave functions, as in Shioura's work [135]. For technical reasons, we assume all of them are rational valued; the relevant complexity parameter $\mu(f)$ is defined in Section 4.2.

Proposition 4.4. *Let $f_1, f_2, \ldots, f_m \colon 2^E \to \mathbb{R}$ be monotone $M^\natural$-concave functions, and let $f = \sum_{j=1}^m f_j$. Then, a $\min_{j=1}^m \mathcal{CG}(f_j)$-approximation algorithm for (4.4) can be obtained in time polynomial in $|E|$, m and $\mu(f)$.*

The simple proof is given in Section 4.3.1. The algorithm is the same as the one by Shioura [135]; we observe that the general $(1 - 1/e)$-bound can be improved to the bound on the correlation gaps.

Combinatorial public projects

A direct application of the model in Proposition 4.4 arises in the *combinatorial public projects* problem, introduced by Papadimitriou, Schapira, and Singer [121], see also Dughmi, Roughgarden, Yan [52]. Let $f_1, f_2, \ldots, f_m \colon 2^E \to \mathbb{R}$ be monotone submodular functions and consider (4.4) for $f = \sum_{j=1}^m f_j$. The interpretation is that a public planner chooses an independent subset from a set of possible projects (e.g., at most k projects), or from a set of shared resources. There are m stakeholders, with valuations $f_1, f_2, \ldots, f_m$. The goal is to maximize the welfare the stakeholders derive from the chosen set of public projects/shared resources.

The problem admits a $(1 - 1/e)$-approximation algorithm for any m as the sum of submodular functions is submodular. This ratio is optimal even for $m = 1$ in the value oracle model. On the other hand, Proposition 4.4 yields stronger guarantees in terms of the correlation gap of these functions, assuming the functions f_j are $M^\natural$-concave.

Concave multicoverage problems

A fundamental special case of the above model is the coverage problem. For m given subsets $E_j \subseteq E$, the corresponding *coverage function* is defined as $f(X) = |\{j \in [m] : E_j \cap X \neq \emptyset\}|$. Note that this is a special case of maximizing the sum of matroid rank functions: $f(X) = \sum_{j=1}^m r_j(X)$, where $r_j(X)$ is the rank function of a rank-1 uniform matroid with support E_j. Even for maximizing under cardinality constraint, there is no better than $(1 - 1/e)$-approximation for this problem unless $P = NP$, see Feige [62].

Recently, tight approximations have been established for the special case when the function values $f_j(S)$ are determined by the size of the set S. Barman, Fawzi, and Fermé [8] studied concave multicoverage problems: for nondecreasing concave $\varphi \colon \mathbb{N} \to \mathbb{R}_+$ and weights $w \in \mathbb{R}_+^m$, the submodular function is defined as $f(X) = \sum_{j=1}^m w_j \varphi(|X \cap E_j|)$.[1] The usual coverage problem corresponds to $\varphi(x) = \min\{1, x\}$; on the other extreme, for $\varphi(x) = x$ we get the trivial problem $f(X) = \sum_{i \in X} |\{j \in [m] : i \in E_j\}|$. In [8], they present a tight approximation guarantee for maximizing such an objective subject to a matroid constraint, parametrized by the *Poisson curvature* of the function φ.

[1]We note that such functions are exactly the one dimensional $M^\natural$-concave functions $f_i \colon \mathbb{N} \to \mathbb{R}_+$.

This extends previous work by Barman, Fawzi, Ghoshal, and Gürpinar [9] which considered $\varphi(x) = \min\{\ell, x\}$ (for $\ell > 1$), motivated by the list decoding problem in coding theory. It also extends the work by Dudycz, Manurangsi, Marcinkowski, and Sornat [50] which considered geometrically dominated concave functions φ, motivated by approval voting rules such as Thiele rules, proportional approval voting, and p-geometric rules. In both cases, the obtained approximation guarantees improve over the $1 - 1/e$ factor.

In Section 4.3.2, we explain how the results in [8, 9, 50] can be derived from Proposition 4.4: the Poisson curvature bound turns out to be equal to the correlation gap of the functions $\varphi(|X \cap E_j|)$.

4.1.2 Sequential Posted Price Mechanisms

Following Yan [159], consider a seller with a set of identical services (or goods), and a set E of agents where each agent is only interested in one service (unit demand). Agent $i \in E$ has a private valuation v_i if they get a service and 0 otherwise, where each v_i is drawn independently from a known distribution F_i over $[0, L]$ for large $L \in \mathbb{R}_+$ with positive smooth density function. The seller can offer the service only to certain subsets of the agents simultaneously; this is captured by a matroid $(E, \mathcal{I})$ where the independent sets represent feasible allocations of the services to the agents.

A mechanism uses an allocation rule $x\colon \mathbb{R}_+^E \to \{0,1\}^E$ to choose the winning set of agents based on the reported valuations $v \in \mathbb{R}_+^E$ of the agents, and uses a payment rule $p\colon \mathbb{R}_+^E \to \mathbb{R}_+^E$ to charge the agents.

Myerson's mechanism [24, 111] guarantees the optimal revenue, but is highly intricate and there has been significant interest in simpler mechanisms such as sequential posted price mechanisms proposed by Chawla, Hartline, Malec, and Sivan [28].

For a given ordering of agents and a price p_i for each agent i, a *sequential posted-price mechanism (SPM)* initializes the allocated set A to be $\emptyset$, and for all agents i in the order, does the following: if serving i is feasible, i.e., $A \cup \{i\} \in \mathcal{I}$, then it offers to serve agent i at the pre-determined price p_i, and adds i to A if agent i accepts.

Thus, the seller makes take-it-or-leave-it price offers to agents one by one. This type of mechanism is easy to run for the sellers, limits agents' strategic behaviour, and keeps the information elicited from agents at a minimum level. Simplicity comes at a cost as it does not deliver optimal revenue, but as it turns out, this cost can be lower bounded by the correlation gap of the underlying matroid $(E, \mathcal{I})$.

Theorem 4.5 ([159, Theorem 3.1])**.** *If the correlation gap of the weighted rank function is at least β for no matter what nonnegative weights, then the expected revenue of greedy-SPM is a β-approximation to that of Myerson's optimal mechanism.*

Note that in Theorem 4.1, we show that the worst case correlation gap of a weighted matroid rank function is achieved with uniform weights. Hence, it suffices in this context to bound the correlation gap of the unweighted rank function.

Similarly, the same paper [159] shows that a greedy-SPM mechanism recovers a constant factor of the VCG mechanism [31, 75, 151] that maximizes the optimal welfare instead of revenue. The factor here is also the correlation gap of the (weighted) rank function of the underlying matroid. The analysis of greedy-SPM in both revenue and welfare maximization settings is tight. For details we refer to [159].

4.1.3 Contention Resolution Schemes

Chekuri, Vondrák, and Zenklusen [30] introduced contention resolution (CR) schemes as a tool for maximizing a general submodular function f (not necessarily monotone) under various types of constraints. For simplicity, let us illustrate it for a single matroid constraint, i.e. (4.4) without the monotonicity assumption on f. It consists of first approximately solving the continuous problem (4.6). After obtaining an approximately optimal solution $x \in \mathcal{P}$ to (4.6), it is rounded to an integral and feasible solution — i.e. an independent set in $\mathcal{J}$ — without losing too much in the objective value. At a high level, given a fractional point $x \in \mathcal{P}$, a CR scheme first generates a random set $R(x)$ by independently including each element i with probability x_i. Then, it removes some elements of $R(x)$ to obtain a feasible solution. We say that a CR scheme is *c-balanced* if, conditioned on $i \in R(x)$, the element i is contained in the final independent set with probability at least c; see [30] for a formal definition. A c-balanced scheme delivers an integer solution with expected cost at least $cF(x)$. Thus, the goal is to design c-balanced CR schemes with the highest possible value of c.

There is a tight relationship between CR schemes and the correlation gap. Namely, the correlation gap of the weighted rank function of $(E, \mathcal{J})$ is equal to the maximum c such that it admits c-balanced CR scheme [30, Theorem 4.6]. We would like to point out that the correlation gap here concerns the matroid in the constraint, unlike in Proposition 4.4 where the correlation gap concerns the objective function. Note that again by Theorem 4.1, it suffices to bound the correlation gap of the (unweighted) rank function of $(E, \mathcal{J})$.

The benefit of CR schemes is that we can obtain good guarantees for submodular function maximization under an intersection of different (downward-closed) constraints, including multiple matroid constrains, knapsack constraints, matching etc. Moreover, in this case the CR scheme can be simply obtained by combining the CR schemes for the individual constraints.[2]

[2]We note however that CR schemes are not optimal for rounding (4.6): for this particular case, pipage or swap rounding finds a feasible integer solution of value $F(x)$, without any loss.

4.1.4 Our Techniques

We now give a high-level overview of the proofs of Theorem 4.1 and Theorem 4.2, and motivate the usage of the rank and girth parameters.

Weighted rank functions The starting point of the proofs of both Theorem 4.1 and Theorem 4.2 is to deduce structural properties of correlation gap minimizer solutions. In Theorem 4.21, we show that there is such a minimizer x in the independent set polytope $\mathcal{P}$. This implies $\hat{r}_w(x) = w^\top x$ for any weighting $w \in \mathbb{R}^E_+$; moreover, we deduce that $x(E)$ must be integral.

To prove Theorem 4.1 (in the restated form of Theorem 4.24), we fix a matroid $\mathcal{M}$ and derive a contradiction for a non-uniform weighting. More precisely, we consider a weighting w and a fractional solution x^* that give a smaller ratio $R_w(x^*)/\hat{r}_w(x^*)$ of the weighted multilinear and weighted concave extensions than achievable by uniform weighting. By the above, we can use the simpler form $R_w(x^*)/\hat{r}_w(x^*) = R_w(x^*)/w^\top x$. We assume that the weight function w has the smallest number of different values. If the number of distinct w_i values is at least 2, we derive a contradiction by showing that a better solution can be obtained by increasing the weights in a carefully chosen value class until they become equal to the next smallest value. The greedy maximization property of matroids is essential for this argument.

Girth and rank To get some understanding of the correlation gap of an (unweighted) matroid rank function, let us first focus on uniform matroids. Let $n = |E|$ and let $\rho = r(E)$ denote the rank of the matroid $\mathcal{M} = (E, \mathcal{I})$. If $\rho = 1$, then it is easy to verify that the solution $x_i = 1/n$ for all $i \in E$ has correlation gap $1 - (1 - 1/n)^n$. This point is in the independent set polytope $\mathcal{P}$ and hence $\hat{r}(x) = x(E) = 1$. If one samples each i with probability $1/n$, the probability of selecting at least one element is $1 - (1 - 1/n)^n$. This yields the multilinear extension at that point, and $1 - (1 - 1/n)^n$ goes to $1 - 1/e$ as $n \to \infty$.

If $\rho > 1$, one can similarly argue that the symmetric solution $x_i = \rho/n$ gives the worst case, and that $\mathcal{CG}(\mathcal{M})$ increases as ρ grows. Clearly, for $\rho = n$ we get $\mathcal{CG}(\mathcal{M}) = 1$. In Section 4.3.2, we show that the bounds in [9] can be interpreted as tight correlation gap bounds for uniform matroids. In Proposition 4.20, we show that these coincide with the bounds in Theorem 4.2 for the case of uniform matroids, i.e., $\gamma = \rho + 1$.

Recall that the girth γ is the smallest size of a dependent set. For the well-studied class of *paving matroids*, we have $\gamma \in \{\rho, \rho + 1\}$. Intuitively, the correlation gap of a paving matroid should be very close to that of a uniform matroid with the same rank, since the rank of any set differs by at most 1 in the two matroids. Decreasing γ leads to more restrictive matroid structures, and hence worse correlation gaps.

It turns out that the correlation gap heavily depends on the relative values of ρ and γ. In Section 4.5, we give simple upper bounds on the correlation gap as parameterized by these parameters. The first construction (Claim 4.25) shows that for a fixed girth γ, increasing the rank ρ of the matroid does not improve the correlation gap. In other words, it is always upper bounded by the correlation gap of a rank-$(\gamma - 1)$ uniform matroid. The second construction (Proposition 4.26) gives a bound which exhibits a similar behaviour to the lower bound in Theorem 4.2: it is a decreasing function of ρ when γ is fixed, and an increasing function of γ when ρ is fixed. We remark however, that these bounds are not tight.

Our results give the first nontrivial bound in terms of these two parameters. We hope that it will motivate further studies into more refined correlation gap bounds, exploring the dependence on further matroid parameters, as well as obtaining tight bounds for special matroid classes.

A continuous time Markov chain analysis To obtain the $(1 - 1/e)$ lower bound on the correlation gap of a monotone submodular function, Calinescu et al. [26] introduced an elegant probabilistic analysis. Instead of sampling all $i \in E$ with probability x_i, we consider n independent *Poisson clocks* of rate x_i that are active during the time interval $[0, 1]$. They may send at most one signal from a Possion process. Let $Q(t)$ be the set of elements where the signal was sent between time 0 and t; the output is $Q(1)$. This process can also be viewed as a continuous time Markov chain. It is easy to see that $\mathbb{E}[f(Q(1))] \le F(x)$.

In [26], they show that the derivative of $\mathbb{E}[F(Q(t))]$ can be lower bounded as $f^*(x) - \mathbb{E}[f(Q(t))]$ for every $t \in [0, 1]$, where

$$f^*(x) := \min_{S \subseteq E} \left(f(S) + \sum_{i \in E} f_S(i) x_i \right) \tag{4.7}$$

is an extension of f such that $f^* \ge \hat{f}$. The bound $\mathbb{E}[f(Q(1))] \ge (1 - 1/e) f^*(x)$ is obtained by solving a differential inequality. Thus, $F(x) \ge \mathbb{E}[f(Q(1))] \ge (1 - 1/e) f^*(x) \ge (1 - 1/e)\hat{f}(x)$ follows.

A two stage approach If f is a matroid rank function, then we have $f^* = \hat{f}$ (see Theorem 4.7). Still, the factor $(1 - 1/e)$ in the analysis cannot be improved. In fact, already for an integer $x \in \mathcal{P}$, we lose a factor $(1 - 1/e)$ due to $\mathbb{E}[f(Q(1))] = (1 - 1/e)F(x)$, even though the extensions coincide: $F(x) = \hat{f}(x)$.

Our analysis in Section 4.6 proceeds in two stages. Given a matroid $\mathcal{M} = (E, \mathcal{I})$ with rank ρ and girth γ, we decompose the rank function as $r = g + h$, where $g(X) = \min\{|X|, \ell\}$ is the rank function of a uniform matroid of rank $\ell = \gamma - 1$. Note that $h := f - g$ may not be submodular, as $h(S) = 0$ for all $|S| \le \ell$. We lower bound the multilinear extensions $G(x)$

and $H(x)$ separately. As above, $G(x)$ can be lower bounded by showing that the worst case is taken at a symmetric solution, i.e., where $x_i = x(E)/n$ for all $i \in E$.

Bounding $H(x)$ is based on a Poisson clock analysis as in [26], but is significantly more involved. Due to the monotonicity of h, directly applying the result in [26] still yields $\mathbb{E}[h(Q(1)] \geq (1 - 1/e)h^*(x)$. However, $h^*(x) = 0$ whenever $\mathcal{M}$ is loopless ($\ell \geq 1$). Indeed, $h(\emptyset) = 0$ and $h_\emptyset(i) = 0$ for all $i \in E$. So, the previous inequality becomes $\mathbb{E}[h(Q(1))] \geq 0$, which is trivial and too weak for our purpose. Nevertheless, one can still show that, conditioned on the event $|Q(t)| \geq \ell$, the derivative of $\mathbb{E}[H(Q(t))]$ is at least $r^*(x) - \ell - \mathbb{E}[H(Q(t))]$. Let $T \geq 0$ be the earliest time such that $|Q(T)| \geq \ell$, which we call the *activation time* of Q. Then, solving a similar differential inequality produces $\mathbb{E}[h(Q(1))|T = t] \geq (1 - e^{-(1-t)})(r^*(x) - \ell)$ for all $t \leq 1$.

To lower bound $\mathbb{E}[h(Q(1))]$, it is left to take the expectation over all possible activation times $T \in [0, 1]$. Let $\bar{h}(x) = (r^*(x) - \ell) \int_0^1 \Pr[T = t](1 - e^{-(1-t)})dt$ be the resulting expression. We prove that $\bar{h}(x)$ is concave in each direction $e_i - e_j$ for $i, j \in E$. This allows us to round x to an integer $x' \in [0, 1]^E$ such that $x'(E) = x(E)$ and $\bar{h}(x') \leq \bar{h}(x)$; recall that $x(E) \in \mathbb{Z}$ by Theorem 4.21. After substantial simplification of $\bar{h}(x')$, we arrive at the formula in Theorem 4.2, except that ρ is replaced by $x(E)$. So, the rounding procedure effectively shifts the dependency of the lower bound from the value of x to the value of $x(E)$. Since $x(E) \leq \rho$ by Theorem 4.21, the final step is to prove that the formula in Theorem 4.2 is monotone decreasing in ρ. This is shown in Lemma 4.38 using the relationship between the Poisson distribution and the incomplete gamma function. Additionally, in Lemma 4.37 we show that the obtained lower bound is always strictly greater than $1 - 1/e$ when $\ell > 1$.

4.1.5 Further Related Work

In the context of submodular maximization (4.4), Proposition 4.4 allows for improved approximation bounds if $f = \sum_{i=1}^m f_i$, where the f_i's are $M^\natural$-concave functions.

A different approach to give fine-grained approximation guarantees for (4.4) is via curvature notions; this is applicable to any submodular function and does not require the form $f = \sum_{i=1}^m f_i$. A well-studied measure is the *total curvature* of the submodular function, namely, $c(f) = 1 - \min_{i \in E}(f(E) - f(E \setminus \{i\}))/f(\{i\})$. Monotonicity and submodularity guarantee $c(f) \in [0, 1]$; the best case $c(f) = 0$ corresponds to additive (modular) functions. For cardinality constraints, such a bound was given by Conforti and Cornuéjols [33], strengthened and extended to matroid constraints by Sviridenko, Vondrák, and Ward [141].

However, there are important cases of submodular functions where the total curvature bound is not tight. For a nondecreasing concave univariate function $\varphi : \mathbb{N} \to \mathbb{R}_+$ with $\varphi(0) = 0$, $f(X) = \varphi(|X|)$ is a submodular function. Exact maximization over matroid constraints is straightforward for such a function, yet the total curvature can be 1. This is a simple example of an $M^\natural$-concave function; submodular function maximization can be done in polynomial time for all such functions (see Proposition 4.10).

Motivated by this, Soma and Yoshida [138] proposed the following generalization of total curvature: assume our monotone submodular function can be decomposed as $f = g + h$, where g is monotone submodular and h is $M^\natural$-concave. They define the h-curvature as $\gamma_h(f) = 1 - \min_{X \subseteq E} h(X)/f(X)$, and provide approximation guarantees in terms of $\gamma_h(f)$. If this is close to 0, then the function can be well-approximated by an $M^\natural$-concave functions. The usual notion of total curvature arises by restricting h to additive (modular) functions.

A common thread in [138] and our approach is to exploit special properties of $M^\natural$-concave functions for submodular maximization. However, there does not appear to be any direct implication between them.

Chapter organization In Section 4.2, we recall the definitions of matroids, $M^\natural$-concave functions, submodular functions, related paremeters, and some classical results that we will use in our proofs. In Section 4.3.1, we recall Shioura's algorithm for maximizing a sum of $M^\natural$-concave functions under matroid constraints and observe that the performance of this algorithm is bounded by the correlation gap of the input functions. Then, in Section 4.3.2, we explain how the results on concave multicoverage problems [8, 9, 50] can be derived using the aforementioned algorithm and that the Poisson curvature bounds are equal to correlation gaps of the same functions.

The rest of the chapter is devoted to showing our two main results. In Section 4.4, we prove Theorem 4.1 and show that the minimizer of the correlation gap can always be found in the independent set polytope of the matroid. Before we prove Theorem 4.2 in Section 4.6; we give upper-bounds on the correlation gap of a matroid with rank ρ and girth γ in Section 4.5.

4.2 Preliminaries

We let $\mathbb{Z}_+$ and $\mathbb{R}_+$ denote the set of nonnegative integers and nonnegative reals, respectively. For $n, k \in \mathbb{Z}_+$, we define $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ if $n \geq k$ and as 0 otherwise.

For a set X and $i \in X$, $j \notin X$, we will use the shorthands $X - i = X \setminus \{i\}$, $X + j = X \cup \{j\}$, $X - i + j = (X \setminus \{i\}) \cup \{j\}$. For $x \in \mathbb{R}^E$ and $S \subseteq E$, we use $x(S) = \sum_{i \in S} x_i$.

All set functions in the chapter will be given by value oracles; our running time bounds will be polynomial in the number of oracle calls and arithmetic operations. We further assume that all set functions are rational valued, and for $f : 2^E \to \mathbb{Q}$, we let $\mu(f)$ denote an upper bound on the encoding length of any value $f(S)$. That is, for any $S \subseteq V$, the oracle returns $f(S) = p/q$ represented by $p, q \in \mathbb{Z}$ such that $\lceil \log_2 p \rceil + \lceil \log_2 q \rceil \leq \mu(f)$.

Matroids For a detailed introduction to matroids, we refer the reader to Oxley's book [120] or Schrijver's book [133]. A matroid $\mathcal{M} = (E, \mathcal{I})$ is given by a set of *independent sets*

$\mathcal{I} \subseteq 2^E$ of a ground set E. We require that $\mathcal{I} \neq \emptyset$, and the following axiom:

$$\forall X, Y \in \mathcal{I} \text{ with } |X| < |Y| : \exists j \in Y \setminus X : X + j \in \mathcal{I} \,. \tag{4.8}$$

A *basis* is an inclusion-wise maximal independent set. Let $\mathcal{B} \subseteq \mathcal{I}$ be the set of bases. The above axiom implies that all bases are of the same size, called the *rank* of $\mathcal{M}$.

The rank function $r : 2^E \to \mathbb{N}$ is defined as $r(S) = \max\{|Z| : Z \subseteq S, Z \in \mathcal{I}\}$. This is a monotone submodular function. A *circuit* is an inclusion-wise minimal dependent set. The size of a smallest circuit is called the *girth* of $\mathcal{M}$.

For a set $T \subseteq E$, $\mathcal{M}|_T$ is the *restriction* of $\mathcal{M}$ to T, that is a matroid on ground set T with independent sets $\mathcal{I}|_T := \{I \in \mathcal{I} : I \subseteq T\}$. For a fixed set $T \subseteq E$, $\mathcal{M}/T$ is the *contraction* of $\mathcal{M}$ by T, that is a matroid on ground set $E \setminus T$ with independent sets $\mathcal{I}/T := \{S \subseteq E \setminus T : S \cup B \in \mathcal{I}$ for some basis B of $\mathcal{M}|_T\}$.

The rank function of $\mathcal{M}/T$ is given by $r_{\mathcal{M}/T}(S) = r(S \cup T) - r(S)$. For a set $S \subseteq E$ and an element $i \in E$, let $r_S(i)$ denote the marginal gain of adding i to S, i.e., $r_S(i) := r(S+i) - r(S)$.

The *closure* of S is defined as $\mathrm{cl}(S) := \{i \in E : r_S(i) = 0\}$. A set $S \subseteq E$ is *spanning*, if $\mathrm{cl}(S) = E$; equivalently, if S contains a basis. A *flat* of $\mathcal{M}$ is a set $S \subseteq E$ where $\mathrm{cl}(S) = S$.

Recall the independent set polytope defined in (4.5).

Theorem 4.6 (Edmonds, [133, Theorem 40.2]). *For a matroid $\mathcal{M} = (E, \mathcal{I})$ with rank function r, $\mathcal{P}(r)$ defined in (4.5) is the convex hull of the incidence vectors of the independent sets in $\mathcal{I}$.*

We also recall another classical result by Edmonds on intersecting the independent set polytope by a box.

Theorem 4.7 (Edmonds, [133, Theorem 40.3]). *Let $r : 2^E \to \mathbb{R}$ be a matroid rank function and $x \in \mathbb{R}_+$. Then,*

$$\max\{y(E) : y \in \mathcal{P}(r),\, y \leq x\} = \min\{r(T) + x(E \setminus T) : T \subseteq E\} \,.$$

$M^\natural$-concave functions A set function $g : 2^V \to \mathbb{R} \cup \{-\infty\}$ is $M^\natural$-concave if

$$\begin{aligned} &\forall X, Y \subseteq \text{ with } |X| < |Y| : \\ &f(X) + f(Y) \leq \max_{j \in Y \setminus X} \{f(X+j) + f(Y-j)\} \end{aligned} \tag{4.9a}$$

$$\begin{aligned} &\forall X, Y \subseteq E \text{ with } |X| = |Y| \text{ and } \forall i \in X \setminus Y : \\ &f(X) + f(Y) \leq \max_{j \in Y \setminus X} \{f(X-i+j) + f(Y+i-j)\}. \end{aligned} \tag{4.9b}$$

We refer the reader to Murota's monography [108] for a comprehensive treatment of $M^\natural$-concave functions. These functions can be defined more generally on the integer lattice $\mathbb{Z}^n$.

In this chapter, we restrict our attention to $M^\natural$-concave set functions, also known as *valuated generalized matroids*. These are closely related to valuated matroids introduced by Dress and Wenzel [49]. The definitions above are from [69, 110] and are equivalent to the standard definition in [108].

The definition can be seen as a generalization of the matroid independence axiom (4.8). Given a matroid $\mathcal{M} = (E, \mathcal{I})$, the indicator function f defined as $f(S) = 0$ if $S \in \mathcal{I}$ and $f(S) = -\infty$ is $M^\natural$-concave. More generally, given also a weight function $w \in \mathbb{R}^E$, the *weighted matroid rank function*

$$r_w(S) := \max\{w(Z) : Z \subseteq S \,, Z \in \mathcal{I}\} \tag{4.10}$$

is $M^\natural$-concave. These functions form a nontrivial subclass of submodular functions [76, 101].

Proposition 4.8 ([108, Theorem 6.19])**.** *Every $M^\natural$-concave function is submodular.*

We recall that submodular functions can be minimized in polynomial time, but submodular maximization is NP-complete. However, it is polynomial time solvable for $M^\natural$-concave functions; in fact, they can be maximized using the greedy algorithm.

Proposition 4.9 ([49])**.** *If $g : 2^E \to \mathbb{R} \cup \{-\infty\}$ is $M^\natural$-concave function and $z \in \mathbb{R}^E$, then $\max_{T \subseteq E} g(T) - z(T)$ can be computed in polynomial time.*

Recall the concave extension $\hat{g}(y)$ defined in (4.2). It is NP-complete to evaluate this function for general submodular functions. However, for $M^\natural$-concave functions, it can be efficiently computed. To see this, we formulate the dual LP, and notice that separation corresponds to maximizing $g(T) - z(T)$.

$$\begin{aligned} \min \quad & z^\top y + \alpha \\ \text{s.t.} \quad & z(T) + \alpha \geq g(T) \qquad \forall T \subseteq U \,. \end{aligned} \tag{4.11}$$

Proposition 4.10. *If $g : 2^E \to \mathbb{R} \cup \{-\infty\}$ is $M^\natural$-concave function and $y \in [0,1]^E$, then $\hat{g}(y)$ can be computed in polynomial time.*

We note that the existence of a concave extension satisfying desirable combinatorial properties is equivalent to the function being $M^\natural$-concave, see [108, Theorem 6.43].

Probability distributions Let $\text{Bin}(n, p)$ denote the binomial distribution with parameters n and p, and $\text{Poi}(\lambda)$ the Poisson distribution with parameter λ. Recall that $P(\text{Poi}(\lambda) = k) = e^{-\lambda}\lambda^k/k!$ for any $k \in \mathbb{Z}_+$.

Definition 4.11. Given random variables X and Y, we say that X *is at least Y in the concave order* if for every concave function $\varphi : \mathbb{R} \to \mathbb{R}$, we have $\mathbb{E}[\varphi(X)] \geq \mathbb{E}[\varphi(Y)]$ whenever the expectations exist. It is denoted as $X \geq_{\text{cv}} Y$.

In particular, we will use the following relation between the binomial and Poission distributions:

Lemma 4.12 ([9, Lemma 2.1])**.** *For any $n \in \mathbb{N}$ and $p \in [0,1]$, we have* $\mathrm{Bin}(n,p) \geq_{cv} \mathrm{Poi}(np)$*.*

4.2.1 Properties of Multilinear Extension

Let $f : 2^E \to \mathbb{R}_+$ be an arbitrary set function, and $F : [0,1]^E \to \mathbb{R}_+$ be its multilinear extension. We will use the following well known properties, see e.g. [25].

Proposition 4.13. *For any $x \in [0,1]^E$ and $i \in E$, the function $\phi(t) := F(x + te_i)$ is linear.*

Proposition 4.14. *If f is monotone, then $F(x) \geq F(y)$ for all $x \geq y$.*

Proposition 4.15. *If f is submodular, then for any $x \in [0,1]^E$ and $i, j \in E$, the function $\phi(t) := F(x + t(e_i - e_j))$ is convex.*

4.3 Correlation Gap Bounds for Submodular Maximization

4.3.1 Maximizing Sum of $M^\natural$-Concave Functions

In this section, we prove Proposition 4.4. Throughout, let $f_1, f_2, \dots, f_m : 2^E \to \mathbb{R}$ be monotone $M^\natural$-concave functions, and let $f = \sum_{j=1}^m f_j$. Let us define $\tilde{f} : [0,1]^E \to \mathbb{R}_+$ as the sum of the concave extensions.

$$\tilde{f}(x) := \sum_{j=1}^{m} \hat{f}_j(x) \tag{4.12}$$

Note that $\tilde{f}(x) \leq \hat{f}(x)$, however, this equality may be strict.

Shioura [135] gave an $(1 - 1/e)$-approximation for (4.4) for a function f in this form. His algorithm starts by solving

$$\max_{x \in \mathcal{P}} \tilde{f}(x) \tag{4.13}$$

This is a convex optimization problem, and is also a relaxation of (4.4), noting that for any $S \subseteq E$, $\tilde{f}(\chi_S) = f(S)$.

The number of constraints in $\mathcal{P}$ is exponential, but can be efficiently separated over. The objective function $\tilde{f}(x)$ can be evaluated by solving m exponential-size linear programs. Shioura showed that (4.13) can be solved using the ellipsoid method by implementing a subgradient oracle. The algorithm returns an exact solution in time polynomial in n, m, and the complexity parameter $\mu(f)$, assuming the functions are rational valued.

Given an optimal solution x^* to (4.13), the pipage rounding technique first introduced by Ageev and Sviridenko [1] can be used to obtain a set $S \in \mathcal{I}$ with $f(S) \geq F(x^*)$. Hence, we obtain an α-approximation for (4.4) as long as we can show $F(x^*) \geq \alpha \tilde{f}(x^*)$. The proof of Proposition 4.4 is complete by the following lemma.

Lemma 4.16. *Let $\alpha := \min_{j=1}^m \mathcal{CG}(f_j)$. Then, for every $x \in [0,1]^E$, $F(x) \geq \alpha \tilde{f}(x)$.*

Proof. Let F_j be the multilinear extension of f_j. Note that $F(x) = \sum_{j=1}^m F_j(x)$. By the definition of the correlation gap,

$$F(x) = \sum_{j=1}^m F_j(x) \geq \alpha \sum_{j=1}^m \hat{f}_j(x) = \alpha \tilde{f}(x)\,. \qquad \square$$

4.3.2 Concave Multicoverage Problems

We now discuss the concave multicoverage model in Barman et al. [8], and show that the Poisson curvature studied in this paper can be interpreted as a correlation gap bound. Further, in Proposition 4.20, we show that the tight bounds in [9] for the maximum multicoverage problems coincide with the correlation gap bound in Theorem 4.2 for uniform matroids, i.e., $\gamma = \rho + 1$.

Let $\mathcal{M} = (E, \mathcal{J})$ be a matroid, and let $\varphi : \mathbb{Z}_+ \to \mathbb{R}_+$ be a normalized nondecreasing concave function, i.e., $\varphi(0) = 0$, $\varphi(1) = 1$, $\varphi(i+1) \geq \varphi(i)$ and $\varphi(i+1) - \varphi(i) \geq \varphi(i+2) - \varphi(i+1)$ for all $i \in \mathbb{Z}_+$. For every $j \in [m]$, we are given a subset $E_j \subseteq E$, a weight $w_j \in \mathbb{R}_+$, and a function $f_j : 2^E \to \mathbb{R}_+$ defined by $f_j(X) := \varphi(|X \cap E_j|)$. In the φ-*MaxCoverage* problem, the goal is to maximize $f(X) := \sum_{j=1}^m w_j f_j(X)$ subject to $X \in \mathcal{J}$. Barman et al. [8] gave an approximation algorithm for this problem, whose approximation factor is the so-called *Poisson concavity ratio* of φ, defined as

$$\alpha_\varphi := \inf_{\lambda \in \mathbb{R}_+} \frac{\mathbb{E}[\varphi(\text{Poi}(\lambda))]}{\hat{\varphi}(\mathbb{E}[\text{Poi}(\lambda)])} = \inf_{\lambda \in \mathbb{R}_+} \frac{\mathbb{E}[\varphi(\text{Poi}(\lambda))]}{\hat{\varphi}(\lambda)}.$$

Here, $\hat{\varphi} : \mathbb{R}_+ \to \mathbb{R}_+$ is the concave extension of φ, i.e. $\hat{\varphi}(\lambda) = \varphi(\lfloor \lambda \rfloor) + (\varphi(\lfloor \lambda \rfloor + 1) - \varphi(\lfloor \lambda \rfloor))(\lambda - \lfloor \lambda \rfloor)$.

In this subsection, we show that the correlation gap of each f_j is at least the Poisson concavity ratio of φ. To this end, fix a $j \in [m]$. The following lemma relates the concave extensions of f_j and φ.

Lemma 4.17. *For any $x \in [0,1]^E$, we have $\hat{f}_j(x) = \hat{\varphi}(x(E_j))$*

Proof. Fix an $x \in [0,1]^E$. Let $\lambda = x(E_j)$ and $\beta = \varphi(\lfloor \lambda \rfloor + 1) - \varphi(\lfloor \lambda \rfloor)$. Based on the LP formulation of $\hat{f}_j$, it suffices to show that $(z, \alpha) := (\beta \chi_{E_j}, \varphi(\lfloor \lambda \rfloor) - \beta \lfloor \lambda \rfloor)$ is an optimal solution to (4.11). Note that $z^\top x + \alpha = \varphi(\lfloor \lambda \rfloor) + \beta(\lambda - \lfloor \lambda \rfloor) = \hat{\varphi}(\lambda)$. Feasibility is straightforward because for any $T \subseteq E$, we have

$$z(T) + \alpha = \beta \chi_{E_j}^\top \chi_T + \varphi(\lfloor \lambda \rfloor) - \beta \lfloor \lambda \rfloor = \varphi(\lfloor \lambda \rfloor) + \beta(|T \cap E_j| - \lfloor \lambda \rfloor) \geq \varphi(|T \cap E_j|) = f_j(T),$$

where the inequality follows from the concavity of $\hat{\varphi}$, and the fact that β is a supergradient of $\hat{\varphi}$ at $\lfloor \lambda \rfloor$. Observe that the inequality is tight if $|T \cap E_j| \in \{\lfloor \lambda \rfloor, \lfloor \lambda + 1 \rfloor\}$. To show optimality, we consider the dual LP (4.2). By complementary slackness, it is left to prove that x be can be written as a convex combination of the indicator vectors of these sets. Define the polytope

$$P := \{y \in [0,1]^E : \lfloor\lambda\rfloor \leq y(E_j) \leq \lfloor\lambda\rfloor + 1\}.$$

It is easy to see that the vertices of P are precisely the aforementioned indicator vectors. Since $x \in P$, it lies in their convex hull. □

The next lemma shows that the multilinear extension F_j is minimized at 'symmetric' points.

Lemma 4.18. *symmetric For any $x \in [0,1]^E$, let $\bar{x} \in [0,1]^E$ be the vector given by*

$$\bar{x}_i := \begin{cases} \frac{x(E_j)}{|E_j|}, & \text{if } i \in E_j \\ x_i, & \text{otherwise.} \end{cases}.$$

Then, $F_j(x) \geq F_j(\bar{x})$.

Proof. Fix an $x \in [0,1]^E$, and let $\bar{x} \in [0,1]^E$ be the vector as defined above. Let $y^* \in \arg\min_{y \in [0,1]^E}\{F_j(y) : y(E_j) = x(E_j)\}$ such that $\|y^* - \bar{x}\|_1$ is minimized. It suffices to prove that $y^* = \bar{x}$. Note that $y^*_i = \bar{x}_i$ for all $i \notin E_j$ because these coordinates do not affect the value of F_j.

