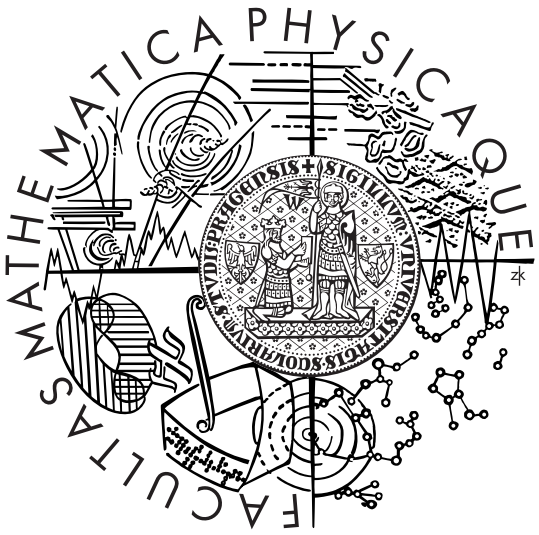
CHARLES UNIVERSITY
FACULTY OF MATHEMATICS AND PHYSICS



HABILITATION THESIS

LINEAR AND EXACT EXTENDED FORMULATIONS

HANS RAJ TIWARY

Department of Applied Mathematics
2016

For Maria and Palomito.

ACKNOWLEDGMENTS

This thesis would not have taken form if not for the numerous wonderful collaborators with whom I had the pleasure and honor to work. My first and foremost thanks to all of them.

I would also like to thank the various mentors I had over the years: Raimund Seidel, Günter M. Ziegler, and Samuel Fiorini. I may not have learnt everything that I could have learnt from them, but the things that I did learn were numerous.

Special thanks also to members of KAM and IUUK for a very friendly and encouraging environment.

Last but not the least, I would like to thank my wife Maria. Without her support I would be utterly lost.

The finger can point to the moon's location.
However, the finger is not the moon.
To look at the moon, it is necessary to gaze beyond the finger, right?"

— Hotei, The laughing Budhha

PREFACE

In recent years there has been a flurry of activity regarding bounds on the extension complexity of combinatorial polytopes. This work covers some of those results in which I took part. At present, this document is part of my habilitation thesis and as such I do not make any attempt to be comprehensive about research on extended formulations. To a knowledgeable reader it may be glaringly obvious that some important aspects of the recent research related to extended formulations are missing from this work. In particular there is essentially no discussion about approximate extended formulations and semidefinite extended formulations. Hopefully a future version of this document will discuss these aspects.

This document – in its current form – is best read as a companion and commentary to ten of the research articles that I have coauthored. These articles are listed after the table of contents. It has been my intent to disassemble and reassemble the contents of these papers to provide the reader with a coherent view of my research in recent years. While individual tastes may differ regarding the value of these lines of inquiry, I have attempted to sew them with a common thread. Naturally, this document also contains results in which I played no part and I have attempted to cite the correct source for those statements. I take full responsibility for any omissions and misattributions, and hope that if a reader notices such an issue they will kindly notify me.

The reader may also notice that this document does not contain any lemmata or theorems, just a sequence of propositions and a few exercises. This choice was made to keep the size of the document reasonable for a habilitation thesis. Listing propositions and exercises had an obvious benefit for me: I could get away with listing only some of the proofs. I have attempted to provide a link to the actual proof where I could find one, but for some propositions and exercises I am not aware of any text that lists them in the same form as stated in this work. This does not mean that they are difficult to prove or are novel, but often a correct proof would require technical discussions that we do not wish to have. I can only hope that the exercises and the propositions that have been stated without proofs are simple enough for someone with basic knowledge of the material. I am also hopeful that a future revision of this work will include the missing proofs.

In its present form, this work assumes a relatively high level of familiarity with polytopes and communication complexity. This does not mean that the reader needs to be an expert in these fields. It is

my estimate that someone pursuing a PhD in theoretical computer science or a related field should be able to follow the text, fill missing proofs, and understand the presented propositions.

It is my hope that anyone interested in extended formulations finds this document helpful. Any comments on how to better this text is most welcome.

Hans Raj Tiwary

CONTENTS

LIST OF FIGURES

PUBLICATIONS

This thesis is based on the following ten research articles that I have coauthored. A copy of each article is attached in the appendices.

[1] David Avis and Hans Raj Tiwary. "A generalization of extension complexity that captures P." In: *Information Processing Letters* 115.6-8 (2015), pp. 588–593. DOI: 10.1016/j.ipl.2015.02.005.

[2] David Avis and Hans Raj Tiwary. "On the extension complexity of combinatorial polytopes." In: *Mathematical Programming* 153.1 (2015), pp. 95–115. DOI: 10.1007/s10107-014-0764-2.

[3] David Avis and Hans Raj Tiwary. "On the $\mathcal{H}$-free extension complexity of the TSP." In: *Optimization Letters* (2016), pp. 1–11. ISSN: 1862-4480. DOI: 10.1007/s11590-016-1029-1.

[4] David Avis, David Bremner, Hans Raj Tiwary, and Osamu Watanabe. "Polynomial size linear programs for non-bipartite matching problems and other problems in P." In: *CoRR* abs/1408.0807 (2014). eprint: arXiv:1408.0807.

[5] Yuri Faenza, Samuel Fiorini, Roland Grappe, and Hans Raj Tiwary. "Extended formulations, nonnegative factorizations, and randomized communication protocols." In: *Mathematical Programming* 153.1 (2015), pp. 75–94. DOI: 10.1007/s10107-014-0755-3.

[6] Samuel Fiorini, Thomas Rothvoß, and Hans Raj Tiwary. "Extended Formulations for Polygons." In: *Discrete & Computational Geometry* 48.3 (2012), pp. 658–668. DOI: 10.1007/s00454-012-9421-9.

[7] Samuel Fiorini, Serge Massar, Sebastian Pokutta, Hans Raj Tiwary, and Ronald de Wolf. "Exponential Lower Bounds for Polytopes in Combinatorial Optimization." In: *Journal of the ACM* 62.2 (2015), p. 17. DOI: 10.1145/2716307.

[8] Jakub Gajarský, Petr Hliněný, and Hans Raj Tiwary. "Parameterized Extension Complexity of Independent Set and Related Problems." In: *CoRR* abs/1511.08841 (2015). eprint: arXiv:1511.08841.

[9] Petr Kolman, Martin Koutecký, and Hans Raj Tiwary. "Extension Complexity, MSO Logic, and Treewidth." In: *Proceedings of the 15th SWAT* To appear (2016). eprint: arXiv:1507.04907.

[10] Hans Raj Tiwary. "Extension Complexity of Formal Languages." In: *ArXiv e-prints* (Mar. 2016). arXiv: 1603.07786 [cs.CC].

"The time has come," the Walrus said,
"To talk of many things:
Of shoes–and ships–and sealing-wax–
Of cabbages–and kings–
And why the sea is boiling hot–
And whether pigs have wings."

— The Walrus and The Carpenter [17]

0 INTRODUCTION

HISTORY

This thesis deals with recent advances in the theory of extended formulations, with emphasis on linear and exact formulations. A linear and exact extended formulation for a polytope P is another polytope Q such that P is a projection of Q. If one wants to optimize a linear function over P one can also obtain the same result by optimizing instead over an extended formulation with the weights modified according to the projection matrix that produces P from Q (cf. Chapter 3). Many polytopes that require an exponential number of facets to describe can be obtained as a projection of a higher dimensional polytope with just polynomially many facets [19, 39, 62]. The importance of extended formulations, thus, is clear for combinatorial optimization.

The directions that are explored in this thesis come from the reverse perspective. In the late eighties Swart attempted to prove that PTIME = NP by writing a polynomial sized linear program (LP) for the traveling salesman problem. Due to the large size of the LP and its complicated nature, it was difficult to find an error in the construction. In a groundbreaking paper Yannakakis showed that any symmetric[1] LP whose feasible region is an extended formulation for the TSP polytope must have exponentially many facets, thus proving that Swart's LP was erroneous. Whether or not the requirement of symmetry could be removed was left by Yannakakis as an open problem. Yannakakis believed that asymmetry should not help one avoid exponential size [63].

This question remained dormant for about two decades when Kaibel, Pashkovich, and Theis showed that asymmetry does play an important role in reducing the size of an extended formulation [40]. They gave explicit polytopes whose symmetric extended formulations required exponential size but for which asymmetric extensions of polynomial size existed. Two years later Rothvoss showed that there are 0/1 polytopes that require exponential size extended formulations [54]. These two results created a fresh interest in proving that symmetry did not help for polytopes related to NP-hard problems such as MAX-CUT or TSP. Soon afterwards in a paper coauthored by Fiorini, Pokutta, Massar, de Wolf, and the present author, it was shown that the TSP polytope requires exponential sized extended formulation [27].

The final piece of the puzzle initiated by Yannakakis was supplied by Rothvoss again who showed that even the perfect matching polytope – which corresponds to the polynomial time solvable problem

1 The reader need not concern themselves with the meaning of a symmetric LP; it suffices to think of it as a technical requirement that Swart's LP satisfied.

of identifying whether a graph has a perfect matching – also requires exponential size extended formulations [55]. The rusults of Rothvoss and Fiorini et al. created a flurry of activity for proving lower bounds for extension complexity of polytopes. While Rothvoss' result implied that even "easy" problems may require large extended formulations, the unconditional nature of the lower bounds – at least for "hard" problems seemed to match the conventional conditional bounds relying on complexity assumptions such as PTIME $\neq$ NP.

At this point the lines of equiry about bounds on the size of smallest extensions of a polytope had branched into widely different directions. The notion of extended formulations was generalized to arbitrary conic lifts [33]; lower bounds were obtained for approximation [9, 10, 18] and semidefinite extensions [12, 25, 46]; and connections with physical theories [28] and information theory [8, 11] were discovered. The previous list of citations do not do any justice to the widely diverse results that have been obtained since then, and we will not even attempt to have a comprehensive citation in order to keep the focus on works where the present author has taken part.

SUMMARY AND ORGANIZATION OF THE THESIS

This thesis summarizes ten articles that the present author has coauthored and that relate to extended formulations. Out of the ten articles six have appeared in peer-reviewed journals [1–3, 24, 26, 27]. The article [44] has been accepted for presentation in a peer-reviewed coference, and [4, 30] and [60] are under peer-review.

This thesis is best read as a commentary to these ten accompanying articles. The contents of these ten articles have been disassembled and reassembled as smaller parts of a larger picture. The rest of thesis is organized into three parts, each containing three chapters.

0.0.1 *Part One: Ingredients*

Part one of the thesis describes the basic objects that we deal with: polytopes, communication complexity, and extended formulations. It is the intent of the author to present the basic notions that are relevant for the accompanying papers in a comprehensive way. We develop common terminology in which later results can be presented without repeated description of the underlying notions. This part of the thesis refers to some results that the present author coauthored in [2, 24, 26] and [27].

In Chapter 1 we collect the basic terminology related to polytopes. We present basic facts related to polytopes, discuss the role of embeddings and describe common operations on polytopes that become relevant in later chapters. This chapter contains mostly classical results about polytopes. The discussion about embeddings becomes relevant when discussing extended formulations because the same polytope may be described in various ways depending on whether the description is minimal or not, and whether the polytope is embedded in

the smallest possible Euclidean space. We develop the notion of slack equivalence of a polytope that serves as an invariant under the actual embedding of a polytope and allows us to later focus on just the combinatorial structure relevant to the extension complexity of the polytope.

In Chapter 2 we describe basic setting of communication complexity and some tools to prove lower bounds for it. In Section 2.1 we introduce nonnegative rank of a matrix which turns out to be closely related to the size of smallest extension of a polytope. We also discuss some tools to lower bound this quantity, and discuss the lower bound for a specific problem: unique disjointness. This problem serves as a canonical ingredient in proving lower bounds for the extension complexity of the CUT polytope. In Section 2.2 we introduce the notion of communication protocol and in Section 2.3 we describe three models of communication complexity, two of which are the classical deterministic and randomized models while the third one – where it is enough for a protocol to work in expectation – is relevant to extension complexity.

In Chapter 3 we finally introduce the notion of extended formulations and extension complexity. This allows us to talk about the size of the smallest possible extended formulation of a polytope. In Section 3.1 we present the notion of extension complexity for a set of related polytopes since one is usually interested not just in the extension complexity of a single polytope but of a family of related polytopes. We have attempted to provide a general language that allows us to later on talk not just about sets of polytopes but also about notions such as the parameterized extension complexity of polytopes. We also discuss some common tricks to bound the extension complexity of a polytope. In Section 3.2 we discuss the effects of applying some common operations of polytopes on their extension complexity. Finally, in Chapter 3.3 we present some canonical examples of polytope families and bounds on their extension complexity. Later on these examples serve us as building blocks for bounds on other families.

0.0.2 *Part two: Recipes*

Part two of the thesis presents more lower bounds for various families of polytopes. We also develop notions allowing us to talk about extension complexity of binary languages and we define the class of languages that admit polynomial size extended formulations. This part of the thesis refers to some results that the present author coauthored in [2, 27] and [60].

In Chapter 4 we present lower bounds for various classes of polytopes. These polytopes correspond to various NP-hard problems and the bounds on their extension complexity are derived using standard NP-hardness reductions together with some simple observations made in earlier chapters. We also discuss some issues in trying to make a meta statement about arbitrary polynomial time reductions.

In Chapter 5 we describe a convenient way to discuss extension complexity of binary languages. In Section 5.1 we discuss various kinds of problems related to an underlying language, and in Section 5.2 we present the class of languages admitting polynomial size extended formulations. We also discuss the impact of small extensions on the computational complexities of various kind of problems related to the underlying language, and present an example of a compact language. Our example is related to walks in directed graphs and we use this example later on to establish that problems efficiently solvable in the streaming model admit small extended formulations. In Section 5.3 we finally discuss some closure properties of compact languages.

In Chapter 6 we use the discussions from the previous chapter to prove that if a language is accepted by an online Turing machine (possibly nondeterministic) using only logarithmic space then it admits a polynomial size extended formulation. In Section 6.3 we discuss various applications of these results. We present lower bounds in the streaming model and upper bounds on extension complexity of some polytopes.

0.0.3 *Part three: Variations*

In this part we discuss some ways in which the notion of extension complexity can be generalized and applied in other settings. This part of the thesis refers to some results that the present author coauthored in [1, 3, 4, 30] and [44].

In Chapter 7 we consider the extension complexities of parameterized problems. In Section 7.1 we define the notion of parameterized extension complexity and in Section 7.2 we apply this definition to study the parameterized extension complexity of the independent set polytope parameterized by the size of the independent sets. We show that this polytope does not admit a fixed-parameter polynomial extended formulation. Finally, in Section 7.3 we given examples of two general classes of polytopes that do admit fixed-parameter polynomial extension complexity: polytopes of assignments for first order logic over graphs of bounded expansion, and polytopes of assignments for monadic second order logic over graphs of bounded treewidth.

In Chapter 8 we consider the following: suppose we identify a subset of facets of a polytope P and show that the extension complexity of the corresponding inequalities with respect to the vertices of P is large. What can be said about the extension complexity of the polytope formed by removing these inequalities from the description of P? This is motivated by practical considerations. Often for hard polytopes – such as the TSP – one can optimize efficiently over a subset of facets and various cutting plane algorithm exploit this in search for violated inequalities. In Section 8.1 we define this notion precisely. In Section 8.2 we provide strong lower bounds for polytopes related to various NP-hard matching polytopes even if we ignore the odd-set

inequalities of Edmonds. In Section 8.3 we discuss similar results for the TSP polytope with respect to a general class of comb inequalities.

In Chapter 9 we consider a weaker form of extended formulations that we call Weak Extended Formulation (WEF). Instead of requiring that the LP formulation of a problem project to the entire feasible set, we only put milder constraints on the formulation. The most important restriction being that the polytope may have some bad vertices, but it must have the right vertices along the right directions. The motivation for this is that under a polyhedral representation of a problem, we are not always interested in optimizing along arbitrary directions but only some. This hold specially true for decision problems. In Section 9.1 we motivate the reader and present the definition of a WEF in Section 9.2. Finally in Section 9.3 we discuss how every problem in P/poly admits a polynomial sized WEF.

NOTATIONS AND CONVENTIONS

We will use the following conventions throughout this document except when specifically stated otherwise.

- Boldface small letters represent column vectors; boldface capital letters represent matrices. The i-th row and the j-th column of a matrix $\mathbf{M}$ will be denoted by $\mathbf{M}_i$ and $\mathbf{M}^j$ respectively.
- Set of vectors will be denoted by capital letters. A set V can also be written as a matrix $\mathbf{V}$. For example, let $V = \{\mathbf{v_1}, \mathbf{v_2}, \ldots\}$ be a set of vectors, then $\mathbf{V}$ is the corresponding matrix whose columns are vectors $\mathbf{v_i}$.
- Similar to previous convention, if $\mathbf{V}$ is an $n \times m$ matrix then V will denote the set of column vectors of $\mathbf{V}$.
- The number of rows of a matrix $\mathbf{V}$ will be denoted by $\text{numrows}(\mathbf{V})$ and the number of columns by $\text{numcols}(\mathbf{V})$.
- For matrices $\mathbf{A}$ and $\mathbf{B}$ we will denote the matrix obtained by concatenating the columns of the matrices by $\begin{bmatrix}\mathbf{A} & \mathbf{B}\end{bmatrix}$. Similarly, the matrix obtained by concatenating the rows will be denoted by $\begin{bmatrix}\mathbf{A} \\ \mathbf{B}\end{bmatrix}$.
- Capital letters such as P will be used for denoting polytopes; $\mathcal{P}$ will denote a family of polytopes, and $\mathbb{P}$ will denote a clan of polytopes (cf. Definition 3.1.10).

SOME LINEAR ALGEBRA

The following basic notion from Linear Algebra will be useful to us later.

Definition 0.0.1. A *linear inequality* is of the form $\mathbf{a}^\top \mathbf{x} \leqslant b$ where $\mathbf{a} \in \mathbb{R}^n$ is a vector and $b \in \mathbb{R}$.

Definition 0.0.2. A linear inequality $a^\top x \leqslant b$ defines the *halfspace* $h(a, b) := \{x \in \mathbb{R}^n \mid a^\top x \leqslant b\}$. A *hyperplane* is the boundary $\{x \in \mathbb{R}^n \mid a^\top x = b\}$ associated with the halfspace $h(a, b)$.

At times – if it is clear from the context – we will not make any distinction between a linear inequality and the associated halfspace.

Definition 0.0.3. Let $V = \{v_1, v_2, \ldots, v_m\}$ be a set of m vectors in $\mathbb{R}^n$. A point $x \in \mathbb{R}^n$ is said to be a *convex combination* of the vectors in V if there exists a vector $\lambda \in \mathbb{R}^m$ such that

$$\begin{aligned} V\lambda &= x, && (1)\\ 1^\top\lambda &= 1, && (2)\\ \lambda &\geqslant 0 && (3) \end{aligned}$$

where 1 and 0 are column vectors of all ones and zeroes respectively.

Dropping the requirement that the sum of λ_i's be one, we get the notion of affine combination.

Definition 0.0.4. The *affine hull* of a set of vectors $V = \{v_1, v_2, \ldots, v_m\}$ – denoted by $\mathrm{aff}(V)$ – is defined to be the set of vectors that are affine combinations of v_i's. That is,

$$\mathrm{aff}(V) := \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{aligned} V\lambda &= x, \\ \lambda &\geqslant 0 \end{aligned} \right\}$$

Finally, dropping the nonnegativity requirement, we get the notion of linear combination.

Definition 0.0.5. The *linear hull* of a set of vectors $V = \{v_1, v_2, \ldots, v_m\}$ – denoted by $\mathrm{lin}(V)$ – is defined to be the set of vectors that are linear combinations of v_i's. That is,

$$\mathrm{aff}(V) := \left\{ x \in \mathbb{R}^n \;\middle|\; V\lambda = x \right\}$$

For a vector $v = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}$ we say that the dimension of v is n. For a set of vectors $V = \{v_1, v_2, \ldots, v_m\}$ with $v_i \in \mathbb{R}^n$ the *ambient dimension* is n, while the dimension – denoted by $\dim(V)$ – is equal to the dimension of their affine hull. That is, the vectors in the set V live in a $\dim(\mathrm{aff}(V))$ dimensional subspace of $\mathbb{R}^n$.

Part I

INGREDIENTS

Whenever Gutei Oshõ was asked about Zen, he simply raised his finger. Once a visitor asked Gutei's boy attendant, "What does your master teach?"

The boy too raised his finger.

Hearing of this, Gutei cut off the boy's finger with a knife. The boy, screaming with pain, began to run away. Gutei called to him, and when he turned around, Gutei raised his finger.

The boy suddenly became enlightened.

— The Gateless Gate: Case 3 [38]

1 POLYTOPES

Polytopes are generalizations of polygons to higher dimensional Euclidean spaces. Whereas polygons are relatively simple objects, their higher dimensional analogs have a much richer structure. In this chapter we collect some basic notions related to polytopes that will be relevant to us. For more details we refer the reader to the excellent textbooks by Grünbaum [35] and Ziegler [65].

1.1 BASIC FACTS ABOUT POLYTOPES

Definition 1.1.1. Let $V = \{v_1, \ldots, v_m\}$ be a subset of $\mathbb{R}^n$. The *convex hull* of V – denoted by $\operatorname{conv}(V)$ – is defined as

$$\operatorname{conv}(V) := \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{rcl} V\lambda & = & x, \\ \mathbf{1}^\top \lambda & = & 1, \\ \lambda & \geqslant & 0 \end{array} \right\}$$

Definition 1.1.2. An *$\mathcal{H}$-polytope* in $\mathbb{R}^n$ is a bounded subset of $\mathbb{R}^n$ that is defined by the intersection of a finite number of halfspaces. A *$\mathcal{V}$-polytope* in $\mathbb{R}^n$ is the convex hull of a finite subset of $\mathbb{R}^n$.

Note that the intersection of a finite number of halfspaces need not always be bounded. However we are only interested in the ones that are. Therefore, we will always assume boundedness unless explicitely stated otherwise.