For the purpose of contradiction, suppose that there exist $a, b \in E_j$ such that $y^*_a < \bar{x}_a$ and $y^*_b > \bar{x}_b$. Let $\phi(t) := F_j(y^* + t(e_a - e_b))$ be the function obtained by restricting F_j along the direction $e_a - e_b$ at y^*. Since f_j is submodular, ϕ is convex by Proposition 4.15. Moreover, $\phi(0) = \phi(y^*_b - y^*_a)$ because F_j becomes a symmetric polynomial after fixing the coordinates in $E \setminus E_j$. It follows that $\phi(t) \leq \phi(0) = F_j(y^*)$ for all $0 \leq t \leq y^*_b - y^*_a$. Thus, if we pick $t = \min\{\bar{x}_a - y^*_a, y^*_b - \bar{x}_b\}$, then $\|y^* + t(e_a - e_b) - \bar{x}\|_1 < \|y^* - \bar{x}\|_1$, which is a contradiction. □

We show that the Poisson concavity ratio is a lower bound on the correlation gap:

Proposition 4.19. *We have $\mathcal{CG}(f_j) \geq \alpha_\varphi$. If $\varphi(x) = o(x)$, then $\mathcal{CG}(f_j) = \alpha_\varphi$ unless $P = NP$.*

Proof. To show the inequality, let $x \in [0,1]^E$ such that $\mathcal{CG}(f_j) = F_j(x)/\hat{f}_j(x)$. Let $\lambda = x(E_j)$ and $n_j = |E_j|$. Define the vector $\bar{x} \in [0,1]^E$ as $\bar{x}_i := \lambda/n_j$ if $i \in E_j$, and $\bar{x}_i := x_i$ otherwise. According to Lemmas 4.12 and 4.18,

$$F_j(x) \geq F_j(\bar{x}) = \sum_{k=0}^{n_j} \varphi(k)\binom{n_j}{k}\left(\frac{\lambda}{n_j}\right)^k\left(1 - \frac{\lambda}{n_j}\right)^{n_j - k} = \mathbb{E}\left[\varphi\left(\text{Bin}\left(n_j, \frac{\lambda}{n_j}\right)\right)\right] \geq \mathbb{E}\left[\varphi(\text{Poi}(\lambda))\right],$$

Moreover, we have $\hat{f}_j(x) = \hat{\varphi}(\lambda)$ by Lemma 4.17. Hence, $\mathcal{CG}(f_j) \geq \mathbb{E}\left[\varphi(\text{Poi}(\lambda))\right]/\hat{\varphi}(\lambda) \geq \alpha_\varphi$.

For the case $\varphi(x) = o(x)$, [8, Theorem 4] shows that it is NP-hard to approximate φ-MaxCoverage by a ratio $\alpha_\varphi + \varepsilon$ for any $\varepsilon > 0$. By Proposition 4.4, we have P = NP in case $\mathcal{CG}(f_j) > \alpha_\varphi$. □

When f_j is the rank function of a rank-ℓ uniform matroid, [9] gave a tight approximation ratio $1 - \frac{e^{-\ell}\ell^\ell}{\ell!}$. We show that this coincides with the lower bound in Theorem 4.2.

Proposition 4.20. *For every $\ell \in \mathbb{N}$, we have*

$$1 - \frac{1}{e} + \frac{e^{-\ell}}{\ell}\left(\sum_{i=0}^{\ell-1}(\ell - i)\left[\binom{\ell}{i}(e-1)^i - \frac{\ell^i}{i!}\right]\right) = 1 - \frac{e^{-\ell}\ell^\ell}{\ell!}.$$

Proof. First, observe that

$$\begin{aligned}\sum_{i=0}^{\ell-1}(\ell-i)\binom{\ell}{i}(e-1)^i &= \ell\sum_{i=0}^{\ell-1}\binom{\ell}{i}(e-1)^i - \sum_{i=1}^{\ell-1}\frac{\ell!}{(i-1)!(\ell-i)!}(e-1)^i \\ &= \ell\left(\sum_{i=0}^{\ell-1}\binom{\ell-1}{i}(e-1)^i + \sum_{i=1}^{\ell-1}\binom{\ell-1}{i-1}(e-1)^i - \sum_{i=1}^{\ell-1}\binom{\ell-1}{i-1}(e-1)^i\right) \\ &= \ell e^{\ell-1}.\end{aligned}$$

Similarly, we have

$$\sum_{i=0}^{\ell-1}(\ell-i)\frac{\ell^i}{i!} = \ell\sum_{i=0}^{\ell-1}\frac{\ell^i}{i!} - \sum_{i=1}^{\ell-1}\frac{\ell^i}{(i-1)!} = \ell\left(\sum_{i=0}^{\ell-1}\frac{\ell^i}{i!} - \sum_{i=0}^{\ell-2}\frac{\ell^i}{i!}\right) = \frac{\ell^\ell}{(\ell-1)!}.$$

Putting them together yields

$$1-e^{-1}+\frac{e^{-\ell}}{\ell}\left(\sum_{i=0}^{\ell-1}(\ell-i)\left[\binom{\ell}{i}(e-1)^i - \frac{\ell^i}{i!}\right]\right) = 1-e^{-1}+\frac{e^{-\ell}}{\ell}\left(\ell e^{\ell-1} - \frac{\ell^\ell}{(\ell-1)!}\right) = 1-\frac{e^{-\ell}\ell^\ell}{\ell!}$$

as desired. □

4.4 Locating the Correlation Gap

In this section, we prove some structural results to locate the minimizer x of the correlation gap $\mathcal{CG}(r_w)$ of a weighted matroid rank function, and prove Theorem 4.1, i.e., the worst case correlation gap over all possible weightings is attained by the uniform weights. Our first goal is to show the following.

Theorem 4.21. *Let $\mathcal{M} = (E, \mathcal{I})$ be a matroid, $w \in \mathbb{R}_+^E$ a weight vector. Let r_w denote the weighted matroid rank function with multilinear extension R_w. Then, there exists an*

$x^* \in \mathcal{P}(r)$ and a set $S^* \subseteq E$ such that $x^*(S^*) = r(S^*)$, $x^*_{E\setminus S} = 0$, and

$$\mathcal{CG}(r) = \frac{R_w(x^*)}{\hat{r}_w(x^*)} = \frac{R_w(x^*)}{w^\top x^*}\,.$$

We start by giving two alternative descriptions of $\hat{r}_w$. Recall the definition of $\hat{r}_w$ in (4.2) and the dual form (4.11). We first show that the equalities in (4.2) can be relaxed to inequalities for any monotone submodular function:

Lemma 4.22. *For any monotone submodular function $f : 2^E \to \mathbb{R}$ and $x \in [0,1]^E$, the concave extension $\hat{f}(x)$ can be equivalently written as*

$$\max\left\{\sum_{S\subseteq E}\lambda_S f(S) : \sum_{S\subseteq E: i\in S}\lambda_S \le x_i \;\forall i\in E\,,\; \sum_{S\subseteq E}\lambda_S = 1\,,\; \lambda\ge 0\right\}. \tag{4.14}$$

Proof. Clearly, the optimal value of (4.14) is at least $\hat{f}(x)$. Take an optimal solution λ to (4.14) such that $\delta(\lambda) := \sum_{i\in E}\left(x_i - \sum_{S\subseteq E: i\in S}\lambda_S\right)$ is minimal. If $\delta = 0$, then λ is also feasible to (4.2), proving the claim. Assume that $\delta > 0$, and take any $i \in E$ for which $x_i > \sum_{S\subseteq E: i\in S}\lambda_S$. Since $x_i \le 1$ and $\sum_{S\subseteq E}\lambda_S = 1$, there exists a set $T \subseteq E$ with $\lambda_T > 0$, $i \notin T$.

Let us modify this solution to λ' defined as $\lambda'_{T+i} = \lambda_{T+i} + \varepsilon$, $\lambda'_T = \lambda_T - \varepsilon$, and $\lambda'_S = \lambda_S$ otherwise. For small enough $\varepsilon > 0$, λ' is also a feasible solution with $\delta(\lambda') < \delta(\lambda)$. Moreover, λ' is also optimal, since $\sum_{S\subseteq E}\lambda'_S f(S) \ge \sum_{S\subseteq E}\lambda_S f(S)$ by the monotonicity of f. This contradicts the choice of λ; consequently, $\delta(\lambda) = 0$ must hold and the claim follows. $\square$

Lemma 4.23. *Let $\mathcal{M} = (E, \mathcal{I})$ be a matroid with rank function r and weight $w \in \mathbb{R}^E_+$. Then, for $x \in [0,1]^E$*

$$\hat{r}_w(x) = \max\{w^\top y : y \in \mathcal{P}(r),\, y \le x\}\,.$$

Proof. Consider an optimal solution λ to the LP in (4.14) for $f = r_w$ with $\sum_{S\subseteq E}\lambda_S|S|$ minimal. We claim that every $S \subseteq E$ with $\lambda_S > 0$ must be independent. Indeed, recall that $r_w(S) = w(X)$ for some independent set $X \subseteq S$. If $S \notin \mathcal{I}$, then we can simply replace S in the combination by this set X. The solution remains feasible with the same objective value, but smaller $\sum_{S\subseteq E}\lambda_S|S|$.

Consequently, we may assume that $r_w(S) = w(S)$ for every $S \in \mathrm{supp}(\lambda)$. Letting $y_i = \sum_{S:\, i\in S}\lambda_i$, the objective of (4.14) can be written as

$$\sum_{S\subseteq V}\lambda_S r_w(S) = w^\top y\,.$$

Note that $y \le x$ and $y \in \mathcal{P}(r)$, since y can be written as a convex combination of incidence vectors of independent sets. Hence, (4.14) for $f = r_w$ is equivalent to maximizing $w^\top y$ over $y \in \mathcal{P}(r)$, $y \le x$, proving the statement. $\square$

Proof of Theorem 4.21. First, we show that the minimum of $R_w(x)/\hat{r}_w(x)$ is taken at some $x \in \mathcal{P}(r)$. Take a minimizer $x \notin \mathcal{P}(r)$. By Lemma 4.23, $\hat{r}_w(x) = w^\top y$ for some $y \in \mathcal{P}(r)$, $y \le x$. Clearly, $\hat{r}_w(y) = w^\top y$. By Proposition 4.14, we have $R_w(y) \le R_w(x)$. This proves that $R_w(x)/\hat{r}_w(x) \ge R_w(y)/\hat{r}_w(y)$, thus, equality must hold and y is also a minimizer of the correlation gap.

For the rest of the proof, consider a minimizer $x \in \mathcal{P}(r)$. Note that $R_w(x)/\hat{r}_w(x) = R_w(x)/w^\top x$. Among such minimizers, let us pick x such that $\mathrm{supp}(x)$ is minimal. The proof is complete by showing that $x(S^*) = r(S^*)$ for $S^* = \mathrm{supp}(x)$.

For a contradiction, assume $x(S^*) < r(S^*)$. We claim that there exists a $j \in S^*$ such that $x + \varepsilon\chi_j \in \mathcal{P}(r)$ for some $\varepsilon > 0$. If no such j exists, then there exists a set T_j for each $j \in S^*$ such that $j \in T_j$, and $x(T_j) = r(T_j)$. A standard uncrossing algorithm shows that $x(T) = r(T)$ for $T = \cup_{j \in S^*} T_j$. Clearly, $S^* \subseteq T$. But this implies $x(S^*) = r(S^*)$ since $x(S^*) = x(T)$ and $r(S^*) \le r(T)$. Thus, there exists a $j \in S^*$ such that $x + \varepsilon\chi_j \in \mathcal{P}(r)$ for some $\varepsilon > 0$.

For $\gamma \in [0,1]$, let x^γ be the vector obtained from x by replacing x_j by γ. Let $\Gamma = \max\{\gamma : x^\gamma \in \mathcal{P}(r)\}$. By the choice of j, $x_j < \Gamma$.

According to Proposition 4.13, $h(\gamma) = R_w(x^\gamma)$ is a linear function in γ; we can write $h(\gamma) = a + b\gamma$ for $a, b \in \mathbb{R}_+$. For $\gamma \in [0,\Gamma]$, $x^\gamma \in \mathcal{P}(r)$, and therefore $\hat{r}(x^\gamma) = w^\top x^\gamma$; this is also a linear function and can be written as $\hat{r}(x^\gamma) = c + d\gamma$, where $c = \sum_{i \ne j} w_i x_i$ and $d = w_j$. Hence, for $\gamma \in [0,\Gamma]$, we can write

$$\frac{R_w(x^\gamma)}{\hat{r}_w(x^\gamma)} = \frac{a + b\gamma}{c + d\gamma} .$$

It is easy to see that if $a/c < b/d$, then the unique minimizer on $\gamma \in [0,\Gamma]$ is $\gamma = 0$; if $a/c > b/d$, then the unique minimizer is $\gamma = \Gamma$. Both these cases contradict the optimal choice of x. Hence, we must have $a/c = b/d$, in which case the ratio is constant on $\gamma \in [0,\Gamma]$. Therefore, x^0 is also a minimizer. This is a contradiction to the minimal choice of $\mathrm{supp}(x)$. □

We are ready to prove Theorem 4.1, restated as follows.

Theorem 4.24. *Let $\mathcal{M} = (E, \mathcal{I})$ be a matroid with rank function r, and r_w the weighted rank function for some weights $w \in \mathbb{R}^E_+$, $w \ne 0$. Then,*

$$\mathcal{CG}(r_w) \ge \mathcal{CG}(r) .$$

Proof. For a contradiction, assume there exists a weight vector $w \ge 0$ and a point $x^* \in [0,1]^E$ such that $R_w(x^*)/\hat{r}_w(x^*) < \mathcal{CG}(r)$. According to Theorem 4.21, we can assume $x^* \in \mathcal{P}(r)$ and thus $\hat{r}_w(x^*) = w^\top x^*$.

Let $w^1 > w^2 > \cdots > w^k \geq 0$ denote the distinct values of w. For each $i \in [k]$, let $E_i \subseteq E$ denote the set of elements with weight w_i. Clearly, $k \geq 2$ as otherwise $R_w(x^*)/\hat{r}_w(x^*) = w^1 R(x^*)/(w^1 x^*(E)) = R(x^*)/x^*(E) \geq \mathcal{CG}(r)$. Let us pick a counterexample with k minimal.

First, we claim that $w_k > 0$. Indeed, if the smallest cost is $w_k = 0$, then $R_w(x^*)/\hat{r}_w(x^*)$ is unchanged by modifying to $w_e = w^1$ and $x^*_e = 0$ for all $e \in E_k$; this contradicts the minimal choice of k.

Let X be the random variable for the set obtained by sampling every element $e \in E$ independently with probability x^*_e. Let $I_X \subseteq X$ denote a maximum weight independent subset of X. Recall the well-known property of matroids that a maximum weight independent set can be selected greedily in decreasing order of the costs w_e. We fix an arbitrary tie-breaking rule inside each set E_i.

The correlation gap of r_w is given by

$$\frac{R_w(x^*)}{\hat{r}_w(x^*)} = \frac{\sum_{S\subseteq E} \Pr(X = S) r_w(S)}{w^\top x^*} = \frac{\sum_{e\in E} w_e \Pr(e \in I_X)}{\sum_{e\in E} w_e x^*_e} = \frac{\sum_{i=1}^k w^i \sum_{e\in E_i} \Pr(e \in I_X)}{\sum_{i=1}^k w^i \sum_{e\in E_i} x^*_e}.$$

Consider the set

$$J := \operatorname*{arg\,min}_{i\in[k]} \frac{\sum_{e\in E_i} \Pr(e \in I_X)}{\sum_{e\in E_i} x^*_e}.$$

We claim that $J \setminus \{1\} \neq \emptyset$. Suppose that $J = \{1\}$ for a contradiction. Define the point $x' \in \mathcal{P}(r)$ as $x'_e := x^*_e$ if $e \in E_1$, and $x'_e := 0$ otherwise. Then, we get a contradiction from

$$\mathcal{CG}(r) \leq \frac{R(x')}{\hat{r}(x')} = \frac{w^1 \sum_{e\in E_1} \Pr(e \in I_X)}{w^1 x^*(E_1)} < \frac{\sum_{i=1}^k w^i \sum_{e\in E_i} \Pr(e \in I_X)}{\sum_{i=1}^k w^i x^*(E_i))} = \frac{R_w(x^*)}{\hat{r}_w(x^*)}.$$

The first equality holds because for each element $e \in E_1$, $\Pr(e \in I_X)$ only depends on $x^*_{E_1} = x'_{E_1}$. This is by the greedy choice of I_X: elements in E_1 are selected regardless of $X \setminus E_1$. The strict inequality is due to $J = \{1\}$, $k \geq 2$ and $w_2 > 0$.

Now, pick any index $j \in J \setminus \{1\}$. Since $w^j > 0$, we have

$$\frac{w^j \sum_{e\in E_j} \Pr(e \in I_X)}{w^j \sum_{e\in E_j} x^*_e} \leq \frac{\sum_{i=1}^k w^i \sum_{e\in E_i} \Pr(e \in I_X)}{\sum_{i=1}^k w^i x^*(E_i)}.$$

So, we can increase w^j to w^{j-1} without increasing the correlation gap. That is, defining $\bar{w} \in \mathbb{R}^E_+$ as $\bar{w}_e := w^{j-1}$ if $e \in E_j$ and $\bar{w}_e := w_e$ otherwise, we get

$$\begin{aligned}\frac{R_w(x^*)}{\hat{r}_w(x^*)} &\geq \frac{\sum_{i\neq j} w^i \sum_{e\in E_i} \Pr(e \in I_X) + w^{j-1} \sum_{e\in E_j} \Pr(e \in I_X)}{\sum_{i\neq j} w^i x^*(E_i) + w^{j-1} x^*(E_j)} \\ &= \frac{\sum_{S\subseteq E} \Pr(X = S) f_{\bar{w}}(S)}{\bar{w}^\top x^*} \geq \min_{x\in[0,1]^E} \frac{F_{\bar{w}}(x)}{\hat{f}_{\bar{w}}(x)}\ .\end{aligned}$$

The equality holds because for every $S \subseteq E$, I_S remains a maximum-weight independent set with the new weights $\bar{w}$. This again contradicts the minimal choice of k. $\square$

4.5 Upper Bounds on the Correlation Gap

Recall that for uniform matroids (with $\gamma = \rho + 1$ and $\ell = \rho$) the bound in Theorem 4.2 simplifies to $1 - \frac{e^{-\ell}\ell^\ell}{\ell!}$, and in this case, the bound is tight. We now give simple upper bounds on the correlation gap of a matroid rank function with a given rank ρ and girth γ. We start with the simple observation that the correlation gap of a uniform rank $\gamma - 1$ matroid gives such an upper bound.

Claim 4.25. *For every $\rho, \ell \in \mathbb{N}$ where $\rho \geq \ell$, there exists a matroid $\mathcal{M} = (E, \mathcal{I})$ with rank ρ and girth $\ell + 1$ whose correlation gap is equal to $1 - \frac{e^{-\ell}\ell^\ell}{\ell!}$.*

Proof. Let $\mathcal{M}_1$ be a rank-ℓ uniform matroid on $n > \ell$ elements with rank function r_1. Let $\mathcal{M}_2$ be a free matroid on $\rho - \ell$ elements with rank function r_2. Consider the matroid $\mathcal{M} = \mathcal{M}_1 \oplus \mathcal{M}_2$ with rank function r. Note that $\mathcal{M}$ has rank ρ and girth $\ell + 1$. By Proposition 4.3, $\mathcal{CG}(r) = \min\{\mathcal{CG}(r_1), \mathcal{CG}(r_2)\}$. Let us define the function $\varphi : \mathbb{Z}_+ \to \mathbb{Z}_+$ as $\varphi(i) := \min\{i, \ell\}$. Clearly, $r_1(S) = \varphi(|S|)$. From Proposition 4.19 and [8], we have $\mathcal{CG}(r_1) \geq \alpha_\varphi = 1 - \frac{e^{-\ell}\ell^\ell}{\ell!}$. Let $x^* \in [0,1]^n$ be the point given by $x^* = \frac{\ell}{n} \cdot \mathbb{1}$. Then,

$$\frac{R(x^*)}{\hat{r}(x^*)} = \frac{\mathbb{E}[\varphi(\mathrm{Bin}(n, \ell/n))]}{\mathbb{1}^\top x^*} = \frac{\mathbb{E}[\varphi(\mathrm{Bin}(n, \ell/n))]}{\ell} \to \frac{\mathbb{E}[\varphi(\mathrm{Poi}(\ell))]}{\ell} = 1 - \frac{e^{-\ell}\ell^\ell}{\ell!}$$

as $n \to \infty$. The first equality is due to Lemma 4.23, while the last equality is by [9, Lemma 2.2]. Thus, $\mathcal{CG}(r_1) = 1 - \frac{e^{-\ell}\ell^\ell}{\ell!}$. It is left to show that $\mathcal{CG}(r_2) = 1$. For any $x \in [0,1]^{\rho-\ell}$, we have $R_2(x) = \mathbb{1}^\top x$. Moreover, $\hat{r}(x) = \mathbb{1}^\top x$ by Lemma 4.23. $\square$

We now give a better, albeit still non-tight upper bound.

Proposition 4.26. *For every $\rho, \ell \in \mathbb{N}$ where $\rho \geq \ell$, there exists a matroid $\mathcal{M} = (E, \mathcal{I})$ with rank ρ and girth $\ell + 1$ whose correlation gap is at most $1 - \frac{1}{e} + \frac{\ell}{e\rho}$.*

Proof. Let $k := \rho - \ell$. For some $n \in \mathbb{N}$, $n \geq \ell + 1$, let the ground set be $E = E_0 \sqcup E_1 \sqcup \cdots \sqcup E_k$, where $|E_0| = \ell n$ and $|E_i| = n$ for all $i \in [k]$. Our matroid $\mathcal{M}$ is constructed as the union of two matroids $\mathcal{M}^u$ and $\mathcal{M}^p$. The first matroid $\mathcal{M}^u = (E, \mathcal{I}^u)$ is the uniform matroid of rank ℓ on ground set E. The second matroid $\mathcal{M}^p = (E, \mathcal{I}^p)$ is the partition matroid on ground set E, where each E_i is a part of rank 1 for all $i \geq 1$; every element of E_0 is a loop in this matroid.

Matroid union is a well known matroid operation where every independent set of the union matroid is the union of two independent sets from each of the two matroids. We can write the rank function of $\mathcal{M}$ as (see e.g., [133, Corollary 42.1a]):

$$r(X) = \sum_{i=1}^{k} \min\{1, |E_i \cap X|\} + \min\left\{\ell, |E_0 \cap X| + \sum_{i=1}^{k} \max\{0, |E_i \cap X| - 1\}\right\} .$$

Note that the rank of the matroid is $r(E) = \gamma + k = \rho$, and the girth is $\gamma = \ell + 1$, since every γ element set is indepedent, but any $\gamma + 1$ element subset of E_0 is dependent.

Let us now fix $F \subseteq E_0$, $|F| = \ell$, and define x as $x_i = 1$ if $i \in F$, $x_i = 0$ if $i \in E_0 \setminus F$, and $x_i = 1/n$ if $i \in E \setminus E_0$. It is easy to verify that $x \in \mathcal{P}(r)$ as it can be written as a convex combination of n bases. Thus, $\hat{r}(x) = x(E) = \ell + k = \rho$.

Let us now compute the multilinear extension $R(x)$. Let $S \subseteq E$ be the random set sampled independently according to the probabilities x_i. We have $F \subseteq S$ with probability 1. From the above rank function expression, we get

$$r(S) = \ell + \sum_{i=1}^{k} \min\{1, |E_i \cap S|\} ,$$

therefore,

$$R(x) = \mathbb{E}\left[r(S)\right] = \ell + \sum_{i=1}^{k} \Pr[|E_i \cap S| \geq 1] = \ell + k\left(1 - \left(1 - \frac{1}{n}\right)^n\right) \to \rho - \frac{\rho - \gamma}{e} ,$$

as $n \to \infty$. From here, we see that

$$\lim_{n \to \infty} \frac{R(x)}{\hat{r}(x)} = 1 - \frac{1}{e} + \frac{\gamma}{e\rho} . \qquad \square$$

4.6 The Correlation Gap Bound for Matroids

This section is dedicated to the proof of Theorem 4.2. For the matroid $\mathcal{M} = (E, \mathcal{I})$, let r denote the rank function, $\rho = r(E)$ the rank, and γ the girth. We have $\gamma > 1$ since the matroid is assumed to be loopless.

According to Theorem 4.21, there exists a point $x^* \in \mathcal{P}(r)$ and a set $S \subseteq E$ such that $x^*(S) = r(S)$, $x^*(E \setminus S) = 0$, and $\mathcal{CG}(r) = R(x^*)/r(x^*)$. For notational convenience, let us define

$$\ell := \gamma - 1 , \qquad \lambda := x^*(E) = x^*(S) = r(S) .$$

Note that if $\lambda < \ell$, then S is independent. As $x^*(S) = r(S) = |S|$ and $x^* \leq \mathbb{1}$, we have $x^*_i = 1$ for all $i \in S$. In this case, it follows that x^* is integral and $R(x^*) = \hat{r}(x^*)$, so the correlation gap is 1. Henceforth, we will assume that $\lambda \geq \ell$.

In this section, we analyze the multilinear extension of r. Let $g\colon 2^E \to \mathbb{Z}_+$ be the rank function of a uniform matroid of rank ℓ over ground set E, and define the function $h := r - g$.

Clearly, $r = g + h$. By linearity of expectation, the multilinear extension of r can be written as

$$R(x) = \mathbb{E}[r(S)] = \mathbb{E}[g(S) + h(S)] = \mathbb{E}[g(S)] + \mathbb{E}[h(S)] = G(x) + H(x) \ , \tag{4.15}$$

where G and H are the multilinear extensions of g and h respectively. To lower bound $R(x^*)$, we will lower bound $G(x^*)$ and $H(x^*)$ separately.

4.6.1 Lower Bounding $G(x^*)$

Observe that G is a symmetric polynomial because g is the rank function of a uniform matroid. As g is submodular, Proposition 4.15 indicates that G is convex along $e_i - e_j$ for all $i, j \in E$. The next lemma is an easy consequence of these two properties. We have already proven it in a more general form in Lemma 4.18.

Lemma 4.27. *For any $x \in [0,1]^E$, we have $G(x) \geq G((x(E)/n) \cdot \mathbb{1})$.*

By Lemma 4.27, we have

$$G(x^*) \geq G\left(\frac{\lambda}{n} \cdot \mathbb{1}\right) = \sum_{k=0}^{n} \min\{k, \ell\} \binom{n}{k} \left(\frac{\lambda}{n}\right)^k \left(1 - \frac{\lambda}{n}\right)^{n-k} = \mathbb{E}\left[\min\left\{\mathrm{Bin}\left(n, \frac{\lambda}{n}\right), \ell\right\}\right] \ .$$

In other words, we can lower bound $G(x^*)$ by the expected value of $\mathrm{Bin}(n, \lambda/n)$ truncated at ℓ. We now use Lemma 4.12 on the concave order of the binomial and Poisson distributions to obtain

$$\mathbb{E}\left[\min\left\{\mathrm{Bin}\left(n, \frac{\lambda}{n}\right), \ell\right\}\right] \geq \mathbb{E}\left[\min\left\{\mathrm{Poi}(\lambda), \ell\right\}\right] = \sum_{k=0}^{\infty} \min\{k, \ell\} \frac{\lambda^k e^{-\lambda}}{k!} \ .$$

Using $\Pr(\mathrm{Poi}(\lambda) \geq j) = \sum_{k=j}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!}$, this amounts to

$$G(x^*) \geq \sum_{j=1}^{\ell} \Pr(\mathrm{Poi}(\lambda) \geq j) = \sum_{j=1}^{\ell} \left(1 - \sum_{k=0}^{j-1} \frac{\lambda^k e^{-\lambda}}{k!}\right) = \ell - \sum_{k=0}^{\ell-1} (\ell - k) \frac{\lambda^k e^{-\lambda}}{k!} \ . \tag{4.16}$$

4.6.2 Lower Bounding $H(x^*)$

Next, we turn to the extension H. We first describe the general setup, which is to incrementally build a set $Q(1)$ as follows. For each element $i \in E$, we put a Poisson clock of rate x_i^* on it. We initialize with $Q(0) = \emptyset$, and start all the clocks simultaneously at time $t = 0$. For $t \in [0,1]$, if the clock on an element rings at time t, we add that element to our current set. This process is terminated at time $t = 1$. This gives rise to the time-dependent set-valued random variable Q such that, for $t \in [0,1]$, $Q(t)$ is the random variable for the set at time t. This process can also be viewed as a continuous-time Markov chain, where the state space is the power set 2^E. From a set/state S, the possible transitions are to those sets S' where

$S \subset S'$ and $|S'| = |S| + 1$. Note that the Markov property is satisfied because both the holding time and transitions only depend on the current state $Q(t)$.

Due to the independence of the Poisson clocks, for every set $S \subseteq E$, we have

$$\Pr[Q(1) = S] = \prod_{i \in S} \Pr[i \in Q(1)] \prod_{i \notin S} \Pr[i \notin Q(1)] = \prod_{i \in S} (1 - e^{-x_i^*}) \prod_{i \notin S} e^{-x_i^*} \ .$$

Since h is monotone and $x_i^* \geq 1 - e^{-x_i^*}$ for all $i \in E$, Proposition 4.14 gives

$$H(x^*) \geq H(1 - e^{-x^*}) = \mathbb{E}[h(Q(1))] \ . \tag{4.17}$$

So, it suffices to lower bound $\mathbb{E}[h(Q(1))]$.

Let $t \in [0,1)$ and consider an infinitesimally small interval $[t, t+dt]$. For every element $i \in E$, the number of times its clock rings is a Poisson random variable with rate $x_i^* dt$. Hence, the probability that an element i is added to our set during this interval is

$$\Pr(\mathrm{Poi}(x_i^* dt) \geq 1) = 1 - e^{-x_i^* dt} = 1 - (1 - x_i^* dt + O(dt^2)) = x_i^* dt + O(dt^2) \ ,$$

where the second equality follows from Taylor's theorem. Observe that the probability of adding two or more elements during this interval is also $O(dt^2)$. Since dt is very small, we can effectively neglect all $O(dt^2)$ terms. Conditioning on the event $Q(t) = S$, the expected increase of $h(Q(t))$ (up to $O(dt^2)$ terms) is

$$\mathbb{E}[h(Q(t+dt)) - h(Q(t)) | Q(t) = S] = \sum_{i \in E} h_S(i) x_i^* dt.$$

From the definition of h, for each element $i \in E$, we have $h_S(i) = r_S(i)$ if $|S| \geq \ell$, and $h_S(i) = 0$ otherwise. This motivates the following definition.

Definition 4.28. We say that Q is *activated at time* t' if $|Q(t)| < \ell$ for all $t < t'$ and $|Q(t)| \geq \ell$ for all $t \geq t'$. We call t' the *activation time* of Q.

We denote the random variable for the activation time of Q by T.

For a fixed $t' \in \mathbb{R}_+$, if we further condition on the event $T = t'$, the expected increase of $h(Q(t))$ (up to $O(dt^2)$ terms) is

$$\mathbb{E}[h(Q(t+dt)) - h(Q(t)) | Q(t) = S \wedge T = t'] = \sum_{i \in E} r_S(i) x_i^* dt \tag{4.18}$$

for all $t \geq t'$ and $S \subseteq E$ where $|S| \geq \ell$. For such a set S, we have

$$h(S) + \sum_{i \in E} r_S(i) x_i^* = r(S) - \ell + \sum_{i \in E} r_S(i) x_i^* \geq r^*(x^*) - \ell = \hat{r}(x^*) - \ell = \mathbb{1}^\top x^* - \ell = \lambda - \ell.$$

The inequality follows from the definition of r^* in (4.7). The second equality is by Theorem 4.7, while the third equality is by Lemma 4.23 because $x^* \in \mathcal{P}(r)$. Hence, (4.18) becomes

$$\mathbb{E}[h(Q(t+dt)) - h(Q(t)) | Q(t) = S \wedge T = t'] \geq (\lambda - \ell - h(S))dt$$

Dividing by dt and taking expectation over S, we obtain for all $t \geq t'$,

$$\frac{1}{dt}\mathbb{E}[h(Q(t+dt)) - h(Q(t)) | T = t'] \geq \lambda - \ell - \mathbb{E}[h(Q(t)) | T = t']. \tag{4.19}$$

Let $\phi(t) := \mathbb{E}[h(Q(t)) | T = t']$. Then, (4.19) can be written as $\frac{d\phi}{dt} \geq \lambda - \ell - \phi(t)$. To solve this differential inequality, let $\psi(t) := e^t \phi(t)$ and consider $\frac{d\psi}{dt} = e^t(\frac{d\phi}{dt} + \phi(t)) \geq e^t(\lambda - \ell)$. Since $\psi(t') = \phi(t') = 0$, we get

$$\psi(t) = \int_{t'}^{t} \frac{d\psi}{ds} ds \geq \int_{t'}^{t} e^s(\lambda - \ell) ds = (e^t - e^{t'})(\lambda - \ell)$$

for all $t \geq t'$. It follows that $\mathbb{E}[h(Q(t)) | T = t'] = \phi(t) = e^{-t}\psi(t) \geq (1 - e^{t'-t})(\lambda - \ell)$ for all $t \geq t'$. In particular, at time $t = 1$, we have $\mathbb{E}[h(Q(1)) | T = t'] \geq (1 - e^{-(1-t')})(\lambda - \ell)$ for all $t' \leq 1$. By the law of total expectation,

$$\begin{aligned}
\mathbb{E}[h(Q(1))] = \mathbb{E}_T[\mathbb{E}[h(Q(1)) | T = t]] &= \int_0^\infty \Pr(T = t)\mathbb{E}[h(Q(1)) | T = t] dt \\
&= \int_0^1 \Pr(T = t)\mathbb{E}[h(Q(1)) | T = t] dt \\
&\geq (\lambda - \ell) \int_0^1 \Pr(T = t)(1 - e^{-(1-t)}) dt. \qquad (4.20)
\end{aligned}$$

Now, the cumulative distribution function of T is given by

$$\begin{aligned}
\Pr(T \leq t) &= 1 - \sum_{\substack{S \subseteq E: \\ |S| < \ell}} \prod_{i \in S}(1 - e^{-x_i^* t}) \prod_{i \notin S} e^{-x_i^* t} \\
&= 1 - \sum_{\substack{S \subseteq E: \\ |S| < \ell}} \sum_{T \subseteq S} (-1)^{|T|} e^{-x^*(T \cup (E \setminus S))t} \\
&= 1 - \sum_{S \subseteq E} \sum_{\substack{T \subseteq S: \\ |T| + |E \setminus S| < \ell}} (-1)^{|T|} e^{-x^*(S)t} && \text{(Change of variables } S \leftarrow T \cup E \setminus S) \\
&= 1 - \sum_{S \subseteq E} \sum_{k=0}^{|S|+\ell-n-1} (-1)^k \binom{|S|}{k} e^{-x^*(S)t} && (|T| \leq \ell - (n - |S|) - 1) \\
&= 1 - \sum_{S \subseteq E} (-1)^{|S|+\ell-n-1} \binom{|S| - 1}{|S| + \ell - n - 1} e^{-x^*(S)t} && \text{(Claim B.1)}
\end{aligned}$$

$$= 1 - \sum_{S \subseteq E} (-1)^{|S|+\ell-n-1} \binom{|S|-1}{n-\ell} e^{-x^*(S)t}.$$

Differentiating with respect to t yields the probability density function of T

$$\Pr(T = t) = \frac{d}{dt} \Pr(T \leq t) = \sum_{S \subseteq E} (-1)^{|S|+\ell-n-1} \binom{|S|-1}{n-\ell} x^*(S) e^{-x^*(S)t}.$$

Note that $\binom{|S|-1}{n-\ell} > 0$ if and only if $|S| \geq n+1-\ell$. Plugging this back into (4.20) gives us

$$\begin{aligned}
\mathbb{E}[h(Q(1))] &\geq (\lambda - \ell) \sum_{S \subseteq E} (-1)^{|S|+\ell-n-1} \binom{|S|-1}{n-\ell} x^*(S) \int_0^1 e^{-x^*(S)t} - e^{-1-(x^*(S)-1)t} dt \\
&= (\lambda - \ell) \sum_{S \subseteq E} (-1)^{|S|+\ell-n-1} \binom{|S|-1}{n-\ell} \left(1 - e^{-1} - \frac{e^{-1} - e^{-x^*(S)}}{x^*(S)-1}\right), \qquad (4.21)
\end{aligned}$$

where the equality is due to

$$\begin{aligned}
x^*(S) \int_0^1 e^{-x^*(S)t} - e^{-1-(x^*(S)-1)t} dt &= x^*(S) \left[-\frac{e^{-x^*(S)t}}{x^*(S)} + \frac{e^{-1-(x^*(S)-1)t}}{x^*(S)-1} \right]_0^1 \\
&= \left[-e^{-x^*(S)t} + \left(1 + \frac{1}{x^*(S)-1}\right) e^{-1-(x^*(S)-1)t} \right]_0^1 \\
&= -e^{-x^*(S)} + 1 + \left(1 + \frac{1}{x^*(S)-1}\right) \left(e^{-x^*(S)} - e^{-1}\right) \\
&= 1 - e^{-1} - \frac{e^{-1} - e^{-x^*(S)}}{x^*(S)-1}.
\end{aligned}$$

Observe that (4.21) is well-defined because whenever $x^*(S) = 1$, L'Hôpital's rule gives us

$$\frac{e^{-1} - e^{-x^*(S)}}{x^*(S)-1} = \lim_{t \to 1} \frac{e^{-1} - e^{-t}}{t-1} = \lim_{t \to 1} \frac{e^{-t}}{1} = e^{-1}.$$

Since $\ell > 0$ (as $\mathcal{M}$ has no loops), Claim B.2 allows us to extract the first part of (4.21) as

$$\sum_{S \subseteq E} (-1)^{|S|+\ell-n-1} \binom{|S|-1}{n-\ell} (1 - e^{-1}) = (1 - e^{-1}) \sum_{k=0}^{n} (-1)^{k+\ell-n-1} \binom{n}{k} \binom{k-1}{n-\ell} = 1 - e^{-1}.$$

Pulling out a factor -1 from the remaining term, (4.21) becomes

$$\mathbb{E}[h(Q(1))] \geq (\lambda - \ell) \left[1 - e^{-1} + e^{-1} \sum_{S \subseteq E} (-1)^{|S|+\ell-n} \binom{|S|-1}{n-\ell} \frac{1 - e^{-(x^*(S)-1)}}{x^*(S)-1} \right]. \qquad (4.22)$$

Rounding x^* to an integer point

Consider the function $\rho\colon \mathbb{R}_+ \to \mathbb{R}_+$ defined by

$$\rho(t) := \frac{1-e^{-t}}{t}.$$

and the last part of (4.22)

$$\begin{aligned}\psi(x) &:= \sum_{S\subseteq E} (-1)^{|S|+\ell-n} \binom{|S|-1}{n-\ell} \frac{1-e^{-(x(S)-1)}}{x(S)-1} \\ &= \sum_{S\subseteq E} (-1)^{|S|+\ell-n} \binom{|S|-1}{n-\ell} \rho(x(S)-1) \ .\end{aligned} \tag{4.23}$$

as a function on $[0,1]^n$. The next observation is well-known, and underpins the pipage rounding technique by Ageev and Sviridenko [1]. For the sake of completeness, we include a proof.