Example 1.1.3. The same octagon described as the intersection of some halfspaces, and as the convex hull of some points.

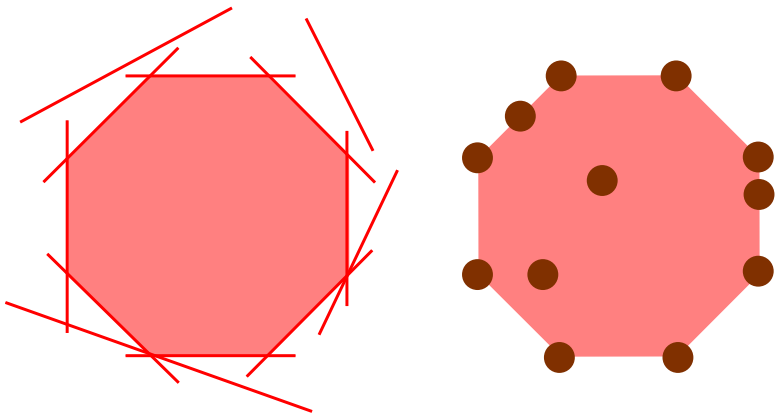


Figure 1: An octagon as an $\mathcal{H}$-polytope (left) and as a $\mathcal{V}$-polytope (right)

An $\mathcal{H}$-polytope $P := \{x \in \mathbb{R}^n \mid Ax \leqslant b\}$ will be often written as $P(A, b)$. Similarly, a $\mathcal{V}$-polytope $P := \operatorname{conv}(V)$ will be written as $P(V)$.

Example 1.1.4. Let $\mathbf{V}$ be an $n \times m$ real matrix. Then,

$$\begin{aligned} P(\mathbf{V}) &= \text{conv}(V), \\ P(\mathbf{V}, \mathbf{1}) &= \{\mathbf{x} \in \mathbb{R}^m \mid \mathbf{V}\mathbf{x} \leqslant \mathbf{1}\}, \\ P(\mathbf{V}^\top, \mathbf{1}) &= \left\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{V}^\top \mathbf{x} \leqslant \mathbf{1}\right\}. \end{aligned}$$

Note again, that $P(\mathbf{V}, \mathbf{1})$ need not always a polytope. In fact, a precise characterization of boundedness of $P(\mathbf{V}, \mathbf{1})$ is possible in terms of the location of the origin with respect to $P(\mathbf{V}))$ and $\dim(\text{aff}(V))$ but the previous example is just to illustrate the notation.

Exercise 1.1.5. Let $\mathbf{V}$ be an $n \times m$ matrix. Show that $P(\mathbf{V}, \mathbf{1})$ is a polytope if and only if $\dim(\text{aff}(V)) = m$ and there exists strictly positive convex multipliers such that $\mathbf{0}$ is a convex combination of vectors in V (columns of $\mathbf{V}$).

Proposition 1.1.6. *P is an $\mathcal{H}$-polytope if and only if it is a $\mathcal{V}$-polytope.*

Proof. See [65], Theorem 1.1. □

Proposition 1.1.6 ensures that every polytope can be represented both by an intersection of a finite number of linear inequalities and as the convex hull of a finite number of points. So we can refer to a *polytope* instead of an $\mathcal{H}$- or a $\mathcal{V}$-polytope, and we can reserve the words $\mathcal{H}$-, $\mathcal{V}$-polytope to refer to a particular representation of a polytope. The shorthands such as $P(\mathbf{V})$ or $P(\mathbf{A}, \mathbf{b})$ will be referred to as a *description* of the associated polytope, with $P(\mathbf{V})$ being a $\mathcal{V}$-description and $P(\mathbf{A}, \mathbf{b})$ being an $\mathcal{H}$-description.

It should be immediately clear that neither $\mathcal{H}$- nor $\mathcal{V}$-descriptions of any polytope are unique, with the empty set and the convex hull of a single point being the only two exceptions for the $\mathcal{V}$-representation.

Definition 1.1.7. A $\mathcal{V}$-description, say $P(\mathbf{V})$, of a polytope is said to be *irredundant* if removing any column from $\mathbf{V}$ results in a different polytope. Otherwise, the description is said to be *redundant*.

An $\mathcal{H}$-description, say $P(\mathbf{A}, \mathbf{b})$, of a polytope P is said to be *irredundant* if removing any inequality from the system $\mathbf{A}\mathbf{x} \leqslant \mathbf{b}$ produces a set of feasible points that is different from P. Otherwise, the description is said to be *redundant*.

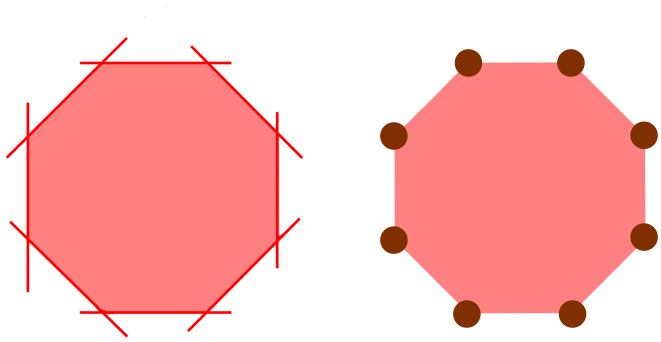


Figure 2: Irredundant $\mathcal{H}$- and $\mathcal{V}$-representation of an octagon

Definition 1.1.8. Let $P = P(\mathbf{V})$ be a polytope. We say that P is a d-polytope in $\mathbb{R}^n$ if $\dim(\text{aff}(V)) = d$ and $\text{numrows}(\mathbf{V}) = n$. Observe that $d \leqslant n$. We call P *full-dimensional* if $d = n$.

1.1.1 *Faces of a polytope*

Definition 1.1.9. An inequality $\mathbf{a}^\top \mathbf{x} \leqslant b$ is said to be valid for a polytope P if $P = P \cap \{\mathbf{x} \mid \mathbf{a}^\top \mathbf{x} \leqslant b\}$.

Definition 1.1.10. $F \subseteq P$ is called a *face* of polytope P if $F = P \cap \{\mathbf{x} \mid \mathbf{a}^\top \mathbf{x} = b\}$ for some $\mathbf{a}, b$ such that $\mathbf{a}^\top \mathbf{x} \leqslant b$ is valid for P.

We will often say that $\mathbf{a}^\top \mathbf{x} \leqslant b$ defines the face F. Observe that for any polytope P, both $\emptyset$ and P are its faces: pick $\mathbf{a} = \mathbf{0}, b = -1$ or $\mathbf{a} = \mathbf{0}, b = 1$. These two faces are called the *trivial* faces, and unless explicitly stated we will use the word "face" to refer only to non-trivial faces. Also, any face of a polytope is itself a polytope: a fact easily proven by observing that any polytope is an $\mathcal{H}$-polytope as well and that adding linear inequalities does not destroy boundedness.

Definition 1.1.11. Let $P = P(\mathbf{V})$ be a d-polytope in $\mathbb{R}^n$. We say that F is an i-face of P if F is an i-polytope and a face of P. Observe that $0 \leqslant i \leqslant d-1$. The 0-faces of a polytope are called the *vertices* and the $(d-1)$-faces are called the *facets*.

Exercise 1.1.12. Let P be a d-polytope in $\mathbb{R}^n$. Let $\mathbf{a}_1^\top \mathbf{x} \leqslant b_1$ and $\mathbf{a}_2^\top \mathbf{x} \leqslant b_2$ define the same facet. Show that there exists $\boldsymbol{\alpha} \in \mathbb{R}^n$ and scalars $\beta, \lambda_1, \lambda_2$ such that

1. $\boldsymbol{\alpha}^\top \mathbf{x} = \beta$ for all $\mathbf{x} \in P$, and
2. $\begin{pmatrix} \mathbf{a}_1 \\ b_1 \end{pmatrix} = \lambda_1 \begin{pmatrix} \mathbf{a}_2 \\ b_2 \end{pmatrix} + \lambda_2 \begin{pmatrix} \boldsymbol{\alpha} \\ \beta \end{pmatrix}$.

Exercise 1.1.13. Let P be a polytope, and let $\mathcal{F}$ be the set of facets of P. Show that, for any description $P(\mathbf{A}, \mathbf{b})$ of P $\text{numrows}(\mathbf{A}) \geqslant |\mathcal{F}|$.

Exercise 1.1.14. Let P be a full-dimensional polytope, and let $\mathcal{F}$ be the set of facets of P. Show that, there exists a description $P(\mathbf{A}, \mathbf{b})$ of P such that $\text{numrows}(\mathbf{A}) = |\mathcal{F}|$.

1.1.2 *Size of a polytope*

Definition 1.1.15. Let P be a k-polytope in $\mathbb{R}^d$. Let $\mathcal{F}$ be the set of facets of P. We define the *size* of P – denoted by $\text{size}(P)$ – as the number $|\mathcal{F}|$.

Exercises 1.1.13 and 1.1.14 are meant for the readers to convince themselves that the above definition makes sense: it assigns a unique number to every polytope and it is possible to describe a full-dimensional polytope with $\text{size}(P)$ inequalities. More precisely,

$$\text{size}(P) = \min_{P = P(\mathbf{A}, \mathbf{b})} \text{numrows}(\mathbf{A}).$$

Proposition 1.1.16. *Let* P *be a* d*-polytope in* $\mathbb{R}^n$*. Then, for every irredundant description* $P(\mathbf{A}, \mathbf{b})$ *of* P *we have that*

$$\text{numrows}(\mathbf{A}) = \text{size}(P) + 2(n - d).$$

In particular, the inequalities in any irredundant description of P can be split in two groups: $\text{size}(P)$ many of the inequalities have a bijection with the facets of P and $2(n-d)$ inequalities correspond to $(n-d)$ equations describing the affine hull of P. For full-dimensional polytopes, the irredundant $\mathcal{H}$- and $\mathcal{V}$-descriptions are unique up to a scaling of inequalities.

Proposition 1.1.17. *Let* P *be a full-dimensional polytope with* $P(\mathbf{V}_1)$, $P(\mathbf{V}_2)$, $P(\mathbf{A}_1, \mathbf{b}_1)$, *and* $P(\mathbf{A}_2, b_2)$ *its irredundant descriptions. Then,* $\mathbf{V}_1 = \mathbf{V}_2$ *up to permutation of columns and* $\begin{bmatrix}\mathbf{A}_1 & \mathbf{b}_1\end{bmatrix} = \mathbf{\Lambda}\begin{bmatrix}\mathbf{A}_2 & \mathbf{b}_2\end{bmatrix}$ *up to permutation of rows, for some diagonal matrix* $\mathbf{\Lambda}$ *with positive diagonal entries.*

1.1.3 *Slack Matrix*

Let $\mathbf{V}$ and $\mathbf{A}$ be matrices and let $\mathbf{b} \in \mathbb{R}^n$ with $\text{numrows}(\mathbf{V}) = \text{numcols}(\mathbf{A}) = n$, such that $P(\mathbf{V}) \subseteq P(\mathbf{A}, \mathbf{b})$. That is, $\mathbf{A}_i\mathbf{V}^j \leqslant \mathbf{b}_i$ for all $i \in [\text{numrows}(\mathbf{A})]$, $j \in [\text{numcols}(\mathbf{V})]$.

Definition 1.1.18. The *slack* of the polytope $P(\mathbf{V})$ with respect to the the polytope $P(\mathbf{A}, \mathbf{b})$ – denoted by $\mathbf{S}(\mathbf{A}, \mathbf{b}, \mathbf{V})$ – is defined to be the $\text{numrows}(\mathbf{A}) \times \text{numcols}(\mathbf{V})$ matrix $\mathbf{S}$ with $\mathbf{S}_{ij} = \mathbf{b}_i - \mathbf{A}_i\mathbf{V}^j$.

When $P(\mathbf{V})$ and $P(\mathbf{A}, \mathbf{b})$ describe the same polytope – say P – then $\mathbf{S}(\mathbf{A}, \mathbf{b}, \mathbf{V})$ is called a slack matrix of P. When $P(\mathbf{A}, \mathbf{b})$ and $P(\mathbf{V})$ are irredundant descriptions, we will say that $\mathbf{S}(\mathbf{A}, \mathbf{b}, \mathbf{V})$ is an irredundant slack matrix of P.

Exercise 1.1.19. Let P be a d-polytope in $\mathbb{R}^n$ with $d \geqslant 1$. Show that no slack matrix of P contains a zero column.

Proposition 1.1.20. *Let* P *be a* d*-polytope in* $\mathbb{R}^n$*. Let* $\mathbf{S}(\mathbf{A}_1, \mathbf{b}_1, \mathbf{V}_1)$ *and* $\mathbf{S}(\mathbf{A}_2, \mathbf{b}_2, \mathbf{V}_2)$ *be irredundant slack matrices of* P*. Then,* $\mathbf{V}_1 = \mathbf{V}_2$ *up to permuting rows, and* $\mathbf{S}(\mathbf{A}_1, \mathbf{b}_1, \mathbf{V}_1) = \mathbf{S}(\mathbf{A}_2, \mathbf{b}_2, \mathbf{V}_2)$ *up to permuting rows and columns and scaling each row by some positive factor.*

Definition 1.1.21. A matrix $\mathbf{S}_2$ is called *slack-equivalent* to a matrix $\mathbf{S}_1$ if $\mathbf{S}_2$ can be obtained from $\mathbf{S}_1$ by any combination of the following operations (in any order):

- Adding or removing convexly dependent rows or columns,
- Adding or removing zero rows or columns,
- Multiplying each row and column by (possibly different) positive scalar values, and
- Applying a permutation of rows and columns.

Proposition 1.1.22. *Let* P *be a polytope. Any two slack matrices of* P *are slack-equivalent to each other.*

Proof. This is easy to prove by noting that any slack matrix of a polytope can be brought to an irredundant form by applying previously described operations that preserve slack-equivalence, and by Proposition 1.1.20 any two irredundant slack matrices of a polytope are slack-equivalent. □

1.2 THE ROLE OF EMBEDDING

Let P be a d-polytope. Whenever we consider a particular description of P, we implicitly impose a specific embedding of P into the Euclidean space $\mathbb{R}^n$ for some $n \geqslant d$. This is the ambient space where P "lives" and in the terminology that we have so far, we say that P is a d-polytope in $\mathbb{R}^n$.

Example 1.2.1. Consider the three dimensional hypercube defined by the inequalities $0 \leqslant x_i \leqslant 1,\ i \in [3]$. The facet $x_1 = 0, 0 \leqslant x_i \leqslant 1,\ i \in \{2,3\}$ is a 2-polytope in $\mathbb{R}^3$. This facet is in fact a square.

Example 1.2.2. Imagine the octagon in Example 1.1.3 as a section of a three dimensional object. The following figure illustrates this.

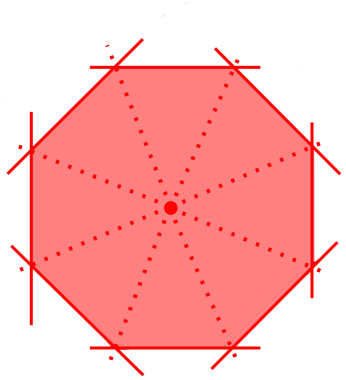


Figure 3: An octagon as a slice of 3-dimensional cone (viewed from top).

To be able to say that two polytopes are the same despite seemingly very different embedding and description – such as in the previous example – we consider various transformations that do not alter the slack matrices of a polytope too much.

1.2.1 *Simple lift*

Definition 1.2.3. Let P be a d-polytope in $\mathbb{R}^n$. A *simple lift* of P into $\mathbb{R}^{n+1}$ is obtained by embedding P into the hyperplane $\{x_{n+1} = 1\}$.

Proposition 1.2.4. *Let P' be obtained by a simple lift of P. Let $\mathbf{S}$ be an irredundant slack matrix of P and $\mathbf{S}'$ be an irredundant slack matrix of P'. Then, $\mathbf{S}'$ and $\mathbf{S}$ are slack equivalent.*

Proof. This follows from the fact that $\mathbf{S}'$ can be obtained from $\mathbf{S}$ by appending two zero rows, scaling each row by some positive factor, and applying some permutation of rows and columns. □

1.2.2 *Variable elimination*

Let P be a d-polytope in $\mathbb{R}^n$. If $d < n$ then any irredundant $\mathcal{H}$-description of P contains $\text{size}(P)$ inequalities and $(n-d)$ equations (cf. Proposition 1.1.16). One can use any of these equations to eliminate one variable resulting in a polytope P' that is an embedding of P in $\mathbb{R}^{n-1}$. This operation can be used to undo a simple lift.

Proposition 1.2.5. *Let P' be obtained from P by reducing the dimension, and let $\mathbf{S}'$ and $\mathbf{S}$ be irredundant slack matrices of P' and P respectively. Then, $\mathbf{S}'$ and $\mathbf{S}$ are slack equivalent.*

Proof. $\mathbf{S}'$ can be obtained from $\mathbf{S}$ by dropping two zero rows, scaling each row by a positive factor, and applying some permutation of rows and columns. □

1.2.3 *Non-degenerate affine transforms*

Definition 1.2.6. Let P be a d-polytope in $\mathbb{R}^n$. A *non-degenerate affine transform* of P is the set $\{\mathbf{T}\mathbf{x} + \mathbf{c} \mid \mathbf{x} \in P\}$ for some invertible matrix $\mathbf{T} \in \mathbb{R}^{n \times n}$ and some vector $\mathbf{c} \in \mathbb{R}^n$.

Exercise 1.2.7. Prove that the image of any non-degenerate affine transform of a polytope is again a polytope with the same number of vertices and facets.

Proposition 1.2.8. *Let P' be obtained from P by a non-degenerate affine transform, and let $\mathbf{S}'$ and $\mathbf{S}$ be slack matrices of P' and P respectively. Then, $\mathbf{S}$ and $\mathbf{S}'$ are slack equivalent.*

Proof. This follows from the fact that $\mathbf{S}$ and $\mathbf{S}'$ are the same up to scaling each row by some positive factor, and permuting rows and columns. □

1.2.4 *Projective scaling*

Definition 1.2.9. Let P be a d-polytope in $\mathbb{R}^n$ with $d < n$. Furthermore, suppose that the origin is not contained in the affine hull $\text{aff}(P)$. A *projective scaling* of P defines a new polytope $P' = \text{conv}(\mathbf{V}')$ where $\mathbf{V}'$ contains each vertex of P scaled by some positive factor such that $\dim(\text{aff}(P')) = \dim(\text{aff}(P))$.

We call this operation a projective scaling because a simple lift of a full-dimensional polytope P followed by a projective scaling creates a simple lift of a polytope projectively isomorphic to P (cf. Subsection 3.2.1).

Proposition 1.2.10. *Let P be a d-polytope in $\mathbb{R}^n$ such that $\text{aff}(P)$ does not contain the origin, and let P' be obtained by a projective scaling of P. If $\mathbf{S}$ and $\mathbf{S}'$ are irredundant slack matrices of P and P' respectively, then $\mathbf{S}'$ is slack-equivalent to $\mathbf{S}$.*

Proof. This follows from the fact that a projective scaling is obtained by a sequence of simple lift, non-degenerate affine transform, and variable elimination. □

1.2.5 *The canonical slack matrix*

When we start discussing extended formulations in Chapter 3 we will see that we are only concerned with slack matrices of a polytope. Propositions 1.2.4, 1.2.5, 1.2.8, and 1.2.10 make it clear that the irredundant slack matrices of a polytope are all the same up to a few zero rows, scaling of rows and columns, and permutation of rows and columns regardless of the choice of the ambient space and the coordinate axes. In fact, the following is true.

Proposition 1.2.11. *Let* P *be a full-dimensional polytope with any irredundant slack matrix* $\mathbf{S}(P)$*. Let* P' *be the polytope obtained by applying any combination of the transformations described in subsections 1.2.1, 1.2.2, 1.2.3, and 1.2.4 and let* $\mathbf{S}'$ *be any slack matrix of* P'*. Then,* $\mathbf{S}'$ *is slack-equivalent to* $\mathbf{S}(P)$*.*

Proof. Follows from Propositions 1.2.4, 1.2.5, 1.2.8, and 1.2.10. □

For our purposes, there will be no distinction between two polytopes that can be obtained from each other via any of the operations described in subsections 1.2.1–1.2.4. That is, a three-dimensional hypercube – for us – remains the same polytope whether embedded in dimension three or thirty; regardless of the position of the origin and the orientation of the coordinate axes; and irrespective of any affine deformations.

We can associate a unique (up to positive scaling and permutation of rows) irredundant slack matrix $\mathbf{S}(P)$ with every polytope P by reducing the dimension of P until we get a full-dimensional polytope P'. Then any two irredundant slack matrices of P' differ only up to reordering and positive scaling (cf. Proposition 1.1.20) and so any irredundant slack matrix of P' is then defined to be $\mathbf{S}(P)$ and is simply referred to as *the* slack matrix of P.

1.3 SOME COMMON OPERATIONS

1.3.1 *Polar duality*

Definition 1.3.1. Let $C \subseteq \mathbb{R}^n$ be a convex set. The polar of C – denoted by C^Δ – is defined as

$$C^\Delta := \left\{ \mathbf{x} \in \mathbb{R}^n \mid \mathbf{y}^\top \mathbf{x} \leqslant 1 \text{ for all } \mathbf{y} \in C \right\}.$$

Let P be a full-dimensional polytope containing origin in its interior. Without loss of generality we can assume that the irredundant descriptions of P are given by $P(\mathbf{V})$ and $P(\mathbf{A}, \mathbf{1})$ for some matrices $\mathbf{V}$ and $\mathbf{A}$. In this case, the polar is also a polytope and the irredundant description of P^Δ is closely related to that of P.

Proposition 1.3.2. $P^\Delta = P(\mathbf{A}^\top) = P(\mathbf{V}^\top, \mathbf{1})$.

1.3.2 *Intersection and Union*

Definition 1.3.3. The *intersection* of two polytopes P_1 and P_2 is the set of common points and is denoted by $P_1 \cap P_2$. That is,

$$P_1 \cap P_2 := \{\mathbf{x} \mid \mathbf{x} \in P_1 \wedge \mathbf{x} \in P_2\}.$$

Proposition 1.3.4. *$P_1 \cap P_2$ is a polytope. Furthermore, if $P_1 = P(\mathbf{A}_1, \mathbf{b}_1)$ and $P_2 = P(\mathbf{A}_2, \mathbf{b}_2)$ then,*

$$P_1 \cap P_2 = P\left(\begin{bmatrix}\mathbf{A}_1\\ \mathbf{A}_2\end{bmatrix}, \begin{pmatrix}\mathbf{b}_1\\ \mathbf{b}_2\end{pmatrix}\right).$$

A simple consequence of this is the following.

Proposition 1.3.5. $\text{size}(P_1 \cap P_2) \leqslant \text{size}(P_1) + \text{size}(P_2)$.

Similarly one may define the union of two polytopes but then the result is not necessarily convex and one may need to take the convex hull of the resulting set to obtain a polytope. We denote this operation by $\uplus$. That is,

$$P_1 \uplus P_2 := \text{conv}\left(\{\mathbf{x} \mid \mathbf{x} \in P_1 \vee \mathbf{x} \in P_2\}\right).$$

For full-dimensional polytopes containing origin in the interior this operation is the polar dual of intersection.

Proposition 1.3.6. *Let P_1 and P_2 be full-dimensional polytopes containing origin in the interior. Then, $(P_1 \cap P_2)^\Delta = P_1^\Delta \uplus P_2^\Delta$ and $(P_1 \uplus P_2)^\Delta = P_1^\Delta \cap P_2^\Delta$.*

1.3.3 *Join and product*

Definition 1.3.7. The *join* of polytopes P_1 and P_2 – denoted by $P_1 * P_2$ – is obtained by embedding them in $\mathbb{R}^d$ for some d such that the affine subspaces $\text{aff}(P_1)$ and $\text{aff}(P_2)$ are skew, and then taking the convex hull of the union.

Any particular choice of the skew subspaces is not very important in this definition since the resulting polytopes are affinely isomorphic (cf. [65]). For polytopes $P(\mathbf{V}_1)$ and $P(\mathbf{V}_2)$ we will take the following canonical embedding to be our definition of the join.

$$P(\mathbf{V}_1) * P(\mathbf{V}_2) := P\left(\begin{bmatrix}\mathbf{V}_1 & \mathbb{O}\\ \mathbb{O} & \mathbf{V}_2\\ -\mathbf{1}^\top & \mathbf{1}^\top\end{bmatrix}\right).$$

Proposition 1.3.8. *Let $P_1 = P(\mathbf{A}_1, \mathbf{b}_1)$ and $P_2 = P(\mathbf{A}_2, \mathbf{b}_2)$ be two polytopes. Then, $\dim(P_1 * P_2) = \dim(P_1) + \dim(P_2) + 1$. Furthermore,*

$$P_1 * P_2 = P\left(\begin{bmatrix}2\mathbf{A}_1 & \mathbb{O} & \mathbf{b}_1\\ \mathbb{O} & 2\mathbf{A}_2 & -\mathbf{b}_2\end{bmatrix}, \begin{bmatrix}\mathbf{b}_1\\ \mathbf{b}_2\end{bmatrix}\right).$$

Once we have the description of the facets of the join of two polytopes, it is a matter of simple substitution of values to relate the canonical slack matrix of the join $\mathbf{S}(P_1 * P_2)$ with the canonical slack matrices of the component polytopes $\mathbf{S}(P_1)$ and $\mathbf{S}(P_2)$.

Proposition 1.3.9. $\mathbf{S}(P_1 * P_2) = \begin{bmatrix} 2\mathbf{S}(P_1) & \mathbf{0} \\ \mathbf{0} & 2\mathbf{S}(P_2) \end{bmatrix}$.

Definition 1.3.10. Let P_1 and P_2 be two polytopes. The *product* – denoted by $P_1 \times P_2$ – is defined as

$$P_1 \times P_2 := \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \middle| x \in P_1, y \in P_2 \right\}.$$

Proposition 1.3.11. *Let* $P_1 = P(\mathbf{A}_1, \mathbf{b}_1)$ *and* $P_2 = P(\mathbf{A}_2, \mathbf{b}_2)$ *be two polytopes. Then,* $\dim(P_1 \times P_2) = \dim(P_1) + \dim(P_2)$. *Furthermore,*

$$P_1 \times P_2 = P\left(\begin{bmatrix} \mathbf{A}_1 & \mathbb{O} \\ \mathbb{O} & \mathbf{A}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix} \right).$$

1.3.4 *Glued-product*

Definition 1.3.12. Let P_1, P_2 be polytopes with $P_1 \subseteq \mathbb{R}^{n_1+d}$ and $P_2 \subseteq \mathbb{R}^{n_2+d}$. The *glued product* of P_1 and P_2 *over the last* d *coordinates* – denoted by $P_1 \otimes_d P_2$ – is defined as

$$P_1 \otimes_d P_2 := \operatorname{conv}\left\{ \begin{pmatrix} x \\ z \\ y \end{pmatrix} \in \mathbb{R}^{n_1+d+n_2} \middle| \begin{array}{l} \begin{pmatrix} x \\ z \end{pmatrix} \in \operatorname{vert}(P_1) \\ \\ \begin{pmatrix} y \\ z \end{pmatrix} \in \operatorname{vert}(P_2) \end{array} \right\}.$$

We also call these special coordinates the glued coordinates.

Example 1.3.13. Let $\mathbf{V}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and let $\mathbf{V}_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$. Then

$$P(\mathbf{V}_1) \otimes_1 P(\mathbf{V}_2) = P\left(\begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 0 \end{bmatrix} \right).$$

It can be shown that the glued product has a very simple description if the glued coordinates have some nice structure. In particular, we have the following.

Proposition 1.3.14. *Let* $P_1 \subseteq \mathbb{R}^{n_1+d}, P_2 \subseteq \mathbb{R}^{n_2+d}$ *be polytopes with descriptions* $\mathbf{A}_1 x + \mathbf{B}_1 z \leqslant \mathbf{c}_1$ *and* $\mathbf{A}_2 y + \mathbf{B}_2 w \leqslant \mathbf{c}_2$ *respectively. Suppose that for every vertex* $(u^\top, z^\top)^\top$ *of* P_1 *or* P_2, z *is a* $0/1$ *vector with at most one* 1. *Then,*

$$P_1 \otimes_d P_2 := \left\{ \begin{pmatrix} x \\ z \\ y \end{pmatrix} \in \mathbb{R}^{n_1+d+n_2} \middle| \begin{array}{rcl} \mathbf{A}_1 x + \mathbf{B}_1 z & \leqslant & \mathbf{c}_1 \\ \mathbf{A}_2 y + \mathbf{B}_2 z & \leqslant & \mathbf{c}_2 \end{array} \right\}.$$

Proof. See [20] Theorem 1. □

2

COMMUNICATION COMPLEXITY

In this chapter we collect some relevant facts about communication complexity. Our focus will be rather narrow and we refer the reader to the excellent text by Kushilevitz and Nisan [45].

Let X, Y, and Z be arbitrary finite sets with $Z \subseteq \mathbb{R}_+$, and let $f : X \times Y \to Z$ be a function. Suppose that there are two players Alice and Bob who wish to compute $f(x, y)$ for some inputs $x \in X$ and $y \in Y$. The players have unlimited computational power. However, Alice knows only x and Bob knows only y. They must therefore exchange information to be able to compute $f(x, y)$.

They could tell each other the inputs that they are holding and thus compute the value $f(x, y)$ but this may not be needed for every function.

Example 2.0.1. Let $f(x, y) = (x + y) \mod 2$. It suffices for Alice and Bob to send one bit each to the other party indicating whether their input is an odd number or not.

Given an ordering $x_1, \ldots, x_m$ of the elements of X, and $y_1, \ldots, y_n$ of the elements of Y, we can visualize the function $f : X \times Y \to Z$ as a $m \times n$ nonnegative matrix $\mathbf{F} = \mathbf{F}(f)$ such that $\mathbf{F}_{ij} = f(x_i, y_j)$ for all $(i, j) \in [m] \times [n]$. The matrix $\mathbf{F}$ is called the *communication matrix* of f. As is natural, we will not always make a distinction between a function and its communication matrix. In fact, for the remainder of the chapter we will use the same notation for a function as for a matrix. For example, if $\mathbf{F}$ denotes a function then both $\mathbf{F}(x, y)$ and $\mathbf{F}_{xy}$ represent the value of the function on input (x, y). Note that this is the same value as the entry in the communication matrix at the row corresponding to x and column corresponding to y.

What features of the communication matrix are relevant for Alice and Bob if they wish to minimize the number of bits that they have to exchange? Surely, we must make precise what is meant when we say that Alice and Bob wish to "compute a function $\mathbf{F}$". We will first discuss a property of matrices – called the nonnegative rank – which will play a crucial role in our discussions and then attempt to take a view of the communication between Alice and Bob in such a way that various notions of computing a function can be handled without requiring much modification.

2.1 NONNEGATIVE RANK

Definition 2.1.1. A *rank-r nonnegative factorization* of a matrix $\mathbf{S}$ is an expression of $\mathbf{S}$ as a product $\mathbf{S} = \mathbf{AB}$ where $\mathbf{A}$ and $\mathbf{B}$ are nonnegative matrices with $\text{numcols}(\mathbf{A}) = \text{numrows}(\mathbf{B}) = r$. The *nonnegative rank* of $\mathbf{S}$, denoted by $\text{rank}_+(\mathbf{S})$, is the minimum nonnegative integer r such that $\mathbf{S}$ admits a rank-r nonnegative factorization.

The nonnegative rank of a matrix S is finite if and only if S is a nonnegative matrix. This is of little consequence for us since we are, in fact, only interested in nonnegative matrices.

Proposition 2.1.2. *The nonnegative rank of a matrix S is the minimum nonnegative integer r such that S is the sum of r nonnegative rank-1 matrices.*

Proof. If $S = AB$, then $S = \sum_{i=1}^{\text{numcols}(A)} A^i B_i$. □

The following is an easy observation which turns out to be very useful when proving lower bounds on the nonnegative matrices.

Proposition 2.1.3. *Let S' be a submatrix of S. Then,* $\text{rank}_+(S) \geqslant \text{rank}_+(S')$.

2.1.1 *Modifying matrices: effect on nonnegative rank*

Now we will see some simple properties of nonnegative rank that will be specially useful for us. Most of the matrices whose nonnegative rank we would like to bound from below will be slack matrices of polytopes. As we will see next, the choice of a canonical slack matrix as done in subsection 1.2.5 was not an ad-hoc choice.

Proposition 2.1.4. *Let F, G be $m \times n$ matrices, then*

1. $\text{rank}_+(F + G) \leqslant \text{rank}_+(F) + \text{rank}_+(G)$
2. $\text{rank}_+(F \circ G) \leqslant \text{rank}_+(F) \cdot \text{rank}_+(G)$

One immediate consequence of this is that the nonnegative rank of a matrix remains unchanged if each row and column of a matrix is scaled independently by a positive factor. One can show something stronger.

Proposition 2.1.5. *Let S', S be matrices such that S' is slack-equivalent to S. Then* $\text{rank}_+(S') = \text{rank}_+(S)$.

Proof. Suppose S' is obtained from S by appending a zero column. Then $\text{rank}_+(S') = \text{rank}_+(S)$ since $[AB\ 0] = A[B\ 0]$. Similarly for appending a zero row.

Suppose S' is obtained from S by appending a convexly dependent column. Then $\text{rank}_+(S') = \text{rank}_+(S)$ since $[AB\ \sum_{i=1}^{\text{numcols}(S)} \lambda_i (AB)^i] = A[B\ \sum_{i=1}^{\text{numcols}(B)} \lambda_i (B)^i]$. Similarly for appending a convexly dependent row.

Finally, suppose S' is obtained by scaling the (i, j)-th entry by $\alpha_i \beta_j$. Let F, G be defined by $F_{ij} = \alpha_i \beta_j$ and $G_{ij} = 1/(\alpha_i \beta_j)$. Clearly, $\text{rank}_+(F) = \text{rank}_+(G) = 1$. Moreover, $S' = S \circ F$ and $S = S' \circ G$. Therefore, by Proposition 2.1.4 we have that $\text{rank}_+(S') \leqslant \text{rank}_+(S)$ and $\text{rank}_+(S) \leqslant \text{rank}_+(S')$. □

Due to the fact that any slack matrix of a polytope P is slack-equivalent to its canonical slack matrix $S(P)$ (Prop. 1.2.11), we get that all slack matrices of a polytope have the same nonnegative rank.

Proposition 2.1.6. *Let* P *be a polytope and* $\mathbf{S}$ *be any slack matrix of* P*. Then,* $\operatorname{rank}_+(\mathbf{S}) = \operatorname{rank}_+(\mathbf{S}(P))$.

At last we can convince ourselves that we do not need to fret over a particular choice of description of a polytope if we are only interested in the nonnegative rank. The nonnegative rank of any slack matrix of a polytope – to a large extent – depends only on the inner geometry and not a particular perspective.

We now describe a combinatorial argument that can sometimes be used to give lower bounds on the nonnegative rank of some matrices. We illustrate the argument by applying it to the slack matrices of joins of polytopes (cf. Subsection 1.3.3). Then in the next subsection we present a related technique that is often used for lower bounding the nonnegative rank of a matrix.

Proposition 2.1.7. *Let* P_1 *and* P_2 *be polytopes with the canonical slack matrices* $\mathbf{S}(P_1)$ *and* $\mathbf{S}(P_2)$*. Let* $\mathbf{S}(P_1 * P_2)$ *be the canonical slack matrix of the join* $P_1 * P_2$*. Then,* $\operatorname{rank}_+(\mathbf{S}(P_1 * P_2)) = \operatorname{rank}_+(\mathbf{S}(P_1)) + \operatorname{rank}_+(\mathbf{S}(P_2))$.

Proof. Due to Proposition 1.3.9 we known that

$$\mathbf{S}(P_1 * P_2) = \begin{bmatrix} 2\mathbf{S}(P_1) & 0 \\ 0 & 2\mathbf{S}(P_2) \end{bmatrix}.$$

Let $\operatorname{numrows}(\mathbf{S}_1) = m_1$, $\operatorname{numrows}(\mathbf{S}_2) = m_2$, $\operatorname{numcols}(\mathbf{S}_1) = n_1$, and $\operatorname{numcols}(\mathbf{S}_2) = n_2$. Also, let $\mathbf{S}(P_1 * P_2) = \mathbf{AB}$ be a rank-r nonnegative factorization with smallest possible r.

We observe that any column of $\mathbf{A}$ cannot contain nonzero entries among the first m_1 rows as well as the last m_2 rows. To see this, let $1 \leqslant k_1 \leqslant m_1$ and $m_1 + 1 \leqslant k_2 \leqslant m_2$ be any two rows. For any column l of $\mathbf{A}$ if $AA_{k_1 l} \neq 0$ then $\mathbf{B}_{ls} = 0$ for all $n_1 + 1 \leqslant s \leqslant n_2$ and if $AA_{k_2 l} \neq 0$ then $\mathbf{B}_{ls} = 0$ for all $1 \leqslant s \leqslant n_1$. Therefore, having nonzero entries within the first n_1 rows and the last m_2 rows of any column of $\mathbf{A}$ would make $\mathbf{B}$ to contain a zero row.

Therefore every column of $\mathbf{A}$ contains zeroes either for all first m_1 rows or for all last m_2 rows. Rearrange columns of $\mathbf{A}$ so that $AA = \begin{bmatrix} \mathbf{A}_1 & 0 \\ 0 & \mathbf{A}_2 \end{bmatrix}$. Arrange the rows of $\mathbf{B}$ accordingly to $\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 & \mathbf{B}_2 \\ \mathbf{B}_3 & \mathbf{B}_4 \end{bmatrix}$ such that their product remains unchanged. Then, $\mathbf{S}_1 = 2\mathbf{A}_1\mathbf{B}_1$ and $\mathbf{S}_2 = 2\mathbf{A}_2\mathbf{B}_4$. Since $\operatorname{numcols}(\mathbf{A}_1) \geqslant \operatorname{rank}_+(\mathbf{S}_1)$ and $\operatorname{numcols}(\mathbf{A}_2) \geqslant \operatorname{rank}_+(\mathbf{S}_2)$ we have that

$$r = \operatorname{numcols}(\mathbf{A}) \geqslant \operatorname{rank}_+(\mathbf{S}(P_1)) + \operatorname{rank}_+(\mathbf{S}(P_2)).$$

Also, $\operatorname{rank}_+(\mathbf{S}(P_1 * P_2)) \leqslant \operatorname{rank}_+(\mathbf{S}(P_1)) + \operatorname{rank}_+(\mathbf{S}(P_2))$. Therefore, equality follows. □

2.1.2 *Rectangle covering bound*

Let $\mathbf{S}$ be an $m \times n$ matrix all whose entries are either zero or one. Such a matrix is often called a 0/1-matrix.

Definition 2.1.8. A combinatorial rectangle (or simply a rectangle) R is a subset of $[m] \times [n]$ such that $R = A \times B$ for some $A \subseteq [m]$ and $B \subseteq [n]$.

Definition 2.1.9. A rectangle R is called a 1-rectangle if for all $(x, y) \in R$ we have that $S_{xy} = 1$. A 0-rectangle is defined similarly. Finally, R is called monochromatic if it is either a 1-rectangle or a 0-rectangle.

Definition 2.1.10. A set of monochromatic rectangles $\mathcal{R}$ is said to cover S if for every $(x, y) \in [m] \times [n]$ there exists a rectangle $R \in \mathcal{R}$ such that $(x, y) \in R$. The rectangle covering number of S, denoted by $\mathrm{rc}(S)$, is the size of smallest $\mathcal{R}$ that covers S.

Let $\mathrm{suppmat}(S)$ be the binary support matrix of S. That is,

$$\mathrm{suppmat}(S)_{ab} = \begin{cases} 1 & \text{if } S_{ab} \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

Proposition 2.1.11. $\mathrm{rank}_+(S) \geqslant \mathrm{rc}(\mathrm{suppmat}(S))$.

Proof. See Theorem 4 in [27] (Appendix A). □

This provides a useful way of proving lower bounds on the nonnegative rank of matrices by means of combinatorial arguments. We illustrate this by an example that will play an important role later on.

2.1.3 *Unique Disjointness*

Consider the following $2^n \times 2^n$ matrix $U = U(n)$ with rows and columns indexed by n-bit strings a and b, and real nonnegative entries:

$$U_{ab} := (a^\top b - 1)^2.$$

An entry U_{ab} of this matrix is zero if and only if the strings a and b are different except at some unique index. A simple combinatorial argument shows the following.

Proposition 2.1.12. $\mathrm{rc}\left(\mathrm{suppmat}(U)\right) \geqslant \left(\frac{3}{2}\right)^n$.

Proof. See [41], Theorem 1. □

Combining Proposition 2.1.12 with Proposition 2.1.11 we get the following theorem.

Proposition 2.1.13. $\mathrm{rank}_+(U) \geqslant \left(\frac{3}{2}\right)^n$.

2.2 COMMUNICATION PROTOCOLS

However Alice and Bob may choose to define what it means to compute a function together, their communication is carried out as a protocol that is agreed upon beforehand by them, on the sole basis of the function f. At the beginning of an *execution* of the protocol Alice and Bob receive their inputs x and y respectively. At each step of the protocol, one of the players has the token. Whoever has the token sends

a bit to the other player. At any point, one of the players outputs a value and the execution stops. The correctness of the protocol is determined by a previously specified relation between the output value and the value of $f(x, y)$.

A protocol can be viewed as a rooted binary tree where each node is marked either Alice or Bob. The leaves have vectors associated with them. An execution of the protocol on a particular input is a path in the tree starting at the root. At a node owned by Alice, following the path to the left subtree corresponds to Alice sending a zero to Bob and taking the right subtree corresponds to Alice sending a one to Bob; and similarly for nodes owned by Bob.

Let X and Y be finite sets and let $f : X \times Y \to \mathbb{R}_+$ be a function that Alice and Bob wish to compute [1].

Definition 2.2.1. A *communication protocol (with private random bits and* nonnegative *outputs)* is a rooted binary tree with some extra information attached to its nodes. Each node of the tree has a *type*, which is either X or Y. To each node v of type X are attached two function $p_{0v}, p_{1v} : X \to [0, 1]$; to each node v of type Y are attached two functions $q_{0v}, q_{1v} : Y \to [0, 1]$; and to each leaf v is attached a nonnegative vector Λ_v that is a column vector of size $|X|$ for leaves of type X and a row vector of size $|Y|$ for leaves of type Y. The functions p_{iv} and q_{jv} define *transition probabilities*, and we assume that $p_{0v}(x) + p_{1v}(x) \leqslant 1$ and $q_{0v}(y) + q_{1v}(y) \leqslant 1$.

Figure 4 shows an example of a protocol.

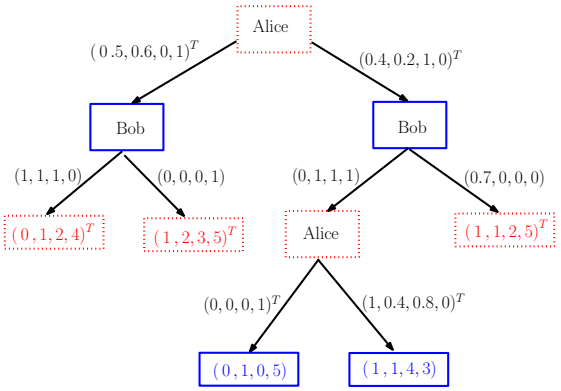


Figure 4: Example of a protocol viewed as a tree

Definition 2.2.2. An *execution* of the protocol on input $(x, y) \in X \times Y$ is a random path that starts at the root and descends to the left child of an internal node v with probability $p_{0v}(x)$ if v is of type X and $q_{0v}(y)$ if v is of type Y, and to the right child of v with probability $p_{1v}(x)$ if v is of type X and $q_{1v}(y)$ if v is of type Y. With probability $1 - p_{0v}(x) - p_{1v}(x)$ and $1 - q_{0v}(y) - q_{1v}(y)$ respectively, the execution stops at v.