Observation 4.29. *Let $f\colon [0,1]^n \to \mathbb{R}$ be a function such that for any $x \in [0,1]^n$ and $i,j \in [n]$,*

$$f_{ij}^x(t) := f(x + t(e_i - e_j))$$

is a concave function on the domain $\{t \in [-1,1]\colon\ x + t(e_i - e_j) \in [0,1]^n\}$. Then, for any $y \in [0,1]^n$ where $\mathbb{1}^\top y \in \mathbb{Z}$, there exists an integral $z \in \{0,1\}^n$ such that $f(y) \geq f(z)$ and $\mathbb{1}^\top y = \mathbb{1}^\top z$.

Proof. We proceed by strong induction on the number k of non-integral coordinates in y. The base case $k = 0$ is trivial by picking $z = y$. Suppose that there exists an $\ell \in \mathbb{Z}_+$ such that the statement is true for all $k \in \{0, 2, 3, \dots, \ell\}$. Consider the case $k = \ell + 1$. Note that $k \neq 1$ because $\mathbb{1}^\top y \in \mathbb{Z}$. Let $i,j \in [n]$ be distinct indices such that $y_i, y_j \in (0,1)$. Since f_{ij}^y is concave, for all $t \geq 0$ or for all $t \leq 0$, we have $f_{ij}^y(t) \leq f_{ij}^y(0)$. Let $\varepsilon' = \min\{1 - y_i, y_j\}$ and $\varepsilon'' = \min\{y_i, 1 - y_j\}$, along with their corresponding points $y' = y + \varepsilon'(e_i - e_j)$ and $y'' = y - \varepsilon''(e_i - e_j)$. Then,

$$\min\{f(y'), f(y'')\} = \min\{f_{ij}^y(\varepsilon'), f_{ij}^y(-\varepsilon'')\} \leq f_{ij}^y(0) = f(y) \ .$$

Let k' and k'' be the number of non-integral coordinates in y' and y'' respectively. Note that $k', k'' \neq 1$ because $\mathbb{1}^\top y' = \mathbb{1}^\top y'' = \mathbb{1}^\top y \in \mathbb{Z}$. Since $k', k'' \leq \ell$, by the inductive hypothesis there exist integral $z', z'' \in \{0,1\}^n$ such that $f(z') \leq f(y')$ and $f(z'') \leq f(y'')$. Thus, our desired $z \in \{0,1\}^n$ can be chosen as

$$z = \operatorname*{arg\,min}_{x \in \{z', z''\}} f(x) \ . \qquad \square$$

We would like to round x^* to a binary vector using Observation 4.29. Hence, we need to prove concavity of ψ along the directions $e_i - e_j$. Taking the second partial derivatives of (4.23) yields

$$\frac{\partial^2 \psi}{\partial x_i \partial x_j}(x) = \sum_{\substack{S \subseteq E: \\ i,j \in S}} (-1)^{|S|+\ell-n} \binom{|S|-1}{n-\ell} \rho''(x(S)-1) \ . \tag{4.24}$$

The following claim provides a closed-form expression for all the derivatives of ρ, and highlights the alternating behaviour of their signs.

Claim 4.30. *For any $k \in \mathbb{Z}_+$, the kth derivative of ρ is given by*

$$\rho^{(k)}(t) = (-1)^k k! \left(\frac{1 - e^{-t} \sum_{i=0}^{k} t^i/i!}{t^{k+1}} \right) \ .$$

Consequently, if k is even, then $\rho^{(k)}(t) > 0$ for all $t \geq 0$. Otherwise, $\rho^{(k)}(t) < 0$ for all $t \geq 0$.

Proof. We prove the first part by induction on $k \geq 0$. The base case $k = 0$ is clear. For the inductive step,

$$\begin{aligned}
\rho^{(k+1)}(t) &= (-1)^k k! \cdot \frac{\left(e^{-t}\sum_{i=0}^{k} t^i/i! - e^{-t}\sum_{i=0}^{k-1} t^i/i!\right) t^{k+1} - \left(1 - e^{-t}\sum_{i=0}^{k} t^i/i!\right)(k+1)t^k}{t^{2k+2}} \\
&= (-1)^k k! \cdot \frac{e^{-t}t^{k+1}/k! - \left(1 - e^{-t}\sum_{i=0}^{k} t^i/i!\right)(k+1)}{t^{k+2}} \\
&= (-1)^k (k+1)! \cdot \frac{e^{-t}t^{k+1}/(k+1)! - 1 + e^{-t}\sum_{i=0}^{k} t^i/i!}{t^{k+2}} \\
&= (-1)^{(k+1)} (k+1)! \cdot \frac{1 - e^{-t}\sum_{i=0}^{k+1} t^i/i!}{t^{k+2}}
\end{aligned}$$

as required. For the second part, note that at $t = 0$, applying L'Hôpital's rule yields

$$\rho^{(k)}(0) = \lim_{t\to 0} (-1)^k k! \left(\frac{e^{-t}\sum_{i=0}^{k} t^i/i! - e^{-t}\sum_{i=0}^{k-1} t^i/i!}{(k+1)t^k} \right) = \lim_{t \to 0} (-1)^k k! \left(\frac{e^{-t}t^k/k!}{(k+1)t^k} \right) = \frac{(-1)^k}{k+1}.$$

Hence, $\rho^{(k)}(0) > 0$ if k is even, and $\rho^{(k)}(0) < 0$ if k is odd. Now, let us rewrite $\rho^{(k)}(t)$ as

$$\rho^{(k)}(t) = (-1)^k \frac{k! e^{-t}}{t^{k+1}} \left(e^t - \sum_{i=0}^{k} \frac{1}{i!} t^i \right).$$

By the Maclaurin series of e^t, for all $t > 0$, we have $\rho^{(k)}(t) > 0$ if k is even, and $\rho^{(k)}(t) < 0$ if k is odd. □

For the proof of concavity, we need the following notion of finite difference.

Definition 4.31. Given a function $\phi\colon \mathbb{R} \to \mathbb{R}$ and a scalar $x \in \mathbb{R}_+$, the *forward difference* of $\phi(t)$ is

$$\Delta_x[\phi](t) := \phi(t+x) - \phi(t) \ .$$

More generally, for a vector $(x_1, \ldots, x_n) = (\tilde{x}, x_n) \in \mathbb{R}^n_+$, the *nth-order forward difference* of $\phi(t)$ is

$$\Delta_x[\phi](t) \coloneqq \Delta_{x_n}[\Delta_{\tilde{x}}[\phi]](t) = \Delta_{x_n}[\Delta_{x_{n-1}}[\cdots \Delta_{x_1}[\phi] \cdots]](t) \ .$$

Claim 4.32. *For any function $\phi\colon \mathbb{R} \to \mathbb{R}$ and vector $x \in \mathbb{R}^n_+$, we have*

$$\Delta_x[\phi](t) = \sum_{S \subseteq [n]} (-1)^{n-|S|} \phi(t + x(S)). \tag{4.25}$$

Proof. The claim follows by induction on n. For $n = 1$, the formula simplifies to $\Delta_x[\phi](t) = (-1)^{1-1}\phi(t+x) - (-1)^{1-0}\phi(t)$, which holds by definition. So, let $n > 1$ and $(x_1, \ldots, x_n) = (\tilde{x}, x_n) \in \mathbb{R}^n_+$. Using the induction hypothesis and linearity of the difference operator, we get

$$\begin{aligned} \Delta^n_x[\phi](t) = \Delta_{x_n}[\Delta_{\tilde{x}}[\phi]](t) &= \sum_{S \subseteq [n-1]} (-1)^{n-1-|S|} \left(\phi(t + \tilde{x}(S) + x_n) - \phi(t + \tilde{x}(S))\right) \\ &= \sum_{S \subseteq [n]} (-1)^{n-|S|} \phi(t + x(S)) \ . \end{aligned}$$

□

Therefore, in the definition of Δ_x, the order in which the difference operators $\{\Delta_{x_i} : i \in [n]\}$ are applied does not matter. The next claim relates the signs of $\phi^{(n)}$ and $\Delta_x[\phi]$.

Claim 4.33. *Let $\phi\colon \mathbb{R} \to \mathbb{R}$ be an n-times differentiable function. For any $x \in \mathbb{R}^n_+$ and $t \in \mathbb{R}$, if $\phi^{(n)}(s) \geq 0$ for all $t \leq s \leq t + \mathbb{1}^\top x$, then $\Delta_x[\phi](t) \geq 0$.*

Proof. We proceed by induction on $n \geq 1$. The base case $n = 1$ is clear as

$$0 \leq \int_t^{t+x} \phi^{(1)}(s) ds = \phi(t+x) - \phi(t) = \Delta_x[\phi](t) \ .$$

For the inductive step, let $x \in \mathbb{R}^n_+$, $y \in \mathbb{R}_+$, and assume that $\phi^{(n+1)}(s) \geq 0$ for all $t \leq s \leq t + \mathbb{1}^\top x + y$. Applying the inductive hypothesis to $\phi^{(1)}$, we have $\Delta_x[\phi^{(1)}](s) \geq 0$ for all $t \leq s \leq t + y$. Using the linearity of the derivative applied to the representation (4.25), we obtain

$$\begin{aligned} 0 \leq \int_t^{t+y} \Delta_x[\phi^{(1)}](s) ds &= \int_t^{t+y} \left(\frac{d}{ds} \Delta_x[\phi](s)\right) ds \\ &= \Delta_x[\phi](t+y) - \Delta_x[\phi](t) = \Delta_y[\Delta_x[\phi]](t) = \Delta_{(x,y)}[\phi](t) \end{aligned}$$

□

We are now ready to show concavity.

Lemma 4.34. *The function $\psi(x)$ is concave along the direction $e_a - e_b$ for all $a, b \in E$.*

Proof. Fix $a, b \in E$ and consider the function $\phi(t) := \psi(x+t(e_a-e_b))$, obtained by restricting ψ along the direction $e_a - e_b$. By substituting $y := x + t(e_a - e_b)$ and applying chain rule, the second derivative of $\phi(t)$ is given by

$$\phi''(t) = \frac{d}{dt}\left(\sum_{i\in E}\frac{\partial\psi}{\partial y_i}\frac{dy_i}{dt}\right) = \frac{d}{dt}\left(\frac{\partial\psi}{\partial y_a} - \frac{\partial\psi}{\partial y_b}\right)$$
$$= \sum_{i\in E}\left(\frac{\partial^2\psi}{\partial y_a\partial y_i} - \frac{\partial^2\psi}{\partial y_b\partial y_i}\right)\frac{dy_i}{dt} = \frac{\partial^2\psi}{\partial y_a^2} + \frac{\partial^2\psi}{\partial y_b^2} - 2\frac{\partial^2\psi}{\partial y_a\partial y_b}.$$

By (4.24), this is equal to

$$\phi''(t) = \sum_{\substack{S\subseteq E:\\ a\in S}}(-1)^{|S|+\ell-n}\binom{|S|-1}{n-\ell}\rho''(y(S)-1) + \sum_{\substack{S\subseteq E:\\ b\in S}}(-1)^{|S|+\ell-n}\binom{|S|-1}{n-\ell}\rho''(y(S)-1)$$
$$- 2\sum_{\substack{S\subseteq E:\\ a,b\in S}}(-1)^{|S|+\ell-n}\binom{|S|-1}{n-\ell}\rho''(y(S)-1)$$
$$= \sum_{\substack{S\subseteq E:\\ a\in S, b\notin S}}(-1)^{|S|+\ell-n}\binom{|S|-1}{n-\ell}\rho''(y(S)-1) + \sum_{\substack{S\subseteq E:\\ a\notin S, b\in S}}(-1)^{|S|+\ell-n}\binom{|S|-1}{n-\ell}\rho''(y(S)-1).$$

We show that each of the two sums above is nonpositive. Let us consider the first sum; the second sum follows by symmetry. In the first sum, every set $S \subseteq E \setminus b$ where $a \in S$ has an associated factor $\binom{|S|-1}{n-\ell}$. It can be interpreted as the number of subsets in $S \setminus a$ of size $n - \ell$. By charging the term associated with S to these subsets, we can rewrite the first sum as

$$\sum_{\substack{S\subseteq E:\\ a\in S, b\notin S}}(-1)^{|S|+\ell-n}\binom{|S|-1}{n-\ell}\rho''(y(S)-1) = \sum_{\substack{C\subseteq E\setminus\{a,b\}:\\ |C|=n-\ell}}\ \sum_{D\subseteq E\setminus(C\cup\{a,b\})}(-1)^{|D|+1}\rho''(y(C\cup D\cup a)-1).$$

Hence, for a fixed set $C \subseteq E \setminus \{a, b\}$ with $|C| = n - \ell$, it suffices to show that

$$\sum_{D\subseteq E\setminus(C\cup\{a,b\})}(-1)^{|D|+1}\rho''(\alpha + y(D)) \tag{4.26}$$

is nonpositive, where we denote $\alpha := y(C \cup a) - 1$. Note that $\alpha \geq 0$ because

$$y(C\cup a) = y(E) - y(E\setminus(C\cup a)) \geq \lambda - |E\setminus(C\cup a)| = \lambda - (n-(n-\ell+1)) = \lambda - \ell + 1 \geq 1,$$

where the first inequality is due to $y(E) = x(E) = \lambda$ and $y \leq \mathbb{1}$. Since $|E \setminus (C \cup \{a, b\})| = n - (n - \ell + 2) = \ell - 2$, we can express (4.26) as the following $(\ell - 2)$th-order forward difference

$$(-1)^{1-\ell} \sum_{D \subseteq E \setminus (C \cup \{a,b\})} (-1)^{\ell-2-|D|} \rho''(\alpha + y(D)) \overset{(4.25)}{=} (-1)^{1-\ell} \Delta_{y_{E \setminus (C \cup \{a,b\})}}[\rho''](\alpha) \ . \tag{4.27}$$

Recall that $\rho^{(k)}(t) > 0$ for all $t \geq 0$ if k is even, and $\rho^{(k)}(t) < 0$ for all $t \geq 0$ if k is odd by Claim 4.30. As $y \in \mathbb{R}_+^{\ell-2}$ and $\alpha \geq 0$, applying Claim 4.33 yields $\Delta_{y_{E \setminus (C \cup \{a,b\})}}[\rho''](\alpha) \geq 0$ if ℓ is even, and $\Delta_{y_{E \setminus (C \cup \{a,b\})}}[\rho''](\alpha) \leq 0$ if ℓ is odd. In both cases, (4.27) is nonpositive. □

Lemma 4.34 allows us to round x^* according to Observation 4.29. In particular, there exists an integral vector $x' \in \{0, 1\}^n$ such that $\psi(x^*) \geq \psi(x')$ and $x^*(E) = x'(E) = \lambda$. Note that x' has exactly λ ones and $n - \lambda$ zeroes; recall that $\lambda \in \mathbb{Z}$ by Theorem 4.21. Let T be the set of elements $i \in E$ where $x'_i = 1$. Then, applying (4.23) yields

$$\begin{aligned} \psi(x') &= \sum_{S \subseteq E} (-1)^{|S|+\ell-n} \binom{|S|-1}{n-\ell} \frac{1 - e^{-(x'(S)-1)}}{x'(S) - 1} \\ &= \sum_{S \subseteq E} (-1)^{|S|+\ell-n} \binom{|S|-1}{n-\ell} \frac{1 - e^{-(|S \cap T|-1)}}{|S \cap T| - 1} \ . \end{aligned} \tag{4.28}$$

Simplifications for an integer point

In (4.28), every term in the sum only depends on the cardinality of S and $S \cap T$, instead of the actual set S. This allows us to rearrange the sum based on $|S \cap T|$ ranging from 0 to $|T| = \lambda$, and $|S \setminus T|$ ranging from 0 to $|E \setminus T| = n - \lambda$:

$$\begin{aligned} \psi(x') &= \sum_{i=0}^{\lambda} \sum_{j=0}^{n-\lambda} \binom{\lambda}{i} \binom{n-\lambda}{j} (-1)^{i+j+\ell-n} \binom{i+j-1}{n-\ell} \frac{1 - e^{-(i-1)}}{i-1} \\ &= \sum_{i=0}^{\lambda} \binom{\lambda}{i} \frac{1 - e^{-(i-1)}}{i-1} \sum_{j=0}^{n-\lambda} (-1)^{i+j+\ell-n} \binom{n-\lambda}{j} \binom{i+j-1}{n-\ell} \\ &= \sum_{i=0}^{\lambda} \binom{\lambda}{\lambda-i} \frac{1 - e^{-(\lambda-i-1)}}{\lambda-i-1} \sum_{j=0}^{n-\lambda} (-1)^{\ell-i-j} \binom{n-\lambda}{n-\lambda-j} \binom{n-i-j-1}{n-\ell} \ . \quad \left(\substack{i \leftarrow \lambda-i \\ j \leftarrow n-\lambda-j}\right) \end{aligned}$$

Recall that, by our convention, a binomial coefficient is zero if the upper part is smaller than the lower part. So, by the last binomial coefficient above, we may restrict to $n - \ell \leq n - i - j - 1 \leq n - i - 1$, which is equivalent to $i \leq \ell - 1$ and $j \leq \ell - 1 - i$. This yields

$$\psi(x') = \sum_{i=0}^{\ell-1} \binom{\lambda}{i} \frac{1 - e^{-(\lambda-i-1)}}{\lambda-i-1} \sum_{j=0}^{\ell-1-i} (-1)^{\ell-i-j} \binom{n-\lambda}{j} \binom{n-i-j-1}{\ell-i-j-1}.$$

Note that we introduced additional terms to the inner sum if $n - \lambda < \ell - 1 - i$, but they are all 0 due to the binomial coefficients $\binom{n-\lambda}{j}$. Applying Claim B.4 with $j \leftarrow \ell - 1 - i$, $k \leftarrow \lambda - 1 - i$, $n \leftarrow n - 1 - i$ to the inner sum gives $(-1)^{\ell-i}\binom{\lambda-i-1}{\ell-i-1}$, leading to

$$\psi(x') = \sum_{i=0}^{\ell-1} (-1)^{\ell-i} \binom{\lambda}{i} \binom{\lambda - i - 1}{\ell - i - 1} \frac{1 - e^{-(\lambda-i-1)}}{\lambda - i - 1} \ . \tag{4.29}$$

Observe that (4.22) evaluates to 0 if $\lambda = \ell$. So, we may now assume that $\lambda > \ell$. This allows us to apply the simple reformulation $\frac{1}{\lambda-i-1}\binom{\lambda-i-1}{\ell-i-1} = \frac{1}{\lambda-i-1}\frac{(\lambda-i-1)!}{(\ell-i-1)!(\lambda-\ell)!} = \frac{(\lambda-i-2)!}{(\ell-i-1)!(\lambda-\ell-1)!}\frac{1}{\lambda-\ell} = \frac{1}{\lambda-\ell}\binom{\lambda-i-2}{\ell-i-1}$ to obtain

$$\psi(x') = \frac{1}{\lambda - \ell} \sum_{i=0}^{\ell-1} (-1)^{\ell-i} \binom{\lambda}{i} \binom{\lambda - i - 2}{\ell - i - 1} \left(1 - e^{-(\lambda-i-1)}\right) \ . \tag{4.30}$$

Applying Claim B.6 with $j \leftarrow \ell - 1$ and $n \leftarrow \lambda$ to $\sum_{i=0}^{\ell-1}(-1)^i\binom{\lambda}{i}\binom{\lambda-i-2}{\ell-i-1}$ gives $(-1)^{\ell-1}\ell$, resulting in

$$\begin{aligned}\psi(x') &= \frac{1}{\lambda - \ell}\left(-\ell - \sum_{i=0}^{\ell-1} (-1)^{\ell-i} \binom{\lambda}{i}\binom{\lambda - i - 2}{\ell - i - 1} e^{-(\lambda-i-1)}\right) \\ &= \frac{1}{\lambda - \ell}\left(-\ell + e^{-\lambda+1} \sum_{i=0}^{\ell-1} (-1)^{\ell-i-1} \binom{\lambda}{i}\binom{\lambda - i - 2}{\ell - i - 1} e^{i}\right) \ .\end{aligned} \tag{4.31}$$

Further simplifications

To simplify the expression in (4.31), we consider the sum as a function of x for $x = e$. More precisely, given integral parameters $\lambda, \ell > 0$, we define the function $w_{\lambda,\ell} : \mathbb{R} \to \mathbb{R}$ as

$$w_{\lambda,\ell}(x) := \sum_{i=0}^{\ell-1} (-1)^{\ell-1-i} \binom{\lambda}{i} \binom{\lambda - 2 - i}{\ell - 1 - i} x^i \ .$$

Note that $w_{\lambda,\ell}$ is a polynomial on $\mathbb{R}$.

Claim 4.35. *For any integers $\lambda > \ell$, we have $w_{\lambda,\ell}(1) = \ell$ and $w'_{\lambda,\ell}(x) = \lambda w_{\lambda-1,\ell-1}(x)$. In particular, $w^{(i)}_{\lambda,\ell}(1) = \frac{\lambda!}{(\lambda-i)!}(\ell - i)$.*

Proof. The first property follows from Claim B.6. For the second property,

$$\begin{aligned}
w'_{\lambda,\ell}(x) &= \sum_{i=1}^{\ell-1}(-1)^{\ell-1-i}\frac{\lambda!}{(i-1)!(\lambda-i)!}\binom{\lambda-2-i}{\ell-1-i}x^{i-1} \\
&= \lambda\sum_{i=1}^{\ell-1}(-1)^{\ell-2-(i-1)}\binom{\lambda-1}{i-1}\binom{(\lambda-1)-2-(i-1)}{(\ell-1)-1-(i-1)}x^{i-1} \\
&= \lambda\sum_{i=0}^{\ell-2}(-1)^{\ell-2-i}\binom{\lambda-1}{i}\binom{(\lambda-1)-2-i}{(\ell-1)-1-i}x^{i} = \lambda w_{\lambda-1,\ell-1}(x).
\end{aligned}$$

The formula for the derivatives follows by induction. □

By Taylor's Theorem and Claim 4.35, we get

$$w_{\lambda,\ell}(x) = \sum_{i=0}^{\ell-1}\frac{w^{(i)}_{\lambda,\ell}(1)}{i!}(x-1)^i = \sum_{i=0}^{\ell-1}\frac{\lambda!}{i!(\lambda-i)!}(\ell-i)(x-1)^i = \sum_{i=0}^{\ell-1}\binom{\lambda}{i}(\ell-i)(x-1)^i\ .$$

Plugging this back into (4.31) gives us

$$\psi(x') = \frac{1}{\lambda-\ell}\left(-\ell+e^{-\lambda+1}w_{\lambda,\ell}(e)\right) = \frac{1}{\lambda-\ell}\left(-\ell+e^{-\lambda+1}\sum_{i=0}^{\ell-1}\binom{\lambda}{i}(\ell-i)(e-1)^i\right)\ . \tag{4.32}$$

Therefore, the multilinear extension of h at x^* is lower bounded by

$$\begin{aligned}
H(x^*) &\geq \mathbb{E}[h(Q(1))] && \text{(by (4.17))}\\
&\geq (\lambda-\ell)\left[1-e^{-1}+e^{-1}\psi(x^*)\right] && \text{(by (4.22))}\\
&\geq (\lambda-\ell)\left[1-e^{-1}+e^{-1}\psi(x')\right] && \text{(rounding via Observation 4.29 and Lemma 4.34)}\\
&= (\lambda-\ell)\left[1-e^{-1}+\frac{e^{-1}}{\lambda-\ell}\left(-\ell+e^{-\lambda+1}\sum_{i=0}^{\ell-1}\binom{\lambda}{i}(\ell-i)(e-1)^i\right)\right] && \text{(by (4.32))}\\
&= (\lambda-\ell)(1-e^{-1})-\ell e^{-1}+e^{-\lambda}\sum_{i=0}^{\ell-1}\binom{\lambda}{i}(\ell-i)(e-1)^i\\
&= \lambda-\ell-\lambda e^{-1}+e^{-\lambda}\sum_{i=0}^{\ell-1}\binom{\lambda}{i}(\ell-i)(e-1)^i\ . && (4.33)
\end{aligned}$$

4.6.3 Putting Everything Together

We are finally ready to lower bound the correlation gap of the matroid rank function r. Recall that we assumed that $\lambda > \ell$ in the previous subsection. Combining the lower bounds in (4.16) and (4.33) gives us

$$\begin{aligned}\mathcal{CG}(r) = \frac{R(x^*)}{\hat{r}(x^*)} &= \frac{G(x^*)+H(x^*)}{\mathbb{1}^\top x^*} \\ &\geq \frac{1}{\lambda}\left[\ell - \sum_{i=0}^{\ell-1}(\ell-i)\frac{\lambda^i e^{-\lambda}}{i!} + \lambda - \ell - \lambda e^{-1} + e^{-\lambda}\sum_{i=0}^{\ell-1}\binom{\lambda}{i}(\ell-i)(e-1)^i\right] \\ &= 1 - e^{-1} + \frac{e^{-\lambda}}{\lambda}\sum_{i=0}^{\ell-1}(\ell-i)\left[\binom{\lambda}{i}(e-1)^i - \frac{\lambda^i}{i!}\right] \end{aligned} \tag{4.34}$$

On the other hand, if $\lambda = \ell$, then $h = 0$. In this case, we obtain

$$\mathcal{CG}(r) = \frac{R(x^*)}{\hat{r}(x^*)} = \frac{G(x^*)+H(x^*)}{\mathbb{1}^\top x^*} = \frac{G(x^*)}{\ell} \overset{(4.16)}{\geq} 1 - \sum_{k=0}^{\ell-1}\left(1-\frac{k}{\ell}\right)\frac{\ell^k e^{-\ell}}{k!} = 1 - \frac{\ell^{\ell-1}e^{-\ell}}{(\ell-1)!},$$

which also agrees with (4.34) by Proposition 4.20.

To better understand the sum in (4.34), consider the function $\phi_\lambda^\xi : [\lambda] \to \mathbb{R}$ defined as

$$\phi_\lambda^\xi(i) := \xi\binom{\lambda}{i}(e-1)^i - \frac{\lambda^i}{i!} \quad , \tag{4.35}$$

with parameters $\xi \in \mathbb{R}_+$ and $\lambda \in \mathbb{N}$. The next claim illustrates the behaviour of ϕ_λ^ξ when $\xi \geq 1/(e-1)$.

Claim 4.36. *Given parameters $\xi \geq \frac{1}{e-1}$ and $\lambda \in \mathbb{N}$, the function ϕ_λ^ξ satisfies the following properties:*

(a) If $1 \leq i \leq (\frac{e-2}{e-1})\lambda + 1$, then $\phi_\lambda^\xi(i) \geq 0$.

(b) If $\phi_\lambda^\xi(i) < 0$, then $\phi_\lambda^\xi(i+1) < 0$.

Proof. Fix parameters $\xi \geq \frac{1}{e-1}$ and $\lambda \in \mathbb{N}$. For $i \in [\lambda]$, we can write

$$\begin{aligned}\phi_\lambda^\xi(i) = \xi\binom{\lambda}{i}(e-1)^i - \frac{\lambda^i}{i!} &= \frac{1}{i!}\left(\xi\prod_{j=0}^{i-1}((e-1)(\lambda-j)) - \lambda^i\right) \\ &= \frac{\lambda}{i!}\left(\xi(e-1)\prod_{j=1}^{i-1}((e-1)(\lambda-j)) - \lambda^{i-1}\right).\end{aligned}$$

To prove the first statement, note that

$$i \leq \left(\frac{e-2}{e-1}\right)\lambda + 1 \iff \lambda \leq (e-1)(\lambda - i + 1).$$

Hence, $\lambda \leq (e-1)(\lambda-j)$ for all $j \in [i-1]$. As we also have $\xi(e-1) \geq 1$, it follows that $\phi_\lambda^\xi(i) \geq 0$. Next, we prove the second statement. Since $\phi_\lambda^\xi(i) < 0$, we obtain $\lambda > (e-1)(\lambda - i + 1)$ by the first statement. Therefore,

$$\begin{aligned}
\phi_\lambda^\xi(i+1) &= \frac{\lambda}{(i+1)!}\left(\xi(e-1)\prod_{j=1}^{i}((e-1)(\lambda-j))-\lambda^i\right)\\
&= \frac{\lambda^2}{(i+1)!}\left(\xi(e-1)\frac{(e-1)(\lambda-i)}{\lambda}\prod_{j=1}^{i-1}((e-1)(\lambda-j))-\lambda^{i-1}\right)\\
&< \frac{\lambda^2}{(i+1)!}\cdot\phi_\lambda^\xi(i) < 0\ . \qquad \square
\end{aligned}$$

Applying Claim 4.36 with $\xi = 1$ allows us to show that the bound in Theorem 4.2 is strictly greater than $1 - 1/e$.

Lemma 4.37. *For every $\lambda, \ell \in \mathbb{N}$ such that $\lambda \geq \ell$, we have*

$$\sum_{i=0}^{\ell-1}(\ell-i)\left[\binom{\lambda}{i}(e-1)^i - \frac{\lambda^i}{i!}\right] > 0\ .$$

Proof. We fix λ and apply Claim 4.36 with $\xi = 1$. If $\frac{\lambda}{\lambda-\ell} < e-1$, then all summands are nonnegative and we are done.

Otherwise, if there is a $k \leq \lambda$ such that $\phi_\lambda^\xi(k) \leq 0$, then we get for $k < j \leq \lambda$

$$\sum_{i=0}^{j}\phi_\lambda^1(i) \geq \sum_{i=0}^{\lambda}\phi_\lambda^1(i) = \sum_{i=0}^{\lambda}\left[\binom{\lambda}{i}(e-1)^i - \frac{\lambda^i}{i!}\right] = (e-1+1)^\lambda - \sum_{i=0}^{\lambda}\frac{\lambda^i}{i!} = e^\lambda - \sum_{i=0}^{\lambda}\frac{\lambda^i}{i!} > 0\ .$$

In particular, this entails

$$\sum_{i=0}^{\ell-1}(\ell-i)\left[\binom{\lambda}{i}(e-1)^i - \frac{\lambda^i}{i!}\right] = \sum_{i=0}^{\ell-1}(\ell-i)\phi_\lambda^1(i) = \sum_{j=0}^{\ell-1}\sum_{i=0}^{j}\phi_\lambda^1(i) > 0,$$

which concludes the proof. $\square$

4.6.4 Monotonicity

To complete the proof of Theorem 4.2, recall that $\lambda \leq \rho$. Hence, we need to show that the expression in (4.34) is monotone decreasing in λ. We derive a stronger statement, noting that the bound is $g(\lambda, \ell)/\lambda$.

Lemma 4.38. *For any fixed $\ell \in \mathbb{N}$, the expression*

$$g(\lambda,\ell) := e^{-\lambda}\sum_{i=0}^{\ell-1}(\ell-i)\left[\binom{\lambda}{i}(e-1)^i - \frac{\lambda^i}{i!}\right],$$

is monotone decreasing in λ.

We will use the following properties of the Poisson distribution. For $k \in \mathbb{Z}_+$ and $x > 0$, let us denote

$$\theta_k(x) := \Pr(\mathrm{Poi}(x) \leq k) = e^{-x} \sum_{i=0}^{k} \frac{x^i}{i!} \ . \tag{4.36}$$

Claim 4.39. *For any fixed $k \in \mathbb{Z}_+$, $\theta_k(x)$ is monotone decreasing with derivative $\theta_k'(x) = -\frac{e^{-x}x^k}{k!}$. Furthermore, $\theta_k(x)$ is convex on the interval (k, ∞).*

Proof. Using $\frac{d}{dx}\left(e^{-x}\frac{x^i}{i!}\right) = -e^{-x}\frac{x^i}{i!} + e^{-x}\frac{x^{i-1}}{(i-1)!}$, the derivative of $\theta_k(x)$ is

$$\theta_k'(x) = -e^{-x}\sum_{i=0}^{k}\frac{x^i}{i!} + e^{-x}\sum_{i=1}^{k}\frac{x^{i-1}}{(i-1)!} = -\frac{e^{-x}x^k}{k!} \ ,$$

which is negative for all $x > 0$. The second derivative of $\theta_k(x)$ is $\theta_k''(x) = e^{-x}$ if $k = 0$, and

$$\theta_k''(x) = \frac{e^{-x}x^{k-1}}{(k-1)!}\left(\frac{x}{k} - 1\right)$$

if $k \geq 1$. In both cases, $\theta_k''(x) > 0$ when $x > k$. □

Claim 4.40. *For every $\lambda \in \mathbb{N}$, we have $\theta_{\lambda+1}(\lambda + 1) \leq \theta_\lambda(\lambda)$.*

Proof. Let $\Gamma \colon \mathbb{R}_{++} \times \mathbb{R}_+ \to \mathbb{R}_+$ be the upper incomplete gamma function (see [118, §8]), i.e.

$$\Gamma(s, x) = \int_x^\infty t^{s-1}e^{-t}dt \ .$$

We will use the property

$$\frac{1}{(s-1)!}\Gamma(s,x) = \frac{1}{(s-1)!}\int_x^\infty t^{s-1}e^{-t}dt = e^{-x}\sum_{i=0}^{s-1}\frac{x^i}{i!} \ , \tag{4.37}$$

which holds for $s \in \mathbb{N}$ and follows by iterated integration by parts.

To show the nonnegativity of $\theta_\lambda(\lambda) - \theta_{\lambda+1}(\lambda+1)$, recall the definition (4.36)

$$\theta_\lambda(\lambda) - \theta_{\lambda+1}(\lambda+1) = e^{-\lambda}\sum_{i=0}^{\lambda}\frac{\lambda^i}{i!} - e^{-(\lambda+1)}\sum_{i=0}^{\lambda}\frac{(\lambda+1)^i}{i!} - \frac{e^{-(\lambda+1)}(\lambda+1)^\lambda}{\lambda!} \ .$$

Applying (4.37) to the two sums yields

$$\begin{aligned}\theta_\lambda(\lambda) - \theta_{\lambda+1}(\lambda+1) &= \frac{1}{\lambda!}\Gamma(\lambda+1,\lambda) - \frac{1}{\lambda!}\Gamma(\lambda+1,\lambda+1) - \frac{e^{-(\lambda+1)}(\lambda+1)^\lambda}{\lambda!} \\ &= \frac{1}{\lambda!}\left(\int_\lambda^\infty t^\lambda e^{-t}dt - \int_{\lambda+1}^\infty t^\lambda e^{-t}dt - e^{-(\lambda+1)}(\lambda+1)^\lambda\right) \\ &= \frac{1}{\lambda!}\left(\int_\lambda^{\lambda+1} t^\lambda e^{-t}dt - e^{-(\lambda+1)}(\lambda+1)^\lambda\right) \ .\end{aligned}$$

The integrand is monotone decreasing in the interval $(\lambda, \lambda+1]$ because $\frac{d}{dt}\left(t^\lambda e^{-t}\right) = (\lambda t^{\lambda-1} - t^\lambda)e^{-t} < 0$ for all $t > \lambda$. Hence, we can lower bound the integral by the value of the integrand at $t = \lambda + 1$

$$\theta_\lambda(\lambda) - \theta_{\lambda+1}(\lambda+1) \geq \frac{1}{\lambda!}\left((\lambda+1)^\lambda e^{-(\lambda+1)} - e^{-(\lambda+1)}(\lambda+1)^\lambda\right) = 0 \ . \qquad \square$$

With these tools we are ready to prove monotonicity.