1 For now, let us not worry about the precise meaning of computing a function.

Definition 2.2.3. The *value* of an execution on the input pair x, y – denoted by $\mathrm{val}(x, y)$ – is defined as follows. For an execution stopping at leaf v with vector $\mathbf{\Lambda}_v$, $\mathrm{val}(x, y)$ is defined as the entry of $\mathbf{\Lambda}_v$ that corresponds to input $x \in X$ if v is of type X, and $y \in Y$ if v is of type Y. For an execution stopping at an internal node, $\mathrm{val}(x, y)$ is defined to be 0.

Definition 2.2.4. The *complexity* of a protocol Π is measured by one of the two parameters. The *depth* of the protocol – denoted by $\mathrm{depth}(\Pi)$ – is the depth of the corresponding protocol tree, and the *size* of the protocol – denoted by $\mathrm{size}(\Pi)$ – is the number of leaves of the corresponding protocol tree.

When presenting a protocol, we shall often say that one of the two players sends an integer k rather than a binary value. This should be interpreted as the player sending the binary encoding of k or, as a (sub)tree of depth $\lceil \lg k \rceil$, or of size k. Finally, our definitions are such that the depth of a protocol equals the number of bits exchanged by Alice and Bob in the worst case.

Exercise 2.2.5. What is the relation between the depth and the size of a protocol, if the protocol tree is balanced?

With every node v of a communication protocol we can associate a nonnegative matrix $\mathbf{P}_v$ that specifies the probability of visiting that node in an execution. Let $v_1, \ldots, v_k$ denote the nodes of type X on the unique path from the root to the parent of v, and let $w_1, \ldots, w_\ell$ denote the nodes of type Y on this path. Then we have

$$\mathbf{P}_v(x, y) = \prod_{i=1}^{k} \mathbf{p}_{\alpha_i v_i}(x) \cdot \prod_{j=1}^{\ell} \mathbf{q}_{\beta_j w_j}(y),$$

where α_i is either 0 or 1 depending on if the path goes the left or right subtree at v_i, and similarly for β_j. Observe that $\mathbf{P}_v$ is a matrix of nonnegative rank one for each node v of the protocol tree as required.

2.2.1 *The expected value of a protocol*

For each input pair (x, y) given to Alice and Bob, $\mathrm{val}(x, y)$ is a random variable whose distribution depends on the transition probabilities at the nodes of the protocol tree. One may therefore talk about the expected value $\mathbb{E}[\mathrm{val}(x, y)]$.

Let L_X and L_Y be the set of all leaves of the protocol that are of type X and Y respectively and let $\mathbf{\Lambda}_v$ denote the (column or row) vector of values at a leaf $v \in L_X \cup L_Y$. We have

$$\mathbb{E}[\mathrm{val}(x, y)] \quad = \quad \sum_{v \in L_X} \mathbf{\Lambda}_v(x) \mathbf{P}_v(x, y) + \sum_{w \in L_Y} \mathbf{P}_w(x, y) \mathbf{\Lambda}_w(y).$$

Regardless of what Alice and Bob may think that they are computing using a protocol Π, we may associate a function $\mathbf{E}_\Pi : X \times Y \to \mathbb{R}_+$ with Π defined by $\mathbf{E}_\Pi(x, y) = \mathbb{E}[[\mathrm{val}(x, y)]]$. Therefore,

$$\mathbf{E}_\Pi = \sum_{v \in L_X} (\mathbf{\Lambda_v} \circ \mathbf{a}_v) \mathbf{b}_v + \sum_{v \in L_Y} \mathbf{a}_w (\mathbf{b}_w \circ \mathbf{\Lambda}_w).$$

Proposition 2.2.6. *Let S be a nonnegative matrix. Then,*

$$\operatorname{rank}_+(S) = \min\{\operatorname{size}(T) \mid T_T = S\}.$$

Proof. Implicit in the proof of Theorem 2 in [24]. □

2.3 COMPLEXITY OF COMPUTING A FUNCTION

To relate a communication protocol to a function f, it remains to establish the relation between $\operatorname{val}(x, y)$ and $f(x, y)$. Each of the following models specify in different way what it means to compute a function.

Definition 2.3.1. The *communication complexity* of a function f is defined to be the minimum depth among all communication protocols that compute f.

A natural reason to define the communication complexity in this way is to imagine Alice and Bob communicating with each other with the goal of selecting a particular leaf in the protocol tree, so that they can output a value that "computes" the function f without any further communication. To reach any leaf it suffices to send one bit at each node on the unique path from the root to the particular leaf indicating if the next node is the left or the right child.

2.3.1 *Classical deterministic model*

In classical deterministic models of communication complexity, the transition probabilities at each node of the protocol tree can take values either zero or one. Thus, $\operatorname{val}(x, y)$ can take only one possible value for each pair (x, y). A protocol is said to compute a function f if and only if $\operatorname{val}(x, y) = f(x, y)$ for all pairs (x, y) of inputs that Alice and Bob may receive.

2.3.2 *Classical randomized model*

In classical randomized models of communication complexity, the transition probabilities at each node of the protocol tree can take values between zero and one. Thus, for each fixed input $(x, y) \in X \times Y$, $\operatorname{val}(x, y)$ is a random variable and can take one of multiple possible values. A protocol is said to compute a function f if and only if $\operatorname{val}(x, y)$ is close to $f(x, y)$ for all pairs x, y of inputs that Alice and Bob may receive. One may further specify whether this happens with high probability for an execution, or for all executions.

We will not clarify the ambiguities in the previous paragraph since this model is not relevant to us. The interested reader may read Chapter 3 of [45]. We leave the discussion with the following food for thought.

Exercise 2.3.2. How does the variance of the random variable $\operatorname{val}(x, y)$ influence the communication complexity in classical randomized models?

2.3.3 *EF model*

As discussed in Section 2.2.1, for each fixed input $(x, y) \in X \times Y$, the value of an execution on input (x, y) is a random variable and one can define the function $\mathbf{E}_\Pi$ of the expected output of the protocol on input (x, y). In the EF model, we say that the protocol computes a function f if $f = \mathbf{E}_\Pi$. As a shorthand we will refer to a communication protocol with the EF model of computation as an *EF-protocol*. For example saying that f is computed by an EF-protocol Π should be understood as: Π is a communication protocol and $\mathbf{E}_\Pi = f$.

Computing a function only in expectation allows us to assume many things about the "smallest" communication protocol available for any given function.

Proposition 2.3.3. *If f can be computed by an EF-protocol of size r, then f can be computed by an EF-protocol of depth $\lceil \log r \rceil$.*

Proof. See [24], Theorem 2 (Appendix B). □

This allows us to measure the communication complexity either in terms of the depth or the size of the smallest protocol computing f. For our purposes, we will measure the communication complexity in the EF model by the smallest size of any EF-protocol for f.

3

EXTENDED FORMULATIONS

Let $P \subset \mathbb{R}^n$ and $Q \subset \mathbb{R}^{n+r}$ be polytopes.

Definition 3.0.1. Q is called an extended formulation (EF) of P if there exists a linear map $\pi : \mathbb{R}^{n+r} \to \mathbb{R}^n$ such that $P = \pi(Q)$.

The map π in the previous definition *projects* Q to P. With a change of basis one can always assume that this projection map just amounts to dropping r coordinates of Q.

Exercise 3.0.2. Let Q be an EF of P. Show that there exists an EF Q' of P such that $\text{size}(Q') = \text{size}(Q)$ and $P = \pi(Q)$ where the map π is defined by $\pi(z) = x$ if $z^\top = (x^\top, y^\top)$.

When the projection map is not specified, we will assume it to be the canonical orthogonal projection: drop-r-coordinates.

Example 3.0.3. A regular octagon can be seen as a projection of a deformed three-dimensional cube.

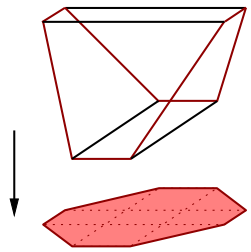


Figure 5: A deformed hypercube projects to a regular octagon.

An extended formulation can also be defined in terms of a certain equivalence in optimization, as follows.

Proposition 3.0.4. *Q is an EF of P if and only if there exists $t \in \mathbb{R}^n$ and an $(n+r) \times n$ matrix R such that*

$$\max_{x \in P} c^\top x = \max_{z \in Q} (Rc)^\top z + c^\top t$$

for all $c \in \mathbb{R}^n$.

Proof. Suppose Q is an EF of P. Then, there exists a linear map $\pi : \mathbb{R}^{n+r} \to \mathbb{R}^n$ such that $\pi(Q) = P$. Let π be defined as $\pi((x^\top, y^\top)^\top) = R^\top (x^\top, y^\top)^\top + t$ where R is an $(n+r) \times n$ matrix and $t \in \mathbb{R}^n$ is a vector. R and t satisfy the requirement of the lemma.

For the other direction, notice that $\alpha^\top x \leqslant \beta$ is valid for P if and only if $(x^\top, y^\top) R\alpha \leqslant \beta - \alpha^\top t$ is valid for Q. Therefore, we can define the linear map $\pi(z) = R^\top (x^\top, y^\top)^\top + t$, if $z^\top = (x^\top, y^\top)$ with $x \in \mathbb{R}^n, y \in \mathbb{R}^r$. For this map we have that $\pi(Q) = P$. □

3.1 EXTENSION COMPLEXITY

Definition 3.1.1. The *extension complexity* of a polytope P – denoted by $\text{xc}(P)$ – is defined to be the size of an extended formulation requiring the fewest number of inequalities. That is,

$$\text{xc}(P) := \min_{Q \text{ is EF of } P} \text{size}(Q)$$

Most often we are interested in the extension complexity of a polytope in terms of the ambient dimension. The ambient dimension, in turn, is often polynomially related to the dimension of the polytope.

Example 3.1.2. The convex hull of the characteristic vectors of all perfect matchings of the complete graph K_n lives in the ambient dimension $\binom{n}{2}$. This polytope, however, is not full-dimension and has dimension $\binom{n}{2} - n$. We may measure the extension complexity of this polytope either in terms of the ambient dimension $\binom{n}{2}$ or the actual dimension $\binom{n}{2} - n$ or n. In all these cases, the expression we will get are essentially equivalent to each other in terms of whether the extension complexity is polynomially bounded or not.

Sometimes we may be interested in the extension complexity of a polytope in terms of other things as well.

Example 3.1.3. Consider the polytope $\text{STAB}_k(G)$ defined as the convex hull of the characteristic vectors of all independent sets of G that are of size k. Are there constant c and function f such that for all graphs G on n vertices, $\text{xc}(\text{STAB}_k(G)) \leqslant f(k) \cdot n^c$?

To talk about such questions succinctly, we may think of k as a parameter of the polytope $\text{STAB}_k(G)$ and talk about its parametrized extension complexity.

Definition 3.1.4. Let $P \subset \mathbb{R}^n$ be a polytope and κ be a fixed number (somehow associated with P). The *parametrized extension complexity* of P is the extension complexity of P expressed as a function of κ and n. The number κ is called a *parameter*.

The above definition does not make a lot of sense since for any fixed polytope P the numbers n, κ, and $\text{xc}(P)$ are fixed numbers. However this definition does make sense for a set of polytopes. This is very convenient since usually we are not interested in the extension complexity of one fixed polytope but a set of related polytopes.

3.1.1 *Extension complexity of multiple polytopes*

Definition 3.1.5. Let $\mathcal{P}$ be a set of polytopes. Let $\mathbf{n}, \kappa : \mathcal{P} \to \mathbb{N}$ be two functions. We say that the extension complexity of $\mathcal{P}$ – denoted by $\text{xc}(\mathcal{P})$ – is a function $\mathbf{f} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ if for every $P \in \mathcal{P}$ we have that $\text{xc}(P) = \mathbf{f}(\mathbf{n}(P), \kappa(P))$. When not specified, the parameter κ is chosen to be $\kappa(P) = 1$ for all $P \in \mathcal{P}$.

The sets of polytopes that are of interest to us, will usually be implicitly described using objects such as graphs, a set of numbers, etc. In such cases, $\mathbf{n}(P)$ will usually be the "size" of these objects and will be polynomially related to the ambient dimension of P. The parameter $\kappa(P)$ will usually be a parameter related to the underlying object used to define P. Let us illustrate this with an example.

Example 3.1.6. For any graph G, consider $\text{CUT}^{\square}(G)$, the convex hull of the characteristic vectors of the edge cuts of G. A natural choice for $\mathbf{n}(\text{CUT}^{\square}(G))$ is the number of vertices of G. A natural parameter $\kappa(\text{CUT}^{\square}(G)$ can the the treewidth of G.

Remark 3.1.7. When the choice of the functions $\mathbf{n}$ and κ is clear and does not create ambiguities, we may say that the extension complexity of the set $\mathcal{P}$ is $\mathbf{f}(n,k)$ for some function $\mathbf{f}$. When κ is not mentioned, we may say that the extension complexity of the set is $\mathbf{g}(n)$ for some function $\mathbf{g}$.

Example 3.1.8. Suppose $\mathcal{MAGIC}$ is a set of polytopes somehow related to graphs. That is, each polytope in this set is defined using a uniquely associated graph. Saying that "$\text{xc}(\mathcal{MAGIC}) = 2^{\tau}n$ where τ is the treewidth and n is the number of vertices of the underlying graph" should be understood to mean that for every graph G with n vertices and treewidth τ the corresponding polytope in the set has extension complexity $2^{\tau}n$.

Example 3.1.9. Let $\mathcal{P}_{\square}$ be the set of full-dimensional hypercubes. Then $\text{xc}(\mathcal{P}_{\square}) = 2n$ where n is the dimension.

Now we will describe two special kinds of sets of polytopes that will help us deduce the parameters $\mathbf{n}$ and κ from the context.

Definition 3.1.10. A *clan* of polytopes is a set of related polytopes. The relation between polytopes will usually be clear from the description. For example, the convex hull of all satisfying assignments of 3CNF formulae defines a clan.

A *family* of polytopes is a countable ordered set $\{P_1, P_2, \ldots\}$ with $P_n \subseteq \mathbb{R}^n$.

Example 3.1.11. A polytope $\text{EP}(G)$ can be defined for every graph G as the convex hull of all perfect matchings of the graph G. For each natural number m, define Edmonds' polytope $\text{EP}(m)$ as the convex hull of characteristic vectors of the perfect matchings of K_n where $m = \binom{n}{2}$.

The set $\{\text{EP}(G) \mid G \text{ is a graph }\}$ defines the perfect matching clan $\mathbb{EP}$ while $\mathcal{EP} = \{\text{EP}(n) | n \in \mathbb{N}\}$ defines a particular family of this clan.

Proposition 3.1.12. $\text{EP}(m)$ *is empty if there is no* n *with* $m = \binom{n}{2}$ *or if such an* n *exists but is odd.*

Proposition 3.1.13 (Rothvoß)**.** *There exists a constant* $0 < c < 1/2$*, such that for every even* $n \in \mathbb{N}$ *we have that* $\text{xc}(\text{EP}(\binom{n}{2})) \geqslant 2^{cn}$.

Proof. See [55], Theorem 1. □

Proposition 3.1.14. *For every* $\epsilon > 0$ *there exists* $n_0 \in \mathbb{N}$ *such that for all* $n \geqslant n_0$ *we have that* $\mathrm{xc}(\mathrm{EP}(\binom{n}{2})) \leqslant 2^{(1/2+\epsilon)n}$.

Proof. See [24], Proposition 3 (Appendix B). □

For a family of polytopes the choice of the parameter $\mathbf{n}$ will most often be the ambient dimension. That is, if $\mathcal{P}$ is a family of polytopes then $\mathbf{n}(P_n) = n$ for $P_n \in \mathcal{P}$. Since a family contains exactly one polytope $P_n \subset \mathbb{R}^n$, the meaning of $\mathrm{xc}(\mathcal{P})$ is clear and a statement such $\mathrm{xc}(\mathcal{P}) = n^3$ is unambiguous with this convention. Note that the polytopes P_n in a family are not required to be full-dimensional (or even non-empty).

Definition 3.1.15. Extension complexity of a clan $\mathbb{P}$ is also denoted by $\mathrm{xc}(\mathbb{P})$ and is defined to be the extension complexity of the family $\mathcal{P} \in \mathbb{P}$ obtained by picking the polytopes with largest extension complexity for each dimension.

More precisely, given a clan $\mathbb{P}$, let $\mathcal{P}_{\max}$ be a family of polytopes such that if $P_n \subset \mathbb{R}^n$ belongs to $\mathcal{P}_{\max}$ then $\mathrm{xc}(P_n) \geqslant \mathrm{xc}(P'_n)$ for all $P'_n \in \mathbb{P}$ with $P'_n \subseteq \mathbb{R}^n$. Moreover, for every n exactly one $P_n \subset \mathbb{R}^n$ belongs to $\mathcal{P}_{\max}$. The extension complexity of clan $\mathbb{P}$ is defined to be equal to $\mathrm{xc}(\mathcal{P})$.

For different values of n, the corresponding polytopes in a family $\mathcal{P}$ may have extension complexities that are not well described by a simple function. Even if exact bounds are known for each polytope in a family of polytopes, it will simplify our lives if we use asymptotic notation to describe the extension complexity of the family. In fact, for a family (or clan) of polytopes, the asymptotic behavior of their extension complexity is what we generally care about. If it is a polynomial function then – at least in principle – the polytopes can be efficiently represented. If the extension complexity of the family (or clan) grows superpolynomially then at least some of the polytopes require large descriptions.

Let $\mathcal{P} = \{P_1, P_2, \ldots\}$ be a family of polytopes with $P_n \subseteq \mathbb{R}^n$. We will say that $\mathrm{xc}(\mathcal{P}) = \mathcal{O}(\mathbf{f})$ to mean that there exists a constant $c > 0$ and a natural number n_0 such that for every polytope $P_n \in \mathcal{P}$ with $n \geqslant n_0$ we have that $\mathrm{xc}(P_n) \leqslant c\mathbf{f}(n)$.

We will say that $\mathrm{xc}(\mathcal{P}) = \Omega(\mathbf{f})$ to mean that there exists a constant $c > 0$ and such that for every natural number n_0 there exists an $n \geqslant n_0$ such that $\mathrm{xc}(P_n) \geqslant c\mathbf{f}(n)$. Note the slight difference from the usual Ω notation used in asymptotic analysis of algorithms [1]. The intent here is to be able to say that $\mathcal{P}$ contains infinitely many polytopes that have high extension complexity.

Finally, we will say that $\mathrm{xc}(\mathcal{P}) = \Theta(\mathbf{f})$ to mean that $\mathrm{xc}(\mathcal{P}) = \mathcal{O}(\mathbf{f})$ as well as $\mathrm{xc}(\mathcal{P}) = \Omega(\mathbf{f})$.

Example 3.1.16. Proposition 3.1.13 can be translated in our setting to the following.

Proposition 3.1.17. $\mathrm{xc}(\mathcal{EP}) = \Omega(c^{\sqrt{n}})$ *for some* $c > 1$.

1 This usage, however, is common among number theorists

One can extend the above notation to provide more information by being able to use functions described in asysmptotic notation as well. We will not go into the details of this point except to present an example that should clarify the point.

Example 3.1.18. Combining Propositions 3.1.13 and 3.1.14 one could say that $\mathrm{xc}(\mathbb{EIP}) = \mathrm{xc}(\mathcal{EP}) = 2^{\Theta(\sqrt{n})}$.

3.1.2 *Bounding extension complexity: some tools*

Before we mention stronger results connecting extension complexity with nonnegative rank, we would like to list few simple facts that follow from the above definition and basic polyhedral properties.

Proposition 3.1.19. *If* P *is the convex hull of* m *points then* $\mathrm{xc}(P) \leqslant m$.

Proof. If $P = P(\mathbf{V})$, then by definition P is a projection of the polytope

$$\left\{ \begin{pmatrix} x \\ \lambda \end{pmatrix} \middle| \begin{array}{rcl} \mathbf{V}\lambda & = & x \\ \mathbf{1}^\top \lambda & = & 1 \\ \lambda & \geqslant & 0 \end{array} \right\}$$

□

Definition 3.1.20. Let Q be a polytope and h be a hyperplane. $Q \cap h$ defines A *slice* of Q.

Proposition 3.1.21. *If* P *is a slice of* Q*, then* $\mathrm{xc}(P) \leqslant \mathrm{xc}(Q)$.

In particular, noting that a polytope is a trivial slice of itself and every face of a polytope P is also a slice of P we get the following simple but important cases.

Proposition 3.1.22. *If* Q *is an* EF *of* P*, then* $\mathrm{xc}(P) \leqslant \mathrm{xc}(Q)$.

Proposition 3.1.23. *If* P *is a face of* Q*, then* $\mathrm{xc}(P) \leqslant \mathrm{xc}(Q)$.

3.1.3 *Yannakakis' characterization of Extension Complexity*

Proposition 3.1.24. *Let* P *be a polytope and* $\mathbf{S}$ *be any slack matrix of* P*. Then,* $\mathrm{xc}(P) = \mathrm{rank}_+(\mathbf{S})$.

Proof. See [24], Theorem 1 (Appendix B). □

Combining Propositions 2.2.6, 2.3.3 and 3.1.24 we get the following.

Proposition 3.1.25. *Let* P *be a polytope. Then, the following are equivalent.*

1. $\mathrm{xc}(P) \leqslant 2^r$.
2. $\mathrm{rank}_+(\mathbf{S}(P)) \leqslant 2^r$.
3. *There exists an EF protocol* Π *with* $\mathbf{E}_\Pi = \mathbf{S}(P)$ *and* $\mathrm{size}(\Pi) \leqslant 2^r$.
4. *There exists an EF protocol* Π *with* $\mathbf{E}_\Pi = \mathbf{S}(P)$ *and* $\mathrm{depth}(\Pi) \leqslant r$.

Already with the discussion so far, the reader should be able to prove bounds on extension complexities of a number of polytopes: some by simply referring to facts already established in previous chapters.

Example 3.1.26. Let $P_{ij} = \left\{x \in \{0,1\}^{n+1} \mid x_{n+1} = x_i \oplus x_j\right\}$ for $1 \leqslant i < j \leqslant n$. Then,

$$P_{ij} = \square_{n-2} \times P\left(\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}\right)$$

upto relabeling of coordinates. Therefore, $\mathrm{xc}(P_{ij}) \leqslant 2n$ (cf. Proposition 3.2.3).

3.1.4 *Combinatorially isomorphic polytopes with different extension complexity*

Before discussing how robust extension complexity is as a measure of intrinsic complexity of representing a polytope, we would like to remark that combinatorial isomorphism of two polytopes is not enough to ensure same extension complexity. This is seen by considering polygons.

Definition 3.1.27. Two polytopes are said to be *combinatorially isomorphic* if the posets of their faces (including trivial ones) ordered by inclusion are isomorphic.

Exercise 3.1.28. Any two n-gons are combinatorially isomorphic.

Proposition 3.1.29. *Let P_n be a regular n-gon. Then $\mathrm{xc}(P_n) \leqslant 2\log n$.*

Proof. See [26], Theorem 2 (Appendix C). □

Proposition 3.1.30. *For every $n \in \mathbb{N}$ there exists an n-gon with extension complexity $\Omega(\sqrt{n})$.*

Proof. See [26], Theorem 3 (Appendix C). □

3.2 EFFECTS OF COMMON OPERATIONS

The extension complexity of a polytope is a fairly robust measure of the inherent complexity of describing a polytope. It does not depend on the ambient space and the choice of a particular coordinate axes. In fact, the extension complexity remains unchanged if the polytope is distorted by a projective transform. Before describing projective transforms formally, we provide a more geometric picture from the excellent textbook "Lectures on Polytopes" by Ziegler.

3.2.1 *Projective transforms*

Let P be a full-dimensional polytope in $\mathbb{R}^n$. Embed this polytope into an affine hyperplane $H \subseteq \mathbb{R}^{n+1}$ and construct the homogenization

of P: $\mathrm{cone}(\{\mathbf{x} \mid \mathbf{x} \in P\})$. Cut this cone with any hyperplane K that intersects all its extreme rays and identify K with $\mathbb{R}^n$. This defines a projective transform of P.

Definition 3.2.1. Let P be a d-polytope in $\mathbb{R}^n$. A *projective transform* of P is defined by a matrix

$$\begin{bmatrix} \mathbf{B} & \mathbf{c} \\ \mathbf{a}^\top & a_{n+1} \end{bmatrix}$$

and a vector $\mathbf{c}'$ with the following conditions:

1. $\det \begin{pmatrix} \mathbf{B} & \mathbf{c} \\ \mathbf{a}^\top & a_{n+1} \end{pmatrix} \neq 0$
2. $\mathbf{a}^\top \mathbf{x} + a_{n+1} > 0$ for all $\mathbf{x} \in P$.

The polytope P' obtained from P via this projective transformation is defined by

$$P' = \left\{ \frac{\mathbf{B}\mathbf{x} + \mathbf{c}}{\mathbf{a}^\top \mathbf{x} + a_{n+1}} + \mathbf{c}' \,\middle|\, \mathbf{x} \in P \right\}.$$

Proposition 3.2.2. *Let P_1 and P_2 be two polytopes that are isomorphic under projective transformations. Then,*

$$\mathrm{xc}(P_1) = \mathrm{xc}(P_2).$$

Proof. See [33], Proposition 2.9. □

3.2.2 *Join, product, and free-sum*

Proposition 3.2.3. *Let P_1, P_2 be polytopes. Then,*

1. $\mathrm{xc}(P_1 * P_2) = \mathrm{xc}(P_1) + \mathrm{xc}(P_2)$.
2. $\mathrm{xc}(P_1 \times P_2) \leqslant \mathrm{xc}(P_1) + \mathrm{xc}(P_2)$.
3. $\mathrm{xc}(P_1 \oplus P_2) \leqslant \mathrm{xc}(P_1) + \mathrm{xc}(P_2)$.

Proof. The first bound follows immediately from Proposition 2.1.7 and Proposition 3.1.24. The second follows from the fact that $P_1 \times P_2$ is a slice of $P_1 * P_2$ (cf. Proposition 3.1.21), while the third follows from the fact that $P_1 \oplus P_2$ is a projection of $P_1 * P_2$ (cf. Proposition 3.1.22). □

3.2.3 *Glued Product*

Proposition 3.2.4 (Margot)**.** *Let $P_1 \subseteq \mathbb{R}^{n_1+d}$ and $P_2 \subseteq \mathbb{R}^{n_2+d}$ be polytopes such that the last d coordinates of any vertex of either polytope is a zero-one vector with at most one 1. Then $\mathrm{xc}(P_1 \otimes_d P_2) \leqslant \mathrm{xc}(P_1) + \mathrm{xc}(P_2)$.*

Proof. See [20], Theorem 1 (cf. [44], Lemma 1, Appendix G). □

3.2.4 *Union*

Proposition 3.2.5. $\mathrm{xc}(P_1 \uplus P_2) \leqslant \mathrm{xc}(P_1) + \mathrm{xc}(P_2)$.

Proof. Let $P_1 = P(\mathbf{V}_1)$ and $P_2 = P(\mathbf{V}_2)$. Consider the projective transform given by the matrix

$$\begin{bmatrix} \mathbf{I} & \mathbf{I} & \mathbf{0} \\ \mathbf{O} & \mathbf{I} & \mathbf{0} \\ \mathbf{0}^\top & \mathbf{0}^\top & 1 \end{bmatrix},$$

where $\mathbf{I}$ is the identity matrix of appropriate size, $\mathbf{O}$ is the matrix of all zeroes, and $\mathbf{0}$ is the zero vector.

This transforms the join $P_1 * P_2$ to

$$P\left(\begin{bmatrix} \mathbf{V}_1 & \mathbf{O} \\ \mathbf{O} & \mathbf{V}_2 \\ -\mathbf{1}^\top & \mathbf{1}^\top \end{bmatrix}\right) \to P\left(\begin{bmatrix} \mathbf{V}_1 & \mathbf{V}_2 \\ \mathbf{O} & \mathbf{V}_2 \\ -\mathbf{1}^\top & \mathbf{1}^\top \end{bmatrix}\right).$$

The later is easily seen to be an EF of $P_1 \uplus P_2$. Combining Propositions 3.2.3, 3.2.2 and 3.1.22 we get that $\mathrm{xc}(P_1 \uplus P_2) \leqslant \mathrm{xc}(P_1 * P_2) \leqslant \mathrm{xc}(P_1) + \mathrm{xc}(P_2)$. □

3.2.5 *Intersection*

Proposition 3.2.6. $\mathrm{xc}(P_1 \cap P_2) \leqslant \mathrm{xc}(P_1) + \mathrm{xc}(P_2)$.

Proof. Let $P_1 = \{\mathbf{x} \mid \mathbf{A}_1\mathbf{x} \leqslant \mathbf{b}_1\}$ and $P_2 = \{\mathbf{x} \mid \mathbf{A}_2\mathbf{x} \leqslant \mathbf{b}_1\}$. Let $Q_1 = \left\{ \begin{pmatrix} \mathbf{x} \\ \mathbf{z} \end{pmatrix} \middle| \mathbf{E}_1\mathbf{x} + \mathbf{F}_1\mathbf{z} \leqslant \mathbf{g}_1 \right\}$ and $Q_2 = \left\{ \begin{pmatrix} \mathbf{x} \\ \mathbf{w} \end{pmatrix} \middle| \mathbf{E}_2\mathbf{x} + \mathbf{F}_2\mathbf{w} \leqslant \mathbf{g}_2 \right\}$ be EFs of P_1 and P_2 respectively. Then,

$$R = \left\{ \begin{pmatrix} \mathbf{x} \\ \mathbf{z} \\ \mathbf{y} \\ \mathbf{w} \end{pmatrix} \middle| \begin{array}{rcl} \mathbf{E}_1\mathbf{x} + \mathbf{F}_1\mathbf{z} & \leqslant & \mathbf{g}_1 \\ \mathbf{E}_2\mathbf{y} + \mathbf{F}_2\mathbf{w} & \leqslant & \mathbf{g}_2 \\ \mathbf{x} & = & \mathbf{y} \end{array} \right\}$$

is an EF of $P_1 \cap P_2$. □

3.3 SOME CANONICAL POLYTOPE FAMILIES

There are obvious clans – such as the one consisting of all polytopes – that have extension complexity unbounded by any function. Considering the clan $\mathbb{POLYGONS}$ of all polygons embedded in all dimensions, by Proposition 3.1.30 we already have that $\mathrm{xc}(\mathbb{POLYGONS})$ is not bounded by any function. One feature of such easily-produced high complexity clans is that describing the members may require unbounded precision. In any case we will be mostly interested in 0/1-polytopes which cannot have arbitrarily high extension complexity.

Proposition 3.3.1. *Let* $\mathbb{ZERO{-}ONE}$ *be the clan of all* $0/1$*-polytopes. Then,* $2^{\frac{n}{2}(1-o(1))} \leqslant xc(\mathbb{ZERO{-}ONE}) \leqslant 2^n$

Proof. The upper bound follows from Proposition 3.1.19 since any $0/1$ polytope in $\mathbb{R}^n$ has at most 2^n vertices. The lower bound follows from the fact that there are $0/1$ polytopes of such extension complexity [54]. □

So we see that the clan of all $0/1$ polytopes has extension complexity $2^{\Theta(n)}$. A family with such complexity can essentially by picked by selecting random polytopes for each dimension. It may be quite intuitive that this will result in a family of large extension complexity but proving it requires some very precise argument controlling the number of bits required to encode any extended formulation. Rothvoss was able to do exactly this and together with an elegant double counting argument was able to show the lower bound.

Such examples may be unsatisfactory because we do not get an explicit family of polytopes that has high extension complexity. A clan with high extension complexity becomes more interesting when we can describe the clan members and a specially hard family in the clan rather succintly. This is what was first done by Fiorini et al. [27] with polytopes related to the maxcut problem and later extended by various authors. We will see some of these clans in Chapter 4. Now we describe three canonical clans of polytopes that will play an important role later: the polytopes of cut vectors of graphs; the polytopes of satisfying assignments of CNF formulae; and the polytopes of non-satisfying assignments of CNF formulae.

3.3.1 *The* $\mathbb{CUT}$ *clan*

An important clan of polytopes that has high extension complexity is the that of Cut polytopes. These polytopes are naturally associated with the familiar NP-hard MAXCUT problem and have a rich history. We direct the reader to the textbook "Geometry of Cuts" by Deza and Laurant [21].

Definition 3.3.2. For a graph G the cut polytope of G – denoted by $CUT^{\square}(G)$ – is defined to be the convex hull of characteristic vectors of all cuts in G. Any polytope P such that $P = CUT^{\square}(G)$ for some graph G is called a cut polytope. The clan $\mathbb{CUT}$ is defined to be the set of all cut polytopes.

Proposition 3.3.3.

$$CUT^{\square}(K_{n+1}) = \left\{ x \in \{0,1\}^{\binom{n}{2}+n} \,\middle|\, x_{ij} = x_i \oplus x_j, i < j \right\}$$

Proof. See [2], proof of Theorem 5 (Appendix D). □

If we take $\mathcal{CUT}$ to be any family of cut polytopes that contains $CUT^{\square}(K_n)$ for each $n \in \mathbb{N}$ then this family has high extension complexity. This family was the first explicit family of polytopes that

was shown to have superpolynomial extension complexity. Here we present a combinatior of ideas that appeared in [2, 27, 41].

The crucial fact used for showing that the $\mathbb{CUT}$ clan has large extension complexity is that a certain matrix $\mathbf{U} = (\mathbf{a}^\top \mathbf{b} - 1)^2$ has large nonnegative rank (cf. Proposition 2.1.13). The next step is to show that this matrix is actually a submatrix of some slack matrix of $\mathrm{CUT}^{\square}(K_n)$. Then by Proposition 2.1.3 and Proposition 3.1.24 we get the desired lower bound on the extension complexity of the $\mathbb{CUT}$ clan.

The first step in embedding $\mathbf{U}(n-1)$ in a slack matrix of $\mathrm{CUT}^{\square}(K_n)$ is to identify some valid inequalities that produce the desired slack. The following lemma describes a set of such inequalities.

Lemma 3.3.4. *For any $n \geqslant 2$, let $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_n$ be any set of n integers. The following inequality is valid for $\mathrm{CUT}^{\square}(K_n)$:*

$$\sum_{1 \leqslant i < j \leqslant n} \mathbf{b}_i \mathbf{b}_j x_{ij} \leqslant \left\lfloor \frac{(\sum_{i=1}^{n} b_i)^2}{4} \right\rfloor \tag{4}$$

Proof. See [2], Lemma 1 (Appendix D). □

The inequality (4) is called *hypermetric* (respectively, of *negative type*) if the integers b_i can be partitioned into two subsets whose sum differs by one (respectively, zero). A simple example of hypermetric inequalities are the triangle inequalities, obtained by setting three of the b_i to be +/- 1 and the rest to be zero. The most basic negative type inequality is non-negativity, obtained by setting one b_i to 1, another one to -1, and the others to zero. We note in passing that Deza and Laurent (see Section 6.1 of [21]) showed that each negative type inequality could be written as a convex combination of hypermetric inequalities, so that none of them are facet inducing for $\mathrm{CUT}^{\square}(K_n)$.

Let $n \geqslant 2$ be an integer. Let $S \subseteq [n-1]$. This defines a cut $\delta(S)$ of K_n and each cut in K_n has such a subset of vertices defining it. Define a vector $\mathbf{b}$ with

$$\mathbf{b}_i = \begin{cases} 1, & \text{if } i \in S \\ 0, & n \neq i \notin S \\ 3 - |S| & \text{i=n} \end{cases}$$

Observe that $|S| = \mathbf{1}^\top \mathbf{b}$. Inequality (4) for this $\mathbf{b}$-vector is easily seen to be of negative type and can be written as

$$\sum_{1 \leqslant i < j \leqslant n-1} \mathbf{b}_i \mathbf{b}_j x_{ij} \leqslant 1 + (\mathbf{1}^\top \mathbf{b} - 3) \sum_{i=1}^{n-1} \mathbf{b}_i x_{in}. \tag{5}$$

Let $C \subseteq [n-1]$ and accordingly $\delta(C)$ be a cut of K_n. Define the vector $\mathbf{a}$ with

$$\mathbf{a}_i = \begin{cases} 1, & \text{if } i \in C \\ 0, & i \notin S \end{cases}$$

Proposition 3.3.5. *Let C and S be subsets of $[n-1]$. Then the slack of the cut $\delta(C)$ with respect to (5) is $(\mathbf{a}^\top \mathbf{b} - 1)^2$ with $\mathbf{a}, \mathbf{b}$ as defined previously.*

Proof.

$$\begin{aligned}
&1 + (\mathbf{1}^\top \mathbf{b} - 3) \sum_{i=1}^{n-1} \mathbf{b}_i \delta(C)_{in} - \sum_{1 \leqslant i < j \leqslant n-1} \mathbf{b}_i \mathbf{b}_j \delta(C)_{ij} \\
&= 1 + (\mathbf{1}^\top \mathbf{b} - 3)\mathbf{a}^\top \mathbf{b} - \mathbf{a}^\top \mathbf{b}(\mathbf{1}^\top \mathbf{b} - \mathbf{a}^\top \mathbf{b} - 1) \\
&= (\mathbf{a}^\top \mathbf{b})^2 - 2\mathbf{a}^\top \mathbf{b} + 1.
\end{aligned}$$

□

This implies that the extension complexity of $\mathrm{CUT}^{\square}_{(}K_n)$ is at least as large as $\mathrm{rank}_+(\mathbf{U}(n-1))$. That is,

Proposition 3.3.6. $\mathrm{xc}(\mathrm{CUT}^{\square}(K_n)) \geqslant 2^{\Omega(n)}$.

This in turn implies that the extension complexity of the $\mathbb{CUT}$ clan is exponential in the ambient dimension.

Proposition 3.3.7. $\mathrm{xc}(\mathbb{CUT}) \geqslant 2^{\Theta(\sqrt{n})}$. *In particular,* $\mathrm{xc}(\mathcal{CUT}) \geqslant 2^{\Theta(\sqrt{n})}$.

It is not a coincidence that the high lower bound of $\mathbb{CUT}$ is obtained by taking the family $\mathcal{CUT}$ of cut polytopes corresponding to the complete graphs. The specific family of complete graphs is in some sense the most general family of graphs for defining a hard family of cut polytopes as evidenced by the following.

Proposition 3.3.8. *Let* G *be any graph on* n *vertices. Then* $\mathrm{CUT}^{\square}(G)$ *is a projection of* $\mathrm{CUT}^{\square}(K_n)$.

Proof. If an edge (i, j) is not present in G, project it out. □

3.3.2 *The $\mathcal{CNF}$-$\mathcal{CERT}$ family*

For any given boolean formula φ with n variables define the polytope $\mathrm{SAT}(\varphi)$ as the convex hull of all satisfying assignments. That is,

$$\mathrm{SAT}(\varphi) := \mathrm{conv}(\{\mathbf{x} \in \{0,1\}^n \mid \varphi(\mathbf{x}) = 1\})$$

Since every 0/1 polytope is trivially the SAT polytope for some CNF formula, the corresponding clan of polytopes is just the clan $\mathbb{ZERO-ONE}$ which has high extension complexity as pointed out in Proposition 3.3.1. We will now show the existence of an easy to construct family of SAT polytopes that has superpolynomial extension complexity. The polytopes in this family will correspond to CNF formulae encoding the cuts of K_n as their satisfying assignments. This family of polytopes will be called the $\mathcal{CNF}$-$\mathcal{CERT}$ family, and will turn out to be a canonical family of polytopes with high extension complexity that will be used in Chapter 4 to give lower bounds for other families.

Let $n \in \mathbb{N}$ and $m = n^2$. For the complete graph K_n define a 3SAT boolean formula φ_m such that $\mathrm{CUT}^{\square}(K_n)$ is a projection of $\mathrm{SAT}(\varphi_m)$. Consider the relation $x_{ij} = x_{ii} \oplus x_{jj}$, where $\oplus$ is the xor operator. The boolean formula

$$(x_{ii} \vee \overline{x}_{jj} \vee x_{ij}) \wedge (\overline{x}_{ii} \vee x_{jj} \vee x_{ij}) \wedge (x_{ii} \vee x_{jj} \vee \overline{x}_{ij}) \wedge (\overline{x}_{ii} \vee \overline{x}_{jj} \vee \overline{x}_{ij})$$

is true if and only if $x_{ij} = x_{ii} \oplus x_{jj}$ for any assignment of the variables x_{ii}, x_{jj} and x_{ij}.

Therefore we define φ_m (with $m = n^2$) as

$$\varphi_m := \bigwedge_{\substack{i,j \in [n] \\ i \neq j}} \begin{bmatrix} (x_{ii} \vee \overline{x}_{jj} \vee x_{ij}) \wedge (\overline{x}_{ii} \vee x_{jj} \vee x_{ij}) \wedge \\ (x_{ii} \vee x_{jj} \vee \overline{x}_{ij}) \wedge (\overline{x}_{ii} \vee \overline{x}_{jj} \vee \overline{x}_{ij}) \end{bmatrix}. \tag{6}$$

This ensures that the the satisfying assignments of φ_m when restricted to the variables x_{ij} with $i \neq j$ are exactly the cut vectors of K_n and every cut vector of K_n can be extended to a satisfying assignment of φ. Consequently, we have that.

Proposition 3.3.9. $\mathrm{SAT}(\varphi_m)$ *is an* EF *of* $\mathrm{CUT}^{\square}(K_{n+1})$ *for each natural number* $m = n^2$.

Proof. This follows from Proposition 3.3.3. In fact, $\mathrm{SAT}(\varphi_m)$ is actually $\mathrm{CUT}^{\square}(K_{n+1})$. □

Definition 3.3.10. The $\mathcal{CNF}$-$\mathcal{CERT}$ family of polytopes is defined by the polytopes $P_m = \mathrm{SAT}(\varphi_m)$ with φ_m described previously by equation 6. For values of m with no corresponding φ_m we have $P_m = \emptyset$.

Note that $\mathrm{SAT}(\varphi_m)$ has $m = n^2$ variables and $4(m-n)$ clauses. Since $\mathrm{CUT}^{\square}(K_n)$ is a projection of $\mathrm{SAT}(\varphi_m)$, we can conclude that $\mathrm{xc}(\mathrm{SAT}(\varphi_m)) \geqslant \mathrm{xc}(\mathrm{CUT}^{\square}_{}) \geqslant 2^{\Omega(n)}$ (cf. Prop. 3.1.22), and thus,

Proposition 3.3.11. $\mathrm{xc}(\mathcal{CNF}\text{-}\mathcal{CERT}) = 2^{\Omega(\sqrt{n})}$.

Proof. This follows from Proposition 3.3.9. □

3.3.3 *The $\mathcal{DNF}$-$\mathcal{CERT}$ family*

Finally, we describe an interesting family of polytopes that has polynomial extension complexity. Similar to, and yet in contrast with the $\mathcal{CNF}$-$\mathcal{CERT}$ family, this family corresponds to the satisfying assignments of DNF formulae. Notice that whereas deciding satisfiability of a CNF formula is an NP-hard problem, deciding the same for a DNF formulae is trivial.

Let $\Phi = \{\varphi_1, \ldots\}$ be a family of DNF formulae where φ_n has n variables and $\mathrm{poly}(n)$ clauses. We will call the family $\{\mathrm{SAT}(\varphi_1), \ldots\}$ of polytopes a $\mathcal{DNF}$-$\mathcal{CERT}$ family. It is not important to pick a canonical representative of this family because as we will see next, the certificates of a polynomial sized DNF formula have polynomial extension complexity.

Proposition 3.3.12. *Let* φ *be a DNF formula with* n *variables and* m *clauses. Then* $\mathrm{xc}(\mathrm{SAT}(\varphi)) \leqslant 2mn$.