Proof of Lemma 4.38. We first prove the cases $\ell \in \{1, 2, 3\}$ separately:

$$g(\lambda, 1) = 0\,, \qquad g(\lambda, 2) = e^{-\lambda}\lambda(e-2)\,, \qquad g(\lambda, 3) = \frac{e^{-\lambda}}{2}\left[e(e-2)\lambda^2 - (e-3)^2\lambda\right] \ .$$

Their derivatives are given by

$$g'(\lambda, 2) = e^{-\lambda}(-\lambda+1)(e-2)\,, \qquad g'(\lambda, 3) = \frac{e^{-\lambda}}{2}\left[-e(e-2)\lambda^2 + ((e-3)^2 + 2e(e-2))\lambda - (e-3)^2\right] \ .$$

It is easy to check that $g'(\lambda, 2) < 0$ for $\lambda \geq 2$, and $g'(\lambda, 3) < 0$ for $\lambda \geq 3$. Henceforth, we will assume that $\ell \geq 4$.

The inequality $g(\lambda+1, \ell) \leq g(\lambda, \ell)$ can be reformulated as

$$\begin{aligned}
&\sum_{i=0}^{\ell-1}(\ell-i)\left[\binom{\lambda+1}{i}(e-1)^i - \frac{(\lambda+1)^i}{i!}\right] \leq e\sum_{i=0}^{\ell-1}(\ell-i)\left[\binom{\lambda}{i}(e-1)^i - \frac{\lambda^i}{i!}\right] \iff \\
&\sum_{i=0}^{\ell-1}(\ell-i)\left[e\frac{\lambda^i}{i!} - \frac{(\lambda+1)^i}{i!}\right] \leq e\sum_{i=0}^{\ell-1}(\ell-i)\binom{\lambda}{i}(e-1)^i - \sum_{i=0}^{\ell-1}(\ell-i)\binom{\lambda+1}{i}(e-1)^i \ .
\end{aligned} \tag{4.38}$$

For the RHS, using $\binom{\lambda+1}{i} = \binom{\lambda}{i} + \binom{\lambda}{i-1}$, we get

$$\begin{aligned}
&\sum_{i=0}^{\ell-1}(\ell-i)\binom{\lambda}{i}(e-1)^{i+1} - \sum_{i=1}^{\ell-1}(\ell-i)\binom{\lambda}{i-1}(e-1)^i \\
=&\sum_{i=0}^{\ell-1}(\ell-i)\binom{\lambda}{i}(e-1)^{i+1} - \sum_{i=0}^{\ell-2}(\ell-i-1)\binom{\lambda}{i}(e-1)^{i+1} \\
=&\sum_{i=0}^{\ell-1}\binom{\lambda}{i}(e-1)^{i+1}
\end{aligned}$$

Using the definition of $\theta_i(\lambda)$ from (4.36), the LHS equals

$$e^{\lambda+1}\sum_{i=0}^{\ell-1}\theta_i(\lambda) - \theta_i(\lambda+1)$$

For every $i = 0, \dots, \ell - 1$, we have $\lambda > i$. Therefore, using the convexity and derivative of $\theta_i(x)$ from Claim 4.39 leads to

$$\theta_i(\lambda) - \theta_i(\lambda+1) \leq \theta_i'(\lambda)(\lambda - (\lambda+1)) = \frac{e^{-\lambda}\lambda^i}{i!} \ .$$

Hence, (4.38) follows by showing

$$0 \leq \sum_{i=0}^{\ell-1} \binom{\lambda}{i}(e-1)^{i+1} - \frac{e\lambda^i}{i!} \ . \tag{4.39}$$

For the sake of brevity, we denote

$$\varphi_\lambda(i) = \binom{\lambda}{i}(e-1)^{i+1} - \frac{e\lambda^i}{i!} \ .$$

Then, our goal is to show that $\sum_{i=0}^{\ell-1} \varphi_\lambda(i) \geq 0$.

Observing that $\varphi_\lambda(i) = e\left(\frac{e-1}{e}\binom{\lambda}{i}(e-1)^i - \frac{\lambda^i}{i!}\right)$, we will apply Claim 4.36 with $\xi = \frac{e-1}{e} > \frac{1}{e-1}$. Consider the following two cases:

Case 1: $\varphi_\lambda(\ell-1) \geq 0$. In this case, $\varphi_\lambda(i) \geq 0$ for all $0 < i < \ell$ by Claim 4.36 (b). Since $\lambda \geq \ell \geq 4$,

$$\begin{aligned}\sum_{i=0}^{\ell-1} \varphi_\lambda(i) &= \varphi_\lambda(0) + \varphi_\lambda(1) + \varphi_\lambda(2) + \sum_{i=3}^{\ell-1} \varphi_\lambda(i) \\ &= -1 + \lambda((e-1)^2 - e) + \frac{\lambda}{2}\left[\lambda((e-1)^3 - e) - (e-1)^3\right] + \sum_{i=3}^{\ell-1} \varphi_\lambda(i) > \sum_{i=3}^{\ell-1} \varphi_\lambda(i) \geq 0 \ .\end{aligned}$$

Case 2: $\varphi_\lambda(\ell-1) < 0$. In this case, $\varphi_\lambda(i) < 0$ for all $\ell \leq i \leq \lambda$ by Claim 4.36 (b). Thus,

$$\begin{aligned}\sum_{i=0}^{\ell-1} \varphi_\lambda(i) > \sum_{i=0}^{\lambda} \varphi_\lambda(i) &= (e-1)\sum_{i=0}^{\lambda}\binom{\lambda}{i}(e-1)^i - \sum_{i=0}^{\lambda}\frac{e\lambda^i}{i!} = (e-1)e^\lambda - e\sum_{i=0}^{\lambda}\frac{\lambda^i}{i!} \\ &= e^{\lambda+1}\left(1 - \frac{1}{e} - \theta_\lambda(\lambda)\right) \overset{\text{Clm. 4.40}}{\geq} e^{\lambda+1}\left(1 - \frac{1}{e} - \theta_4(4)\right) > 0.\end{aligned}$$

The second inequality holds due to Claim 4.40 together with the assumption $\lambda \geq \ell \geq 4$, whereas the last inequality follows from $1 - 1/e - \Pr(\mathrm{Poi}(4) \leq 4) > 0$. □

Chapter 5

Parity Games: Strategy Iteration with Universal Trees

5.1 Introduction

A *parity game* is an infinite duration game between two players Even and Odd. It takes place on a sinkless directed graph $G = (V, E)$ equipped with a *priority* function $\pi : V \to \{1, 2, \dots, d\}$. Let $n = |V|$ and $m = |E|$. The node set V is partitioned into $V_0 \sqcup V_1$ such that nodes in V_0 and V_1 are owned by Even and Odd respectively. The game starts when a token is placed on a node. In each turn, the owner of the current node moves the token along an outgoing arc to the next node, resulting in an infinite walk. If the highest priority occurring infinitely often in this walk is even, then Even wins. Otherwise, Odd wins.

By the positional determinacy of parity games [59], there exists a partition of V into two subsets from which Even and Odd can force a win respectively. The main algorithmic problem of parity games is to determine this partition, or equivalently, to decide the winner given a starting node. This is a notorious problem that lies in NP ∩ co-NP [60], and also in UP ∩ co-UP [86], with no known polynomial algorithm to date.

Due to its intriguing complexity status, as well as its fundamental role in automata theory and logic [60, 97], parity games have been intensely studied over the past three decades. Prior to 2017, algorithms for solving parity games, e.g. [162, 87, 152, 13, 130, 89, 132, 107, 10], are either exponential or subexponential. In a breakthrough result, Calude et al. [27] gave the first quasi-polynomial algorithm. Since then, many other quasi-polynomial algorithms [61, 88, 100, 122, 11] have been developed. Most of them have been unified by Czerwiński et al. [35] via the concept of a *universal tree*. A universal tree is an ordered tree into which every ordered tree of a certain size can be isomorphically embedded. They proved a quasi-polynomial lower bound on the size of a universal tree.

Value iteration The starting point of this chapter is the classic *progress measure* algorithm [87, 88] for solving parity games. It belongs to a broad class of algorithms called *value iteration* – a well-known method for solving more general games on graphs such as mean payoff games and stochastic games. In value iteration, every node v in G is assigned a value $\mu(v) \in \mathcal{V}$ from some totally ordered set $\mathcal{V}$, and the values are locally improved until we reach the *least fixed point* of a set of operators associated with the game. The set $\mathcal{V}$ is called the *value domain*, which is usually a bounded set of real numbers or integers. For the progress measure algorithm, its value domain is the set of leaves $L(T)$ in a universal tree T. As the values are monotonically improved, the running time is proportional to $|L(T)|$. The first progress measure algorithm of Jurdziński [87] uses a perfect n-ary tree, which runs in exponential time. Its subsequent improvement by Jurdziński and Lazić [88] uses a quasi-polynomial-sized tree, which runs in $n^{\log(d/\log n)+O(1)}$ time.

Despite having good theoretical efficiency, the progress measure algorithm is not robust against its worst-case behaviour. In fact, it is known to realize its worst-case running time on very simple instances. As an example, let (G, π) be an arbitrary instance with maximum priority d, with d being even. For a small odd constant k, if we add two nodes of priority k as shown in Figure 5.1, then the progress measure algorithm realizes its worst-case running time. This is because the values of those nodes are updated superpolynomially many times.

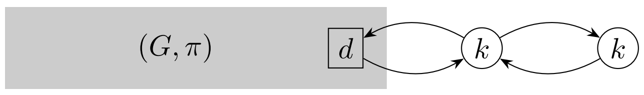

Fig. 5.1 A worst-case construction for the progress measure algorithm. Nodes in V_0 and V_1 are drawn as squares and circles, respectively.

Strategy iteration A different but related method for solving games on graphs is *strategy iteration*. For a parity game (G, π), a *(positional) strategy* τ for a player (say Odd) is a choice of an outgoing arc from every node in V_1. Removing the unchosen outgoing arcs from every node in V_1 results in a *strategy subgraph* $G_\tau \subseteq G$. A general framework for strategy iteration is given, e.g., in [68]. Following that exposition, to rank the strategies for Odd, one fixes a suitable value domain $\mathcal{V}$ and associates a valuation $\mu : V \to \mathcal{V}$ to each strategy. This induces a partial order over the set of strategies for Odd. Note that most valuations used in the literature can be thought of as fixed points of a set of operators associated with the 1-player game (G_τ, π) for Even. In every iteration, the algorithm maintains a strategy τ for Odd and its corresponding valuation $\mu : V \to \mathcal{V}$. Based on a *pivot rule*, it modifies τ to a better strategy τ', and updates μ to the valuation μ' of τ'. Note that $\mu' \geq \mu$. This process is repeated until we reach the optimal strategy for Odd.

Originally introduced by Hoffman and Karp for stochastic games [81], variants of strategy iteration for parity games have been developed [124, 152, 13, 130]. They usually perform well in practice, but tedious constructions of their worst case (sub)exponential complexity are known [67]. Motivated by the construction of small universal trees [88, 43], a natural question is whether there exists a strategy iteration algorithm with value domain $L(T)$ for a universal tree T. It is not hard to see that with value domain $L(T)$, unfortunately, the fixed point of a 1-player game (G_τ, π) may not be unique. Moreover, in a recent thesis [117], Ohlmann showed that a valuation that is fit for strategy iteration cannot be defined using $L(T)$.

Our contribution We show that an adaptation of strategy iteration with value domain $L(T)$ is still possible. To circumvent the impossibility result of Ohlmann [117], we slightly alter the strategy iteration framework as follows. After pivoting to a strategy τ' in an iteration, we update the current node labeling μ to the least fixed point of $(G_{\tau'}, \pi)$ that is *pointwise at least* μ. In other words, we force μ to increase (whereas this happens automatically in the previous framework). Since the fixed point of a 1-player game may not be unique, this means that we may encounter a strategy more than once during the course of the algorithm. The motivation of our approach comes from tropical geometry, as discussed in Section 1.4.

To carry out each iteration efficiently, we give a combinatorial method for computing the least fixed point of 1-player games with value domain $L(T)$. It relies on adapting the classic techniques of label-correcting and label-setting from the shortest path problem to the setting of ordered trees. When T is instantiated as a specific universal tree constructed in the literature, we obtain the following running times:

- The universal tree of Jurdziński and Lazić [88] takes $O(mn^2 \log n \log d)$.
- The Strahler universal tree of Daviaud et al. [43] takes $O(mn^2 \log^3 n \log d)$.
- The perfect n-ary tree of height $d/2$ takes $O(d(m + n \log n))$.

The total number of strategy iterations is trivially bounded by $n|L(T)|$, the same bound for the progress measure algorithm. Whereas we do not obtain a strict improvement over previous running time bounds, it is conceivable that our algorithm would terminate in fewer iterations than the progress measure algorithm on most examples. Moreover, our framework provides large flexibility in the choice of pivot rules. Identifying a pivot rule that may provide strictly improved (and possibly even polynomial) running time is left for future research.

5.1.1 Computing the Least Fixed Point of 1-Player Games

Let (G_τ, π) be a 1-player game for Even, and μ^* be its least fixed point with value domain $L(T)$ for some universal tree T. Starting from $\mu(v) = \min L(T)$ for all $v \in V$, the progress measure algorithm successively lifts the label of a node based on the labels of its out-neighbours until μ^* is reached. However, this is not polynomial in general, even on 1-player games. So, instead of approaching μ^* from below, we approach it from above. This is reminiscent of shortest path algorithms, where node labels form upper bounds on the shortest path distances throughout the algorithm. In a label-correcting method like the Bellman–Ford algorithm, to compute shortest paths to a target node t, the label at t is initialized to 0, while the label at all other nodes is initialized to $+\infty$. By iteratively checking if an arc violates feasibility, the node labels are monotonically decreased. We refer to Ahuja et al. [4] for an overview on label-correcting and label-setting techniques for computing shortest paths.

In our setting, the role of the target node t is replaced by a (potentially empty) set of *even* cycles in G_τ. A cycle is said to be *even* if its maximum priority is even. However, this set is not known to us a priori. To overcome this issue, we define *base nodes* as candidate target nodes. A node $w \in V$ is a *base node* if it *dominates* an even cycle in G_τ, that is, it is a node with the highest priority in the cycle. Note that $\pi(w)$ is even.

To run a label-correcting method, we need to assign initial labels ν to the nodes in G_τ. The presumably obvious choice is to set $\nu(w) \leftarrow \min L(T)$ if w is a base node, and $\nu(w) \leftarrow \top$ otherwise, where $\top$ is bigger than every element in $L(T)$ ($\top$ is analogous to $+\infty$ for real numbers). However, this only works when T is a perfect n-ary tree. For a more complicated universal tree, the number of children at each internal vertex of T is not the same. Hence, it is possible to have $\nu(w) < \mu^*(w)$ for a base node w. We also cannot make $\nu(w)$ too large, as otherwise we may converge to a fixed point that is not pointwise minimal.

To correctly initialize $\nu(w)$ for a base node w, let us consider the cycles dominated by w in G_τ. Every such cycle C induces a subgame (C, π) on which Even wins because C is even. The least fixed point of (C, π) consists of leaves of an ordered tree T_C of height $j := \pi(w)/2$. Initializing $\nu(w)$ essentially boils down to finding such a cycle C with the 'narrowest' T_C. To this end, let $\mathcal{T}_j$ be the set of *distinct* subtrees of height j of our universal tree T. We will exploit the fact that $\mathcal{T}_j$ is a poset with respect to the partial order of embeddability. In particular, let $\mathcal{C}_j$ be a set of chains covering $\mathcal{T}_j$, and fix a chain $\mathcal{C}_j^k$ in $\mathcal{C}_j$. We define the *width* of a cycle C as the 'width' of the smallest tree in $\mathcal{C}_j^k$ into which T_C is embeddable. Then, we show that our problem reduces to finding a minimum width cycle dominated by w in G_τ.

To solve the latter problem, we construct an arc-weighted *auxiliary digraph* D on the set of base nodes. Every arc uv in D represents a path from base node u to base node v in G_τ, in such a way that minimum bottleneck cycles in D correspond to minimum width cycles in G_τ. It follows that the desired cycle C can be obtained by computing a minimum bottleneck cycle in D containing w. After getting C, we locate the corresponding subtree T' of T into which T_C is embeddable. Then, the label at w is initialized as $\nu(w) \leftarrow \min L(T')$.

With these initial labels, we show that a generic label-correcting procedure returns the desired least fixed point μ^* in $O(mn)$ time. The overall running time of this label-correcting method is dominated by the initialization phase, whose running time is proportional to the size of the chain cover $\mathcal{C}_j$. We prove that the quasi-polynomial universal trees constructed in the literature [88, 43] admit small chain covers. Using this result, we then give efficient implementations of our method for these trees.

In Section 5.5, we also develop a label-setting method for computing μ^*, which is faster but only applicable when T is a perfect n-ary tree. Unlike the label-correcting approach, in a label-setting method such as Dijkstra's algorithm, the label of a node is fixed in each iteration. In the shortest path problem, Dijkstra's algorithm selects a node with the smallest label to be fixed in every iteration. When working with labels given by the leaves of a universal tree, this criterion does not work anymore. Let H be the subgraph of G_τ obtained by deleting all the base nodes. For $p \in \mathbb{N}$, let H_p be the subgraph of H induced by the nodes with priority at most p. We construct a suitable potential function by interlacing each node label with a tuple that encodes the topological orders in $H_2, H_4, \dots$. In every iteration, a node with the smallest potential is selected, and its label is fixed.

Chapter organization In Section 5.2, we introduce notation and provide the necessary preliminaries on parity games and universal trees. Section 5.3 contains our strategy iteration framework based on universal trees. In Section 5.4, we give a label-correcting method for computing the least fixed point of 1-player games. The label-setting method is given in Section 5.5.

5.2 Preliminaries

A parity game instance is given by (G, π), where $G = (V, E)$ is a sinkless directed graph with $V = V_0 \sqcup V_1$, and $\pi : V \to [d]$ is a priority function. Without loss of generality, we may assume that d is even. In this chapter, we are only concerned with positional strategies. A *strategy* for Odd is a function $\tau : V_1 \to V$ such that $v\tau(v) \in E$ for all $v \in V_1$. Its *strategy subgraph* is $G_\tau = (V, E_\tau)$, where $E_\tau := \{vw \in E : v \in V_0\} \cup \{v\tau(v) : v \in V_1\}$. A strategy for Even and its strategy subgraph are defined analogously. We always denote a strategy for Even as σ, and a strategy for Odd as τ. If we fix a strategy τ for Odd, the resulting instance (G_τ, π) is a *1-player game* for Even.

For the sake of brevity, we overload the priority function π as follows. Given a subgraph $H \subseteq G$, let $\pi(H)$ be the highest priority in H. The subgraph H is said to be *even* if $\pi(H)$ is even, and *odd* otherwise. For a fixed π, we denote by $\Pi(H)$ the set of nodes with the highest priority in H. If $v \in \Pi(H)$, we say that v *dominates* H. For $p \in [d]$, H_p refers to the subgraph of H induced by nodes with priority at most p. For a node v, let $\delta^-_H(v)$ and $\delta^+_H(v)$ be the incoming and outgoing arcs of v in H respectively. Similarly, let $N^-_H(v)$ and $N^+_H(v)$ be the in-neighbors and out-neighbors of v in H respectively. When H is clear from context, we will omit it from the subscripts.

The win of a player can be certified by *node labels* from a *universal tree*, as stated in Theorem 5.3. We give the necessary background for this now.

5.2.1 Ordered Trees and Universal Trees

An *ordered tree* T is a prefix-closed set of tuples, whose elements are drawn from a linearly ordered set M. The linear order of M lexicographically extends to T. Equivalently, T can be thought of as a rooted tree, whose root we denote by r. Under this interpretation, elements in M correspond to the branching directions at each vertex of T (see Figures 5.2 and 5.3 for examples). Every tuple then corresponds to a vertex $v \in V(T)$. This is because the tuple can be read by traversing the unique r-v path in T. Observe that v is an h-tuple if and only if v is at depth h in T. In particular, r is the empty tuple.

In this chapter, we always use the terms 'vertex' and 'edge' when referring to an ordered tree T. The terms 'node' and 'arc' are reserved for the game graph G.

Given an ordered tree T of height h, let $L(T)$ be the set of leaves in T. For convenience, we assume that every leaf in T is at depth h throughout. The tuple representing a leaf $\xi \in L(T)$ is denoted as $\xi = (\xi_{2h-1}, \xi_{2h-3}, \dots, \xi_1)$, where $\xi_i \in M$ for all i. We refer to ξ_{2h-1} as the *first* component of ξ, even though it has index $2h-1$. For a fixed $p \in [2h]$, the *p-truncation* of ξ is

$$\xi|_p := \begin{cases} (\xi_{2h-1}, \xi_{2h-3}, \dots, \xi_{p+1}), & \text{if } p \text{ is even} \\ (\xi_{2h-1}, \xi_{2h-3}, \dots, \xi_p), & \text{if } p \text{ is odd.} \end{cases}$$

In other words, the p-truncation of a tuple is obtained by deleting the components with index less than p. Note that a truncated tuple is an ancestor of the untruncated tuple in T.

Definition 5.1. Given ordered trees T and T', we say that T *embeds into* T' (denoted $T \sqsubseteq T'$) if there exists an *injective* and *order-preserving* homomorphism from T to T' such that leaves in T are mapped to leaves in T'. Formally, this is an injective function $f : V(T) \to V(T')$ which satisfies the following properties:

1. For all $u, v \in V(T)$, $uv \in E(T)$ implies $f(u)f(v) \in E(T')$;
2. For all $u, v \in V(T)$, $u \leq v$ implies $f(u) \leq f(v)$.
3. $f(u) \in L(T')$ for all $u \in L(T)$.

We write $T \equiv T'$ if $T \sqsubseteq T'$ and $T' \sqsubseteq T$. Also, $T \sqsubset T'$ if $T \sqsubseteq T'$ and $T \not\equiv T'$.

In the definition above, since f is order-preserving, the children of every vertex in T are mapped to the children of its image injectively such that their order is preserved. As an example, the tree in Figure 5.3 embeds into the tree in Figure 5.2. It is easy to verify that $\sqsubseteq$ is a partial order on the set of all ordered trees.

Definition 5.2. An (ℓ, h)-*universal tree* is an ordered tree T' of height h such that $T \sqsubseteq T'$ for every ordered tree T of height h and with at most ℓ leaves, all at depth exactly h.

The simplest example of an (ℓ, h)-universal tree is the perfect ℓ-ary tree of height h, which we call a *perfect universal tree*. The linearly ordered set M for this tree can be chosen as $\{0, 1, \dots, \ell - 1\}$ (see Figure 5.2 for an example). It has ℓ^h leaves, which grows exponentially with h. Jurdziński and Lazić [88] constructed an (ℓ, h)-universal tree with at most $\ell^{\log h + O(1)}$ leaves, which we call a *succinct universal tree*. In this tree, every leaf ξ corresponds to an h-tuple of binary strings with at most $\lfloor \log(\ell) \rfloor$ bits in total[1]. We use $|\xi|$ and $|\xi_i|$ to denote the total number of bits in ξ and ξ_i respectively. The linearly ordered set M for this tree consists of finite binary strings, where $\varepsilon \in M$ is the empty string (see Figure 5.3 for an example). For any pair of binary strings $s, s' \in M$ and a bit b, the linear order on M is defined as $0s < \varepsilon < 1s'$ and $bs < bs' \iff s < s'$.

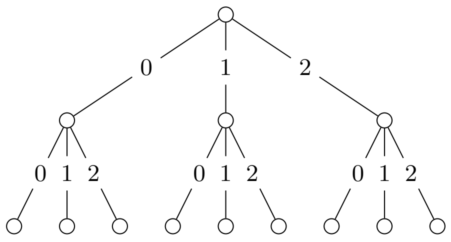

Fig. 5.2 The perfect (3,2)-universal tree.

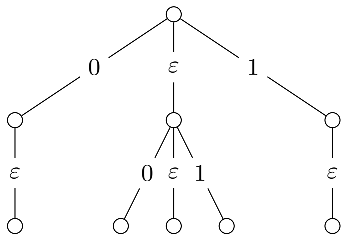

Fig. 5.3 The succinct (3,2)-universal tree.

5.2.2 Node Labelings from Universal Trees

Let (G, π) be a parity game instance and T be an ordered tree of height $d/2$. We augment the set of leaves with an extra *top* element $\top$, denoted $\bar{L}(T) := L(T) \cup \{\top\}$, such that $\top > v$ for all $v \in V(T)$. We also set $\top|_p := \top$ for all $p \in [d]$. A function $\mu : V \to \bar{L}(T)$ which maps the nodes in G to $\bar{L}(T)$ is called a *node labeling*. For a subgraph H of G, we say that μ is *feasible in* H if there exists a strategy $\sigma : V_0 \to V$ for Even with $v\sigma(v) \in E(H)$ whenever $\delta^+_H(v) \neq \emptyset$, such that the following condition holds for every arc vw in $H \cap G_\sigma$:

- If $\pi(v)$ is even, then $\mu(v)|_{\pi(v)} \geq \mu(w)|_{\pi(v)}$.
- If $\pi(v)$ is odd, then $\mu(v)|_{\pi(v)} > \mu(w)|_{\pi(v)}$ or $\mu(v) = \mu(w) = \top$.

An arc vw which does not satisfy the condition above is called *violated* (with respect to μ). On the other hand, if $\mu(v)$ is the smallest element in $\bar{L}(T)$ such that vw is non-violated, then vw is said to be *tight*. Any arc which is neither tight nor violated is called *loose*. We say that a subgraph is tight if it consists of tight arcs.

In the literature, a node labeling which is feasible in G is also called a *progress measure*. The node labeling given by $\mu(v) = \top$ for all $v \in V$ is trivially feasible in G. However, we are primarily interested in progress measures with minimal top support, i.e. such that the set of nodes having label $\top$ is inclusion-wise minimal.

[1]A slightly looser bound of $\lceil \log \ell \rceil$ was derived in [88, Lemma 1]. It can be strengthened to $\lfloor \log \ell \rfloor$ with virtually no change in the proof.

Theorem 5.3 ([87, Corollaries 7–8]). *Given an $(n, d/2)$-universal tree T, let $\mu^* : V \to \bar{L}(T)$ be a node labeling which is feasible in G and has minimal top support. Then, Even wins from $v \in V$ if and only if $\mu^*(v) \neq \top$.*

The above theorem formalizes the following intuition: nodes with smaller labels are more advantageous for Even to play on. Note that if μ is a minimal node labeling which is feasible in G, i.e. μ' is infeasible in G for all $\mu' < \mu$, then there exists a strategy σ for Even such that $v\sigma(v)$ is tight for all $v \in V_0$. The next observation is well-known (see, e.g., [88, Lemma 2]) and follows directly from the definition of feasibility.

Lemma 5.4 (Cycle Lemma). *If a node labeling μ is feasible in a cycle C, then $\mu(v)|_{\pi(C)} = \mu(w)|_{\pi(C)}$ for all $v, w \in V(C)$. Furthermore, if $\mu(v) \neq \top$ for some $v \in V(C)$, then C is even.*

We assume to have access to the following algorithmic primitive, whose running time we denote by $\gamma(T)$. Its implementation depends on the ordered tree T. For instance, $\gamma(T) = O(d)$ if T is a perfect $(n, d/2)$-universal tree. If T is a succinct $(n, d/2)$-universal tree, Jurdziński and Lazić [88, Theorem 7] showed that $\gamma(T) = O(\log n \log d)$.

Subroutine 5.2.1. TIGHTEN(μ, vw)

Given a node labeling $\mu : V \to \bar{L}(T)$ and an arc $vw \in E$, return the unique element $\xi \in \bar{L}(T)$ such that vw is tight after setting $\mu(v)$ to ξ.

Given a node labeling $\mu : V \to \bar{L}(T)$ and an arc $vw \in E$, let $\mathrm{lift}(\mu, vw)$ be the smallest element $\xi \in \bar{L}(T)$ such that $\xi \geq \mu(v)$ and vw is not violated after setting $\mu(v)$ to ξ. Observe that if vw is violated, $\mathrm{lift}(\mu, vw)$ is given by TIGHTEN(μ, vw). Otherwise, it is equal to $\mu(v)$. Hence, it can be computed in $\gamma(T)$ time.

5.2.3 Fixed Points in Lattices

We recall a fundamental result on the existence of fixed points. Let $\mathcal{L}$ be a non-empty finite lattice, and let $\mu, \nu \in \mathcal{L}$. An operator $\phi \colon \mathcal{L} \to \mathcal{L}$ is *monotone* if $\mu \leq \nu \Rightarrow \phi(\mu) \leq \phi(\nu)$, and it is *inflationary* if $\mu \leq \phi(\mu)$. Given a family $\mathcal{G}$ of inflationary monotone operators on $\mathcal{L}$, we denote $\mu^{\mathcal{G}}$ as the *least* (simultaneous) fixed point of $\mathcal{G}$ which is pointwise *at least* μ.

Proposition 5.5 (Knaster–Tarski). *Let $\mathcal{G}$ be a family of inflationary monotone operators on $\mathcal{L}$. For any $\mu \in \mathcal{L}$ and $\mathcal{H} \subseteq \mathcal{G}$,*

(i) The least fixed point $\mu^{\mathcal{H}}$ exists.

(ii) The least fixed point is non-decreasing with the set of operators: $\mu^{\mathcal{H}} \leq \mu^{\mathcal{G}}$.

(iii) The least fixed point is monotone: if $\mu \leq \nu$ then $\mu^{\mathcal{H}} \leq \nu^{\mathcal{H}}$.

An operator $\psi \colon \mathcal{L} \to \mathcal{L}$ is *deflationary* if $\mu \geq \psi(\mu)$. Considering the lattice through an order-reversing poset isomorphism implies that the analogous statements of Proposition 5.5 hold for deflationary monotone operators. Given a family $\mathcal{G}$ of deflationary monotone operators on $\mathcal{L}$, we denote $\mu^{\mathcal{G}}$ as *greatest* (simultaneous) fixed point of $\mathcal{G}$ which is pointwise *at most* μ.

In this chapter, $\mathcal{L}$ will be the finite lattice of node labelings mapping V to $\bar{L}(T)$ for a universal tree T. For a sinkless subgraph $H \subseteq G$, consider the following operators. For every node $v \in V_0$, define $\text{Lift}_v : \mathcal{L} \times V \to \bar{L}(T)$ as

$$\text{Lift}_v(\mu, u) := \begin{cases} \min_{vw \in E(H)} \text{lift}(\mu, vw), & \text{if } u = v \\ \mu(u), & \text{otherwise.} \end{cases}$$

For every arc $vw \in E(H)$ where $v \in V_1$, define $\text{Lift}_{vw} : \mathcal{L} \times V \to \bar{L}(T)$ as

$$\text{Lift}_{vw}(\mu, u) := \begin{cases} \text{lift}(\mu, vw), & \text{if } u = v \\ \mu(u), & \text{otherwise.} \end{cases}$$

We denote $\mathcal{H}^{\uparrow} := \{\text{Lift}_v : v \in V_0\} \cup \{\text{Lift}_{vw} : v \in V_1\}$ as the operators in H. Since they are inflationary and monotone, for any $\mu \in \mathcal{L}$, the least fixed point $\mu^{\mathcal{H}^{\uparrow}}$ exists by Proposition 5.5 (i). Note that a node labeling is a fixed point of $\mathcal{H}^{\uparrow}$ if and only if it is feasible in H. The *progress measure algorithm* [87, 88] is an iterative application of the operators in $\mathcal{G}^{\uparrow}$ to μ to obtain $\mu^{\mathcal{G}^{\uparrow}}$.

5.3 Strategy Iteration with Tree Labels

In this section, we present a strategy iteration algorithm (Algorithm 9) whose pivots are guided by a universal tree. It takes as input an instance (G, π), a universal tree T, and an initial strategy τ_1 for Odd. Throughout, it maintains a node labeling $\mu : V \to \bar{L}(T)$, initialized as the least simultaneous fixed point of $\mathcal{G}^{\uparrow}_{\tau_1}$. At the start of every iteration, the algorithm maintains a strategy τ for Odd, and a node labeling $\mu : V \to \bar{L}(T)$ which is feasible in G_τ. Furthermore, there are no loose arcs in G_τ with respect to μ. So, every arc in G_τ is either tight (usable by Even in her counterstrategy σ) or violated (not used by Even). Note that our initial node labeling satisfies these conditions with respect to τ_1.

For $v \in V_1$, we call a violated arc $vw \in E$ with respect to μ *admissible* (as it admits Odd to perform an improvement). If there are no admissible arcs in G, then the algorithm terminates. In this case, μ is feasible in G. Otherwise, Odd pivots to a new strategy τ' by switching to admissible arc(s). The choice of which admissible arc(s) to pick is governed by a *pivot rule*. Then, μ is updated to $\mu^{\mathcal{G}^{\uparrow}_{\tau'}}$. Due to the minimality of $\mu^{\mathcal{G}^{\uparrow}_{\tau'}}$, there are no loose arcs in $G_{\tau'}$ with respect to $\mu^{\mathcal{G}^{\uparrow}_{\tau'}}$, so this invariant continues to hold in the next iteration.

Algorithm 9: Strategy iteration with tree labels

```
  Input  : Parity game instance $(G, \pi)$, universal tree $T$, and initial strategy $\tau_1$ for
           Odd
  Output : A strategy for Odd and node labeling from $T$
1 $\mu(v) \leftarrow \min L(T)\ \forall v \in V$
2 $\tau \leftarrow \tau_1$, $\mu \leftarrow \mu^{\mathcal{G}_\tau^\uparrow}$
3 while $\exists$ an admissible arc in $G$ with respect to $\mu$ do
4     Pivot to a strategy $\tau'$ by selecting admissible arc(s)   ▷ requires a pivot rule
5     $\tau \leftarrow \tau'$, $\mu \leftarrow \mu^{\mathcal{G}_\tau^\uparrow}$
6 return $\tau$, $\mu$
```

We remark that a strategy τ may occur more than once during the course of the algorithm, as mentioned in the description of strategy iteration in Section 5.1. This is because the fixed points of $\mathcal{G}_\tau^\uparrow$ are not necessarily unique. See Figure 5.4 for an example run with a perfect universal tree and a succinct universal tree.

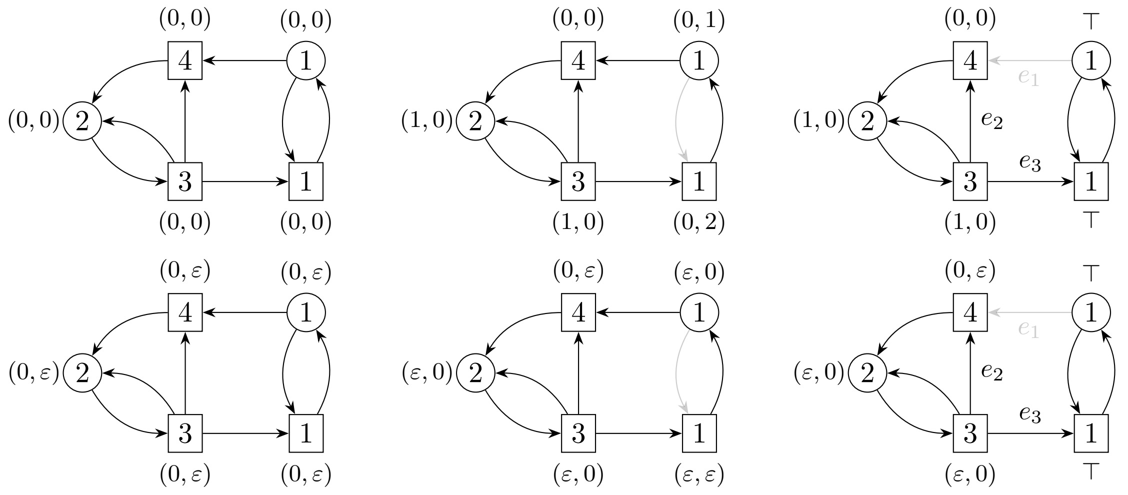

Fig. 5.4 The top and bottom rows illustrate an example run of Algorithm 9 with the perfect (3,2)-universal tree and the succinct (3,2)-universal tree respectively. In each row, the left figure depicts the game instance (nodes in V_0 and V_1 are drawn as squares and circles respectively). The next two figures show Odd's strategy and the node labeling at the start of Iteration 1 and 2. The arcs not selected by Odd are greyed out. In the right figure, e_1 is loose, e_2 is tight, and e_3 is violated.