Proof. If φ consists of a single clause then it is just a conjunction of some literals. In this case $\mathrm{SAT}(\varphi)$ is a face of the n-hypercube and has $\mathrm{xc}(\mathrm{SAT}(\varphi)) \leqslant 2n$. Furthermore, for DNF formulae φ_1, φ_2 we have that $\mathrm{SAT}(\varphi_1 \vee \varphi_2) = \mathrm{SAT}(\varphi_1) \uplus \mathrm{SAT}(\varphi_2))$. Therefore, using Proposition 3.2.5 repeatedly we obtain that for a DNF formula φ with n variables and m clauses $\mathrm{SAT}(\varphi) \leqslant 2mn$. □

As a consequence we obtain the following.

Proposition 3.3.13. *Let* $\mathcal{DNF}\text{-}\mathcal{CERT} = \{P_1, P_2, \ldots\}$ *be a family of polytopes where* $P_n = \mathrm{SAT}(\boldsymbol{\varphi_n})$ *and* $\boldsymbol{\varphi_n}$ *a* DNF *formula with* n *varibles and* $\mathrm{poly}(n)$ *clauses. Then,* $\mathrm{xc}(\mathcal{DNF}\text{-}\mathcal{CERT}) = \mathrm{poly}(n)$.

Part II

RECIPES

會則事同一家　*With realization, things make one family;*
不會萬別千差　*Without realization, things are separated in a thousand ways.*
不會事同一家　*Without realization, things make one family;*
會則萬別千差　*With realization, things are separated in a thousand ways.*

— The Gateless Gate: Case 16 [38]

4

TURING REDUCTIONS

In Section 3.1 we saw some simple observations that make it possible to translate bounds on extension complexity of a polytope P to that of another polytope Q simply by demonstrating that Q essentially contains P (cf. Propositions 3.1.21, 3.1.22, and 3.1.23). Now we will see actual examples where these observations are put to use.

4.1 RELATIVES OF CUT POLYTOPES

4.1.1 *Cut polytope for minors of a graph*

Definition 4.1.1. Let $G = (V, E)$ be a graph. A graph $H = (V', E')$ is called a *minor* of G if an isomorphic copy of H can be obtained from G by a sequence of the following operations.

- *Vertex deletion* : $V' = V \setminus \{v\}$ and $E' = E \setminus \{e \in E \mid v \in e\}$ for some vertex $v \in V$.
- *Edge deletetion* : $V' = V$ and $E' = E \setminus \{e\}$ for some edge $e \in E$.
- *Edge contraction* : $V' = (V \setminus \{u, v\}) \cup \{w\}$ and $E' = E \setminus \{e \in E | u \in e \vee v \in e\} \cup \{(w, z) | (x, z) \in E, x \in \{u, v\}, z \notin \{u, v\}\}$ for some $u, v \in V$ and $w \notin V$.

It turns out that extension complexity is a monotone property under taking minors.

Proposition 4.1.2. *Let G be a graph and let H be a minor of G, then some face of $\mathrm{CUT}^{\square}(G)$ is an EF of $\mathrm{CUT}^{\square}(H)$. In particular,*

$$\mathrm{xc}(\mathrm{CUT}^{\square}(G)) \geqslant \mathrm{xc}(\mathrm{CUT}^{\square}(H)).$$

Proof. See [2], Theorem 12 (Appendix D). □

Using this together with Proposition 3.3.6 we can conclude the following.

Proposition 4.1.3. *The extension complexity of $\mathrm{CUT}^{\square}(G)$ for a graph G with a K_n minor is at least $2^{\Omega(n)}$.*

The cut polytope of the tripartite graph $K_{1,n,n}$ is called the Bell inequality polytope and plays an important role in Quantum Physics for the study of quantum entanglement [5]. We can conclude that this polytope cannot be represented by a polynomial number of inequalities even if we allow extra variables.

Proposition 4.1.4. $\mathrm{xc}(\mathrm{CUT}^{\square}(K_{1,n,n})) = 2^{\Omega(n)}$.

Proof. Pick any matching of size n between the vertices in each of the two parts of cardinality n. Contracting the edges in this matching yields K_{n+1} and the result follows. □

So we see that a large clique as a minor is sufficient for the cut polytope of a graph to have high extension complexity. Is it also necessary? Before we answer this question (in the negative) we discuss a polytope whose relation to cut polytope will become clear in Subsection 4.1.3.

4.1.2 *Stable set for cubic planar graphs*

Definition 4.1.5. Let $G = (V, E)$ be a graph. The convex hull of characteristic vectors of the independent sets in G is called the *stable set polytope* of G and is denoted by $\mathrm{STAB}(G)$. Any polytope P such that $P = \mathrm{STAB}(G)$ for some graph G is called a STAB polytope.

Since finding the largest independent set in arbitrary graphs is an NP-hard problem, it would be very surprising if the clan $\mathbb{STAB}$ of STAB polytopes had polynomial extension complexity. Indeed this is not the case and this clan was also one of the first to be shown to have superpolynomial extension complexity.

Proposition 4.1.6. $\mathrm{xc}(\mathbb{STAB}) = 2^{\Omega(\sqrt{n})}$.

Proof. See [27], Theorem 10 (Appendix A). □

In fact a family of STAB polytopes of cubic planar graphs can already has superpolynomial extension complexity.

Proposition 4.1.7. *There exists a family $\mathcal{STAB}$ of* STAB *polytopes of cubic planar graphs such that* $\mathrm{xc}(\mathcal{STAB}) = 2^{\Omega(\sqrt[4]{n})}$.

Proof. See [2], Corollary 5 (Appendix D). □

4.1.3 *Cut Polytope for K_6 minor-free graphs*

In Subsection 4.1.1 we saw that a large clique as a minor is sufficient for the cut polytope of a graph to have high extension complexity. Now we will see that a large clique minor is not necessary for high extension complexity of the cut polytope. In particular, there are K_6-minor free graphs whose cut polytopes have large extension complexity. Note that if a n-vertex graph G has no K_5-minor then $\mathrm{CUT}^{\square}(G)$ has $\mathcal{O}(n^3)$ extension complexity [21]. Contrast this with the fact that the MAXCUT problem is solvable in polynomial time on K_5 minor-free graphs but becomes NP-hard on K_6 minor-free graphs.

Definition 4.1.8. Let $G = (V, E)$ be any graph with $V = \{1, \ldots, n\}$. The *suspension* G' of G is obtained by adding an extra vertex labeled 0 with edges to all vertices V.

The operation of creating suspension of a graph is actually what relates the CUT and the STAB polytopes with each other.

Proposition 4.1.9. *Let* $G = (V, E)$ *be a graph and let* G' *be a suspension over* G. *Then* $\mathrm{STAB}(G)$ *is the projection of a face of* $\mathrm{CUT}^{\square}(G')$.

Proof. See [2], Theorem 13 (Appendix D). □

By Proposition 4.1.6 we we have a family of cubic planar graphs whose STAB polytopes give a family with superpolynomial extension complexity. Planar graphs do not contain K_5 as a minor and so the suspension of any planar graph is K_6 minor-free. This gives us a family of K_6-minor graphs whose cut polytopes have high extension complexity.

Proposition 4.1.10. *There exists a family* $\mathcal{CUT}'$ *of* CUT *polytopes of* K_6 *minor-free graphs such that* $\mathrm{xc}(\mathcal{CUT}') = 2^{\Omega(\sqrt[4]{n})}$.

This provides a sharp contrast for the complexity of the cut polytope for graphs in terms of their minors. As noted earlier, for any K_5 minor-free graph G with n vertices $\mathrm{CUT}^{\square}(G)$ has an extension of size $\mathcal{O}(n^3)$ whereas the above result shows that there are K_6 minor-free graphs whose cut polytope has superpolynomial extension complexity.

4.2 EMBEDDING ARGUMENTS FROM TURING REDUCTIONS

The central technique used in the previous section was to argue that a polytope P can be obtained as a projection of some face of polytope Q and then using the fact that $\mathrm{xc}(P)$ is large to argue that $\mathrm{xc}(Q)$ must be large too. How difficult is it to come up with a reduction that shows such an embedding?

Surprisingly it is quite common that for a polytope family related to an NP-hard problem the standard NP-hardness reduction also gives the desired embedding of one polytope family into another. In fact, the proofs of Propositions 4.1.7 and 4.1.9 are based on standard NP-hardness reductions for the associated problems. Now we present some more examples where the standard reductions suffice.

4.2.1 *Traveling Salesman*

Definition 4.2.1. Let G be a graph. The *traveling salesman polytope* of G – denoted by $\mathrm{TSP}(G)$ – is the convex hull of all Hamiltonian cycles of G. Any polytope P such that $P = \mathrm{TSP}(G)$ for some graph G will be referred to as a TSP polytope. The clan of all TSP polytopes is denoted by $\mathbb{TSP}$.

Similar to the $\mathbb{CUT}$ clan, complete graphs a canonical hard class of graphs for the extension complexity of the $\mathbb{TSP}$ clan. This is because of the following.

Proposition 4.2.2. *Let* G *be a graph on* n *vertices. Then* $\mathrm{TSP}(G)$ *is a face of* $\mathrm{TSP}(K_n)$.

Proof. For any edge (i, j) missing in G, restrict to the face of $\mathrm{TSP}(K_n)$ defined by the valid inequality $x_{ij} = 0$. □

The TSP problem asks whether a given graph contains a tour visiting every vertex exactly once and is known to be NP-hard. In fact, the standard NP-hardness reduction can be used to show superpolynomial lower bound on the extension complexity of the $\mathbb{TSP}$ clan.

Proposition 4.2.3. $\mathrm{xc}(\mathbb{TSP}) = 2^{\Omega(\sqrt[4]{n})}$.

Proof. See [27], Theorem 12 (Appendix A). □

Rothvoß [55] has improved the above bound to show that in fact $\mathrm{xc}(\mathbb{TSP}) = 2^{\Omega(\sqrt{n})}$. His bound again uses a standard embedding argument from perfect matching to TSP, but his lower bound for the perfect matching polytope uses new tools which are out of scope for us.

4.2.2 *Subset sum*

Definition 4.2.4. Given n integers $a^\top = (a_1, \ldots, a_n)$ and another integer b, the subset sum problems asks whether any subset of the set $\{a_1, \ldots, a_n\}$ sums exactly to b. Define the subset sum polytope $\mathrm{SUBSETSUM}(a, b)$ as the convex hull of all characteristic vectors of the subsets of a whose sum is exactly b.

$$\mathrm{SUBSETSUM}(a, b) := \mathrm{conv}\left(\left\{x \in \{0,1\}^n \mid a^\top x = b\right\}\right)$$

A polytope P which is $\mathrm{SUBSETSUM}(a, b)$ for some integers a, b will be a SUBSETSUM polytope.

The subset sum problem is, then, equivalent to asking whether the SUBSETSUM polytope for a given integer vector a and integer b is empty.

A related knapsack polytope can be defined as

$$\mathrm{KNAPSACK}(a, b) := \mathrm{conv}\left(\left\{x \in \{0,1\}^n \mid a^\top x \leqslant b\right\}\right)$$

Using the standard NP-hardness reduction for the subset sum problem one can show the following.

Proposition 4.2.5. *For every 3SAT formula φ with n variables and m clauses, there exists an integer vector $a(\varphi)^\top = (a_1, \ldots, a_{2n+2m})$ and integer $b(\varphi)$ such that $\mathrm{SAT}(\varphi)$ is a projection of $\mathrm{SUBSETSUM}(a, b)$.*

Proof. See [2], Theorem 6 (Appendix D). □

It immediately follows using Proposition 3.3.11 that there is a family of SUBSETSUM polytopes with high extension complexity.

Proposition 4.2.6. *Let $\mathbb{SUBSETSUM}$ be the clan of all SUBSETSUM polytopes. Then, $\mathrm{xc}(\mathbb{SUBSETSUM}) = 2^{\Omega(\sqrt{n})}$.*

Since the polytope $\mathrm{SUBSETSUM}(a, b)$ is a face of $\mathrm{KNAPSACK}(a, b)$, we have the following.

Proposition 4.2.7. *Let $\mathbb{KNAPSACK}$ be the clan of of KNAPSACK polytopes. Then, $\mathrm{xc}(\mathbb{KNAPSACK}) = 2^{\Omega(\sqrt{n})}$.*

4.2.3 *3d-matching*

Definition 4.2.8. Consider a hypergraph $G = ([n], E)$, where E contains triples (i, j, k) for some distinct $i, j, k \in [n]$. A subset $E' \subseteq E$ is said to be a 3-dimensional matching if all the triples in E' are disjoint. The 3d-matching polytope $3DM(G)$ is defined as the convex hull of the characteristic vectors of every 3d-matching of G. That is,

$$3DM(G) := \text{conv}(\{\chi(E') \mid E' \subseteq E \quad \text{is a 3d-matching}\})$$

It is often customary to consider only hypergraphs defined over three disjoint set of vertices X, Y, Z such that the hyperedges are subsets of $X \times Y \times Z$. Observe that any hypergraph G can be converted into a hypergraph H in such a form by making three copies of the vertex set V, V', V'' and using a hyperedge (i, j', k'') in H if and only if (i, j, k) is a hyperedge in G.

Exercise 4.2.9. Show that $xc(3DM(G)) = \Theta(xc(3DM(H)))$.

Definition 4.2.10. A polytope P is said to be a 3DM polytope if $P = 3DM(G)$ for some hypergraph G.

The 3d-matching problem asks: given a hypergraph G, does there exist a 3d-matching that covers all vertices? This problem is known to be NP-complete and was one of Karp's 21 problems proved to be NP-complete [31, 42]. This problem can be solved by linear optimization over the polytope $3DM(G)$ and therefore it is to be expected that $3DM(G)$ would not have a polynomial size EF for every hypergraph G.

Proposition 4.2.11. *For the clan* $3\mathbb{DM}$ *of* 3DM *polytopes we have that* $xc(3\mathbb{DM}) = 2^{\Omega(\sqrt[4]{n})}$.

This follows from the following Proposition whose proof relies on the standard NP-hardness reduction for the 3d-matching problem.

Proposition 4.2.12. *Let* φ *be a CNF formula with* n *variables and* m *clauses. Then there exists a hypergraph* $H = (V, E)$ *with* $|V| = \mathcal{O}(nm)$ *and* $|E| = \mathcal{O}(nm)$ *such that* $SAT(\varphi)$ *is the projection of a face of* $3DM(H)$.

Proof. See [2], Corollary 3 (Appendix D). □

4.2.4 *Induced matchings*

Definition 4.2.13. A matching in a graph $G = (V, E)$ is called induced if there is no edge in G between any pair of matching edges.

Stockmeyer and Vazirani [58] and Cameron [15] proved that the problem of finding a maximum cardinality induced matching is NP-hard.

Definition 4.2.14. Let G be a graph. The convex hull of all induced matchings G is called the *induced matching polytope* of G and is denoted by $\text{IndMatch}(G)$. A polytope P is said to be an IndMatch polytope if there exists a graph G such that $P = \text{IndMatch}(G)$.

Using the reduction in [15] one can show the following.

Proposition 4.2.15. *For every n there exists a bipartite graph G_n with $\mathcal{O}(n)$ edges and vertices such that* $\mathrm{xc}(\mathrm{IndMatch}(G)) = 2^{\Omega(\sqrt[4]{n})}$.

Proof. See [1], Theorem 1 (Appendix H). □

This implies the existence of a family with high extension complexity.

Proposition 4.2.16. *There exists a family $\mathcal{INDMATCH}$ of* IndMatch *polytopes such that* $\mathrm{xc}(\mathcal{INDMATCH}) = 2^{\Omega(\sqrt[4]{n})}$.

4.2.5 *Maximal matchings*

Definition 4.2.17. A matching in a graph $G = (V, E)$ is called *maximal* if its edge set is not included in a larger matching.

It is known that finding the minimum maximal matching is NP-hard [64].

Definition 4.2.18. Let G be a graph. The convex hull of all maximal matchings of G is called the maximal matching polytope of G and is denote it by $\mathrm{MaxMatch}(G)$. Accordingly a polytope P is called a MaxMatch polytope if $P = \mathrm{MaxMatch}(G)$ for some graph G.

Since the perfect matching polytope of K_{2n} has extension complexity $2^{\Omega(n)}$, we clearly have that the clan of MaxMatch polytopes has high extension complexity. However one can use the standard proof of NP-hardness of minimum maximal matching to show superpolynomial bound as well.

The only hurdle in using the reduction of [64] is that the reduction is from CNF formulae of special kind, namely formulae with at most one nonnegated literal and at most two negated literals in each clause. High extension complexity for the SAT polytopes of formulae of this specific kind can be shown by a fairly simple reduction.

Proposition 4.2.19. *For every n there exists a 3-CNF formula φ_n with $\mathcal{O}(n)$ variables and clauses such* $\mathrm{xc}(\mathrm{SAT}(\varphi_n)) \neq \mathrm{poly}(n)$. *Furthermore, in every clause of φ_n every variable appears at most twice non-negated and at most once negated.*

Proof. See [1], Theorem 2 (Appendix H). □

The reduction of [64] then gives the following.

Proposition 4.2.20. *For every n there exists a bipartite graph $G = (V_1 \cup V_2, E)$ with $\mathcal{O}(n)$ vertices and edges, such that* $\mathrm{xc}(\mathrm{MaxMatch}(G)) \neq \mathrm{poly}(n)$.

Proof. See [1], Theorem 3 (Appendix H). □

Thus we have the following.

Proposition 4.2.21. *Let $\mathbb{MAXMATCH}$ be the clan of* MaxMatch *polytopes. Then,* $\mathrm{xc}(\mathbb{MAXMATCH}) \neq \mathrm{poly}(n)$.

4.2.6 *Edge disjoint matching and perfect matching*

Given a bipartite graph $G(V_1 \cup V_2, E)$ and a natural number k, it is NP-hard to decide whether G contains a perfect matching M and a matching M' of size k such that M and M' do not share an edge [52].

For a given graph G with n vertices and m edges consider an encoding of a perfect matching and a matching using variables $x_1, \ldots, x_m$, $y_1, \ldots, y_m$ as follows. For a subset of edges encoding a perfect matching M and a matching M' of size k we construct a vector with

$$x_i = \begin{cases} 1, & \text{if } e_i \in M \\ 0, & \text{if } e_i \notin M \end{cases}, \quad y_i = \begin{cases} 1, & \text{if } e_i \in M' \\ 0, & \text{if } e_i \notin M' \end{cases}$$

Definition 4.2.22. Let $G = (V, E)$ be a graph. Define the polytope $\mathrm{MPM}(G, k)$ to be the convex hull of all the vectors encoding an edge disjoint perfect matching and a matching of size at least k. As usual, a polytope P such that $P = \mathrm{MPM}(G, k)$ for some graph G and natural number k is called an MPM polytope.

We would like to remark that one can also define a "natural" polytope here without using separate variables for a matching and a perfect matching and instead using the characteristic vectors of all subsets of edges that are an edge-disjoint union of a matching and a perfect matching. However, the formulation that we consider allows different cost functions to be applied to the matching and the perfect matching.

Using the reduction in [52], one can show that for every n there exists a bipartite graph G_n with $\mathcal{O}(n)$ vertices and a constant $0 < c < \frac{1}{2}$ such that $\mathrm{MPM}(G, cn)$ has extension complexity super polynomial in n. Again the reduction in [52] is from MAX-2-SAT, so we first need to prove a super polynomial lower bound for the SAT polytopes of 2-CNF formulas.

Proposition 4.2.23. *For every n there exists a 2-SAT formula φ_n with n variables such that* $\mathrm{xc}(\mathrm{SAT}(\varphi_n)) = 2^{\Omega(\sqrt[4]{n})}$.

Proof. See [1], Theorem 4 (Appendix H). □

As a side remark, the 2-SAT instances required in the above theorem are always satisfiable.

Proposition 4.2.24. *For every n there exists a bipartite graph G_n on n vertices and a constant $0 < c < \frac{1}{2}$ such that* $\mathrm{xc}(\mathrm{MPM}(G, cn)) \neq \mathrm{poly}(n)$.

Proof. See [1], Theorem 5 (Appendix H). □

Thus, we have the following.

Proposition 4.2.25. *There exists a family $\mathcal{MPM}$ of* MPM *polytopes such that* $\mathrm{xc}(\mathcal{MPM}) \neq \mathrm{poly}(n)$.

4.3 DIFFICULITIES IN HANDLING GENERAL REDUCTIONS

Seeing so many NP-hardness reductions yield superpolynomial extension complexity lower bounds for the associated polytopes, it was suspected[1] that it may be possible to prove a meta result. A result similar to: "A problem is in PTIME if and only if the associated polytope has polynomial extension complexity". Notwithstanding what "associated polytope of a problem" meant, this was shown to be impossible in a remarkable paper [55] that showed that the perfect matching polytope has exponential extension complexity (See Proposition 3.1.13).

It should be noted that it may still be possible to have the polytopes associated with NP-complete problems always have superpolynomial extension complexity. However a more likely scenario may be that the answer depends crucially on how one associates polytopes with problems, and for various choices of such associations the extension complexity may be trivially exponential while for others trivially polynomial, and for yet others very difficult to determine.

In any case, for all we know PTIME reductions that are allowed for proving NP-hardness may be as powerful as NP itself and so being able to translate superpolynomial lower bounds on extension complexity using arbitrary polynomial reductions may be too much to ask.

1 At least by the present author

5

COMPACT LANGUAGES

Let φ be a boolean formula. Consider the following languages:

$$\begin{aligned} L &= \{x \mid x \text{ encodes a satisfiable boolean formula}\} \\ L((\varphi)) &= \{x \mid \varphi(x) = 1\} \end{aligned}$$

The former language consists of all strings that encode[1] all satisfiable boolean formulae, while the later language consists of all satisfying assignments of a given boolean formula. Which of these represents the boolean satisfiability problem *more naturally*?

Reasonable people will agree that there is no correct choice of a natural polytope for a problem. One complication is that there various kinds of problems: decision, optimization, enumeration, etc, and very similar problems can have very different behaviour if the notion of problem changes.

Example 5.0.1. Checking whether a bipartite graph has a perfect matching can be solved by a simple polynomial time algorithm. A related problem where one wants to count the number of bipartite matchings is #P-hard.

Therefore one can pick any clan of polytopes, that they consider reasonable, as representing a given problem but it can be asked whether the extension complexities of that particular choice of polytopes reflect some underlying complexity measure of the problems. Often the most immediate choice of polytopes does not really correspond well to the computational complexity.

Example 5.0.2. The clan $\mathbb{EP}$ of perfect matching polytopes is a natural choice for the underlying decision problem: given a graph G, does it have a perfect matching? While the computational problem is polynomial time solvable, the extension complexity of Edmonds' polytopes is exponential.

One reason for such complication is that wildly different kinds of problems are defined over the same set of objects. For example, over the set of graphs and their perfect matchings, we can ask natural decision, optimization, and counting questions. The first two are polynomial time solvable while the last one is #P-hard.

Our perspective of the situation will be as follows. Algorithms will be identified with Turing machines with five tapes[2].

- A two-way read-only input tape.

1 Assume that some (arbitrary but fixed) encoding of boolean formulae as binary strings.

2 This does not make the Turing machines special.

- An auxiliary read-only input tape than can be read only from left to right.
- A two-way read-write work tape.
- A two-way write-only output tape.
- An auxiliary write-only output tape that can be written only from left to right.

Computational problems are then questions about existence of Turing machines of the above kind with various restrictions on the consumption of resources such as space consumed on the work tape or the overall time, and on the relation between the contents of the input tapes and the output tapes when the machine halts.

5.1 PROBLEMS AS LANGUAGES

For us computational problems will just be questions about an underlying language. Various natural problems can be modeled in this way. Usually a specific computational problem comes with an underlying language $\mathbf{L}$. The specific problem at hand is then some question about the language $\mathbf{L}$ (or about strings of this language).

5.1.1 *Membership Problem*

The membership problems asks whether a given string belongs to a particular language. We will denote such problems by $\text{mem}(\mathbf{L})$.

Example 5.1.1. Let $\mathbf{L} = \{x \mid x \text{ encodes a satisfiable boolean formula}\}$. The problem $\text{mem}(\mathbf{L})$ then is just the familiar boolean satisfiability problem where an encoding of the input has been agreed upon.

5.1.2 *Optimization Problem*

The optimization problem for a particular language $\mathbf{L}$ comes equipped with a function that assigns a real number to every string in the language and one is interested in finding the "best" string from $\mathbf{L}$.

A particularly important class of such problems is one of linear optimization. Given a cost vector $\mathbf{c} \in \mathbb{R}^n$ one is interested in a string $\mathbf{x}^* \in \mathbf{L}$ with $|\mathbf{x}^*| = n$ such that for every $\mathbf{x} \in \mathbf{L}$ with $|\mathbf{x}| = n$ we have that $\mathbf{c}^\top \mathbf{x}^* \geqslant \mathbf{c}^\top \mathbf{x}$.

We will denote by $\text{opt}(\mathbf{L})$ the problem of maximizing a linear function over all members of $\mathbf{L}$ that have length n.

Example 5.1.2. Given a linear cost function and a boolean formula φ, find a satisfying assignment (if any) of minimum cost.

5.1.3 *Enumeration Problem*

The canonical enumeration problem for a particular language $\mathbf{L}$ – denoted by $\text{enum}(\mathbf{L})$ – is as follows: given a number n enumerate all members of $\mathbf{L}$ that have length n.

Example 5.1.3. Enumerate all satisfying assignments of a given boolean formula.

5.1.4 *Sampling Problem*

The canonical sampling problem for a particular language $\mathbf{L}$ – denoted by sample($\mathbf{L}$) – is as follows: given a number n produce a length n member of $\mathbf{L}$ with a given probability distribution.

Example 5.1.4. Given a boolean formula, produce a satisfying assignment (if any) uniformly at random.

5.1.5 *Counting problem*

The canonical counting problem for a particular language $\mathbf{L}$ – denoted by count($\mathbf{L}$) – is as follows: given a number n how many members of $\mathbf{L}$ have length exactly n?

Example 5.1.5. How many satisfying assignents does a given boolean formula have?

5.2 COMPACT LANGUAGES

5.2.1 *Languages to Polytopes*

For every natural number n define the set $\mathbf{L}(n) := \{\mathbf{x} \in \{0,1\}^n \mid \mathbf{x} \in \mathbf{L}\}$. Viewing each string $\mathbf{x} \in \mathbf{L}(n)$ as a column vector, and ordering the strings lexicographically, we can view the set $\mathbf{L}(n)$ as a matrix of size $n \times |\mathbf{L}(n)|$. Thus we are in a position to naturally associate a family of polytopes with a given language and the extension complexity of these polytopes can serve as a natural measure of how hard is it to model these languages as Linear Programs.

That is, one can associate with $\mathbf{L}$, the family of polytopes $\mathcal{P}(\mathbf{L}) = \{P(\mathbf{L}(1)), P(\mathbf{L}(2)), \ldots\}$ and the extension complexity $\mathrm{xc}(\mathcal{P}(\mathbf{L}))$ is then an intrinsic measure of complexity of the language $\mathbf{L}$.

Example 5.2.1. Matching polytope of complete graphs with a canonical (say lexicographic) ordering on the edges. The associated language consists of the characteristic vectors of all perfect matchings of K_n for $n \in \mathbb{N}$.

Definition 5.2.2. The *extension complexity* of a language $\mathbf{L}$ – denoted by $\mathrm{xc}(\mathbf{L})$ – is defined by $\mathrm{xc}(\mathbf{L}) := \mathrm{xc}(\mathcal{P}(\mathbf{L}))$.

Now we are ready to define the class of languages that we are interested in: namely, the languages that have small extension complexities.

Definition 5.2.3. $\mathcal{CF}$ is the class of languages admitting $\mathcal{C}$ompact extended $\mathcal{F}$ormulations and is defined as

$$\mathcal{CF} \quad = \quad \{\mathbf{L} \subseteq \{0,1\}^* \mid \exists c > 0 \text{ s.t. } \mathrm{xc}(\mathbf{L}) \leqslant n^c\}$$

5.2.2 *Easy problems for Compact languages*

Let $\mathbf{L}$ be a compact language. That is for each $n \in \mathbb{N}$ the polytope $P(\mathbf{L}(n))$ has small extension complexity. What does that give us in terms of solving computational problems related to $\mathbf{L}$? Naturally, one would need such an extended formulation itself to be efficiently computable. It may very well happen that a polytope has a small extension but the actual numbers needed to represent any small sized extension require very large precision. In fact it is unknown whether small extension complexity using real numbers implies small extension complexity using rational numbers.

Notwithstanding the previous discussion, let us assume that language $\mathbf{L}$ has a small extension that is also efficiently constructible. By efficiently constructible, we mean that given n we can construct an extension of $P(\mathbf{L}(n))$ requiring $\text{poly}(n)$ bits to describe in time $\text{poly}(n)$. What do we gain in this case?

Proposition 5.2.4. *Let $\mathbf{L}$ have an efficiently constructible extended formulation of size $s(n)$. Then, mem$(\mathbf{L})$ and opt$(\mathbf{L})$ can be solved in time $\text{poly}(s(n) + n)$.*

Proof. Let $Q = \{(\mathbf{x}, \mathbf{y} \mid \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leqslant \mathbf{c}\}$ be an EF of $P(\mathbf{L}(n)$. Checking whether a given $\mathbf{x}^*$ belongs to $\mathbf{L}$ or not can be done by checking the feasibility of $Q \cap \{\mathbf{x} = \mathbf{x}^*\}$. The problem opt$(()\mathbf{L})$ is also easily solved by standard Linear Programming. □

Proposition 5.2.5. *Let $\mathbf{L}$ have an efficiently constructible extended formulation of size $s(n)$. Then, enum$(\mathbf{L})$ and sample$(()\mathbf{L})$ can be solved in time $\text{poly}(s(n) + n + h(n))$ where $h(n)$ is the number of strings of length n.*

Proof. This follows from the fact that vertices of a zero-one polytope can be enumerated in strongly polynomial time [14]. For the method of Bussiek and Lübbecke to work, we only need to check whether a given face of the zero-one polytope is empty or not.

For the sampling problem one can just start the enumeration and select any of the output string uniformly at random on the fly. Note that one does not have to store the entire output to select an element with uniform distribution. □

Proposition 5.2.6. *The problem count$(\mathbf{L})$ may be #P-hard even if $P(\mathbf{L}(n))$ has small size.*

Proof. This follows from the fact that the polytope of perfect matchings of bipartite graphs has small description, but counting the number of perfect matchings in bipartite graphs is #P-hard. □

5.3 CLOSURE PROPERTIES

Now we discuss the closure properties of the class $\mathcal{CF}$ with respect to some common operations on formal languages. The operations that we consider are as follows.

- **Complement** : $\overline{L} = \{x \mid x \notin L\}$
- **Union** : $L_1 \cup L_2 = \{x \mid x \in L_1 \vee x \in L_2\}$
- **Intersection** : $L_1 \cap L_2 = \{x \mid x \in L_1 \wedge x \in L_2\}$
- **Set difference** : $L_1 \setminus L_2 = \{x \mid x \in L_1 \wedge x \notin L_2\}$
- **Concatenation** : $L_1 L_2 = \{xy \mid x \in L_1 \wedge y \in L_2\}$
- **Kleene star** : $L^* = L \cup LL \cup LLL \cup LLLL \cup \ldots$

Proposition 5.3.1. *$\mathcal{CF}$ is not closed under taking complement.*

Proof. Consider the family $\mathcal{CNF}$-$\mathcal{CERT}$ of polytopes discussed in Subsection 3.3.2. Let $\mathbf{\Phi}$ be the family of boolean formulae corresponding to this polytope family, and let L be the language consisting of all binary strings that correspond to a vertex of some polytope in $\mathcal{CNF}$-$\mathcal{CERT}$. That is, L is the language of all cut vectors of K_n for $n \in \mathbb{N}$, viewed as binary strings.

Now consider the family of DNF formulae $\overline{\mathbf{\Phi}} = \{\overline{\varphi} \mid \varphi \in \mathbf{\Phi}\}$. Notice that the language of all certifying assignments of formulae in $\overline{\mathbf{\Phi}}$ is precisely $\overline{L}$.

We see that $\mathcal{P}(\overline{L}) = \mathcal{DNF}$-$\mathcal{CERT}$ while $\mathcal{P}(L) = \mathcal{CNF}$-$\mathcal{CERT}$. Propositions 3.3.11 and 3.3.13 state that $\text{xc}(\mathcal{DNF}\text{-}\mathcal{CERT}) = \text{poly}(n)$ while $\text{xc}(\mathcal{CNF}\text{-}\mathcal{CERT}) \neq \text{poly}(n)$. Therefore $\overline{L} \in \mathcal{CF}$ and $L \notin \mathcal{CF}$. □

Proposition 5.3.2. *$\mathcal{CF}$ is closed under taking union.*

Proof. Let L_1 and L_2 be two languages. Then, $\text{xc}(L_1 \cup L_2) \leqslant \text{xc}(L_1) + \text{xc}(L_2)$ (cf. Proposition 3.2.5). □

Proposition 5.3.3. *$\mathcal{CF}$ is not closed under taking intersection.*

Proof. Let L_1 be a language such that a string $x \in L_1$ if and only if it satisfies the following properties.

- $|x| = (n+1)\binom{n}{2}$ for some natural number n, and
- $x_{ij(n+1)} = x_{iji} \oplus x_{ijj}$ if the characters are indexed as x_{ijk} with $1 \leqslant i < j \leqslant n$, $1 \leqslant k \leqslant n+1$.

We claim that $\text{xc}(L_1) = \mathcal{O}(n^3)$. Indeed $P\left(L_1\left((n+1)\cdot\binom{n}{2}\right)\right)$ is the product of polytopes

$$P_{ij} = \left\{x \in \{0,1\}^{n+1} \mid x_{n+1} = x_i \oplus x_j\right\}$$

for $1 \leqslant i < j \leqslant n$ and $\text{xc}(P_{ij}) = \mathcal{O}(n)$ (cf. Example 3.1.26).

Now let L_2 be a language such that a string $x \in L_2$ if and only if it satisfies the following properties.

- $|x| = (n+1)\binom{n}{2}$ for some natural number n, and
- $x_{i_1 j_1 k} = x_{i_2 j_2 k}$ for all $k \in [n], i \neq j \in [n]$

Each polytope $P\left(L_1\left((n+1)\cdot\binom{n}{2}\right)\right)$ is just an embedding of $\square_{n+\binom{n}{2}}$ in $\mathbb{R}^{(n+1)\binom{n}{2}}$ and therefore, $\text{xc}(L_2) = \mathcal{O}(n^2)$.

Finally, observe that for $m = (n+1)\binom{n}{2}$ the polytope $P((L_1 \cap L_2)(m))$ when projected to the coordinates labelled $x_{ij(n+1)}$ is just the polytope $\text{CUT}^{\square}_n$ (cf. Proposition 3.3.3). Therefore, $\text{xc}(L_1 \cap L_2) = 2^{\Omega(n)}$ and even though $L_1, L_2 \in \mathcal{CF}$, the intersection $L_1 \cap L_2 \notin \mathcal{CF}$. □

Proposition 5.3.4. *$\mathcal{CF}$ is not closed under taking set difference.*

Proof. The complete language $\{0,1\}^*$ clearly belongs to $\mathcal{CF}$. For any language $\mathbf{L}$ we have $\overline{\mathbf{L}} = \{0,1\}^* \setminus \mathbf{L}$. If $\mathcal{CF}$ were closed under taking set-difference, it would also be closed under taking complements. But as pointed out in Proposition 5.3.1, it is not. □

Proposition 5.3.5. *$\mathcal{CF}$ is closed under concatenation.*

Proof. $P(\mathbf{L}_1\mathbf{L}_2(n))$ is the union of the polytopes $P(\mathbf{L}_1(i)) \times P(\mathbf{L}_2(n-i))$ for $i \in [n]$. Therefore, using Propositions 3.2.3 and 3.2.5 we have that $\mathrm{xc}(\mathbf{L}_1\mathbf{L}_2) \leqslant n(\mathrm{xc}(\mathbf{L}_1) + \mathrm{xc}(\mathbf{L}_2))$. □

Proposition 5.3.6. *$\mathcal{CF}$ is closed under taking Kleene star.*

Proof. Let $\mathbf{L} \in \mathcal{CF}$. For $0 \leqslant k \leqslant n$, consider the polytope P_k defined as

$$P_k := \mathrm{conv}\left(\left\{ \begin{pmatrix} e^{n+1}_{i+1} \\ 0^i \\ x \\ 0^{n-i-k} \\ e^{n+1}_{i+|x|+1} \end{pmatrix} \in \{0,1\}^{3n+2} \;\middle|\; \begin{array}{ll} & x \in \mathbf{L} \\ \wedge & |x| = k \\ \wedge & 0 \leqslant i \leqslant n-k \end{array} \right\}\right)$$

Define $P := \cup_{j=0}^{n} P_j$. Then, $\mathrm{xc}(P) \leqslant \sum_{k=0}^{n} \mathrm{xc}(P_k) \leqslant \sum_{k=0}^{n} (n\,\mathrm{xc}(P(\mathbf{L}(k)))) \leqslant \mathcal{O}(n^2\,\mathrm{xc}(\mathbf{L}))$.

Let S_0 be the face of P defined by the first n coordinates being 0 and the $(n+1)$-th coordinate being 1. Construct S_{i+1} by taking the glued product of S_i with P over the last $n+1$ coordinates of S_i and the first $n+1$ coordinates of Q.

Take the face R of S_n defined by the last n coordinates being 0 and the $(n+1)$-th penultimate coordinate being 1. Then, R is an EF for $P(\mathbf{L}^*(n))$. Moreover, $\mathrm{xc}(R) \leqslant \mathrm{xc}(S_n) \leqslant (n+1)\,\mathrm{xc}(P) \leqslant \mathcal{O}(n^3\,\mathrm{xc}(\mathbf{L}))$.

Therefore, $\mathrm{xc}(\mathbf{L}^*) = \mathcal{O}(n^3\,\mathrm{xc}(\mathbf{L}))$ and $\mathbf{L}^* \in \mathcal{CF}$. □

6 ONE-PASS LANGUAGES

6.1 ONLINE TURING MACHINES

An online Turing machine is a two tape Turing machine where one of the tapes stores the input and can be read only from left to right. The second tape is the work tape and the machine can read and write freely on it and the head is free to move in any direction. The space consumed on the worktape in the worst case is the measure of space complexity.

6.1.1 *History*

Online Turing machines that require only logarithmic space on the work tape were considered by Hartmanis, Immerman, and Mahaney [36] to restrict the power of reductions between two problems. Traditionally, for establishing equivalence of problems arbitrary polynomial time reduction between them is allowed. In the light of the fact that we do not know whether PTIME is different from NP, such reductions may be misleading. In fact, even the possibly smaller class of LOGSPACE problems are not known to be different from the class NP and so even logspace reductions between problems may be misleading about their true complexity.

It is known that one-pass logspace Turing machines can only accept regular languages [57, 59] and therefore if two problems are reducible to each other using only one-pass logspace reduction, they are equivalent in a stronger sense.

6.1.2 *Determinism vs. Non-determinism*

For online Turing machines requiring at least logarithmic space, nondeterminism allows provably stronger machines. Non-regular languages can be accepted by non-deterministic machines using logarithmic space while any one-pass deterministic logspace Turing machine can only accept regular languages [59].

6.2 EXTENSION COMPLEXITY OF ONE-PASS LANGUAGES

Definition 6.2.1. The complexity class $k\text{-NSPACE}(s(n))$ is the class of languages accepted by a k-pass non-deterministic Turing machines using space $s(n)$. Similarly, the complexity class $k\text{-DSPACE}(s(n))$ is the class of languages accepted by a k-pass deterministic Turing machine using space $s(n)$.

What is the extension complexity of any language in this class? Before we answer this we note the following.

Proposition 6.2.2. $\mathbf{L} \in \text{k-NSPACE}(s(n)) \implies \mathbf{L} \in \text{1-NSPACE}(ks(n))$.

Proof. Let M_n be the Turing machine that accepts strings of length n. We will simulate M_n using a multi-tape single pass nondeterministic Turing machine called the simulator S. S is supplied with $p(n)$ work tapes. S starts by guessing the initial work state of M_n at the start of i-th pass and writing them on the i-th work tape. S then simulates (using extra space on each work tape) each of the passes independently starting from their respective initial configuration. Once the entire input has been scanned, the simulator verifies that the work space of M_n on the i-th tape at the end of the pass matches the guess for the initial content for the $(i+1)$-th tape. S will accept only if the last tape is in an accepting state.

To store the content of work tape and the current state, S needs $s(n) + o(s(n))$ space for each pass. Thus S uses a single pass and total space of $p(n)s(n)(1 + o(1))$. By Proposition 6.2.8 the extension complexity of the strings accepted by M_n is then $2^{\mathcal{O}(p(n)s(n))}n$. □

Thus for our purposes it suffices to restrict our attention to single pass TMs. In the next subsection we describe the polytope associated with walks in a directed graph, that will help us bound the extension complexity of such languages.

6.2.1 *Walks in directed graphs*

Definition 6.2.3. Let $D = (V, A)$ be a directed graph with every edge labeled either zero or one. Consider two nodes $u, v \in V$ and a walk ω of length n from u to v. The *signature* of ω – denoted by σ_ω – is the sequence of edge labels along the walk ω. The node u is called the *source* of the walk and the node v the *destination*.

Definition 6.2.4. Consider the convex hull of all zero-one vectors of the form (u, σ, v) where u and v are indices of two nodes in D and σ is the signature of some walk of length n from u to v. This polytope – denoted by $P_{\text{markov}}(D, n)$ – is called the *Markovian polytope of* D.

Proposition 6.2.5. *Let $D = (V, A)$ be directed graph (possibly with self-loops and multiple edges) with every edge labeled either zero or one. Then, $P_{\text{markov}}(D, n)$ has extension complexity at most $2|V| + |A| \cdot n$.*

Proof. Let us encode every vertex of D with a zero-one vector of length V such that the unit vector e_i represents vertex i.

Define polytope $P_{\text{trans}} \subset \{0,1\}^{|V|+1+|V|}$ with $(a, z, b) \in \{0,1\}^{|V|+1+|V|}$ a vertex of P_{trans} if and only if it encodes a possible transition in D. That is, a and b encode vertices of V, and the coordinate z represents the label of the edge following which one can move from a to b. Since P_{trans} has at most $|E|$ vertices $\text{xc}(P_{\text{trans}}) \leqslant |E|$ (cf: Proposition 3.1.19).

Let P_0 be the convex hull of (i, e_i) for $i \in V$ and P_f be the convex hull of (e_i, i) for $i \in V$. Observe that the two polytopes are the same except for relabeling of coordinates. Also, $\text{xc}(P_0) = \text{xc}(P_f) \leqslant |V|$.

Let $P_1 = P_{\text{trans}}$. For $2 \leqslant i \leqslant n$, construct the polytope P_i by glueing the last $|V|$ coordinates of P_{i-1} with the first $|V|$ coordinates of P_{trans}. By Proposition 3.2.4 we have that $\text{xc}(P_n) \leqslant |E| \cdot n$.

Finally, let P be the polytope obtained by glueing last $|V|$ coordinates of P_0 with the first $|V|$ coordinates of P_n, and then glueing the last $|V|$ vertices of the result with the first $|V|$ coordinates of P_f. Note that $xc(P) \leqslant 2|V| + |E| \cdot n$.

To complete the proof, notice that P is an extended formulation for $P_{markov}(D, n)$. In particular, projecting out every coordinate except the ones corresponding to the source node in P_0, the ones corresponding to the destination node in P_f, and ones that correspond to the z coordinates in all the copies of P_{trans} produces exactly the vertices of $P_{markov}(D, n)$. The z-coordinate corresponding to the i-th copy of P_{trans} corresponds to the i-th index of signatures in the vectors in $P_{markov}(D, n)$. □

6.2.2 *Extension complexity of single-pass machines*

Definition 6.2.6. The *configuration graph* for input of length n for a given one-pass Turing machine (deterministic or non-determinisitic) is constructed as follows. For each fixed n, consider the directed graph whose nodes are marked with a label consisting of $s(n) + \lceil \log(s(n)) \rceil$ characters. The labels encode the complete configuration of the Turing machine: the content of the worktape and head position on the worktape. We make directed edges between two nodes u and v if the machine can reach from configuration u to configuration v by a sequence of transitions with exactly one input bit read in between. The directed edge is labeled by the input bit read during this sequence of transition.