The correctness of Algorithm 9 is an easy consequence of Proposition 5.5.

Theorem 5.6. *Algorithm 9 returns the pointwise minimal node labeling $\mu^* : V \to \bar{L}(T)$ which is feasible in G.*

Proof. The node labeling μ is monotone increasing in every iteration. Since $\mathcal{L}$ is finite and the all-top node labeling is feasible in G, the algorithm terminates. Let μ^* be the pointwise minimal node labeling which is feasible in G. Note that μ^* is the least simultaneous fixed point of $\mathcal{G}^\uparrow$ in $\mathcal{L}$. By induction and Proposition 5.5 (ii), we have $\mu^{\mathcal{G}_\tau^\uparrow} \leq \mu^*$ in every iteration. As the algorithm terminates with a simultaneous fixed point of $\mathcal{G}^\uparrow$, it terminates with μ^*. $\square$

Thus, by Theorem 5.3, the algorithm correctly determines the winning positions for Even.

Runtime In Algorithm 9, a naive method for computing the least fixed point $\mu^{\mathcal{G}_\tau^\uparrow}$ is to iterate the operators in $\mathcal{G}_\tau^\uparrow$ on μ until convergence. The operators Lift_v and Lift_{vw} can be implemented to run in $O(|\delta^+(v)|\gamma(T))$ time. Since μ is monotone increasing throughout, the total running time of Algorithm 9 is

$$O\left(\sum_{v\in V} |\delta^+(v)|\gamma(T)|L(T)|\right) = O(m\gamma(T)|L(T)|),$$

which matches progress measure algorithms [87, 88]. However, this method of computing $\mu^{\mathcal{G}_\tau^\uparrow}$ can take $\Omega(\gamma(T)|L(T)|)$ time; recall that $|L(T)|$ is at least quasi-polynomial for a universal tree T [35]. Our goal is to compute $\mu^{\mathcal{G}_\tau^\uparrow}$ in polynomial time. Nevertheless, running this method in parallel with a more efficient algorithm for computing $\mu^{\mathcal{G}_\tau^\uparrow}$ is still useful in ensuring that we are not slower than the progress measure algorithm overall.

5.3.1 The Least Fixed Point of 1-Player Games

Let (G_τ, π) be a 1-player game for Even, and let $\mu \in \mathcal{L}$ be a node labeling such that there are no loose arcs in G_τ. In the rest of the chapter, we focus on developing efficient methods for computing $\mu^{\mathcal{G}_\tau^\uparrow}$. We know that applying the operators in $\mathcal{G}_\tau^\uparrow$ to μ is not polynomial in general. So, we will approach $\mu^{\mathcal{G}_\tau^\uparrow}$ from above instead.

Given a node labeling $\nu : V \to \bar{L}(T)$ and an arc $vw \in E$, let $\mathrm{drop}(\nu, vw)$ be the largest element $\xi \in \bar{L}(T)$ such that $\xi \leq \nu(v)$ and vw is not loose after setting $\nu(v)$ to ξ. Observe that if vw is loose, then $\mathrm{drop}(\nu, vw)$ is given by $\textsc{Tighten}(\nu, vw)$. Otherwise, it is equal to $\nu(v)$. Hence, it can be computed in $\gamma(T)$ time.

We are ready to define the deflationary counterpart of Lift_{vw}. For every arc $vw \in E_\tau$, define the operator $\mathrm{Drop}_{vw} : \mathcal{L} \times V \to \bar{L}(T)$ as

$$\mathrm{Drop}_{vw}(\nu, u) := \begin{cases} \mathrm{drop}(\nu, vw), & \text{if } u = v \\ \nu(v), & \text{otherwise.} \end{cases}$$

For a subgraph $H \subseteq G_\tau$, we denote $\mathcal{H}^\downarrow := \{\mathrm{Drop}_e : e \in E(H)\}$ as the operators in H. Since they are deflationary and monotone, for any $\nu \in \mathcal{L}$, the *greatest* simultaneous fixed point $\nu^{\mathcal{H}^\downarrow}$ exists by Proposition 5.5 (i). Note that a node labeling is a simultaneous fixed point of $\mathcal{H}^\downarrow$ if and only if there are no loose arcs in H with respect to it.

Our techniques are inspired by the methods of *label-correcting* and *label-setting* for the shortest path problem. In the shortest path problem, we have a designated target node t whose label is initialized to 0. For us, the role of t is replaced by a (potentially empty) set of even cycles in G_τ, which we do not know a priori. So, we define a set of candidates nodes called *base nodes*, whose labels need to be initialized properly.

Definition 5.7. Given a 1-player game (G_τ, π) for Even, we call $v \in V$ a *base node* if $v \in \Pi(C)$ for some even cycle C in G_τ. Denote $B(G_\tau)$ as the set of base nodes in G_τ.

The base nodes can be found by recursively decomposing G_τ into strongly connected components (SCCs). Initially, for each SCC K of G_τ, we delete $\Pi(K)$. If $\pi(K)$ is even and $|V(K)| > 1$, then $\Pi(K)$ are base nodes and we collect them. Otherwise, we ignore them. Then, we are left with a smaller subgraph of G, so we repeat the process. Using Tarjan's SCCs algorithm [146], this procedure takes $O(dm)$ time.

In Section 5.4, we develop a label-correcting method for computing $\mu^{\mathcal{G}_\tau^\uparrow}$, and apply it to the quasi-polynomial universal trees constructed in the literature [88, 43]. The label-setting method, which is faster but only applicable to perfect universal trees, is given in Section 5.5.

5.4 Label-Correcting Method for Computing the Least Fixed Point

The Bellman–Ford algorithm for the shortest path problem is a well-known implementation of the generic label-correcting method [4]. We start by giving its analogue for ordered trees. Algorithm 10 takes as input a 1-player game (G_τ, π) for Even and a node labeling $\nu : V \to \bar{L}(T)$ from some ordered tree T. Like its classical version for shortest paths, the algorithm runs for $n - 1$ iterations. In each iteration, it replaces the tail label of every arc $e \in E_\tau$ by $\text{drop}(\nu, e)$. Clearly, the running time is $O(mn\gamma(T))$. Moreover, if ν' is the returned node labeling, then $\nu' \geq \nu^{\mathcal{G}_\tau^\downarrow}$.

Algorithm 10: BELLMAN–FORD

```
Input  : 1-player game $(G_\tau, \pi)$ for Even, node labeling $\nu : V \to \bar{L}(T)$ from an
         ordered tree $T$
Output : Node labeling from $T$
1 for $i = 1$ to $n - 1$ do
2     foreach $vw \in E$ do                                   ▷ In any order
3         $\nu(v) \leftarrow \text{drop}(\nu, vw)$
4 return $\nu$
```

Recall that we have a node labeling $\mu \in \mathcal{L}$ such that G_τ does not have loose arcs, and our goal is to compute $\mu^{\mathcal{G}_\tau^\uparrow}$. We first state a sufficient condition on the input node labeling ν

such that Algorithm 10 returns $\mu^{\mathcal{G}_\tau^\uparrow}$. In the shortest path problem, we set $\nu(t) = 0$ at the target node t, and $\nu(v) = \infty$ for all $v \in V \setminus \{t\}$. When working with node labels given by an ordered tree, one has to ensure that the algorithm does not terminate with a fixed point larger than $\mu^{\mathcal{G}_\tau^\uparrow}$, motivating the following definition.

Definition 5.8. Given a node labeling $\mu \in \mathcal{L}$, the *threshold label* of a base node $v \in B(G_\tau)$ is

$$\widehat{\mu}(v) := \min_{\tilde{\mu} \in \mathcal{L}} \{\tilde{\mu}(v) : \tilde{\mu}(v) \geq \mu(v) \text{ and } \tilde{\mu} \text{ is feasible in a cycle dominated by } v \text{ in } G_\tau\} \enspace .$$

The next lemma follows directly from the pointwise minimality of $\mu^{\mathcal{G}_\tau^\uparrow}$.

Lemma 5.9. *Let $\mu \in \mathcal{L}$ be a node labeling such that G_τ does not have loose arcs. For every base node $v \in B(G_\tau)$, we have $\widehat{\mu}(v) \geq \mu^{\mathcal{G}_\tau^\uparrow}(v)$.*

Proof. Fix a base node v. Let C be a cycle dominated by v in G_τ. Let $\tilde{\mu} \in \mathcal{L}$ be a node labeling which is feasible in C and satisfies $\tilde{\mu}(v) \geq \mu(v)$. It suffices to prove that $\tilde{\mu}(v) \geq \mu^{\mathcal{G}_\tau^\uparrow}(v)$, as C and $\tilde{\mu}$ were chosen arbitrarily. Without loss of generality, we may assume that $\tilde{\mu}(w) = \top$ for all $w \in V \setminus V(C)$. Then, $\tilde{\mu}$ is feasible in G_τ. Since there are no loose arcs in C with respect to μ and $\tilde{\mu}(v) \geq \mu(v)$, we also have $\tilde{\mu}(w) \geq \mu(w)$ for all $w \in V(C)$. It follows that $\tilde{\mu} \geq \mu$. Hence, $\tilde{\mu}$ is a fixed point of $\mathcal{G}_\tau^\uparrow$ that is pointwise at least μ. From the pointwise minimality of $\mu^{\mathcal{G}_\tau^\uparrow}$, we get $\tilde{\mu} \geq \mu^{\mathcal{G}_\tau^\uparrow}$. □

The next theorem shows that if we initialize the base nodes with their corresponding threshold labels, then Algorithm 10 returns $\mu^{\mathcal{G}_\tau^\uparrow}$. Even more, it suffices to have an initial node labeling $\nu \in \mathcal{L}$ such that $\mu^{\mathcal{G}_\tau^\uparrow}(v) \leq \nu(v) \leq \widehat{\mu}(v)$ for all $v \in B(G_\tau)$. For the other nodes $v \notin B(G_\tau)$, we can simply set $\nu(v) \leftarrow \top$.

Theorem 5.10. *Let $\mu \in \mathcal{L}$ be a node labeling such that G_τ does not have loose arcs. Given input $\nu \in \mathcal{L}$ where $\nu \geq \mu^{\mathcal{G}_\tau^\uparrow}$ and $\nu(v) \leq \widehat{\mu}(v)$ for all $v \in B(G_\tau)$, Algorithm 10 returns $\mu^{\mathcal{G}_\tau^\uparrow}$.*

Proof. Given input $\nu \in \mathcal{L}$, let ν' be the node labeling returned by Algorithm 10. Then, we have

$$\mu \leq \mu^{\mathcal{G}_\tau^\uparrow} \leq \nu^{\mathcal{G}_\tau^\downarrow} \leq \nu' \leq \nu.$$

The second inequality is justified as follows. Since G_τ does not have loose arcs with respect to μ, it also does not have loose arcs with respect to $\mu^{\mathcal{G}_\tau^\uparrow}$ due to the pointwise minimality of $\mu^{\mathcal{G}_\tau^\uparrow}$. Hence, $\mu^{\mathcal{G}_\tau^\uparrow}$ is a fixed point of $\mathcal{G}_\tau^\downarrow$. As $\mu^{\mathcal{G}_\tau^\uparrow} \leq \nu$, we get $\mu^{\mathcal{G}_\tau^\uparrow} \leq \nu^{\mathcal{G}_\tau^\downarrow}$ by the pointwise maximality of $\nu^{\mathcal{G}_\tau^\downarrow}$. The third inequality, on the other hand, follows from the fact that $\nu^{\mathcal{G}_\tau^\downarrow}$ and ν' are obtained by iterating the operators in $\mathcal{G}_\tau^\downarrow$ on ν.

First, we show that $\mu^{\mathcal{G}_\tau^\uparrow} = \nu^{\mathcal{G}_\tau^\downarrow}$. Consider the set $S := \{v \in V : \mu^{\mathcal{G}_\tau^\uparrow}(v) < \nu^{\mathcal{G}_\tau^\downarrow}(v)\}$, and suppose that $S \neq \emptyset$ for the sake of contradiction. Observe that every arc in $\delta^+_{G_\tau}(S)$ is violated with respect to $\mu^{\mathcal{G}_\tau^\uparrow}$ because there are no loose arcs in G_τ with respect to $\nu^{\mathcal{G}_\tau^\downarrow}$. Since $\mu^{\mathcal{G}_\tau^\uparrow}$ is feasible in G_τ, there exists a strategy σ for Even such that $G_{\sigma\tau}$ does not contain violated arcs with respect to $\mu^{\mathcal{G}_\tau^\uparrow}$. Hence, it follows that $\delta^+_{G_{\sigma\tau}}(S) = \emptyset$. As $G_{\sigma\tau}$ is a sinkless graph, there exists a cycle C in $G_{\sigma\tau}[S]$. By the Cycle Lemma, C is even because $\mu^{\mathcal{G}_\tau^\uparrow}(v) < \top$ for all $v \in S$. Thus, every $w \in \Pi(C)$ is a base node and satisfies $\widehat{\mu}(w) \leq \mu^{\mathcal{G}_\tau^\uparrow}(w)$. However, we obtain the following contradiction

$$\widehat{\mu}(w) \leq \mu^{\mathcal{G}_\tau^\uparrow}(w) < \nu^{\mathcal{G}_\tau^\downarrow}(w) \leq \nu(w) \leq \widehat{\mu}(w).$$

Next, we show that $\nu^{\mathcal{G}_\tau^\downarrow} = \nu'$, or equivalently, ν' is a fixed point of $\mathcal{G}_\tau^\downarrow$. Let $T = (n-1)|E_\tau|$ be the total number of steps carried out by the algorithm. For each $0 \leq t \leq T$, let $\nu_t \in \mathcal{L}$ be the node labeling at the end of step t. Note that $\nu_0 = \nu$ and $\nu_T = \nu'$. We will prove that for each $0 \leq t \leq T$, if an arc uv is loose with respect to ν_t, then there exists a tight path P with respect to ν_t from v to some $w \in V$ such that $u \notin V(P)$ and $\nu_t(w) = \nu(w)$. This would then imply the absence of loose arcs with respect to $\nu_T = \nu'$. Indeed, if there were a loose arc uv, concatenating uv with the tight v-w path P yields a u-w path. However, this u-w path certifies that $\nu'(u)$ would have been smaller because $\nu'(w) = \nu(w)$ and the algorithm ran for $n-1$ iterations; a contradiction.

We now prove the statement by induction on $t \geq 0$. The base case $t = 0$ is trivially true with the singleton path $P = \{v\}$. Suppose that the statement is true for some $t \geq 0$, and let uv be the arc processed in step $t+1$. We may assume that uv is loose with respect to ν_t, as otherwise $\nu_{t+1} = \nu_t$ and we are done. By the inductive hypothesis, there exists a tight path P with respect to ν_t from v to some $w \in V$ such that $u \notin V(P)$ and $\nu_t(w) = \nu(w)$. Pick a shortest such P. Then, $\nu_t(s) < \nu(s)$ for all $s \in V(P) \setminus (w)$.

Consider the u-w path $P' := P \cup \{uv\}$. At the end of step $t+1$, P' is tight with respect to ν_{t+1} and $\nu_{t+1}(w) = \nu(w)$. Let qu be a loose arc with respect to ν_{t+1}. To finish the proof, it suffices to show that $q \notin V(P')$. For the purpose of contradiction, suppose that $q \in V(P')$. Let C' be the unique cycle in $P' \cup \{qu\}$. Since $\nu_{t+1}(s) < \top$ for all $s \in V(P')$, C' is even by the Cycle Lemma. Let $p \in \Pi(C')$, which is a base node. If $p = q$, then $\widehat{\mu}(p) < \nu_{t+1}(p) \leq \nu(p) \leq \widehat{\mu}(p)$, where the strict inequality is due to qu being loose. If $p \neq q$, then $\widehat{\mu}(p) \leq \nu_{t+1}(p) < \nu(p) \leq \widehat{\mu}(p)$, where the strict inequality is due to $p \neq w$. Both cases result in a contradiction. □

Our strategy for computing such a node labeling ν is to find the cycles in Definition 5.8. In particular, for every base node $v \in B(G_\tau)$, we aim to find a cycle C dominated by v in G_τ such that $\widehat{\mu}(v)$ can be extended to a node labeling that is feasible in C. To accomplish this goal, we first introduce the notion of *width* in Section 5.4.1, which allows us to evaluate how 'good' a cycle is. It is defined using chains in the poset of subtrees of T, where the partial order is given by $\sqsubseteq$. Then, in Section 5.4.2, we show how to obtain the desired cycles by computing minimum bottleneck cycles on a suitably defined auxiliary digraph.

5.4.1 Width from a Chain of Subtrees in T

Two ordered trees T' and T'' are said to be *distinct* if $T' \not\equiv T''$ (not isomorphic in the sense of Definition 5.1). Let h be the height of our universal tree T. For $0 \leq j \leq h$, denote $\mathcal{T}_j$ as the set of distinct (whole) subtrees rooted at the vertices of depth $h - j$ in T. For example, $\mathcal{T}_h = \{T\}$, while $\mathcal{T}_0$ contains the trivial tree with a single vertex. Since we assumed that all the leaves in T are at the same depth, every tree in $\mathcal{T}_j$ has height j. We denote $\mathcal{T} = \cup_{j=0}^{h} \mathcal{T}_j$ as the union of all these subtrees. The sets $\mathcal{T}$ and $\mathcal{T}_j$ form posets with respect to the partial order $\sqsubseteq$. The next definition is the usual chain cover of a poset, where we additionally require that the chains form an indexed tuple instead of a set.

Definition 5.11. For $0 \leq j \leq h$, let $\mathcal{C}_j = (\mathcal{C}_j^0, \mathcal{C}_j^1, \dots, \mathcal{C}_j^\ell)$ be a tuple of chains in the poset $(\mathcal{T}_j, \sqsubseteq)$. We call $\mathcal{C}_j$ a *cover of* $\mathcal{T}_j$ if $\cup_{k=0}^{\ell} \mathcal{C}_j^k = \mathcal{T}_j$. A *cover of* $\mathcal{T}$ is a tuple $\mathcal{C} = (\mathcal{C}_0, \mathcal{C}_1, \dots, \mathcal{C}_h)$ where $\mathcal{C}_j$ is a cover of $\mathcal{T}_j$ for all $0 \leq j \leq h$. We refer to $\mathcal{C}_j$ as the *jth-subcover of* $\mathcal{C}$. Given a cover $\mathcal{C}$ of $\mathcal{T}$, we denote the trees in the chain $\mathcal{C}_j^k$ as $T_{0,j}^k \sqsubset T_{1,j}^k \sqsubset \cdots \sqsubset T_{|\mathcal{C}_j^k|-1,j}^k$.

An example of an ordered tree with its cover is given in Figure 5.5. We are ready to introduce the key concept of this subsection.

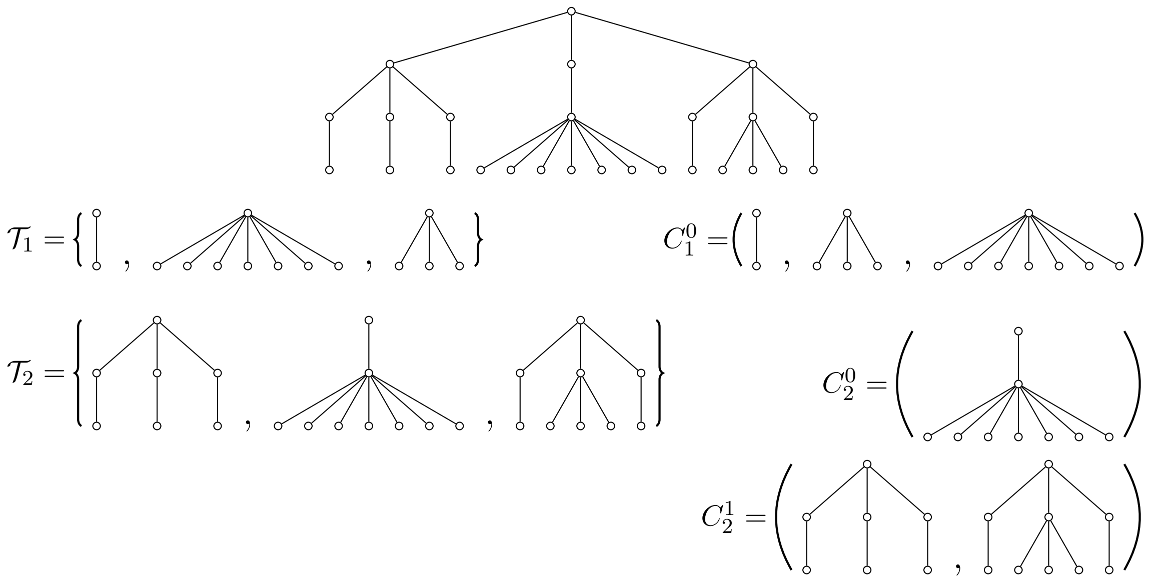

Fig. 5.5 An ordered tree T of height 3, and a cover $\mathcal{C}_j$ of $\mathcal{T}_j$ for all $0 < j < 3$. Recall that $\mathcal{T}_j$ is the set of distinct subtrees of T rooted at depth $3 - j$, while $\mathcal{C}_j^k$ is the kth chain in $\mathcal{C}_j$.

Definition 5.12. Let $\mathcal{C}$ be a cover of $\mathcal{T}$. Let H be a subgraph of G_τ and $j = \lceil \pi(H)/2 \rceil$. For a fixed chain $\mathcal{C}_j^k$ in $\mathcal{C}_j$, the *kth-width* of H, denoted $\alpha_{\mathcal{C}}^k(H)$, is the smallest integer $i \geq 0$ such that there exists a node labeling $\nu : V(H) \to L(T^k_{i,j})$ which is feasible in H. If i does not exist, then $\alpha^k_{\mathcal{C}}(H) = \infty$.

Note that $T^k_{i,j}$ is the $(i+1)$-th smallest tree in the chain $\mathcal{C}^k_j$. We are mainly interested in the case when H is a cycle, and write $\alpha^k(H)$ whenever the cover $\mathcal{C}$ is clear from context. Observe that the definition above requires $\nu(v) \neq \top$ for all $v \in V(H)$. Hence, an odd cycle has infinite kth-width by the Cycle Lemma. As $(\mathcal{C}^k_j, \sqsubseteq)$ is a chain, for all finite $i \geq \alpha^k(H)$, there exists a node labeling $\nu : V(H) \to L(T^k_{i,j})$ which is feasible in H. The next lemma illustrates the connection between the kth-width of an even cycle and its path decomposition.

Lemma 5.13. *Let $\mathcal{C}$ be a cover of $\mathcal{T}$. For an even cycle C, let $\Pi(C) = \{v_1, v_2, \dots, v_\ell\}$ and $j = \pi(C)/2$. Decompose C into arc-disjoint paths $P_1, P_2, \dots, P_\ell$ such that each P_i ends at v_i. Then, $\alpha^k(C) = \max_{i\in[\ell]} \alpha^k(P_i)$ for all $0 \leq k < |\mathcal{C}_j|$.*

Proof. Fix a chain $\mathcal{C}^k_j$ and let $\alpha^* = \max_{i\in[\ell]} \alpha^k(P_i)$. Clearly, any node labeling which is feasible in C is also feasible in P_i for all $i \in [\ell]$. So, $\alpha^k(C) \geq \alpha^*$. Next, we prove the reverse inequality. For each $i \in [\ell]$, there exists a node labeling $\nu_i : V(P_i) \to L(T^k_{\alpha^*,j})$ which is feasible in P_i. Without loss of generality, we may assume that $\nu_i(v_i) = \min L(T^k_{\alpha^*,j})$ for all $i \in [\ell]$. Let us define a new node labeling ν as follows. For $v \in V(C)$, set $\nu(v) := \nu_i(v)$ where $i \in [\ell]$ is the unique index such that $\delta^-_{P_i}(v) \neq \emptyset$. Then, ν is feasible in C because $\nu(v) \neq \top$ for all $v \in V(C)$. Hence, $\alpha^k(C) \leq \alpha^*$. ☐

For a base node $v \in B(G_\tau)$, let us consider the cycles in G_τ which are dominated by v. Among these cycles, we are interested in finding one with the smallest kth-width. So, we extend the notion of kth-width to base nodes in the following way.

Definition 5.14. Let $\mathcal{C}$ be a cover of $\mathcal{T}$. Let $v \in B(G_\tau)$ be a base node and $j = \pi(v)/2$. For $0 \leq k < |\mathcal{C}_j|$, the *kth-width* of v is defined as

$$\alpha^k_{\mathcal{C}}(v) := \min\left\{\alpha^k_{\mathcal{C}}(C) : C \text{ is a cycle dominated by } v \text{ in } G_\tau\right\}.$$

Again, we write $\alpha^k(v)$ whenever the cover $\mathcal{C}$ is clear from context. Observe that $T^k_{\alpha^k(v),\pi(v)/2}$ is the smallest tree in the chain $\mathcal{C}^k_{\pi(v)/2}$ which can encode a node labeling that is feasible on some cycle dominated by v.

Given a leaf $\xi \in L(T)$ and integers $i, j, k \in \mathbb{Z}_{\geq 0}$, the following subroutine locates a member of the chain $\mathcal{C}^k_j$ in T whose leaves are at least ξ and into which $T^k_{i,j}$ is embeddable.

Subroutine 5.4.1. Raise(ξ, i, j, k)

Given a leaf $\xi \in L(T)$ and integers $i, j, k \in \mathbb{Z}_{\geq 0}$, return the smallest leaf $\xi' \in L(T)$ such that (1) $\xi' \geq \xi$; and (2) ξ' is the smallest leaf in the subtree $T^k_{i',j}$ for some $i' \geq i$. If ξ' does not exist, then return $\top$.

This subroutine allows us to relate the kth-width $\alpha^k(v)$ of a base node v to its threshold label $\widehat{\mu}(v)$. In particular, for any $0 \leq k < |\mathcal{C}_{\pi(v)/2}|$, the element returned by Raise$(\mu(v), \alpha^k(v), \frac{\pi(v)}{2}, k)$ is at least $\widehat{\mu}(v)$. Moreover, the smallest such element over all k is precisely $\widehat{\mu}(v)$.

Lemma 5.15. *Let $\mu \in \mathcal{L}$ be a node labeling such that G_τ does not have loose arcs. Let $v \in B(G_\tau)$ be a base node and $j = \pi(v)/2$. If $\xi^k \in \bar{L}(T)$ is the element returned by* Raise$(\mu(v), \alpha^k(v), j, k)$*, then*

$$\widehat{\mu}(v) = \min_{0 \leq k < |\mathcal{C}_j|} \xi^k.$$

Proof. First, we prove that $\widehat{\mu}(v) \leq \xi^k$ for all $0 \leq k < |\mathcal{C}_j|$. Fix a k and assume that $\xi^k \neq \top$. Then, ξ^k is the smallest leaf in a subtree $T^k_{i,j}$ of T for some $i \geq \alpha^k(v)$. From the definition of $\alpha^k(v)$, there exists a cycle C dominated by v in G_τ and a node labeling $\nu : V(C) \to L(T^k_{i,j})$ which is feasible in C. Since $\pi(v) = 2j$ and $\nu(w) \neq \top$ for all $w \in V(C)$, we may assume that $\nu(v) = \min L(T^k_{i,j})$. Let us define a new node labeling $\tilde{\mu} : V \to \bar{L}(T)$ as follows. If $w \in V(C)$, set $\tilde{\mu}(w)$ as the concatenation of $\xi^k|_{2j}$ and $\nu(v)$, which is a leaf in T. Otherwise, set $\tilde{\mu}(w) := \top$. Then, $\tilde{\mu}$ is feasible in C and $\tilde{\mu}(v) = \xi^k \geq \mu(v)$, where the equality is due to our assumption $\nu(v) = \min L(T^k_{i,j})$. Hence, $\tilde{\mu}(v) \geq \widehat{\mu}(v)$ from the definition of $\widehat{\mu}(v)$.

It is left to show that $\widehat{\mu}(v) \geq \xi^k$ for some $0 \leq k < |\mathcal{C}_j|$. We may assume that $\widehat{\mu}(v) \neq \top$. Let $T^k_{i,j}$ be the subtree of T rooted at $\widehat{\mu}(v)|_{2j}$. We claim that $\widehat{\mu}(v) \geq \xi^k$. From the definition of $\widehat{\mu}(v)$, there exists a node labeling $\tilde{\mu} \in \mathcal{L}$ and a cycle C dominated by v in G_τ such that $\tilde{\mu}$ is feasible in C and $\tilde{\mu}(v) = \widehat{\mu}(v) \geq \mu(v)$. By the Cycle Lemma, $\tilde{\mu}(v)|_{2j} = \tilde{\mu}(w)|_{2j}$ for all $w \in V(C)$. Since $\tilde{\mu}$ is feasible in C and $\tilde{\mu}(w) \neq \top$ for all $w \in V(C)$, it follows that $i \geq \alpha^k(C) \geq \alpha^k(v)$. So, the only way to get $\widehat{\mu}(v) < \xi^k$ is when $\widehat{\mu}(v) = \mu(v)$ and $\mu(v)$ is not the smallest leaf in the subtree of T rooted at $\mu(v)|_{2j}$. However, this cannot happen. Indeed, denoting s as the out-neighbour of v in C, we have $\mu(v)|_{2j} = \tilde{\mu}(v)|_{2j} = \tilde{\mu}(s)|_{2j}$. So, if $\tilde{\mu}(s)|_{2j} \geq \mu(s)|_{2j}$, then vs is loose with respect to μ. Otherwise, the s-v path in C contains a loose arc with respect to μ. □

The necessary number of chains in the subcover $\mathcal{C}_{\pi(v)/2}$ can be large if T is an arbitrary ordered tree. Fortunately, the universal trees constructed in the literature admit covers with small subcovers. We prove that a succinct (n, h)-universal tree has a cover with only 1 chain per subcover in Section 5.4.4, whereas a succinct Strahler (n, h)-universal tree (introduced by Daviaud et al. [43]) has a cover with at most $\log n$ chains per subcover in Section 5.4.5.

Let $\rho(T, \mathcal{C})$ denote the running time of Raise. We provide efficient implementations of Raise for succinct universal trees and succinct Strahler universal trees in Sections 5.4.4–5.4.5. They have the same running time as Tighten, i.e., $\rho(T, \mathcal{C}) = O(\log n \log h)$.

5.4.2 Estimating the Width of Base Nodes

In light of the previous discussion, we can now focus on computing the kth-width of a base node $w \in B(G_\tau)$. Fix a $0 \leq k < |\mathcal{C}_{\pi(w)/2}|$. Since we ultimately need a label that lies between $\mu^{\mathcal{G}_\tau^\uparrow}(w)$ and $\widehat{\mu}(w)$ in order to initialize Algorithm 10, it suffices to compute a 'good' under-estimation of $\alpha^k(w)$. In this subsection, we reduce this problem to computing a minimum bottleneck cycle in an auxiliary digraph D with nonnegative arc costs $c^k \geq 0$.

For a base node $w \in B(G_\tau)$, let K_w denote the SCC containing w in $(G_\tau)_{\pi(w)}$, the subgraph of G_τ induced by nodes with priority at most $\pi(w)$. Let $K'_w \subseteq K_w$ be the subgraph obtained by deleting the incoming arcs $\delta^-(v)$ for all $v \in \Pi(K_w) \setminus \{w\}$. Then, we define J_w as the subgraph of K'_w induced by those nodes which can reach w in K'_w. These are the nodes which can reach w in K_w without encountering an intermediate node of priority $\pi(w)$.

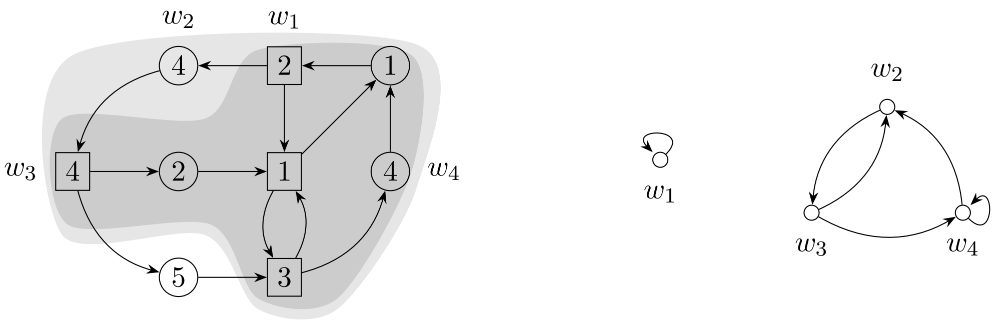

Fig. 5.6 An example of a 1-player game (G_τ, π) for Even is given on the left, with its auxiliary digraph D on the right. Nodes in V_0 and V_1 are drawn as squares and circles respectively. Base nodes are labeled as w_1, w_2, w_3, w_4. The light gray region is K_{w_4}, while the dark gray region is J_{w_4}.

The auxiliary digraph D is constructed as follows. Its node set is $B(G_\tau)$. For every ordered pair (v, w) of base nodes where $\pi(v) = \pi(w)$, add the arc vw if v has an outgoing arc in J_w. Note that if $(v, w) \in D$, then v can reach w by only seeing smaller priorities on the intermediate nodes. As ordered pairs of the form (v, v) are also considered, D may contain self-loops. Observe that D is a disjoint union of SCCs, each of which consists of base nodes with the same priority (see Figure 5.6 for an example). For $w \in B(G_\tau)$, we denote D_w as the component in D which contains w.

To finish the description of D, it is left to assign the arc costs c^k. Note that the graph structure of D is independent of k. We give a range in which the cost of each arc should lie. Fix a base node $w \in B(G_\tau)$ and let $j = \pi(w)/2$. Recall that $\mathcal{J}_w^\downarrow = \{\mathrm{Drop}_e : e \in E(J_w)\}$ is the set of Drop operators in the subgraph $J_w \subseteq G_\tau$. For each $0 \leq i < |\mathcal{C}_j^k|$, let $\lambda_{i,w}^k : V(J_w) \to \bar{L}(T_{i,j}^k)$ be the greatest simultaneous fixed point of $\mathcal{J}_w^\downarrow$ subject to $\lambda_{i,w}^k(w) = \min L(T_{i,j}^k)$. Then, for each arc $vw \in E(D)$, the lower and upper bounds of $c^k(vw)$ are given by

$$\begin{aligned} \underline{c}^k(vw) &:= \min\left\{i : \lambda_{i,w}^k(u) \neq \top \text{ for some } u \in N_{J_w}^+(v)\right\} \\ \overline{c}^k(vw) &:= \min\left\{\alpha^k(P) : P \text{ is a } u\text{-}w \text{ path in } J_w \text{ where } u \in N_{J_w}^+(v)\right\} \end{aligned} \tag{5.1}$$

respectively. The lower bound $\underline{c}^k(vw)$ is the smallest integer $i \geq 0$ such that the greatest simultaneous fixed point $\lambda^k_{i,w}$ assigns a non-top label to an out-neighbor of v in J_w. On the other hand, the upper bound $\overline{c}^k(vw)$ is the minimum kth-width of a path from an out-neighbor of v to w in J_w. Note that these quantities could be equal to $+\infty$.

Lemma 5.16. *For every arc $vw \in E(D)$, we have $\underline{c}^k(vw) \leq \overline{c}^k(vw)$.*

Proof. We may assume that $\overline{c}^k(vw) < \infty$. Let P be a u-w path in J_w where $u \in N^+_{J_w}(v)$ and $\alpha^k(P) = \overline{c}^k(vw)$. Let $j = \pi(w)/2$. From the definition of $\alpha^k(P)$, there exists a node labeling $\nu : V(P) \to L(T^k_{\alpha^k(P),j})$ which is feasible in P. Note that $\nu(s) \neq \top$ for all $s \in V(P)$. Moreover, $\nu(s) \geq \lambda^k_{\alpha^k(P),w}(s)$ for all $s \in V(P)$ because $\nu(w) \geq \min L(T^k_{\alpha^k(P),j}) = \lambda^k_{\alpha^k(P),w}(w)$ and there are no loose arcs in P with respect to $\lambda^k_{\alpha^k(P),w}$. It follows that $\lambda^k_{\alpha^k(P),w}(u) \leq \nu(u) < \top$, which gives $\underline{c}^k(vw) \leq \alpha^k(P)$ as desired. □

The next lemma concerns the greatest simultaneous fixed point of $\mathcal{J}^{\downarrow}_w$ when comparable trees are used as codomains.

Lemma 5.17. *For every node $u \in V(J_w)$, if $\lambda^k_{i,w}(u) \neq \top$, then $\lambda^k_{i',w}(u) \neq \top$ for all $i' \geq i$.*

Proof. Let $j = \pi(w)/2$ and fix integers $i' \geq i$. Define the node labelings $\nu^{(0)}_i : V(J_w) \to \bar{L}(T^k_{i,j})$ and $\nu^{(0)}_{i'} : V(J_w) \to \bar{L}(T^k_{i',j})$ by $\nu^{(0)}_i(w) = \min L(T^k_{i,j})$, $\nu^{(0)}_{i'}(w) = \min L(T^k_{i',j})$, and $\nu^{(0)}_i(u) = \nu^{(0)}_{i'}(u) = \top$ for all $u \neq w$. We know that $\lambda^k_{i,w}$ can be obtained by applying a sequence R of Drop operators from $\mathcal{J}_w$ to $\nu^{(0)}_i$. For each $0 \leq \ell \leq |R|$, let R_ℓ be the sequence consisting of the first ℓ elements in R. Let $\nu^{(\ell)}_i$ and $\nu^{(\ell)}_{i'}$ be the node labelings obtained by applying R_ℓ to $\nu^{(0)}_i$ and $\nu^{(0)}_{i'}$ respectively. Since $T^k_{i,j} \sqsubseteq T^k_{i',j}$, there exists an injective and order-preserving homomorphism $f : V(T^k_{i,j}) \to V(T^k_{i',j})$. Let us extend f by setting $f(\top) := \top$. As $\lambda^k_{i',w} \leq \nu^{(|R|)}_{i'}$, it suffices to show that $\nu^{(\ell)}_{i'}(u) \leq f(\nu^{(\ell)}_i(u))$ for all $u \in V(J_w)$ and $0 \leq \ell \leq |R|$. We proceed by induction on ℓ. The base case $\ell = 0$ is true by construction. Suppose that this is true for some $\ell \geq 0$, and let $\mathrm{Drop}_{uv}(\cdot,\cdot)$ be the $(\ell+1)$-th element in R. Since f is order-preserving and Drop_{uv} is monotone with respect to its input node labeling, we obtain $\nu^{(\ell+1)}_{i'}(u) \leq f(\nu^{(\ell+1)}_i(u))$ by the inductive hypothesis. □

For a cycle C in D, its (bottleneck) c^k*-cost* is defined as $c^k(C) := \max_{e\in E(C)} c^k(e)$. Note that self-loops in D are considered cycles. The next theorem enables us to obtain the desired initial node labeling ν for Algorithm 10 by computing minimum bottleneck cycles in D.

Theorem 5.18. *Let $\mathcal{C}$ be a cover of $\mathcal{T}$. Let $\mu \in \mathcal{L}$ be a node labeling such that G_τ does not have loose arcs. For a base node w, let c^k be arc costs in D_w such that $\underline{c}^k \leq c^k \leq \overline{c}^k$ for all $0 \leq k < |\mathcal{C}_{\pi(w)/2}|$. For each k, let i^k be the minimum c^k-cost of a cycle containing w in D_w, and ξ^k be the label returned by* $\textsc{Raise}(\mu(w), i^k, \frac{\pi(w)}{2}, k)$.[2] *Then, $\mu^{\mathcal{G}^{\uparrow}_{\tau}}(w) \leq \min_k \xi^k \leq \widehat{\mu}(w)$.*

[2]We set $\xi^k = \top$ if $i^k = \infty$.

Proof. Let $j = \pi(w)/2$. We first prove the lower bound $\mu^{\mathcal{G}_\tau^\uparrow}(w) \leq \xi^k$ for all $0 \leq k < |\mathcal{C}_j|$. Fix a k and assume that $\xi^k \neq \top$. Let C be a minimum c^k-cost cycle containing w in D. Denote $C = (w_1, w_2, \dots, w_\ell)$ where $w = w_1 = w_\ell$. For every $s \in [\ell]$, let $\nu_s : V \to \bar{L}(T)$ be the node labeling defined by $\nu_s(w_s) := \xi^k$ and $\nu_s(v) := \top$ for all $v \neq w_s$. Consider the greatest simultaneous fixed point $\nu_s^{\mathcal{J}_{w_s}^\downarrow}$. Since $\pi(J_{w_s}) = \pi(w_s) = 2j$, we have $\nu_s^{\mathcal{J}_{w_s}^\downarrow}(w_s) = \nu_s(w_s) = \xi^k$ because ξ^k is the smallest leaf in the subtree of T rooted at $\xi^k|_{2j}$. This also implies that $\nu_s^{\mathcal{J}_{w_s}^\downarrow}(v) \geq \xi^k$ for all $v \in V$. Furthermore, $\nu_s^{\mathcal{J}_{w_s}^\downarrow}$ is feasible in $G_\tau \setminus \delta^+(w_s)$, as ν_s is feasible in $G_\tau \setminus \delta^+(w_s)$.

Claim 5.19. *For each $1 < s \leq \ell$, there exists a node $u_s \in N^+_{J_{w_s}}(w_{s-1})$ such that $\nu_s^{\mathcal{J}_{w_s}^\downarrow}(u_s)|_{2j} = \xi^k|_{2j}$.*

Proof. Fix an s and let $i = \underline{c}^k(w_{s-1}w_s)$. From the definition of $\underline{c}^k$ in (5.1), there exists a node $u_s \in N^+_{J_{w_s}}(w_{s-1})$ such that $\lambda^k_{i,w_s}(u_s) \neq \top$. Recall that $\lambda^k_{i,w_s} : V(J_{w_s}) \to \bar{L}(T^k_{i,j})$ is the greatest simultaneous fixed point of $\mathcal{J}_{w_s}^\downarrow$ subject to $\lambda^k_{i,w_s}(w_s) = \min L(T^k_{i,j})$. We will show that $\nu_s^{\mathcal{J}_{w_s}^\downarrow}(u_s)|_{2j} = \xi^k|_{2j}$. First, observe that $\nu_s^{\mathcal{J}_{w_s}^\downarrow} : V \to \bar{L}(T)$ is the greatest simultaneous fixed point of $\mathcal{J}_{w_s}^\downarrow$ subject to $\nu_s^{\mathcal{J}_{w_s}^\downarrow}(w_s) = \xi^k$. Since ξ^k is returned by $\textsc{Raise}(\mu(w), i^k, j, k)$, it is the smallest leaf in a copy of $T^k_{i',j}$ in the main tree T for some $i' \geq i^k$. As $c^k(C) = i^k$ and $w_{s-1}w_s \in E(C)$, we also have $i^k \geq c^k(w_{s-1}w_s) \geq \underline{c}^k(w_{s-1}w_s) = i$, where the last inequality is due to our choice of the arc costs c^k. It follows that $i' \geq i$ and $T^k_{i,j} \sqsubseteq T^k_{i',j}$. Thus, we obtain $\nu_s^{\mathcal{J}_{w_s}^\downarrow}(u_s)|_{2j} = \xi^k|_{2j}$ by Lemma 5.17 because $\lambda^k_{i,w_s}(u_s) \neq \top$. □

Now, consider the node labeling ν defined by $\nu(v) := \min_{s \in [\ell]} \nu_s^{\mathcal{J}_{w_s}^\downarrow}(v)$ for all $v \in V$. Note that $\nu(w_s) = \xi^k$ for all $s \in [\ell]$. It suffices to show that ν is feasible in G_τ and $\nu \geq \mu$. This is because it would then imply $\nu \geq \mu^{\mathcal{G}_\tau^\uparrow}$ by the pointwise minimality of $\mu^{\mathcal{G}_\tau^\uparrow}$. In particular, $\xi^k = \nu(w) \geq \mu^{\mathcal{G}_\tau^\uparrow}(w)$.

We first prove that ν is feasible in G_τ. Notice that ν is feasible in $G_\tau \setminus \cup_{s=1}^{\ell} \delta^+(w_s)$ because ν_s is feasible in $G_\tau \setminus \delta^+(w_s)$ for all $s \in [\ell]$. So, it suffices to show that every w_s has a non-violated outgoing arc in G_τ with respect to ν. Fix an $1 < s \leq \ell$. By Claim 5.19, there exists a node $u_s \in N^+_{J_{w_s}}(w_{s-1})$ such that $\nu_s^{\mathcal{J}_{w_s}^\downarrow}(u_s)|_{2j} = \xi^k|_{2j}$. Hence, $\nu(u_s)|_{2j} = \xi^k|_{2j}$. As $\nu(w_{s-1}) = \xi^k$ and $\pi(w_{s-1}) = 2j$, the arc $w_{s-1}u_s$ is non-violated with respect to ν.

Next, we prove that $\nu_s^{\mathcal{J}_{w_s}^\downarrow} \geq \mu$ for all $s \in [\ell]$, which will then imply $\nu \geq \mu$ as desired. We proceed by induction on s. For the base case $s = \ell$, we know that $\nu_\ell^{\mathcal{J}_{w_\ell}^\downarrow}$ is the greatest simultaneous fixed point of $\mathcal{J}_{w_\ell}^\downarrow$ subject to $\nu_\ell^{\mathcal{J}_{w_\ell}^\downarrow}(w_\ell) = \xi^k$. Observe that μ is also a simultaneous fixed point of $\mathcal{J}_{w_\ell}^\downarrow$ because there are no loose arcs in G_τ with respect to μ. As $\nu_\ell^{\mathcal{J}_{w_\ell}^\downarrow}(w_\ell) = \xi^k \geq \mu(w_\ell)$ due to $w_\ell = w$, we obtain $\nu_\ell^{\mathcal{J}_{w_\ell}^\downarrow} \geq \mu$. For the inductive step, suppose that $\nu_s^{\mathcal{J}_{w_s}^\downarrow} \geq \mu$ for some $1 < s \leq \ell$. By Claim 5.19, there exists a node $u_s \in N^+_{J_{w_s}}(w_{s-1})$ such that $\nu_s^{\mathcal{J}_{w_s}^\downarrow}(u_s)|_{2j} = \xi^k|_{2j}$. Then, $\mu(w_{s-1}) \leq \nu_s^{\mathcal{J}_{w_s}^\downarrow}(w_{s-1}) = \xi^k$, where equality follows from the tightness of $w_{s-1}u_s$ with respect to $\nu_s^{\mathcal{J}^{\downarrow}_{w_s}}$. Since $\nu_{s-1}^{\mathcal{J}^{\downarrow}_{w_{s-1}}}$ is the greatest simultaneous fixed point of $\mathcal{J}^{\downarrow}_{w_{s-1}}$ subject to $\nu_{s-1}^{\mathcal{J}^{\downarrow}_{w_{s-1}}}(w_{s-1}) = \xi^k \geq \mu(w_{s-1})$, we get $\nu_{s-1}^{\mathcal{J}^{\downarrow}_{w_{s-1}}} \geq \mu$ because μ is also a simultaneous fixed point of $\mathcal{J}^{\downarrow}_{w_{s-1}}$.

It is left to show that $\xi^k \leq \widehat{\mu}(w)$ for some $0 \leq k < |\mathcal{C}_j|$. We may assume that $\widehat{\mu}(w) \neq \top$. By Lemma 5.15, $\widehat{\mu}(w)$ is returned by $\textsc{Raise}(\mu(w), \alpha^k(w), j, k)$ for some $0 \leq k < |\mathcal{C}_j|$. Let H be a cycle in G_τ such that $w \in \Pi(H)$ and $\alpha^k(H) = \alpha^k(w)$. We denote $\Pi(H) = \{w_1, w_2, \dots, w_r\}$ and decompose H into arc-disjoint paths $P_1, P_2, \dots, P_r$ such that P_s is a w_{s-1}-w_s path for all $s \in [r]$, with the convention $w_0 := w_r$. Since each P_s lies in the subgraph J_{w_s}, we have $w_{s-1}w_s \in E(D)$ for all $s \in [r]$, and their union induces a cycle H' containing w in D. Then,

$$i^k \leq c^k(H') = \max_{s\in[r]} c^k(w_{s-1}w_s) \leq \max_{s\in[r]} \overline{c}^k(w_{s-1}w_s) \leq \max_{s\in[r]} \alpha^k(P_s) = \alpha^k(H) = \alpha^k(w).$$

The third inequality follows from the definition of $\overline{c}^k$ in (5.1), while the second equality is due to Lemma 5.13. Therefore, $\xi^k \leq \widehat{\mu}(w)$ because $\textsc{Raise}$ is monotone with respect to its second argument. □

5.4.3 The Label-Correcting Algorithm

The overall algorithm for computing $\mu^{\mathcal{G}^{\uparrow}_{\tau}}$ is given in Algorithm 11. The main idea is to initialize the labels on base nodes via the recipe given in Theorem 5.18, before running Algorithm 10. The labels on $V \setminus B(G_\tau)$ are initialized to $\top$. The auxiliary graph D serves as a condensed representation of the 'best' paths between base nodes. The arc costs are chosen such that minimum bottleneck cycles in D give a good estimate on the width of base nodes.

Algorithm 11: Label-correcting algorithm for computing the least fixed point

```
Input  : 1-player game $(G_\tau, \pi)$ for Even, universal tree $T$ with cover $\mathcal{C}$, node
         labeling $\mu : V \to \bar{L}(T)$ with no loose arc in $G_\tau$
Output : $\mu^{\mathcal{G}^{\uparrow}_{\tau}}$
1  $\nu(v) \leftarrow \top$ for all $v \in V$
2  Construct auxiliary digraph $D$
3  foreach component $H$ in $D$ do
4      for $k = 0$ to $|\mathcal{C}_{\pi(H)/2}| - 1$ do
5          Assign arc costs $c^k$ to $H$ where $\underline{c}^k \leq c^k \leq \overline{c}^k$
6          foreach $w \in V(H)$ do
7              $i^k \leftarrow$ minimum $c^k$-cost of a cycle containing $w$ in $H$
8              if $i^k < \infty$ then
9                  $\nu(w) \leftarrow \min(\nu(w), \textsc{Raise}(\mu(w), i^k, \frac{\pi(H)}{2}, k))$
10 $\nu \leftarrow \textsc{BellmanFord}((G_\tau, \pi), \nu)$
11 return $\nu$
```

In the next two paragraphs, we elaborate on how the arc costs c^k and minimum bottleneck cycles are computed.

Computing arc costs Fix a component H in D and let $j = \pi(H)/2$. If the chain $\mathcal{C}_j^k$ is short, then for every node $w \in V(H)$, we can obtain c^k for its incoming arcs $\delta_D^-(w)$ by running Algorithm 10 on the subgraph J_w $|\mathcal{C}_j^k|$ times. In every run $0 \leq i < |\mathcal{C}_j^k|$, we use the initial node labeling $\nu_i : V(J_w) \to \bar{L}(T_{i,j}^k)$ given by $\nu_i(w) := \min L(T_{i,j}^k)$ and $\nu_i(u) := \top$ for all $u \neq w$. Let ν_i' be the returned node labeling. Observe that $\nu_i' \geq \lambda_{i,w}^k$. Moreover, for each node $u \in V(J_w)$, if $\nu_i'(u) = \top$, then $\alpha^k(P) > i$ for all u-w paths P in J_w. This is due to our choice of the initial node labeling ν_i, and the fact that Algorithm 10 ran for $n-1$ iterations.

For every arc $vw \in \delta_D^-(w)$, we set its cost as

$$c^k(vw) := \min_{u \in N_{J_w}^+(v)} \{i : \nu_i'(u) \neq \top\}.$$

Since $\lambda_{c^k(vw),w}^k(u) \leq \nu_{c^k(vw)}'(u) \neq \top$ for some $u \in N_{J_w}^+(v)$, we have $c^k(vw) \geq \underline{c}^k(vw)$. We also have $c^k(vw) \leq \overline{c}^k(vw)$ because $\nu_i'(u) = \top$ for all $i < c^k(vw)$ and $u \in N_{J_w}^+(v)$.

If the chain $\mathcal{C}_j^k$ is long, then we combine the above approach with binary search. Start by selecting the middle tree $T_{i,j}^k$ in the chain, i.e. $i = \lfloor |\mathcal{C}_j^k|/2 \rfloor$. Run Algorithm 10 on the subgraph J_w with the initial node labeling $\nu_i : V(J_w) \to \bar{L}(T_{i,j}^k)$ as defined above, and let ν_i' be the returned node labeling. For each node $u \in V(J_w)$, if $\nu_i'(u) = \top$, then we restrict ourselves to the trees in the chain which are bigger than $T_{i,j}^k$. Otherwise, we disregard these trees. This process is repeated until we find the optimal tree $T_{i,j}^k$ for every node $u \in V(J_w)$. Hence, the time taken to compute c^k for the component H is $O(|V(H)| mn\gamma(T) \cdot \min\{|\mathcal{C}_j^k|, n \log |\mathcal{C}_j^k|\})$.

Computing Minimum Bottleneck Cycles It is well-known how to compute a minimum bottleneck directed cycle in a digraph with arbitrary arc costs. For the sake of completeness, we give a brief description. Let H be a component in D and $c_1^k < c_2^k < \cdots < c_\ell^k$ be the distinct arc costs in H. Consider the subgraph of H induced by arcs with cost at most $c_{\lceil \ell/2 \rceil}^k$, and let $K_1, K_2, \ldots, K_t$ be its SCCs. For each $i \in [t]$, if $|E(K_i)| > 0$, then every node in K_i has a cycle going through it with cost at most $c_{\lceil \ell/2 \rceil}^k$. Otherwise, K_i is a singleton with no self-loops. Hence, every cycle going through it has cost greater than $c_{\lceil \ell/2 \rceil}^k$. This observation allows us to split the instance into two. The first instance is given by the disjoint union of $K_1, K_2, \ldots, K_t$. The second instance is obtained by contracting each K_i into a node in H, destroying any self-loops on the resulting node. Then, the procedure is repeated on these two smaller instances. Since H contains $O(|V(H)|^2)$ arcs, and each arc appears in at most $O(\log \ell) = O(\log |V(H)|)$ instances, the running time is $O(|V(H)|^2 \log |V(H)|)$ using Tarjan's SCCs algorithm.

We are ready to prove a generic bound on the running time of Algorithm 11 for an arbitrary universal tree T with an arbitrary cover $\mathcal{C}$.

Theorem 5.20. *In $O(mn^2\gamma(T) \cdot \max_{j,k} |\mathcal{C}_j| \min\{|\mathcal{C}_j^k|, n\log|\mathcal{C}_j^k|\} + n\rho(T,\mathcal{C}) \cdot \max_j |\mathcal{C}_j|)$ time, Algorithm 11 returns $\mu^{\mathcal{G}_\mathcal{T}^\uparrow}$.*

Proof. Correctness follows immediately from Theorem 5.18 and Theorem 5.10. In terms of running time, identifying the base nodes takes $O(dm)$ time (see Section 5.3.1), while constructing the auxiliary digraph D takes $O(mn)$ time. Next, for every component H in D and $0 \leq k < |\mathcal{C}_{\pi(H)/2}|$, computing the arc costs c^k takes $O(|V(H)|mn\gamma(T) \cdot \min\{|\mathcal{C}^k_{\pi(H)/2}|, n\log|\mathcal{C}^k_{\pi(H)/2}|\})$. Then, finding minimum c^k-cost cycles in H takes $O(|V(H)|^2 \log|V(H)|)$ time, while applying Raise takes $O(|V(H)|\rho(T,\mathcal{C}))$ time. Finally, Bellman–Ford runs in $O(mn\gamma(T))$ time. $\square$

In the next two subsections, we apply Algorithm 11 to the quasi-polynomial universal trees constructed in the literature [88, 43]. It runs in time $O(mn^2 \log n \log d)$ time for a succinct $(n, d/2)$-universal tree, and $O(mn^2 \log^3 n \log d)$ for a succinct Strahler $(n, d/2)$-universal tree. The vertices in these trees are encoded using tuples of binary strings. For working with these tuples, we introduce the following notation.

Definition 5.21. Given a tuple $\xi = (\xi_{2h-1}, \xi_{2h-3}, \dots, \xi_1)$ of binary strings, denote $\zeta(\xi)$ as the number of leading zeroes in ξ_{2h-1}. We also define $\zeta(\top) := -1$.

For a pair of tuples ξ, ξ' of binary strings, note that if $\xi \geq \xi'$, then $\zeta(\xi) \leq \zeta(\xi')$ by the lexicographic order on tuples.

Definition 5.22. Given a tuple $\xi = (\xi_{2h-1}, \xi_{2h-3}, \dots, \xi_1)$ of binary strings and an integer $\kappa \geq 0$, let ξ^κ be the tuple obtained by deleting κ leading zeroes from ξ_{2h-1}. If $\kappa > \zeta(\xi)$, then $\xi^\kappa := \top$. We also define $\top^\kappa := \top$.

5.4.4 Application to Succinct Universal Trees

Let T be a succinct (n,h)-universal tree. Recall that every leaf $\xi \in L(T)$ corresponds to an h-tuple of binary strings where $|\xi| \leq \lfloor \log n \rfloor$. First, we show that each $\mathcal{T}_j$ has a cover of size 1. Equivalently, each $(\mathcal{T}_j, \sqsubseteq)$ is a chain.

Lemma 5.23. *There exists a cover $\mathcal{C}$ of $\mathcal{T}$ such that $|\mathcal{C}_j| = 1$ for all $0 \leq j \leq h$.*

Proof. Fix $0 \leq j \leq h$ and pick two vertices $r_1, r_2 \in V(T)$ at depth $h - j$. Let T_1 and T_2 be the subtrees of T rooted at r_1 and r_2 respectively. Every leaf $\xi_1 \in L(T_1)$ and $\xi_2 \in L(T_2)$ corresponds to a j-tuple of binary strings where $|\xi_1| \leq \lfloor \log n \rfloor - |r_1|$ and $|\xi_2| \leq \lfloor \log n \rfloor - |r_2|$. Without loss of generality, assume that $|r_1| \geq |r_2|$. Then, the identity map from $V(T_1)$ to $V(T_2)$ is an order-preserving and injective homomorphism. Hence, $T_1 \sqsubseteq T_2$. $\square$

Since each subcover $\mathcal{C}_j$ of $\mathcal{C}$ consists of a single chain, we write $\mathcal{C}_j = \mathcal{T}_j$ and omit the superscript k. The subtrees in $\mathcal{T}_j$ are $T_{0,j} \sqsubset T_{1,j} \sqsubset \cdots \sqsubset T_{\lfloor \log n \rfloor, j}$. Observe that every leaf $\xi \in L(T_{i,j})$ corresponds to a j-tuple of binary strings where $|\xi| \leq i$.

Next, we give an efficient implementation of the Raise subroutine.

Lemma 5.24. *For a succinct $(n, d/2)$-universal tree T with cover $\mathcal{C} = (\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_{d/2})$, the* Raise$(\xi, i, j, k)$ *subroutine runs in $O(\log n \log d)$ time.*

Proof. We may assume that ξ is the smallest leaf in the subtree rooted at $\xi|_{2j}$. Otherwise, we can set it to the smallest leaf of the next subtree rooted at that depth using Tighten. Recall that the Tighten subroutine for T also runs in $O(\log n \log d)$ time. It follows that $\xi_{2j-1} \in \{0 \cdots 0, \varepsilon\}$ and $\xi_q = \varepsilon$ for all odd $q < 2j - 1$. If $|\xi_{2j-1}| \geq i$, then we simply return ξ. Otherwise, let $p \in [d]$ be the smallest even integer such that $\xi|_p$ has a child bigger than $\xi|_{p-1}$ with at most $\lfloor \log n \rfloor - i$ bits. If p does not exist, then we return $\top$. Otherwise, $p > 2j$. Let $r = \lfloor \log n \rfloor - i - |\xi|_{p-1}|$. There are two cases.

Case 1: $r > 0$. Return

$$(\xi_{d-1}, \ldots, \xi_{p-1} 1 \underbrace{0 \cdots 0}_{r-1}, \varepsilon, \ldots, \varepsilon, \underbrace{0 \cdots 0}_{i}, \varepsilon, \ldots, \varepsilon)$$

where the string of i zeroes is at index $2j - 1$.

Case 2: $r \leq 0$. Denote $\xi_{p-1} = b_1 b_2 \cdots b_\ell$ where $b_q \in \{0, 1\}$ for all $q \in [\ell]$. Note that $\ell \geq 1$ by our choice of p. Furthermore, there exists a largest $t \in [\ell]$ such that $b_t = 0$ and $r' := r + \ell - t + 1 \geq 0$. Then,

- If $p = 2j + 2$, return

$$(\xi_{d-1}, \ldots, \xi_{p+1}, b_1 \cdots b_{t-1}, \underbrace{0 \cdots 0}_{i+r'}, \varepsilon, \ldots, \varepsilon)$$

- If $p > 2j + 2$, return

$$(\xi_{d-1}, \ldots, \xi_{p+1}, b_1 \cdots b_{t-1}, \underbrace{0 \cdots 0}_{r'}, \varepsilon, \ldots, \varepsilon, \underbrace{0 \cdots 0}_{i}, \varepsilon, \ldots, \varepsilon)$$

  where the string of i zeroes is at index $2j - 1$.

□

Due to the structure of succinct universal trees, the running time of Algorithm 11 given in Theorem 5.20 can be improved. The following lemma yields a faster method for computing arc costs for the auxiliary digraph D_τ. The key observation is that for any pair of trees in a chain $\mathcal{T}_j$, the smaller tree can be obtained from the larger tree by deleting vertices in decreasing lexicographic order. For example, a succinct (3,2)-universal tree can be obtained from a succinct (7,2)-universal tree by deleting vertices whose first component does not contain a leading zero (compare Figures 5.3 and 5.7).

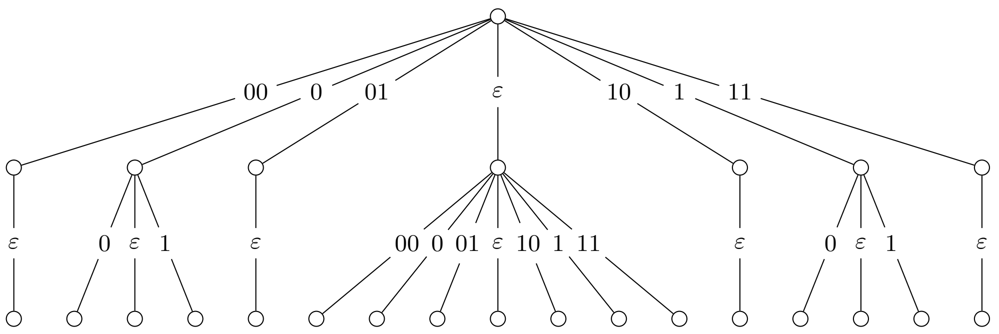

Fig. 5.7 The succinct $(7,2)$-universal tree

Lemma 5.25. *Given integers $0 \leq i_1 \leq i_2$ and $j \geq 0$, let $\nu_1 : V \to \bar{L}(T_{i_1,j})$ and $\nu_2 : V \to \bar{L}(T_{i_2,j})$ be node labelings such that $\nu_1(u) = \nu_2(u)^{i_2-i_1}$ for all $u \in V$. For any arc $vw \in E_\tau$ where $\pi(v) < 2j$, we have* $\operatorname{drop}(\nu_1, vw) = \operatorname{drop}(\nu_2, vw)^{i_2-i_1}$.

Proof. Let $\xi_1 = \operatorname{drop}(\nu_1, vw)$ and $\xi_2 = \operatorname{drop}(\nu_2, vw)$. First, assume that vw is violated with respect to ν_1. Then, $\nu_1(v) \neq \top$, which implies that $\nu_2(v) \neq \top$. In particular, $\zeta(\nu_2(v)) \geq i_2 - i_1$. We claim that vw is also violated with respect to ν_2. This is clear if $\nu_1(w) \neq \top$. If $\nu_1(w) = \top$, then $\zeta(\nu_2(w)) < i_2 - i_1$. As $\nu_2(v)|_{2j-1} < \nu_2(w)|_{2j-1}$ and $\pi(v) < 2j$, the arc vw is indeed violated with respect to ν_2. Hence, $\xi_1 = \nu_1(v)$ and $\xi_2 = \nu_2(v)$.

Next, assume that vw is not violated with respect to ν_1. If $\nu_1(w) \neq \top$, then vw is also not violated with respect to ν_2. It is easy to verify that $\xi_1 = \xi_2^{i_2-i_1}$. On the other hand, if $\nu_1(w) = \top$, then $\nu_1(v) = \top$. So, we have $\zeta(\nu_2(v)) < i_2 - i_1$ and $\zeta(\nu_2(w)) < i_2 - i_1$. Since $\pi(v) < 2j$, we also have $\zeta(\xi_2) < i_2 - i_1$. Thus, $\xi_1 = \top = \xi_2^{i_2-i_1}$ as required. $\square$

Pick a node $w \in V(H)$ and let $j = \pi(w)/2$. For each $0 \leq i \leq \lfloor \log n \rfloor$, let $\nu_i : V(J_w) \to \bar{L}(T_{i,j})$ be the node labeling defined by $\nu_i(w) := \min L(T_{i,j})$ and $\bar{\nu}(u) := \top$ for all $u \neq w$. To compute the cost of incoming arcs $\delta^-_{D_\tau}(w)$, we only need to run Algorithm 10 on J_w once, with the input node labeling $\nu_{\lfloor \log n \rfloor}$. Let ν be the returned node labeling. Note that $\nu(u) \neq \top$ for all $u \in V(J_w)$ because $T_{\lfloor \log n \rfloor, j}$ is an (n,j)-universal tree. Without loss of generality, we may assume that Algorithm 10 is run on the subgraph $J_w \setminus \delta^+(w)$ instead, as $\nu(w) = \nu_{\lfloor \log n \rfloor}(w)$. Then, by Lemma 5.25, for any $0 \leq i \leq \lfloor \log n \rfloor$, $\nu^{\lfloor \log n \rfloor - i}$ is the node labeling returned by Bellman–Ford on J_w with input node labeling ν_i. Hence, the cost of each arc $vw \in \delta^-_{D_\tau}(w)$ is set as $c(vw) := \lfloor \log n \rfloor - \max_{u \in N^+_{J_w}(v)} \zeta(\nu(u))$.

We are ready to prove the running time of Algorithm 11 for succinct universal trees.

Theorem 5.26. *For a succinct $(n, d/2)$-universal tree T with cover $\mathcal{C} = (\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_{d/2})$, Algorithm 11 runs in $O(mn^2 \log n \log d)$ time.*

Proof. By Lemma 5.23, we have $|\mathcal{C}_j| = 1$ for all $0 \leq j \leq h$. Computing arc costs for the auxiliary digraph D_τ takes $O(mn^2\gamma(T))$ time. Hence, the running time of Algorithm 11 becomes $O(n\rho(T,\mathcal{C})+mn^2\gamma(T))$. The result then follows from $\gamma(T) = O(\log n \log d) = \rho(T,\mathcal{C})$, where the latter equality is due to Lemma 5.24. □

5.4.5 Application to Succinct Strahler Universal Trees

In this subsection, we apply Algorithm 11 to succinct Strahler universal trees. Let us start by introducing the necessary definitions. The *Strahler number* of a rooted tree T is the largest height of a perfect binary tree that is a minor of T. For example, a perfect (ℓ, h)-universal tree has Strahler number 0 if $\ell = 1$, and h otherwise.

Definition 5.27. A *g-Strahler (ℓ, h)-universal tree* is an ordered tree T' such that $T \sqsubseteq T'$ for every ordered tree T of Strahler number at most g, height at most h, and with at most ℓ leaves.

In the definition above, we may assume that $g \le \min(h, \lfloor \log \ell \rfloor)$. Daviaud et al. [43] constructed a g-Strahler (ℓ, h)-universal tree with $\ell^{O(1)}(h/g)^g$ leaves. Note that this is quasipolynomial in ℓ and h by our previous remark. We call it a *succinct g-Strahler (ℓ, h)-universal tree*. Every leaf $\xi = (\xi_{2h-1}, \xi_{2h-3}, \dots, \xi_1)$ in this tree corresponds to an h-tuple of binary strings which satisfies the following three properties:

1. There are g nonempty bit strings, i.e. $|\{i : \xi_i \neq \varepsilon\}| = g$;
2. The total number of bits $|\xi|$ is at most $g + \lfloor \log \ell \rfloor$;
3. For each odd $i \in [2h]$,
   (a) If there are $f < g$ nonempty bit strings in $\xi|_i$ and $|\xi|_i| = f + \lfloor \log \ell \rfloor$, then $\xi_i = 0$.
   (b) If $x_j \neq \varepsilon$ for all odd $j \in [i]$, then x_j starts with 0 for all odd $j \in [i]$.

This is the construction of $\mathcal{B}^k_{t,h}$ in [43, Definition 19]. In each string ξ_i, the first bit is called the *leading bit*, while the remaining bits are the *non-leading bits*. Properties 1 and 2 imply that ξ contains exactly g leading bits and at most $\lfloor \log \ell \rfloor$ non-leading bits. Observe that if $g = h$, then the tree is identical to a succinct (ℓ, h)-universal tree. Indeed, one can arrive at the encoding of Jurdziński and Lazić [88] by removing the leading zero in every string.

Let T be a succinct g-Strahler (ℓ, h)-universal tree. Let $v \in V(T)$ be a vertex at depth $h - j$ for some $0 \le j \le h$. If v has k nonempty strings and i non-leading bits, then the subtree rooted at v is a succinct $(g-k)$-Strahler $(2^{\lfloor \log \ell \rfloor - i}, j)$-universal tree. Note that if $i = \lfloor \log \ell \rfloor$, then the subtree is a path. This fact is actually independent of k. Indeed, varying k only yields different encodings of the same path.

The crucial fact for obtaining a small cover is that the subtrees of T with fixed height j and fixed Strahler number k form a chain. This leads to the following statement.

Lemma 5.28. *There exists a cover $\mathcal{C}$ of $\mathcal{T}$ such that $|\mathcal{C}_j| \le g$ for all $0 \le j \le h$.*

Proof. Fix a $0 \leq j \leq h$. For each $0 \leq k \leq g$, let $\mathcal{C}_j^k$ be the set of k-Strahler $(\cdot, j)$-universal trees in $\mathcal{T}_j$. Then, $\cup_k \mathcal{C}_j^k = \mathcal{T}_j$. Note that $\mathcal{C}_j^k \neq \emptyset$ if and only if $\max(0, g-j) \leq k \leq \min(j, g)$. It is left to show that $(\mathcal{C}_j^k, \sqsubseteq)$ is a chain for all k. Fix a k and pick two vertices $r_1, r_2 \in V(T)$ such that they each have $g-k$ nonempty bit strings. Let T_1 and T_2 be the subtrees of T rooted at r_1 and r_2 respectively. Observe that T_1 is a succinct k-Strahler $(\lfloor \log n \rfloor - |r_1| + g - k, j)$-universal tree. Similarly, T_2 is a succinct k-Strahler $(\lfloor \log n \rfloor - |r_2| + g - k, j)$-universal tree. Without loss of generality, assume that $|r_1| \geq |r_2|$. Then, the identity map from $V(T_1)$ to $V(T_2)$ is an order-preserving and injective homomorphism. Hence, $T_1 \sqsubseteq T_2$. □

Let $\mathcal{C}$ be the cover given in the proof of Lemma 5.28, i.e. $\mathcal{C}_j = (\mathcal{C}_j^0, \mathcal{C}_j^1, \dots, \mathcal{C}_j^g)$ for all $0 \leq j \leq h$. Note that $T_{0,j}^k = T_{0,j}^{k'}$ for all k, k' where $\mathcal{C}_j^k, \mathcal{C}_j^{k'} \neq \emptyset$. The next lemma shows that the Raise subroutine can be implemented efficiently using this cover.

Lemma 5.29. *For a succinct g-Strahler $(n, d/2)$-universal tree T with cover $\mathcal{C}$, the* Raise(ξ, i, j, k) *subroutine runs in $O(\log n \log d)$ time.*

Proof. We may assume that ξ is the smallest leaf in the subtree of T rooted at $\xi|_{\pi(v)}$. Otherwise, we can set it to the smallest leaf of the next subtree rooted at that depth using Tighten. Recall that the Tighten subroutine for T also runs in $O(\log n \log d)$ time. It follows that $\xi_{2j-1} \in \{0 \cdots 0, \varepsilon\}$ and $\xi_q \in \{0, \varepsilon\}$ for all $q < 2j-1$. If $i = 0$, then we output ξ because $T_{0,j}^k$ is identical for all $0 \leq k < |\mathcal{C}_j|$. So, let us assume that $i > 0$ from now on.

If there are at least i non-leading bits in ξ_{2j-1} and exactly k non-empty strings among $\xi_{2j-1}, \dots, \xi_1$, then we simply return ξ. Otherwise, let $p \in [d]$ be the smallest even integer such that $\xi|_p$ has a child bigger than $\xi|_{p-1}$ with at most $\lfloor \log n \rfloor - i$ non-leading bits and exactly x nonempty strings for some $g - k - (p/2 - j) + 1 \leq x \leq g - k$. If p does not exist, then we return $\top$. Otherwise, $p > 2j$.

Our goal is to increase ξ minimally such that it can accommodate a label from $T_{i',j}^k$ for some $i' \geq i$, using only the components after $\xi|_p$. Let z be the number of non-empty strings in $\xi|_{p-1}$ and let y be the number of non-leading bits in $\xi|_{p-1}$. We set $s = g - k - z$, representing the discrepancy on the number of nonempty bit strings (Property 1). We also set $r = \lfloor \log n \rfloor - i - y$, representing the discrepancy on the number of non-leading bits (Property 2). We split the remaining analysis into cases based on the emptiness of ξ_{p-1} and the signs of r, s.

Case 1: $\xi_{p-1} = \varepsilon$. Note that $y \leq \lfloor \log n \rfloor - i$ and $z < g - k$. So, $r \geq 0$ and $s > 0$. Return

$$(\xi_{d-1}, \dots, \xi_{p+1}, 1\underbrace{0\cdots0}_{r}, \underbrace{0, \dots, 0}_{s-1}, \varepsilon, \dots, \varepsilon, \underbrace{0\cdots0}_{i+1}, \underbrace{0, \dots, 0}_{k-1}, \varepsilon, \dots, \varepsilon)$$

where the string of $i+1$ zeroes is at index $2j-1$. Property 3b holds because there is at least one empty string among the last $p/2$ components. As all the other properties are also fulfilled by construction, this is a valid tuple encoding a leaf.

Case 2: $\xi_{p-1} \neq \varepsilon$ and $s < 0$. Note that $s = -1$ and ξ_{p-1} has a leading zero due to our choice of p. Let t be the number of non-leading bits in ξ_{p-1}. Then, $y - t \leq \lfloor \log n \rfloor - i$, which implies that $r + t \geq 0$. Return

$$(\xi_{d-1}, \ldots, \xi_{p+1}, \varepsilon, \varepsilon, \ldots, \varepsilon, \underbrace{0 \cdots 0}_{i+1+r+t}, \underbrace{0, \ldots, 0}_{k-1}, \varepsilon, \ldots, \varepsilon)$$

where the string of $i + 1 + r + t$ zeroes is at index $2j - 1$.

Case 3: $\xi_{p-1} \neq \varepsilon$, $r > 0$ and $s \geq 0$. Return

$$(\xi_{d-1}, \ldots, \xi_{p-1}1\underbrace{0 \cdots 0}_{r-1}, \underbrace{0, \ldots, 0}_{s}, \varepsilon, \ldots, \varepsilon, \underbrace{0 \cdots 0}_{i+1}, \underbrace{0, \ldots, 0}_{k-1}, \varepsilon, \ldots, \varepsilon)$$

where the string of $i + 1$ zeroes is at index $2j - 1$.

Case 4: $\xi_{p-1} \neq \varepsilon$, $r \leq 0$ and $s \geq 0$. Denote $\xi_{p-1} = b_1 b_2 \cdots b_\ell$ where $b_q \in \{0, 1\}$ for all $q \in [\ell]$. By our choice of p, there exists a largest $t \in [\ell]$ such that $b_t = 0$ and $r' := r + \ell - \max(t - 1, 1) \geq 0$. Then,

- If $t = 1$, return

$$(\xi_{d-1}, \ldots, \xi_{p+1}, \varepsilon, \underbrace{0 \cdots 0}_{r'+1}, \underbrace{0, \ldots, 0}_{s}, \varepsilon, \ldots, \varepsilon, \underbrace{0 \cdots 0}_{i+1}, \underbrace{0, \ldots, 0}_{k-1}, \varepsilon, \ldots, \varepsilon)$$

  where the string of $i + 1$ zeroes is at index $2j - 1$.

- If $s = 0$ and $t > 1$, return

$$(\xi_{d-1}, \ldots, \xi_{p+1}, b_1 \cdots b_{t-1}, \varepsilon, \ldots, \varepsilon, \underbrace{0 \cdots 0}_{i+1+r'}, \underbrace{0, \ldots, 0}_{k-1}, \varepsilon, \ldots, \varepsilon)$$

  where the string of $i + 1 + r'$ zeroes is at index $2j - 1$.

- If $s > 0$ and $t > 1$, return

$$(\xi_{d-1}, \ldots, \xi_{p+1}, b_1 \cdots b_{t-1}, \underbrace{0 \cdots 0}_{r'+1}, \underbrace{0, \ldots, 0}_{s-1}, \varepsilon, \ldots, \varepsilon, \underbrace{0 \cdots 0}_{i+1}, \underbrace{0, \ldots, 0}_{k-1}, \varepsilon, \ldots, \varepsilon)$$

  where the string of $i + 1$ zeroes is at index $2j - 1$.

□

We are ready to prove the running time of Algorithm 11 for succinct Strahler universal trees. Recall that $g \leq \log n$.

Theorem 5.30. *For a succinct g-Strahler $(n, d/2)$-universal tree T with cover $\mathcal{C}$, Algorithm 11 runs in $O(mn^2 g \log^2 n \log d)$ time.*

Proof. By Lemma 5.28, we have $|\mathcal{C}_j| \leq g$ for all $0 \leq j \leq h$. Furthermore, $|\mathcal{C}_j^k| \leq \log n$ for all $0 \leq j \leq h$ and $0 \leq k < |\mathcal{C}_j|$. As $\gamma(T) = O(\log n \log d) = \rho(T, \mathcal{C})$, where the latter equality is due to Lemma 5.29, the result follows from Theorem 5.20. □

5.5 Label-Setting Method for Computing the Least Fixed Point

The epitome of a label-setting algorithm is none other than Dijkstra's algorithm [47]. In this section, we develop its analogue for ordered trees. Despite having a faster running time than Algorithm 10, it requires prior knowledge of $\mu^{\mathcal{G}_\tau^\uparrow}(v)$ for all $v \in B(G_\tau)$ in order to compute $\mu^{\mathcal{G}_\tau^\uparrow}$. Nevertheless, we demonstrate its applicability to perfect universal trees.

The algorithm takes as input a 1-player game (G_τ, π) for Even and a node labeling $\nu : V \to \bar{L}(T)$ from some ordered tree T. During its execution, the node labeling $\nu : V \to \bar{L}(T)$ is updated. The algorithm also maintains a growing node set $S \subseteq V$ such that $\nu(v)$ remains fixed for all $v \in S$. For the sake of brevity, let us denote $H := G_\tau \setminus B(G_\tau)$.

In every iteration, a new node is added to S, whose label is fixed. To determine this node, we introduce a label function Φ, which remains fixed throughout the algorithm. It encodes a family of topological orders in H induced by the even priorities. The node is then selected using a *potential* function Φ^ν, defined based on the labels Φ and ν. This selection criteria accounts for the fact that the representation of a parity game as a mean payoff game could have negative arc weights.

To describe Φ, we define a family of functions Φ_p, parametrized by the even priorities in H. For an even $p \in [d]$, Φ_p encodes the topological order of nodes in the subgraph H_p. Recall that H_p is the subgraph of H induced by the nodes with priority at most p. Formally, $\Phi_p : V(H) \to \mathbb{Z}_+$ is any function which satisfies the following three properties:

- $\Phi_p(v) = 0$ if and only if $\pi(v) > p$;
- $\Phi_p(v) \leq \Phi_p(w)$ if v can be reached from w in H_p;
- $\Phi_p(v) = \Phi_p(w) > 0$ if and only if v and w are strongly connected in H_p.

The label function $\Phi : V(H) \to \mathbb{Z}_+^{d/2}$ is then defined as

$$\Phi(v) := (\Phi_d(v), \Phi_{d-2}(v), \ldots, \Phi_2(v)).$$

A linear order on $\Phi(v)$ is obtained by extending the linear order of its components lexicographically.

Remark 5.31. Given a pair of nodes v and w, comparing $\Phi(v)$ and $\Phi(w)$ amounts to finding the largest $p \in [d]$ such that $\Phi_p(v) \neq \Phi_p(w)$. Observe that if $\Phi_q(v) = \Phi_q(w) > 0$ for some

$q \in [d]$, then $\Phi_r(v) = \Phi_r(w)$ for all $r \geq q$. On the other hand, if $\Phi_q(v) = \Phi_q(w) = 0$, then $\Phi_r(v) = \Phi_r(w) = 0$ for all $r \leq q$. Hence, such a p can be computed in $O(\log d)$ time via binary search.

Given a node labeling $\nu : V \to \bar{L}(T)$, the potential function $\Phi^\nu : V(H) \to (\mathbb{Z}_+^{d/2} \times L(T)) \cup \{\infty\}$ is obtained by interlacing the components of Φ and ν in the following way

$$\Phi^\nu(v) := \begin{cases} (\Phi_d(v), \nu(v)_{d-1}, \ldots, \Phi_2(v), \nu(v)_1), & \text{if } \nu(v) \neq \top. \\ \infty & \text{otherwise.} \end{cases}$$

A linear order on $\Phi^\nu(v)$ is acquired by extending the linear order of its components lexicographically. For any $p \in [d]$, the *p-truncation* of $\Phi(v)$ and $\Phi^\nu(v)$, denoted $\Phi(v)|_p$ and $\Phi^\nu(v)|_p$ respectively, are obtained by deleting the components with index less than p, with the convention $\infty|_p := \infty$.

We are ready to state Dijkstra's algorithm for ordered trees (Algorithm 12). First, it initializes the node set S as $B(G_\tau)$, and sets $\nu(v) := \top$ for all $v \in V \setminus S$. Then, for each even $p \in [d]$, it computes the topological order Φ_p by running Tarjan's SCCs algorithm on H_p. Next, ν is updated by dropping the tail labels of the incoming arcs $\delta^-(S)$. At the start of every iteration, the algorithm selects a node u with minimum potential $\Phi^\nu(u)$ among all the nodes in $V \setminus S$ (ties are broken arbitrarily). Then, it adds u to S and updates ν by dropping the tail labels of the incoming arcs $\delta^-(u) \cap \delta^-(S)$. The algorithm terminates when $S = V$.

Algorithm 12: Dijkstra

```
   Input  : 1-player game $(G_\tau, \pi)$ for Even, node labeling $\nu : V \to \bar{L}(T)$ from an
            ordered tree $T$
   Output : Node labeling from $T$
1  $S \leftarrow B(G_\tau)$
2  $\nu(v) \leftarrow \top$ for all $v \in V \setminus S$
3  Compute $\Phi_p$ for all even $p \in [d]$                ▷ Using Tarjan's SCCs algorithm
4  foreach $vw \in \delta^-(S)$ do
5      $\nu(v) \leftarrow \mathrm{drop}(\nu, vw)$
6  while $S \subsetneq V$ do
7      $u \in \arg\min_{v \in V \setminus S} \Phi^\nu(v)$                ▷ Break ties arbitrarily
8      $S \leftarrow S \cup \{u\}$
9      foreach $vu \in \delta^-(u)$ where $v \notin S$ do
10         $\nu(v) \leftarrow \mathrm{drop}(\nu, vu)$
11 return $\nu$
```

An efficient implementation of Dijkstra's algorithm using Fibonacci heaps was given by Fredman and Tarjan [66]. Its running time is $O(m + n \log n)$ when the keys in the heap are real numbers, assuming that each elementary operation on the reals takes constant time. In our setting, the keys are the node potentials Φ^ν. Since computing $\mathrm{drop}(\nu, e)$ takes $\gamma(T)$ time while comparing the potential of two nodes takes $\gamma(T) + \log d$ time, their result translates to $O(\gamma(T)m + (\gamma(T) + \log d)n \log n)$ time here. We also need to compute the base nodes $B(G_\tau)$ and the topological orders Φ_p, which take $O(dm)$ time. Hence, the total running time of Algorithm 12 is $O((\gamma(T) + d)m + (\gamma(T) + \log d)n \log n)$.

Before proving the algorithm's correctness, we illustrate an important connection between the potential function Φ^ν and even cycles in H.

Lemma 5.32. *Let $vw \in E(H)$ be a tight arc with respect to some node labeling ν. If $\Phi^\nu(v) < \Phi^\nu(w)$, then there exists an even cycle C in H such that $vw \in E(C)$ and $\pi(C) = \pi(v)$.*

Proof. First, note that $\nu(v) \neq \top$ because $\Phi^\nu(v) < \Phi^\nu(w)$. This in turn implies that $\nu(w) \neq \top$ as vw is tight with respect to ν. Next, since v can reach w in H, we have $\Phi(v)|_{\pi(v)} \geq \Phi(w)|_{\pi(v)}$. We also have $\nu(v)|_{\pi(v)} \geq \nu(w)|_{\pi(v)}$ due to the tightness of vw. Then, combining these two inequalities yield $\Phi^\nu(v)|_{\pi(v)} \geq \Phi^\nu(w)|_{\pi(v)}$. In fact, we get $\Phi^\nu(v)|_{\pi(v)} = \Phi^\nu(w)|_{\pi(v)}$ because $\Phi^\nu(v) < \Phi^\nu(w)$. Since vw is tight, we conclude that $\pi(v)$ is even. It follows that $0 < \Phi_{\pi(v)}(v) = \Phi_{\pi(v)}(w)$, which implies that v and w are strongly connected in $H_{\pi(v)}$. Thus, there exists a cycle C in $H_{\pi(v)}$ such that $vw \in E(C)$. Clearly, $\pi(C) = \pi(v)$. □

The next theorem shows that Algorithm 12 returns the pointwise minimal node labeling which is 'almost' feasible in G_τ. The key observation is that the sequence of node potentials admitted to S during the algorithm is monotonically nondecreasing.

Theorem 5.33. *Given initial node labeling ν, Algorithm 12 returns the pointwise minimal node labeling ν^* which is feasible in $G_\tau \setminus \cup_{v \in B(G_\tau)} \delta^+(v)$ and satisfies $\nu^*(v) = \nu(v)$ for all $v \in B(G_\tau)$.*

Proof. For every $i \geq 1$, let ν_i be the node labeling at the start of iteration i. So, the algorithm returns ν_n. To show the feasibility of ν_n, we prove that ν_i is feasible in $G_\tau \setminus \cup_{v \in B(G_\tau)} \delta^+(v)$ by induction on $i \geq 1$. The base case $i = 1$ is clearly true, and the inductive step is straightforward. Furthermore, it is easy to see that $\nu_n(v) = \nu(v)$ for all $v \in B(G_\tau)$ due to our initialization.

For every $i \geq 1$, let u_i be the node added to S in iteration i. We start by showing that $\Phi^{\nu_n}(u_i) \geq \Phi^{\nu_n}(u_{i-1})$ for all $i > 1$. For the purpose of contradiction, suppose that $\Phi^{\nu_n}(u_i) < \Phi^{\nu_n}(u_{i-1})$ for some $i > 1$. We claim that $u_i u_{i-1} \in E(H)$ and it is tight with respect to ν_i. Suppose otherwise for a contradiction. Then, $\nu_i(u_i) = \nu_{i-1}(u_i)$. As $\nu_n(u_i) = \nu_i(u_i)$ and $\nu_n(u_{i-1}) = \nu_{i-1}(u_{i-1})$, it follows that $\Phi^{\nu_{i-1}}(u_i) < \Phi^{\nu_{i-1}}(u_{i-1})$. This is a contradiction because u_i would have been added to S in iteration $i - 1$ instead of u_{i-1}. By the claim above and Lemma 5.32, there exists an even cycle C in H. However, $\Pi(C) \subseteq B(G_\tau)$, which is a contradiction.

Next, we show that there are no loose arcs in $G_\tau \setminus \cup_{v \in B(G_\tau)} \delta^+(v)$ with respect to ν_n. It suffices to prove that $u_i u_j$ is not loose with respect to ν_n for all $1 \leq i < j$ where $u_i u_j \in E(H)$.

For the purpose of contradiction, let $u_iu_j \in E(H)$ be a loose arc with respect to ν_n. Note that $\nu_n(u_i) \neq \top$, as otherwise it would imply $\nu_n(u_j) = \top$ because $\Phi^{\nu_n}(u_i) \leq \Phi^{\nu_n}(u_j)$. Since the label $\nu_n(u_i)$ was given by TIGHTEN, it is the smallest leaf in the subtree of T rooted at $\nu_n(u_i)|_{\pi(u_i)}$. Therefore, $\nu_n(u_i)|_{\pi(u_i)} > \nu_n(u_j)|_{\pi(u_i)}$. Let p be the smallest even integer such that $p \geq \pi(u_i)$. Then, $\Phi(u_i)|_p \geq \Phi(u_j)|_p$ because either $\pi(u_j) > p$ or $u_iu_j \in E(H_p)$. However, these two inequalities yield $\Phi^{\nu_n}(u_i) > \Phi^{\nu_n}(u_j)$, which is a contradiction.

It is left to show the pointwise minimality of ν_n. For the purpose of contradiction, let $\nu' : V \to \bar{L}(T)$ be a node labeling feasible in $G_\tau \setminus \cup_{v \in B(G_\tau)} \delta^+(v)$ such that $\nu'(v) = \nu(v)$ for all $v \in B(G_\tau)$ and $\nu'(u) < \nu_n(u)$ for some $u \in V(H)$. Note that $\nu'(u) \neq \top$. From the definition of feasibility, there exists a strategy σ for Even such that $G_{\sigma\tau} \setminus \cup_{v \in B(G_\tau)} \delta^+(v)$ does not contain violated arcs with respect to ν'. In this subgraph, u can reach a node in $B(G_\tau)$. Indeed, if it reaches a cycle C, then C is even by the Cycle Lemma because $\nu'(v) \neq \top$ for all $v \in V(C)$. So, $\Pi(C) \subseteq B(G_\tau)$. Let P be a u-w path in this subgraph for some $w \in B(G_\tau)$. Since $\nu_n(u) > \nu'(u)$ and $\nu_n(w) = \nu'(w)$, there exists a loose arc in P with respect to ν_n. We have reached a contradiction. □

Consequently, if we can determine $\mu^{\mathcal{G}_\tau^\uparrow}(v)$ for all $v \in B(G_\tau)$ beforehand, then we can use Algorithm 12 to compute $\mu^{\mathcal{G}_\tau^\uparrow}$.

Corollary 5.34. *Given initial node labeling ν where $\nu(v) = \mu^{\mathcal{G}_\tau^\uparrow}(v)$ for all $v \in B(G_\tau)$, Algorithm 12 returns $\mu^{\mathcal{G}_\tau^\uparrow}$.*

Proof. Let ν be the node labeling returned by Algorithm 12. By Theorem 5.33, we have $\nu \leq \mu^{\mathcal{G}_\tau^\uparrow}$. Since $\nu(v) = \mu^{\mathcal{G}_\tau^\uparrow}(v)$ for all $v \in B(G_\tau)$, this implies that ν is feasible in G_τ. To show that $\nu \geq \mu^{\mathcal{G}_\tau^\uparrow}$, it suffices to prove that $\nu \geq \mu$ due to the pointwise minimality of $\mu^{\mathcal{G}_\tau^\uparrow}$. For the purpose of contradiction, suppose that $\nu(u) < \mu(u)$ for some $u \in V(H)$. Let σ be a strategy for Even such that $G_{\sigma\tau}$ does not contain violated arcs with respect to ν. Since $\nu(u) \neq \top$, there exists a u-w path P in $G_{\sigma\tau}$ for some $w \in B(G_\tau)$. As $\mu(u) > \nu(u)$ and $\mu(w) \leq \mu^{\mathcal{G}_\tau^\uparrow}(w) = \nu(w)$, there exists a loose arc in P with respect to μ. This contradicts our assumption on μ. □

5.5.1 Application to Perfect Universal Trees

In this subsection, we show that Algorithm 12 yields a faster method for computing $\mu^{\mathcal{G}_\tau^\uparrow}$ than Algorithm 11, when a perfect universal tree is used.

Theorem 5.35. *Given a 1-player game (G_τ, π) for Even, let $\mu : V(G_\tau) \to \bar{L}(T)$ be a node labeling with no loose arc in G_τ. If T is a perfect $(n, d/2)$-universal tree, then $\mu^{\mathcal{G}_\tau^\uparrow}$ can be computed in $O(d(m + n \log n))$ time.*

Proof. First, we show that for every $v \in V$, we may assume that $\mu(v)$ is either $\top$ or the smallest leaf in the subtree of T rooted at $\mu(v)|_{\pi(v)}$. If a node w violates this condition, then we know that $\mu^{\mathcal{G}_\tau^\uparrow}(w) > \mu(w)$ because there are no loose arcs in G_τ with respect to μ. Hence, we can set $\mu(w)$ as the smallest leaf of the next subtree rooted at that depth using Tighten. Recall that the Tighten subroutine for T takes $O(d)$ time. In the worst case, we incur $O(dn)$ extra time.

By Corollary 5.34, it suffices to prove that $\mu(v) = \mu^{\mathcal{G}_\tau^\uparrow}(v)$ for all $v \in B(G_\tau)$. Fix a base node $w \in B(G_\tau)$. We may assume that $\mu(w) \neq \top$. Let C be a cycle dominated by w in G_τ, and consider the path $P := C \setminus wu$ where $wu \in E(C)$. Let $\bar{\mu} : V \to \bar{L}(T)$ be a node labeling such that $\bar{\mu}(v) = \top$ for all $v \notin V(P)$, $\bar{\mu}(w) = \mu(w)$ and P is tight with respect to $\bar{\mu}$. Then, $\bar{\mu}$ is feasible in $G_\tau \setminus \delta^+(w)$. Moreover, $\bar{\mu} \geq \mu$ because there are no loose arcs in P with respect to μ. Now, recall that $\mu(w)$ is the smallest leaf in the subtree of T rooted at $\mu(w)|_{\pi(w)}$. Since $\pi(v) \leq \pi(w)$ for all $v \in V(P)$, we have $\bar{\mu}(u)|_{\pi(w)} = \bar{\mu}(w)|_{\pi(w)}$ because $|V(P)| \leq n$. As $\pi(w)$ is even, the arc wu is tight with respect to $\bar{\mu}$. It follows that $\bar{\mu}$ is feasible in G_τ. Thus, $\mu \leq \mu^{\mathcal{G}_\tau^\uparrow} \leq \bar{\mu}$. In particular, $\mu(w) = \mu^{\mathcal{G}_\tau^\uparrow}(w)$. $\square$

Chapter 6

Conclusions and Future Work

An Accelerated Newton–Dinkelbach Method

In Chapter 2, we have presented an accelerated version of the Newton–Dinkelbach method for univariate concave functions, and illustrated its utility on three application domains. For linear fractional combinatorial optimization, we obtain an improved $O(m \log m)$ iteration bound. For 2VPI LP feasibility, we get a strongly polynomial label-correcting algorithm which runs in $O(mn)$ iterations. Finally, the method yields a simplified analysis of the parametric submodular function minimization result by Goemans et al. [71].

The key idea is to analyze the Newton–Dinkelbach method using the Bregman divergence; previous work [125, 156] analyzed the gradient and function value of the iterates. With the look-ahead step, we show that the Bregman divergence halves every two iterations. The Bregman divergence has a useful interpretation in terms of a 'modified' cost function, which allows us to derive convergence bounds for various problems. We expect that this accelerated method and its analysis find more applications in other fractional/parametric optimization problems.

For 2VPI LP feasibility, every iteration of the Newton–Dinkelbach method takes $O(mn)$ time, which results in a total running time of $O(m^2n^2)$ for the overall label-correcting algorithm. We do not know whether our analysis is tight; it may be possible to amortize over the $O(mn)$ iterations. It is interesting to see whether the running time can be lowered to match the $O(mn^2 \log m)$ bound of Hochbaum and Naor [80].

Circuit Diameter Bounds for Polyhedra

In Chapter 3, we have derived circuit diameter bounds for polyhedra in standard equality form (P) and capacitated form (Cap-P). For a constraint matrix $A \in \mathbb{R}^{m\times n}$, our bounds are polynomial in m, n and $\log(\kappa_A)$, where κ_A is the circuit imbalance measure of A. It is independent of the costs $c \in \mathbb{R}^n$, the RHS vector $b \in \mathbb{R}^m$, and the capacities $u \in \mathbb{R}^n$.

Since $\log(\kappa_A)$ is polynomial in the encoding size of A, this yields a weakly polynomial circuit diameter bound. In particular, it is strongly polynomial if all the entries in A have polynomially bounded encoding length in n. Moreover, we have developed circuit-augmentation algorithms for LP with similar iteration complexity.

The key idea of the bounds is to analyze the following procedure. In every iteration, we first decompose the difference between the target vertex x^* and the current point $x^{(t)}$ into conformal circuits. Then, we pick one which yields the largest gain per unit step, and move along it maximally. This ensures geometric decay in the 'distance' to the target vertex. The analysis involves considering coordinates of x^* which are large with respect to the current 'distance', and showing that this set grows monotonically every $O(n \log(\kappa_A + n)$ iterations.

Since we now have a weakly polynomial circuit diameter bound, it is natural to wonder whether the combinatorial diameter also admits a weakly polynomial bound. Another interesting line of research is to derive a strongly polynomial bound on the circuit diameter. In particular, is the circuit analogue of Hirsch conjecture true?

Correlation Gap Bounds for Matroids

In Chapter 4, we have derived an improved lower bound on the correlation gap of matroid rank functions, as parametrized by the rank and girth of the matroid. Parametrizing by the rank or the girth alone does not lead to a correlation gap better than $1 - 1/e$, the lower bound that applies to all monotone submodular functions. We have also shown that for any matroid, the smallest correlation gap of its weighted rank function is attained under uniform weights. Consequently, our bound applies to weighted matroid rank functions as well.

The key ideas are as follows. Let $r : 2^{[n]} \to \{0, 1, \dots, n\}$ be the rank function of a matroid with girth γ. We first show that the correlation gap is realized inside the independent set polytope, which allows us to replace the concave extension $\hat{r}(x)$ by $\mathbb{1}^\top x$. To lower bound the multilinear extension $R(x)$, we split the rank function into $r = g + h$, where g is the rank function of a uniform matroid of rank-$(\gamma - 1)$. Then, $R = G + H$ by linearity of expectation. Since $G(x) \geq G(\frac{\mathbb{1}^\top x}{n} \cdot \mathbb{1})$, this lends itself to an analysis via the binomial distribution. On the other hand, $H(x)$ is analyzed by extending the Poisson clock argument of Calinescu et al. [26] to handle periods with zero expected marginal gain.

Since our lower and upper bounds on the correlation gap do not match, an obvious next step is to derive tighter bounds with respect to these two parameters. It is also interesting to identify other matroid parameters which better capture its correlation gap.

Strategy Iteration with Universal Trees

In Chapter 5, we have presented a strategy iteration framework for parity games that works with valuations from a universal tree. We have also demonstrated the efficiency of our framework by applying it to known constructions of universal trees [88, 43]. In particular, every iteration takes $O(mn^2 \log n \log d)$ time for the universal tree of Jurdzinski–Lazić [88], and $O(mn^2 \log^3 n \log d)$ time for the Strahler-universal tree of Daviaud et al. [43]. As these trees have quasi-polynomial size, this immediately yields strategy iteration algorithms with quasi-polynomial worst-case complexity.

The valuation of a strategy can be seen as the least fixed point of a set of operators associated with the corresponding strategy subgraph. Ohlmann [117] showed that valuations from a universal tree are not compatible with the standard strategy iteration framework [68]. In order to circumvent this impossibility result, we force the valuation to increase in every iteration (whereas this happens automatically in the standard framework). To compute the valuation in every iteration, we adapt label-correcting and label-setting techniques from the shortest path problem to our setting. The key observation that we often exploit is that subtrees of a universal tree form a poset with respect to embedability. This allows us to take advantage of the recursive nature of universal trees in [88, 43] to obtain fast running times.

Unlike value iteration algorithms which are known to realize their worst-case complexity over all possible runs on an instance [87, 64], our framework provides large flexibility in the choice of pivot rules. The most tantalizing open question is whether there exist a pivot rule and a universal tree such that when instantiated with our framework runs in subquasi-polynomial time, i.e. $n^{O(\log^{1-\varepsilon} d)}$ for some constant $\varepsilon > 0$.

References

[1] A. A. Ageev and M. Sviridenko. “Pipage Rounding: A New Method of Constructing Algorithms with Proven Performance Guarantee”. In: *J. Comb. Optim.* 8.3 (2004), pp. 307–328.

[2] S. Agrawal, Y. Ding, A. Saberi, and Y. Ye. “Price of correlations in stochastic optimization”. In: *Operations Research* 60.1 (2012), pp. 150–162.

[3] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows - Theory, Algorithms and Applications*. Prentice Hall, 1993.

[4] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows. Theory, algorithms, and applications*. English. Englewood Cliffs, NJ: Prentice Hall, 1993, pp. xvi + 846. ISBN: 0-13–617549-X.

[5] M. Akian, S. Gaubert, and A. E. Guterman. “Tropical Polyhedra are Equivalent to mean Payoff Games”. In: *Int. J. Algebra Comput.* 22.1 (2012).

[6] A. Asadpour, R. Niazadeh, A. Saberi, and A. Shameli. “Sequential Submodular Maximization and Applications to Ranking an Assortment of Products”. In: *EC ’22: The 23rd ACM Conference on Economics and Computation*. 2022, p. 817.

[7] B. Aspvall and Y. Shiloach. “A Polynomial Time Algorithm for Solving Systems of Linear Inequalities with Two Variables per Inequality”. In: *SIAM J. Comput.* 9.4 (1980), pp. 827–845.

[8] S. Barman, O. Fawzi, and P. Fermé. “Tight Approximation Guarantees for Concave Coverage Problems”. In: *38th International Symposium on Theoretical Aspects of Computer Science (STACS)*. Vol. 187. LIPIcs. Saarbrücken, Germany, 2021, 9:1–9:17.

[9] S. Barman, O. Fawzi, S. Ghoshal, and E. Gürpinar. “Tight approximation bounds for maximum multi-coverage”. In: *Math. Program.* 192.1 (2022), pp. 443–476.

[10] M. Benerecetti, D. Dell’Erba, and F. Mogavero. “Solving parity games via priority promotion”. In: *Formal Methods Syst. Des.* 52.2 (2018), pp. 193–226.

[11] M. Benerecetti, D. Dell’Erba, F. Mogavero, S. Schewe, and D. Wojtczak. “Priority Promotion with Parysian Flair”. In: *ArXiv* abs/2105.01738 (2021).

[12] A. Bhalgat, T. Chakraborty, and S. Khanna. “Mechanism design for a risk averse seller”. In: *International Workshop on Internet and Network Economics*. Springer. 2012, pp. 198–211.

[13] H. Björklund, S. Sandberg, and S. G. Vorobyov. “A Discrete Subexponential Algorithm for Parity Games”. In: *20th Annual Symposium on Theoretical Aspects of Computer Science, STACS*. Vol. 2607. Lecture Notes in Computer Science. Berlin, Germany, 2003, pp. 663–674.

[14] R. G. Bland. “On the generality of network flow theory”. Presented at the ORSA/TIMS Joint National Meeting, Miami, FL. 1976.

[15] R. G. Bland, D. Goldfarb, and M. J. Todd. “Feature Article - The Ellipsoid Method: A Survey”. In: *Oper. Res.* 29.6 (1981), pp. 1039–1091.

[16] N. Bonifas, M. Di Summa, F. Eisenbrand, N. Hähnle, and M. Niemeier. “On subdeterminants and the diameter of polyhedra”. In: *Discrete & Computational Geometry* 52.1 (2014), pp. 102–115.

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

[18] S. Borgwardt, E. Finhold, and R. Hemmecke. “On the circuit diameter of dual transportation polyhedra”. In: *SIAM Journal on Discrete Mathematics* 29.1 (2015), pp. 113–121.

[19] S. Borgwardt, T. Stephen, and T. Yusun. “On the circuit diameter conjecture”. In: *Discrete & Computational Geometry* 60.3 (2018), pp. 558–587.

[20] S. Borgwardt and C. Viss. “An implementation of steepest-descent augmentation for linear programs”. In: *Operations Research Letters* 48.3 (2020), pp. 323–328.

[21] J. Bradfield and I. Walukiewicz. “The mu-calculus and Model Checking”. In: *Handbook of Model Checking.* Ed. by E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem. Cham: Springer International Publishing, 2018, pp. 871–919.

[22] J. van den Brand, Y. P. Liu, Y.-T. Lee, T. Saranurak, A. Sidford, Z. Song, and D. Wang. “Minimum Cost Flows, MDPs, and L1-Regression in Nearly Linear Time for Dense Instances”. In: *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC).* 2021, pp. 859–869.

[23] T. Brunsch and H. Röglin. “Finding short paths on polytopes by the shadow vertex algorithm”. In: *Proceedings of the 40th International Colloquium on Automata, Languages, and Programming (ICALP).* Springer. 2013, pp. 279–290.

[24] J. Bulow and J. Roberts. “The simple economics of optimal auctions”. In: *Journal of political economy* 97.5 (1989), pp. 1060–1090.

[25] G. Călinescu, C. Chekuri, M. Pál, and J. Vondrák. “Maximizing a Monotone Submodular Function Subject to a Matroid Constraint”. In: *SIAM J. Comput.* 40.6 (2011), pp. 1740–1766.

[26] G. Călinescu, C. Chekuri, M. Pál, and J. Vondrák. “Maximizing a Submodular Set Function Subject to a Matroid Constraint (Extended Abstract)”. In: *12th International Conference on Integer Programming and Combinatorial Optimization (IPCO).* Vol. 4513. Lecture Notes in Computer Science. Ithaca, NY, USA, 2007, pp. 182–196.

[27] C. S. Calude, S. Jain, B. Khoussainov, W. Li, and F. Stephan. “Deciding parity games in quasipolynomial time”. In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC.* Montreal, Canada, 2017, pp. 252–263.

[28] S. Chawla, J. D. Hartline, D. L. Malec, and B. Sivan. “Multi-parameter mechanism design and sequential posted pricing”. In: *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010.* Ed. by L. J. Schulman. 2010, pp. 311–320.

[29] C. Chekuri and V. Livanos. “On Submodular Prophet Inequalities and Correlation Gap”. In: *14th International Symposium on Algorithmic Game Theory, SAGT.* Vol. 12885. Lecture Notes in Computer Science. 2021, p. 410.

[30] C. Chekuri, J. Vondrák, and R. Zenklusen. “Submodular function maximization via the multilinear relaxation and contention resolution schemes”. In: *SIAM Journal on Computing* 43.6 (2014), pp. 1831–1879.

[31] E. H. Clarke. “Multipart pricing of public goods”. In: *Public choice* (1971), pp. 17–33.

[32] E. Cohen and N. Megiddo. “Improved Algorithms for Linear Inequalities With Two Variables per Inequality”. In: *SIAM J. Comput.* 23.6 (1994), pp. 1313–1347.

[33] M. Conforti and G. Cornuéjols. “Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the Rado-Edmonds theorem”. In: *Discret. Appl. Math.* 7.3 (1984), pp. 251–274.

[34] W. Cook, A. M. Gerards, A. Schrijver, and É. Tardos. “Sensitivity theorems in integer linear programming”. In: *Mathematical Programming* 34.3 (1986), pp. 251–264.

[35] W. Czerwinski, L. Daviaud, N. Fijalkow, M. Jurdzinski, R. Lazic, and P. Parys. “Universal trees grow inside separating automata: Quasi-polynomial lower bounds for parity games”. In: *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*. San Diego, California, USA, 2019, pp. 2333–2349.

[36] D. Dadush and N. Hähnle. “On the shadow simplex method for curved polyhedra”. In: *Discrete & Computational Geometry* 56.4 (2016), pp. 882–909.

[37] D. Dadush, S. Huiberts, B. Natura, and L. A. Végh. “A scaling-invariant algorithm for linear programming whose running time depends only on the constraint matrix”. In: *Proceedings of the 52nd Annual ACM Symposium on Theory of Computing (STOC)*. 2020, pp. 761–774.

[38] D. Dadush, Z. K. Koh, B. Natura, and L. A. Végh. “An Accelerated Newton-Dinkelbach Method and Its Application to Two Variables per Inequality Systems”. In: *29th Annual European Symposium on Algorithms, ESA 2021*. Vol. 204. LIPIcs. 2021, 36:1–36:15.

[39] D. Dadush, Z. K. Koh, B. Natura, and L. A. Végh. “On Circuit Diameter Bounds via Circuit Imbalances”. In: *23rd International Conference on Integer Programming and Combinatorial Optimization, IPCO 2022*. Vol. 13265. Lecture Notes in Computer Science. 2022, pp. 140–153.

[40] D. Dadush, B. Natura, and L. A. Végh. “Revisiting Tardos’s Framework for Linear Programming: Faster Exact Solutions using Approximate Solvers”. In: *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science*. 2020, pp. 931–942.

[41] D. Dadush, B. Natura, and L. A. Végh. “Revisiting Tardos’s Framework for Linear Programming: Faster Exact Solutions using Approximate Solvers”. In: *Proc. 61st IEEE Symposium on Foundations of Computer Science (FOCS)*. 2020, pp. 931–942.

[42] D. Dadush, L. A. Végh, and G. Zambelli. “Geometric Rescaling Algorithms for Submodular Function Minimization”. In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA ’18. New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2018, pp. 832–848. ISBN: 9781611975031.

[43] L. Daviaud, M. Jurdzinski, and K. S. Thejaswini. “The Strahler Number of a Parity Game”. In: *47th International Colloquium on Automata, Languages, and Programming, ICALP*. Vol. 168. LIPIcs. Saarbrücken, Germany, 2020, 123:1–123:19.

[44] J. A. De Loera, R. Hemmecke, and M. Köppe. *Algebraic and Geometric Ideas in the Theory of Discrete Optimization*. USA: Society for Industrial and Applied Mathematics, 2012. ISBN: 1611972434.

[45] J. A. De Loera, R. Hemmecke, and J. Lee. “On Augmentation Algorithms for Linear and Integer-Linear Programming: From Edmonds–Karp to Bland and Beyond”. In: *SIAM Journal on Optimization* 25.4 (2015), pp. 2494–2511.

[46] J. A. De Loera, S. Kafer, and L. Sanità. “Pivot Rules for Circuit-Augmentation Algorithms in Linear Optimization”. In: *arXiv preprint arXiv:1909.12863* (2019).

[47] E. W. Dijkstra. “A note on two problems in connexion with graphs”. In: *Numerische Mathematik* 1 (1959), pp. 269–271.

[48] W. Dinkelbach. “On Nonlinear Fractional Programming”. In: *Management Science* 13.7 (1967), pp. 492–498.

[49] A. W. M. Dress and W. Wenzel. “Valuated matroids”. In: *Advances in Mathematics* 93.2 (1992), pp. 214–250. ISSN: 0001-8708.

[50] S. Dudycz, P. Manurangsi, J. Marcinkowski, and K. Sornat. “Tight Approximation for Proportional Approval Voting”. In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020.* Ed. by C. Bessiere. 2020, pp. 276–282.

[51] S. Dughmi. “Matroid Secretary Is Equivalent to Contention Resolution”. In: *13th Innovations in Theoretical Computer Science Conference, ITCS.* Vol. 215. LIPIcs. 2022, 58:1–58:23.

[52] S. Dughmi, T. Roughgarden, and Q. Yan. “Optimal mechanisms for combinatorial auctions and combinatorial public projects via convex rounding”. In: *Journal of the ACM (JACM)* 63.4 (2016), pp. 1–33.

[53] M. Dyer and A. Frieze. “Random walks, totally unimodular matrices, and a randomised dual simplex algorithm”. In: *Mathematical Programming* 64.1 (1994), pp. 1–16.

[54] H. Edelsbrunner, G. Rote, and E. Welzl. “Testing the Necklace Condition for Shortest Tours and Optimal Factors in the Plane”. In: *Theor. Comput. Sci.* 66.2 (1989), pp. 157–180.

[55] J. Edmonds and R. M. Karp. “Theoretical improvements in algorithmic efficiency for network flow problems”. In: *Journal of the ACM (JACM)* 19.2 (1972), pp. 248–264.

[56] A. Ehrenfeucht and J. Mycielski. “Positional strategies for mean payoff games”. In: *Int. J. Game Theory* 8 (1979), pp. 109–113.

[57] F. Eisenbrand and S. Vempala. “Geometric random edge”. In: *Mathematical Programming* 164.1-2 (2017), pp. 325–339.

[58] F. Ekbatani, B. Natura, and L. A. Végh. “Circuit imbalance measures and linear programming”. In: *Surveys in Combinatorics 2022.* London Mathematical Society Lecture Note Series. Cambridge University Press, 2022, pp. 64–114.

[59] E. A. Emerson and C. S. Jutla. “Tree Automata, Mu-Calculus and Determinacy”. In: *32nd Annual Symposium on Foundations of Computer Science, FOCS.* San Juan, Puerto Rico, 1991, pp. 368–377.

[60] E. A. Emerson, C. S. Jutla, and A. P. Sistla. “On Model-Checking for Fragments of μ-Calculus”. In: *5th International Conference on Computer-Aided Verification, CAV.* Vol. 697. Lecture Notes in Computer Science. Elounda, Greece, 1993, pp. 385–396.

[61] J. Fearnley, S. Jain, B. de Keijzer, S. Schewe, F. Stephan, and D. Wojtczak. “An ordered approach to solving parity games in quasi-polynomial time and quasi-linear space”. In: *Int. J. Softw. Tools Technol. Transf.* 21.3 (2019), pp. 325–349.

[62] U. Feige. “A threshold of $\ln n$ for approximating set cover”. In: *Journal of the ACM (JACM)* 45.4 (1998), pp. 634–652.

[63] E. A. Feinberg and J. Huang. “The value iteration algorithm is not strongly polynomial for discounted dynamic programming”. In: *Oper. Res. Lett.* 42.2 (2014), pp. 130–131.

[64] N. Fijalkow. “An Optimal Value Iteration Algorithm for Parity Games”. In: *ArXiv* abs/1801.09618 (2018).

[65] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. "An analysis of approximations for maximizing submodular set functions – II". In: *Polyhedral combinatorics*. Springer, 1978, pp. 73–87.

[66] M. L. Fredman and R. E. Tarjan. "Fibonacci heaps and their uses in improved network optimization algorithms". In: *J. ACM* 34.3 (1987), pp. 596–615.

[67] O. Friedmann. "An Exponential Lower Bound for the Parity Game Strategy Improvement Algorithm as We Know it". In: *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science, LICS*. Los Angeles, CA, USA, 2009, pp. 145–156.

[68] O. Friedmann. "Exponential Lower Bounds for Solving Infinitary Payoff Games and Linear Programs". PhD thesis. University of Munich, 2011. URL: http://files.oliverfriedmann.de/theses/phd.pdf.

[69] S. Fujishige and H. Hirai. "Compression of $\mathrm{M}^\natural$-convex Functions – Flag Matroids and Valuated Permutohedra". arXiv:2005.12526. 2020.

[70] J. B. Gauthier and J. Desrosiers. "The Minimum Mean Cycle-Canceling Algorithm for Linear Programs". In: *European Journal of Operational Research* (2021).

[71] M. X. Goemans, S. Gupta, and P. Jaillet. "Discrete Newton's algorithm for parametric submodular function minimization". In: *Proceedings of the 19th International Conference on Integer Programming and Combinatorial Optimization*. Waterloo, ON, Canada, 2017, pp. 212–227.

[72] A. V. Goldberg and R. E. Tarjan. "Finding minimum-cost circulations by canceling negative cycles". In: *Journal of the ACM (JACM)* 36.4 (1989), pp. 873–886.

[73] A. V. Goldberg and R. E. Tarjan. "Finding minimum-cost circulations by canceling negative cycles". In: *J. ACM* 36.4 (1989), pp. 873–886.

[74] M. Grötschel, L. Lovász, and A. Schrijver. "The ellipsoid method and its consequences in combinatorial optimization". In: *Comb.* 1.2 (1981), pp. 169–197.

[75] T. Groves. "Incentives in teams". In: *Econometrica: Journal of the Econometric Society* (1973), pp. 617–631.

[76] F. Gul and E. Stacchetti. "Walrasian equilibrium with gross substitutes". In: *Journal of Economic theory* 87.1 (1999), pp. 95–124.

[77] T. D. Hansen, H. Kaplan, and U. Zwick. "Dantzig's pivoting rule for shortest paths, deterministic MDPs, and minimum cost to time ratio cycles". In: *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*. Portland, OR, USA, 2014, pp. 847–860.

[78] J. D. Hartline. "Mechanism design and approximation". In: ().

[79] D. S. Hochbaum, N. Megiddo, J. Naor, and A. Tamir. "Tight bounds and 2-approximation algorithms for integer programs with two variables per inequality". In: *Math. Program.* 62 (1993), pp. 69–83.

[80] D. S. Hochbaum and J. Naor. "Simple and Fast Algorithms for Linear and Integer Programs With Two Variables per Inequality". In: *SIAM J. Comput.* 23.6 (1994), pp. 1179–1192.

[81] A. J. Hoffman and R. M. Karp. "On Nonterminating Stochastic Games". In: *Manage. Sci.* 12.5 (1966), pp. 359–370.

[82] E. Husić, Z. K. Koh, G. Loho, and L. A. Végh. "On the Correlation Gap of Matroids". In: *CoRR* abs/2209.09896 (2022). arXiv: 2209.09896. URL: https://arxiv.org/abs/2209.09896.

[83] S. Iwata. “Submodular function minimization”. In: *Mathematical Programming* 112.1 (2008), pp. 45–64.

[84] S. Iwata and J. B. Orlin. “A simple combinatorial algorithm for submodular function minimization”. In: *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*. SIAM. 2009, pp. 1230–1237.

[85] H. Jiang. “Minimizing convex functions with integral minimizers”. In: *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM. 2021, pp. 976–985.

[86] M. Jurdzinski. “Deciding the Winner in Parity Games is in UP $\cap$ co-UP”. In: *Inf. Process. Lett.* 68.3 (1998), pp. 119–124.

[87] M. Jurdzinski. “Small Progress Measures for Solving Parity Games”. In: *17th Annual Symposium on Theoretical Aspects of Computer Science, STACS*. Vol. 1770. Lecture Notes in Computer Science. Lille, France, 2000, pp. 290–301.

[88] M. Jurdzinski and R. Lazic. “Succinct progress measures for solving parity games”. In: *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS*. Reykjavik, Iceland, 2017, pp. 1–9.

[89] M. Jurdzinski, M. Paterson, and U. Zwick. “A Deterministic Subexponential Algorithm for Solving Parity Games”. In: *SIAM J. Comput.* 38.4 (2008), pp. 1519–1532.

[90] S. Kafer, K. Pashkovich, and L. Sanità. “On the circuit diameter of some combinatorial polytopes”. In: *SIAM Journal on Discrete Mathematics* 33.1 (2019), pp. 1–25.

[91] G. Kalai. “A subexponential randomized simplex algorithm”. In: *Proceedings of the 24th annual ACM Symposium on Theory of Computing*. 1992, pp. 475–482.

[92] G. Kalai and D. J. Kleitman. “A quasi-polynomial bound for the diameter of graphs of polyhedra”. In: *Bulletin of the American Mathematical Society* 26.2 (1992), pp. 315–316.

[93] A. Karczmarz. “Improved Strongly Polynomial Algorithms for Deterministic MDPs, 2VPI Feasibility, and Discounted All-Pairs Shortest Paths”. In: *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2022, pp. 154–172.

[94] R. M. Karp. “A characterization of the minimum cycle mean in a digraph”. In: *Discret. Math.* 23.3 (1978), pp. 309–311.

[95] A. S. Kelso Jr and V. P. Crawford. “Job matching, coalition formation, and gross substitutes”. In: *Econometrica: Journal of the Econometric Society* (1982), pp. 1483–1504.

[96] Z. K. Koh and G. Loho. “Beyond Value Iteration for Parity Games: Strategy Iteration with Universal Trees”. In: *47th International Symposium on Mathematical Foundations of Computer Science, MFCS 2022*. Vol. 241. LIPIcs. 2022, 63:1–63:15.

[97] O. Kupferman and M. Y. Vardi. “Weak Alternating Automata and Tree Automata Emptiness”. In: *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, STOC*. 1998, pp. 224–233.

[98] J. Lee. “Subspaces with well-scaled frames”. In: *Linear Algebra and its Applications* 114 (1989), pp. 21–56.

[99] Y. T. Lee, A. Sidford, and S. C.-w. Wong. “A faster cutting plane method and its implications for combinatorial and convex optimization”. In: *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE. 2015, pp. 1049–1065.

[100] K. Lehtinen. “A modal μ perspective on solving parity games in quasi-polynomial time”. In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS*. Oxford, UK, 2018, pp. 639–648.

[101] R. P. Leme. “Gross substitutability: An algorithmic survey”. In: *Games and Economic Behavior* 106 (2017), pp. 294–316.

[102] M. L. Littman, T. L. Dean, and L. P. Kaelbling. “On the Complexity of Solving Markov Decision Problems”. In: *Proceedings of the 11th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*. Montreal, Quebec, Canada, 1995, pp. 394–402.

[103] O. Madani. “On Policy Iteration as a Newton’s Method and Polynomial Policy Iteration Algorithms”. In: *Proceedings of the 18th National Conference on Artificial Intelligence*. Edmonton, AB, Canada, 2002, pp. 273–278.

[104] S. T. McCormick and A. Shioura. “Minimum ratio canceling is oracle polynomial for linear programming, but not strongly polynomial, even for networks”. In: *Operations Research Letters* 27.5 (2000), pp. 199–207.

[105] N. Megiddo. “Combinatorial Optimization with Rational Objective Functions”. In: *Math. Oper. Res.* 4.4 (1979), pp. 414–424.

[106] N. Megiddo. “Towards a Genuinely Polynomial Algorithm for Linear Programming”. In: *SIAM J. Comput.* 12.2 (1983), pp. 347–353.

[107] M. Mnich, H. Röglin, and C. Rösner. “New deterministic algorithms for solving parity games”. In: *Discret. Optim.* 30 (2018), pp. 73–95.

[108] K. Murota. *Discrete convex analysis*. Vol. 10. SIAM monographs on discrete mathematics and applications. SIAM, 2003.

[109] K. Murota. “On basic operations related to network induction of discrete convex functions”. In: *Optim. Methods Softw.* 36.2-3 (2021), pp. 519–559.

[110] K. Murota and A. Shioura. “Simpler exchange axioms for M-concave functions on generalized polymatroids”. In: *Japan Journal of Industrial and Applied Mathematics* 35.1 (2018), pp. 235–259. ISSN: 0916-7005.

[111] R. B. Myerson. “Optimal auction design”. In: *Mathematics of operations research* 6.1 (1981), pp. 58–73.

[112] K. Nagano. “A strongly polynomial algorithm for line search in submodular polyhedra”. In: *Discrete Optimization* 4.3-4 (2007), pp. 349–359.

[113] G. L. Nemhauser and L. A. Wolsey. “Best Algorithms for Approximating the Maximum of a Submodular Set Function”. In: *Math. Oper. Res.* 3.3 (1978), pp. 177–188.

[114] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. “An analysis of approximations for maximizing submodular set functions – I”. In: *Math. Program.* 14.1 (1978), pp. 265–294.

[115] E. Nikolova. “Approximation algorithms for reliable stochastic combinatorial optimization”. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2010, pp. 338–351.

[116] N. Nisan, T. Roughgarden, É. Tardos, and V. V. Vazirani, eds. *Algorithmic Game Theory*. Cambridge University Press, 2007. ISBN: 9780511800481.

[117] P. Ohlmann. “Monotonic graphs for parity and mean-payoff games”. PhD thesis. Université Paris Cité, 2021. URL: https://tel.archives-ouvertes.fr/tel-03771185.

[118] F. Olver, D. Lozier, R. Boisvert, and C. Clark. *The NIST Handbook of Mathematical Functions*. Cambridge University Press, New York, NY, 2010.

[119] N. Olver and L. A. Végh. “A Simpler and Faster Strongly Polynomial Algorithm for Generalized Flow Maximization”. In: *J. ACM* 67.2 (2020), 10:1–10:26.

[120] J. G. Oxley. *Matroid theory.* Oxford University Press, 1992. ISBN: 978-0-19-853563-8.

[121] C. Papadimitriou, M. Schapira, and Y. Singer. “On the hardness of being truthful”. In: *2008 49th Annual IEEE Symposium on Foundations of Computer Science.* IEEE. 2008, pp. 250–259.

[122] P. Parys. “Parity Games: Zielonka’s Algorithm in Quasi-Polynomial Time”. In: *44th International Symposium on Mathematical Foundations of Computer Science, MFCS.* Vol. 138. LIPIcs. Aachen, Germany, 2019, 10:1–10:13.

[123] I. Post and Y. Ye. “The Simplex Method is Strongly Polynomial for Deterministic Markov Decision Processes”. In: *Math. Oper. Res.* 40.4 (2015), pp. 859–868.

[124] A. Puri. “Theory of hybrid systems and discrete event systems”. PhD thesis. EECS Department, University of California, Berkeley, 1995.

[125] T. Radzik. “Fractional Combinatorial Optimization”. In: *Handbook of Combinatorial Optimization: Volume 1–3.* Ed. by D.-Z. Du and P. M. Pardalos. Springer US, 1998, pp. 429–478.

[126] T. Radzik. “Newton’s Method for Fractional Combinatorial Optimization”. In: *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science.* Pittsburgh, PA, USA, 1992, pp. 659–669.

[127] T. Radzik and A. V. Goldberg. “Tight bounds on the number of minimum-mean cycle cancellations and related results”. In: *Algorithmica* 11.3 (1994), pp. 226–242.

[128] A. Rubinstein and S. Singla. “Combinatorial prophet inequalities”. In: *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms.* SIAM. 2017, pp. 1671–1687.

[129] F. Santos. “A counterexample to the Hirsch conjecture”. In: *Annals of Mathematics* (2012), pp. 383–412.

[130] S. Schewe. “An Optimal Strategy Improvement Algorithm for Solving Parity and Payoff Games”. In: *22nd International Workshop on Computer Science Logic, CSL.* Vol. 5213. Lecture Notes in Computer Science. Bertinoro, Italy, 2008, pp. 369–384.

[131] S. Schewe. “From Parity and Payoff Games to Linear Programming”. In: *34th International Symposium on Mathematical Foundations of Computer Science (MFCS).* Vol. 5734. Lecture Notes in Computer Science. 2009, pp. 675–686.

[132] S. Schewe. “Solving parity games in big steps”. In: *J. Comput. Syst. Sci.* 84 (2017), pp. 243–262.

[133] A. Schrijver. *Combinatorial optimization: polyhedra and efficiency.* Vol. 24. Springer, 2003.

[134] A. S. Schulz and R. Weismantel. “An oracle-polynomial time augmentation algorithm for integer programming”. In: *Proceedings of the 10th annual ACM-SIAM Symposium on Discrete Algorithms (SODA).* 1999, pp. 967–968.

[135] A. Shioura. “On the Pipage Rounding Algorithm for Submodular Function Maximization - a View from Discrete Convex Analysis”. In: *Discret. Math. Algorithms Appl.* 1.1 (2009), pp. 1–24.

[136] R. E. Shostak. “Deciding Linear Inequalities by Computing Loop Residues”. In: *J. ACM* 28.4 (1981), pp. 769–779.

[137] S. Smale. “Mathematical problems for the next century”. In: *The Mathematical Intelligencer* 20 (1998), pp. 7–15.

[138] T. Soma and Y. Yoshida. “A New Approximation Guarantee for Monotone Submodular Function Maximization via Discrete Convexity”. In: *45th International Colloquium on Automata, Languages, and Programming (ICALP)*. Vol. 107. LIPIcs. Prague, Czech Republic, 2018, 99:1–99:14.

[139] B. Sturmfels and R. R. Thomas. “Variation of cost functions in integer programming”. In: *Mathematical Programming* 77.2 (1997), pp. 357–387.

[140] N. Sukegawa. “Improving bounds on the diameter of a polyhedron in high dimensions”. In: *Discrete Mathematics* 340.9 (2017), pp. 2134–2142.

[141] M. Sviridenko, J. Vondrák, and J. Ward. “Optimal Approximation for Submodular and Supermodular Optimization with Bounded Curvature”. In: *Math. Oper. Res.* 42.4 (2017), pp. 1197–1218.

[142] É. Tardos. “A Strongly Polynomial Algorithm to Solve Combinatorial Linear Programs”. In: *Operations Research* 34.2 (1986), pp. 250–256.

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

[144] É. Tardos. “A strongly polynomial minimum cost circulation algorithm”. In: *Combinatorica* 5.3 (1985), pp. 247–256.

[145] É. Tardos. “A strongly polynomial minimum cost circulation algorithm”. In: *Combinatorica* 5.3 (Sept. 1985), pp. 247–255.

[146] R. E. Tarjan. “Depth-First Search and Linear Graph Algorithms”. In: *SIAM J. Comput.* 1.2 (1972), pp. 146–160.

[147] D. M. Topkis. “Minimizing a submodular function on a lattice”. In: *Operations research* 26.2 (1978), pp. 305–321.

[148] L. Tunçel. “Approximating the complexity measure of Vavasis-Ye algorithm is NP-hard”. In: *Mathematical Programming* 86.1 (Sept. 1999), pp. 219–223.

[149] S. A. Vavasis and Y. Ye. “A primal-dual interior point method whose running time depends only on the constraint matrix”. In: *Mathematical Programming* 74.1 (1996), pp. 79–120.

[150] L. A. Végh. “A Strongly Polynomial Algorithm for Generalized Flow Maximization”. In: *Math. Oper. Res.* 42.1 (2017), pp. 179–211.

[151] W. Vickrey. “Counterspeculation, auctions, and competitive sealed tenders”. In: *The Journal of finance* 16.1 (1961), pp. 8–37.

[152] J. Vöge and M. Jurdzinski. “A Discrete Strategy Improvement Algorithm for Solving Parity Games”. In: *12th International Conference on Computer-Aided Verification, CAV*. Vol. 1855. Lecture Notes in Computer Science. Chicago, USA, 2000, pp. 202–215.

[153] J. Vondrák. “Optimal approximation for the submodular welfare problem in the value oracle model”. In: *Proceedings of the fortieth annual ACM symposium on Theory of computing*. 2008, pp. 67–74.

[154] C. Wallacher. “A generalization of the minimum-mean cycle selection rule in cycle canceling algorithms”. unpublished manuscript, Institute für Angewandte Mathematik, Technische Universität Braunschweig. 1989.

[155] C. Wallacher and U. T. Zimmermann. “A polynomial cycle canceling algorithm for submodular flows”. In: *Mathematical programming* 86.1 (1999), pp. 1–15.

[156] Q. Wang, X. Yang, and J. Zhang. “A Class of Inverse Dominant Problems under Weighted ℓ_∞ Norm and an Improved Complexity Bound for Radzik’s Algorithm”. In: *J. Global Optimization* 34.4 (2006), pp. 551–567.

[157] K. D. Wayne. “A polynomial combinatorial algorithm for generalized minimum cost flow”. In: *Mathematics of Operations Research* (2002), pp. 445–459.

[158] S. J. Wright. *Primal-Dual Interior-Point Methods.* Other Titles in Applied Mathematics. SIAM, 1997.

[159] Q. Yan. “Mechanism design via correlation gap”. In: *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms.* SIAM. 2011, pp. 710–719.

[160] Y. Ye. “A new complexity result on solving the Markov decision problem”. In: *Math. Oper. Res.* 30.3 (2005), pp. 733–749.

[161] Y. Ye. “The simplex and policy-iteration methods are strongly polynomial for the Markov decision problem with a fixed discount rate”. In: *Math. Oper. Res.* 36.4 (2011), pp. 593–603.

[162] W. Zielonka. “Infinite Games on Finitely Coloured Graphs with Applications to Automata on Infinite Trees”. In: *Theor. Comput. Sci.* 200.1-2 (1998), pp. 135–183.

Appendix A

Further 2VPI Explanations

A.1 Reducing 2VPI to M2VPI

Following [54, 79], the idea is to replace each variable y_u with $(y_u^+ - y_u^-)/2$, where y_u^+ and y_u^- are newly introduced variables. Then, an inequality $ay_u + by_v \leq c$ becomes

$$a\left(\frac{y_u^+ - y_u^-}{2}\right) + b\left(\frac{y_v^+ - y_v^-}{2}\right) \leq c,$$

which contains four variable, but will be adjusted based on the signs of a and b: If a or b is zero, then the resulting inequality is already monotone and contains two variables. Next, if $\operatorname{sgn}(a) = \operatorname{sgn}(b)$, then we replace the inequality with $ay_u^+ - by_v^- \leq c$ and $-ay_u^- + by_v^+ \leq c$. Otherwise, we replace it with $ay_u^+ + by_v^+ \leq c$ and $-ay_u^- - by_v^- \leq c$. Observe that every inequality in the new system is monotone and supported on exactly two variables. If $\hat{y}$ is a feasible solution to the original system, then setting $y^+ = \hat{y}$ and $y^- = -\hat{y}$ yields a feasible solution to the new system. Conversely, if $(\hat{y}^+, \hat{y}^-)$ is a feasible solution to the new system, then setting $y = (\hat{y}^+ - \hat{y}^-)/2$ yields a feasible solution to the original system. It follows that the two systems are equivalent.

A.2 Non-Existence of Shortest Paths

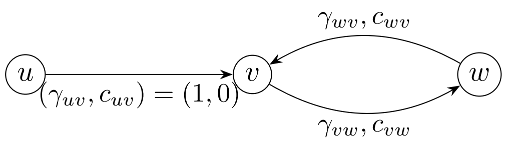

Fig. A.1 A shortest path from u with respect to node labels y may not exist.

Consider Figure A.1. We will sketch three different scenarios in which a shortest path from u with respect to node labels $y \in \mathbb{R}^3$ does not exist. Throughout, let C be the unique directed cycle and C^k be the v-v walk that traverses C exactly $k \in \mathbb{N}$ times.

Negative unit gain cycle Let $\gamma_{wv} = \gamma_{vw} = 1$ and $c_{wv} = c_{vw} = -1$. Then the cycle C fulfils $\gamma(C) = 1$ and $c(C) = -2 < 0$. The concatenation of (u, v) and C^k leads to arbitrarily short walks from u. In particular, there exists no shortest path from u. This observation is independent of the node labels y. Recall as well, that the existence of such a cycle renders the M2VPI instance infeasible (Theorem 2.13).

Flow-absorbing cycle for large node labels Let $\gamma_{vw} = 1$ and $\gamma_{wv} = 1/2$. Then $\gamma(C) = \gamma_{vw}\gamma_{wv} = 1/2$, so C is flow-absorbing. Let further $c_{wv} = c_{vw} = 0$ and $y_w = y_v = 1$. Label-correcting for the cycle C then updates y_v and y_w in two strictly decreasing sequences, which both converge towards 0. Again, the concatenation of (u, v) and C^k leads to a sequence of u-v walks that have no smallest element.

Flow-generating cycle for small node labels Let $\gamma_{vw} = 1$ and $\gamma_{wv} = 2$. Then $\gamma(C) = \gamma_{vw}\gamma_{wv} = 2$, so C is flow-generating. Let further $c_{wv} = -1, c_{vw} = 0$ and $y_w = y_v = 0$. Label-correcting for the cycle C then updates y_v and y_w in two strictly decreasing and unbounded sequences. Again, the concatenation of (u, v) and C^k leads to a sequence of u-v walks that have no smallest element.

A.3 From $y^{\max}$ to a Finite Feasible Solution

In this section, we show how to convert the node labels $y \in \bar{\mathbb{R}}^n$ obtained from Algorithm 3 into a finite feasible solution or an infeasibility certificate of the M2VPI system (G, c, γ) in question. We summarize the classical arguments already used by Aspvall and Shiloach [7]. If y is finite, then we are done because there are no violated arcs in G with respect to y. In fact, y is the pointwise maximal solution by Theorem 2.18. So, we may assume that $y_u = \infty$ for some $u \in V$.

Define $y^{\min} \in \bar{\mathbb{R}}^n$ as the pointwise minimal solution to (G, c, γ) if the system is feasible, where $y_v^{\min} := -\infty$ if and only if the variable y_v is unbounded from below. Consider the reversed graph $\overleftarrow{G} = (V, \overleftarrow{E})$, where $\overleftarrow{E} := \{vu : uv \in E\}$ denotes the set of reversed arcs. The cost and gain factor of each arc $vu \in \overleftarrow{E}$ are given by $\overleftarrow{c}_{vu} := c_{uv}/\gamma_{uv}$ and $\overleftarrow{\gamma}_{vu} := 1/\gamma_{uv}$ respectively. The M2VPI system defined by $(\overleftarrow{G}, \overleftarrow{c}, \overleftarrow{\gamma})$ is equivalent to the original system (G, c, γ), which can be verified by performing the change of variables $z = -y$. Let us run Algorithm 3 on $(\overleftarrow{G}, \overleftarrow{c}, \overleftarrow{\gamma})$. By Theorem 2.18, if it returns node labels $z \in \bar{\mathbb{R}}^n$, then $z = -y^{\min}$ if the system is feasible. Otherwise, the system is infeasible. If z is finite, then we are again done because there are no violated arcs in $\overleftarrow{G}$ with respect to z. So, we may assume that $z_v = \infty$ for some $v \in V$.

If $y_w = z_w = \infty$ for some $w \in V$, then we know that w cannot reach a flow-absorbing cycle in G and $\overleftarrow{G}$. The inability to reach a flow-absorbing cycle in $\overleftarrow{G}$ is equivalent to the inability to be reached by a flow-generating cycle in G. Denote $W := \{w \in V : y_w = z_w = \infty\}$. Observe that every node $w \in W$ is not strongly connected to any $v \notin W$ in G. Thus, checking the feasibility of the system amounts to checking whether there exists a negative unit-gain cycle in $G[W]$. This can be done by running Grapevine on $G[W]$. Let $C_1, C_2, \ldots, C_k$ be the sink components in the strongly connected component decomposition of $G[W]$, and pick any $v_i \in V(C_i)$ for all $i \in [k]$. Then, the input node labels $y' \in \bar{\mathbb{R}}^W$ to Grapevine are set as $y'_{v_i} \in \mathbb{R}$ for all $i \in [k]$ and $y'_v := \infty$ for all other nodes. Let $z' \in \mathbb{R}^W$ be the returned node labels. It is easy to see that there exists a negative unit-gain cycle in $G[W]$ if and only if there exists a violated arc in $G[W]$ with respect to z'.

If the check above reveals that the system is feasible, then we have $y = y^{\max}$ and $-z = y^{\min}$ by Theorem 2.18. Then, we can apply a result of Aspvall and Shiloach which states that the interval $[y_u^{\min}, y_u^{\max}]$ is the projection of the feasible region onto the coordinate y_u for every $u \in V$. To obtain a feasible solution, we simply fix a coordinate $y_u \in [y_u^{\min}, y_u^{\max}]$, update $y^{\min}$ and $y^{\max}$ using a generic label-correcting algorithm like Grapevine, and repeat.

Appendix B

Identities for Alternating Sums of Binomial Coefficients

Claim B.1. *For any $0 \leq \ell \leq n$, we have*

$$\sum_{k=0}^{\ell} (-1)^k \binom{n}{k} = (-1)^\ell \binom{n-1}{\ell}.$$

Proof. We proceed by induction on ℓ. The base case $\ell = 0$ is trivial. For the inductive step, let $\ell \geq 1$. Then,

$$\begin{aligned}
\sum_{k=0}^{\ell} (-1)^k \binom{n}{k} &= (-1)^\ell \binom{n}{\ell} + \sum_{k=0}^{\ell-1} (-1)^k \binom{n}{k} \\
&= (-1)^\ell \binom{n}{\ell} + (-1)^{\ell-1} \binom{n-1}{\ell-1} \\
&= (-1)^\ell \left(\binom{n-1}{\ell-1} + \binom{n-1}{\ell} \right) + (-1)^{\ell-1} \binom{n-1}{\ell-1} = (-1)^\ell \binom{n-1}{\ell}.
\end{aligned}$$

□

Claim B.2. *For any $0 \leq j < n$, we have*

$$\sum_{k=0}^{n} (-1)^{k-1-j} \binom{n}{k} \binom{k-1}{j} = 1 \ .$$

Proof. We proceed by induction on $n - j \geq 1$. For the base case $n - j = 1$, we have

$$\sum_{k=0}^{n} (-1)^{k-1-j} \binom{n}{k} \binom{k-1}{j} = (-1)^{n-1-(n-1)} \binom{n}{n} \binom{n-1}{n-1} = 1.$$

For the inductive step, assume that $n - j > 1$. Then,

$$\begin{aligned}
\sum_{k=0}^{n}(-1)^{k-1-j}\binom{n}{k}\binom{k-1}{j} &= \sum_{k=0}^{n}(-1)^{k-1-j}\left(\binom{n-1}{k-1}+\binom{n-1}{k}\right)\binom{k-1}{j} \\
&= \sum_{k=0}^{n-1}(-1)^{k-j}\binom{n-1}{k}\binom{k}{j} + \sum_{k=0}^{n-1}(-1)^{k-1-j}\binom{n-1}{k}\binom{k-1}{j} \\
&= \sum_{i=0}^{n-1-j}(-1)^{i}\binom{n-1}{i+j}\binom{i+j}{j} + \sum_{k=0}^{n-1}(-1)^{k-1-j}\binom{n-1}{k}\binom{k-1}{j} \\
&= \binom{n-1}{j}\sum_{i=0}^{n-1-j}(-1)^{i}\binom{n-1-j}{i} + \sum_{k=0}^{n-1}(-1)^{k-1-j}\binom{n-1}{k}\binom{k-1}{j} \\
&= \sum_{k=0}^{n-1}(-1)^{k-1-j}\binom{n-1}{k}\binom{k-1}{j} = 1.
\end{aligned}$$

The second last equality is due to $n - 1 - j > 0$, while the last equality is by the inductive hypothesis. □

Claim B.3. *For any $0 < j \le n$, we have*

$$\sum_{i=0}^{j}(-1)^{i}\binom{n}{i}\binom{n-i}{j-i} = 0.$$

Proof. Using $\binom{n}{i}\binom{n-i}{j-i} = \frac{n!}{i!(n-i)!}\frac{(n-i)!}{(j-i)!(n-j)!} = \frac{n!}{i!}\frac{j!}{j!}\frac{1}{(j-i)!(n-j)!} = \binom{n}{j}\binom{j}{i}$, we get

$$\sum_{i=0}^{j}(-1)^{i}\binom{n}{i}\binom{n-i}{j-i} = \sum_{i=0}^{j}(-1)^{i}\binom{n}{j}\binom{j}{i} = \binom{n}{j}(1-1)^{j} = 0\ .$$

□

Claim B.4. *For any $0 \le j \le k \le n$, we have*

$$\sum_{i=0}^{j}(-1)^{i}\binom{n-k}{i}\binom{n-i}{j-i} = \binom{k}{j}$$

Proof. Let $\{A, B\}$ be a partition of $[n]$ such that $|A| = k$ and $|B| = n - k$. In the sum, every set $S \subseteq [n]$ of size j is counted $\sum_{i=0}^{|S\cap B|}(-1)^{i}\binom{|S\cap B|}{i}$ times. If $|S \cap B| = 0$, then S is counted once. Otherwise, it is counted 0 times. Thus, every set $S \subseteq A$ of size j is counted once. □

Claim B.5. *For any $0 \le j \le n - 1$, we have*

$$\sum_{i=0}^{j}(-1)^{i}\binom{n}{i}\binom{n-1-i}{j-i} = (-1)^{j}.$$

Proof. We proceed by induction on $j \geq 0$. The base case $j = 0$ is clear as

$$(-1)^0 \binom{n}{0}\binom{n-1}{0} = 1.$$

For the inductive step, assume that $j > 0$. Then,

$$\begin{aligned}
\sum_{i=0}^{j} (-1)^i \binom{n}{i}\binom{n-1-i}{j-i} &= \sum_{i=0}^{j} (-1)^i \binom{n}{i}\left(\binom{n-i}{j-i} - \binom{n-1-i}{j-1-i}\right) \\
&= -\sum_{i=0}^{j} (-1)^i \binom{n}{i}\binom{n-1-i}{j-1-i} && \text{(Claim B.3)} \\
&= -\sum_{i=0}^{j-1} (-1)^i \binom{n}{i}\binom{n-1-i}{j-1-i} = (-1)^j. && \text{(Inductive hypothesis)}
\end{aligned}$$

□

Claim B.6. *For any $0 \leq j \leq n-2$, we have*

$$\sum_{i=0}^{j} (-1)^i \binom{n}{i}\binom{n-2-i}{j-i} = (-1)^j (j+1).$$

Proof. We proceed by induction on $j \geq 0$. The base case $j = 0$ is clear as

$$(-1)^0 \binom{n}{0}\binom{n-2}{0} = 1.$$

For the inductive step, assume that $j > 0$. Then,

$$\begin{aligned}
\sum_{i=0}^{j} (-1)^i \binom{n}{i}\binom{n-2-i}{j-i} &= \sum_{i=0}^{j} (-1)^i \binom{n}{i}\left(\binom{n-1-i}{j-i} - \binom{n-2-i}{j-1-i}\right) \\
&= (-1)^j - \sum_{i=0}^{j} (-1)^i \binom{n}{i}\binom{n-2-i}{j-1-i} && \text{(Claim B.5)} \\
&= (-1)^j - \sum_{i=0}^{j-1} (-1)^i \binom{n}{i}\binom{n-2-i}{j-1-i} \\
&= (-1)^j - (-1)^{j-1} j = (-1)^j (1+j). && \text{(Inductive hypothesis)}
\end{aligned}$$

□