Finally, we add two special nodes: a start node with a directed edge to each possible starting configuration of the machine, and a finish node with a directed edge from each possible accepting configuration. Each of these directed edges are labeled by zero.

Proposition 6.2.7. *The configuration graph for input of length n for a one-pass Turing machine has $\mathcal{O}(2^{s(n)}s(n))$ nodes. If the Turing machine is non-deterministic, this graph has $\mathcal{O}(4^{s(n)}(s(n))^2)$ edges. If the Turing machine is deterministic then this graph has $O(2^{s(n)}s(n))$ edges.*

Proof. The bound for number of nodes is clear from the construction of the configuration graph. We can have at most two transition edges between any two (possibly non-distinct) nodes: one corresponding to reading a zero on the input tape, and one corresponding to reading a one. Therefore, asymptotically the configuration graph can have at most square of the number of nodes.

For deterministic Turing machine, each node in the configuration graph has exactly two outgoing edges (possibly to the same node). Therefore the number of edges is asymptotically the same as the number of vertices. □

Now Proposition 6.2.5 can be used to bound the extension complexity of language accepted by one-pass machines.

Proposition 6.2.8. *Let $L \in$ 1-NSPACE$(s(n))$. Then,*

$$xc(L) = \mathcal{O}(4^{s(n)}(s(n))^2 \cdot n).$$

Proof. Let $\mathbf{L} \in$ 1-NSPACE$(s(n))$ be a language. That is, there exists a Turing machine that when supplied with a string on the one-way input tape uses at most $s(n)$ cells on the worktape, makes a single pass over the input and then accepts or rejects the input. If the input string is in $\mathbf{L}$, some sequence of non-deterministic choices lead the machine to an accepting state, otherwise the machine always rejects.

The length-n strings that are accepted by such a Turing machine correspond exactly to the signatures of length $n+2$ walks on the corresponding configuration graph D. The first and the last character of these strings is always zero. Therefore, an extended formulation for $P(\mathbf{L}(n))$ is obtained by taking the face of $P_{\text{markov}}(D, n+2)$ corresponding to walks that start and the start node and finish at the finish node. By Proposition 6.2.5 $P_{\text{markov}}(D, n+2)$ has extension complexity $\mathcal{O}(4^{s(n)}(s(n))^2 \cdot n$, and by Proposition 3.1.23 so does the desired face. □

If $\mathbf{L}$ is accepted by a one-pass deterministic TM then one can do better because the configuration graph has fewer edges.

Proposition 6.2.9. *Let* $\mathbf{L} \in$ 1-DSPACE$(s(n))$. *Then,*

$$\text{xc}(\mathbf{L}) = \mathcal{O}(2^{s(n)}s(n) \cdot n).$$

6.2.3 *Extensions for multiple-pass machines*

Proposition 6.2.10. *Let* $\mathbf{L} \in$ p-NSPACE$(s(n))$. *Then,*

$$\text{xc}(\mathbf{L}) = 2^{\mathcal{O}(p(n)s(n))}n.$$

Proof. This follows immediately from Propositions 6.2.2 and 6.2.8. □

Proposition 6.2.11. *Let* $\mathcal{M}$ *be a (not necessarily uniform) family of deterministic online Turing machines. Let the number of passes and the space used by the family be bounded by functions,* $p(n), s(n)$ *respectively. Let* $\mathbf{L}(M)$ *be the language accepted by* M. *Then,* $\text{xc}(L(\mathcal{M})) \leqslant 2^{\mathcal{O}(p(n)s(n))}n$.

Proposition 6.2.12. *If* $\mathbf{L}$ *is accepted by a fixed-pass non-deterministic logspace Turing machine then* $\mathbf{L} \in \mathcal{CF}$.

We end this section with the following remark. For a language to be compact (that is, to have polynomial extension complexity), it is sufficient to be accepted by an online Turing machine (deterministic or not) that requires only logarithmic space. However, this requirement is clearly not necessary. This can be proved by contradiction: Suppose that the condition is necessary. Then the class of compact languages must be closed under taking intersection. (Simply chain the two accepting machines and accept only if both do). Since we have already established (cf. Proposition 5.3.3) that the class of compact languages is not closed under taking intersection, we have a contradiction.

6.3 APPLICATIONS

6.3.1 *Streaming lower bounds*

Reading Proposition 6.2.10 in reverse readily yields lower bounds in the streaming model of computation. We illustrate this by an example.

Example 6.3.1. We know that the perfect matching polytope of the complete graph K_n has extension complexity $2^{\Omega(n)}$. Any $p(n)$-pass algorithm requiring space $s(n)$, that correctly determines whether a given stream of $\binom{n}{2}$ is the characteristic vector of a perfect matching in K_n, must have $p(n)s(n) = \Omega(n)$. This bound applies even to nondeterministic algorithms.

In fact Proposition 6.2.5 provides an even stronger lower bound.

Definition 6.3.2. Let $\mathbf{L} \subseteq \{0,1\}^n$ be a language. $\mathbf{L}$ is said to be online μ-magic if there exists a Turing machine T that accepts $\mathbf{L}$ with the following oracle access. On an input of length n on the one-way input tape, the machine T scans the input only once. T may prepare its working tape to describe any well-formed function $f : \{0,1\}^{\mu(n)} \to \{0,1\}^{m u(n)}$ and a particular input x and invoke the oracle that changes the contents of the work-tape to $f(x)$. The machine must always reject strings not in $\mathbf{L}$. For strings in $\mathbf{L}$ there must be some possible execution resulting in accept.

Notice that even the working of such a machine can be encoded in terms of the configuration graph where the transitions may depend arbitrarily but in a well-formed way on the contents of the work-tape.

Proposition 6.3.3. *If the set of characteristic vectors of perfect matchings in K_n are accepted by an online μ-magic Turing machine, then $\mu(n) = \Omega(n)$.*

Thus we see that extension complexity lower bounds highlight deep limitations of the streaming model: even powerful oracles do not help solve in sublinear space problems that are LOGSPACE solvable if the one-way restriction on the input is removed.

6.3.2 *Upper bounds from online algorithms*

Parity Polytope

As an example, consider the language containing strings where the last bit indicates the parity of the previous bits. This language can be accepted by a deterministic logspace turing machine requiring a single pass over the input and a single bit of space. Therefore, the parity polytope has extension complexity $O(n)$.

The parity polytope is known to have extension complexity at most $4n - 4$ [16].

Integer Partition Polytope

For non-negative integer n the Integer Partition Polytope, P_n, is defined as

$$P_n := \text{conv}\{x \in \mathbb{Z}_+^n | \sum_{k=1}^{n} kx_k = n\}.$$

It is known that $\text{xc}(P_n) = \mathcal{O}(n^3)$ [50].

Consider the polytope in $\mathbb{R}^{\lceil \log n \rceil \times n}$ that encodes each x_i as a binary string. For example, for $n = 4$ the vector $(2, 1, 0, 0)$ is encoded as $(1, 0, 0, 1, 0, 0, 0, 0)$. This polytope is clearly an extended formulation of the Integer Partition Polytope. Call this polytope BIPP_n. The following single pass deterministic algorithm accepts a string $(x_1, x_2, \ldots, x_n) \in \{0, 1\}^{\lceil \log n \rceil \times n}$ if and only if the string represents a vertex of BIPP_n.

```
Data : Binary string of length n⌈log n⌉
Result : Accept if the input encodes a vertex of the BIPP_n
s = 0; i = 0; l = 0;
while i < n do
    b =read_next_bit;
    if (s + (i + 1)2^l b) > n then
        reject;
    else
        s = (s + (i + 1)2^l b);
        l = (l + 1)%⌈log n⌉;
        if l == 0 then
            i++;
        end
    end
end
if s == n then
    accept;
else
    reject;
end
```

Algorithmus 1 : One pass algorithm for accepting vertices of BIPP_n.

The above algorithm together with Proposition 6.2.9 shows that $\text{xc}(\text{IPP}_n) \leqslant \text{xc}(\text{BIPP}_n) \leqslant \mathcal{O}(n^3 \log^2 n)$.

Knapsack Polytopes

Let $(a, b) = (a_1, a_2, \ldots, a_n, b)$ be a given sequence of (non-negative) integers. The Knapsack polytope $\text{KS}(a, b)$ is defined as

$$\text{KS}(a, b) := \{x \in \{0, 1\}^n | \sum_{i=1}^{n} a_i x_i \leqslant b\}.$$

The Knapsack polytope is known to have extension complexity super-polynomial in n. However, optimizing over $\text{KS}(a, b)$ can be done via dynamic programming in time $O(nW)$ where W is the largest number among $a_1, \ldots, a_n, b$.

Suppose the integers a_i, b are arriving in a stream with a bit in between indicating whether $x_i = 0$ or $x_i = 1$. With a space of W bits, an online Turing machine can store and update $\sum_{i=1}^{n} a_i x_i$. At the end, it can subtract b and accept or reject depending on whether the result is 0 or not. Any overflow during intermediate steps can be

used to safely reject the input. Therefore, the extension complexity of the Knapsack polytope is $O(nW \log W)$. Note however the extension obtained this way is actually an extended formulation of a polytope encoding all the instances together with their solutions.

*Languages in co-*DLIN

Let $\mathbf{L}$ be a language generated by a determinisitic linear grammar [37]. The following result was proved by Babu, Limaye, and Varma [6].

Proposition 6.3.4 (BLV)**.** *Let* $\mathbf{L} \in$ DLIN*. Then there exists a probabilistic one-pass streaming algorithm using* $\mathcal{O}(\log n)$ *space that accepts every string in* $\mathbf{L}$ *and rejects every other string with probability at least* $1/n^c$.

Using the above algorithm together with Proposition 6.2.10 we get the following.

Proposition 6.3.5. *If* $\mathbf{L} \in$ DLIN*, then* $\overline{\mathbf{L}} \in \mathcal{CF}$.

Part III

VARIATIONS

A novice was trying to fix a broken Lisp machine by turning the power off and on.

Knight, seeing what the student was doing, spoke sternly: "You cannot fix a machine by just power-cycling it with no understanding of what is going wrong."

Knight turned the machine off and on. The machine worked.

— Tom Knight and the Lisp Machine [53]

7

FPT EXTENDED FORMULATIONS

7.1 PARAMETERIZED EXTENSION COMPLEXITY

Most of the bounds seen so far (specially in Chapter 4) were only in terms of the ambient dimension n. For example the perfect matching polytope $EP(K_n)$ for the complete graph K_n was seen to have extension complexity $2^{\Theta(n)}$. What about other graphs on n vertices? It was shown by Barahona [7] that for planar graphs the perfect matching polytope has polynomial extension complexity. Gerard [32] showed that if G is a n-vertex graph of genus g, then $xc(EP(G)) \leqslant n^{\mathcal{O}(g)}$.

In this chapter we shall see results of similar type. We have already seen some results on parametrized extension complexity although it was not made explicit so far.

Example 7.1.1. Let each 0/1 polytope $P \in \mathbb{R}^n$ be parameterized by the minimum number of clauses in any DNF formula φ such that $P = SAT(\varphi)$. Denoting this parameter by μ, one can use Proposition 3.3.13 to conclude that $xc(\mathbb{ZERO}-\mathbb{ONE}) = \mathcal{O}(\mu n)$.

In other words, the clan of 0/1 polytopes has polynomial extension complexity when *parameterized* by the size of the smalled DNF formula describing the vertex set of the polytopes.

Definition 3.1.5 can be used to formally speak about parametrized extension complexity of a family (and therefore a clan) of polytopes. Recall that for a family of polytopes there is exactly[1] one polytope $P_n \subset \mathbb{R}^n$ in the family for each $n \in \mathbb{N}$.

Definition 7.1.2. Let $\mathbb{P}$ be a clan of polytopes and $\kappa : \mathbb{P} \to \mathbb{N}$ be a parameter. Let $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be a function. We will say that the *parameterized extension complexity* of $\mathbb{P}$ is $g(\kappa, n)$ if for every polytope $P \in \mathbb{P}$ such that $P \subseteq \mathbb{R}^n$ we have that $xc(P) = g(\kappa(P), n)$.

We will say that extension complexity of $\mathbb{P}$ or $xc(\mathbb{P})$ is FPT if there exists $f : \mathbb{N} \to \mathbb{N}$ and a constant c such that $xc(\mathbb{P}) \leqslant f(\kappa)n^c$, and if no such function or constant exist then we will say that the extension complexity of $\mathbb{P}$ is not FPT.

As described in Subsection 3.1.1, we will mostly use asymptotic descriptions of functions whenever we use the above definition.

Example 7.1.3. The extension complexity of the clan $\mathbb{ZERO}-\mathbb{ONE}$ parameterized by the size of the smallest DNF formula describing the vertices is FPT.

1 In case there is no such polytope explicitely present, the empty polytope can play the desired role.

7.2 THE INDEPENDENT SET POLYTOPE

The k-independent set problem asks one to decide whether a graph has independent set of size at most k. When parameterized by k this problem is W[1]—hard but is fixed-parameter tractable for graphs of bounded expansion. The corresponding polytope has analogous behavior with respect to the extension complexity [30].

7.2.1 *The k-independent set polytope*

Definition 7.2.1. Let $G = (V, E)$ be a graph on n vertices. The $k^{\leqslant}$-*independent set polytope of* G – denoted by $\mathrm{STAB}_{k^{\leqslant}}(G)$ – is defined to be the convex hull of the independent sets of G that have size at most k.

Alternatively, one could define the $k^{=}$-*independent set polytope of* G – denoted by $\mathrm{STAB}_{k^{=}}(G)$ – to be the convex hull of all independent sets of size exactly equal to k.

As far as extension complexity is concerned, either definition can be used to define the clan of stable set polytopes parameterized by the size of the independent set. This is because the extension complexities of the two polytopes defined above are within a polynomial factor of each other.

Proposition 7.2.2.

$$\mathrm{xc}(\mathrm{STAB}_{k^{=}}(G)) \leqslant \mathrm{xc}(\mathrm{STAB}_{k^{\leqslant}}(G)) \leqslant \sum_{i=0}^{k} \mathrm{xc}(\mathrm{STAB}_{i^{=}}(G)).$$

Proof. Clearly, $\mathrm{STAB}_{k^{=}}(G)$ is a face of $\mathrm{STAB}_{k^{\leqslant}}(G)$. Therefore, we have that $\mathrm{xc}(\mathrm{STAB}_{k^{=}}(G)) \leqslant \mathrm{xc}(\mathrm{STAB}_{k^{\leqslant}}(G))$ (cf. Proposition 3.1.23).

On the other hand, $\mathrm{STAB}_{k^{\leqslant}}(G) = \mathrm{conv}(\bigcup_{i=1}^{k} \mathrm{STAB}_{i^{=}}(G))$, and therefore $\mathrm{xc}(\mathrm{STAB}_{k^{\leqslant}}(G)) \leqslant \sum_{i=0}^{k} \mathrm{xc}(\mathrm{STAB}_{i^{=}}(G))$ by Proposition 3.2.5. □

Therefore any bounds (whether lower or upper) that are valid for $\mathrm{xc}(\mathrm{STAB}_{k^{=}}(G))$ are also asymptotically valid for $\mathrm{xc}(\mathrm{STAB}_{k^{\leqslant}}(G))$. To simplify our notations, instead of either $\mathrm{STAB}_{k^{\leqslant}}(G)$ or $\mathrm{STAB}_{k^{=}}(G)$, we will use $\mathrm{STAB}_{k}(G)$. In the rest of the chapter $\mathrm{STAB}_{k}(G)$ represents $\mathrm{STAB}_{k^{=}}(G)$ but with minor adjustments the same arguments can be made using $\mathrm{STAB}_{k^{\leqslant}}(G)$.

Buchanan [13] showed that the extension complexity of the stable set polytopes parametrized by the treewidth τ of the underlying graph is $2^{\mathcal{O}(\tau)}n$.

Proposition 7.2.3. *Let* $\mathbb{STAB}$ *be the clan of stable set polytopes parametrized by the treewidth of the underlying graph. That is, let* $\tau : \mathbb{STAB} \to \mathbb{N}$ *be defined as* $\tau(P) = \min\{\tau(G) \mid P = \mathrm{STAB}(G)\}$, *where* $\tau(G)$ *is the treewidth of graph* G. *Then,* $\mathrm{xc}(\mathbb{STAB}) \leqslant 2^{\mathcal{O}(\tau)} \cdot n$.

Proof. See [13], Theorem 3. □

Buchanan also asked if the extension complexity of the stable set polytope parametrized by the size of the independent set is FPT. We next present the answer to this question (in the negative). The proof relies on encoding cuts of the complete graph on roughly $k \log n$ vertices as independent sets of size k^2 in another graph whose size is not too big. The result then follows from the fact that the cut polytope of the initial graph has size $\Omega(n^k)$.

7.2.2 *Paired Local-Cut Graphs*

Given positive integers k and n, we define a graph called a *Paired Local-Cut Graph* and denoted by $\mathrm{PLC}(k, n)$.

First we create $k2^{\lfloor \log n \rfloor}$ vertices labeled with tuples (i, S) for $i \in [k]$ and $S \subseteq [\lfloor \log n \rfloor]$. These vertices will be called *cut vertices*. Then we create $2\binom{k}{2}2^{2\lfloor \log n \rfloor}$ vertices labeled with tuples (i, j, S_1, S_2) where $1 \leqslant i \neq j \leqslant k$ and $S_1, S_1 \subseteq [\lfloor \log n \rfloor]$. These vertices will be called *pairing vertices*.

We add edges to these vertices of $\mathrm{PLC}(k, n)$ as follows. For each fixed $i \in [k]$ we add the edges between all cut nodes that have labels (i, S). Furthermore, for each fixed pair $i, j \in [k]$ we add the edges between all pairing nodes that have labels (i, j, S_1, S_2). Finally, let u be a cut vertex labeled (i, S) and let v be a pairing vertex labeled (j_1, j_2, S_1, S_2). If $i = j_1$ but $S \neq S_1$ we add edge uv. Symmetrically, if $i = j_2$ but $S \neq S_2$ we add edge uv.

For ease of exposition we will identify vertices of $\mathrm{PLC}(k, n)$ with their labels whenever convenient.

Proposition 7.2.4. *The number of vertices of the graph* $\mathrm{PLC}(k, n)$ *equals* $2\binom{k}{2}2^{2\lfloor \log n \rfloor} + k2^{\lfloor \log n \rfloor} \leqslant (kn)^2$.

Proposition 7.2.5. *Let* (i, S) *and* (j_1, j_2, S_1, S_2) *be two vertices of* $\mathrm{PLC}(k, n)$ *that are not joined by an edge. If* $i = j_1$ *then* $S = S_1$*, and if* $i = j_2$ *then* $S = S_2$.

This together with the next proposition will ensure that in any independent set I of $\mathrm{PLC}(k, n)$ that has size k^2, every index $i \in [k]$ can be uniquely associated with a subset $S_i \subseteq \lfloor \log n \rfloor$.

Proposition 7.2.6. *Let* I *be an independent set in* $\mathrm{PLC}(k, n)$*. Then,* $|I| \leqslant k^2$*. Moreover, an equality holds if and only if* I *contains exactly one cut vertex for each* $1 \leqslant i \leqslant k$ *and exactly one pairing vertex for each* $1 \leqslant i \neq j \leqslant k$.

Proof. Clearly, the set I can contain at most k cut vertices – at most one vertex (i, S_i) for each $1 \leqslant i \leqslant k$. Also, set I can contain at most $2\binom{k}{2} = k^2 - k$ pairing vertices – at most one vertex (i, j, S_i, S_j) for each ordered pair $1 \leqslant i, j \leqslant k$. □

The vertices of $\mathrm{STAB}_{k^2}(\mathrm{PLC}(k, n))$ are related to the vertices of the *polytope* $\mathrm{CUT}^{\square}(K_r)$ where $r = k\lfloor \log n \rfloor$, in the following way. Denote the vertices and edges of K_r by V_r and E_r respectively, and group the vertices of K_r into k groups, each of size $\lfloor \log n \rfloor$. Label the vertices v_j^i where $1 \leqslant i \leqslant k$ and $1 \leqslant j \leqslant \lfloor \log n \rfloor$. Finally, order the vertices lexicographically according to their labels.

A cut vector of K_r – corresponding to a cut C – is a $0/1$ vector of length $\binom{r}{2}$ whose coordinates correspond to whether an edge of K_r is in the cut C or not. The edges of K_r are labeled with pairs (i_1, j_1, i_2, j_2) where $1 \leqslant i_1, i_2 \leqslant k$; $1 \leqslant j_1, j_2 \leqslant \lfloor \log n \rfloor$, and $(i_1, j_1) \leqslant (i_2, j_2)$ lexicographically. So, if z is a cut vector corresponding to a given cut $C \subset E_r$, then $z_{i_1,j_1,i_2,j_2} = 1$ if and only if the edge (i_1, j_1, i_2, j_2) is in C. $\mathrm{CUT}^{\square}(K_r)$ is the convex hull of all such cut vectors.

Similarly, an independent-set vector of $\mathrm{PLC}(k, n)$ – corresponding to an independent set I – is a $0/1$ vector of length $2\binom{k}{2}2^{2\lfloor \log n \rfloor} + k2^{\lfloor \log n \rfloor}$ (cf. Prop. 7.2.4) whose coordinates correspond to whether the corresponding vertex is in I or not. Recall that the cut vertices of $\mathrm{PLC}(k, n)$ are labeled with a pair consisting of an index from $[k]$, and a subset of $[\lfloor \log n \rfloor]$. Also, the pairing vertices of $\mathrm{PLC}(k, n)$ are labeled with a tuple consisting of two indices from $[k]$ and two subsets of $[\lfloor \log n \rfloor]$.

Let $\mathcal{C}$ be the set of all cuts in K_r, and let $\mathbf{I}$ be the set of all independent sets of size k^2 in $\mathrm{PLC}(k, n)$. Any cut $C \in \mathcal{C}$ creates a bipartition $(S, \overline{S})$ of the vertices of K_r. Recall that the vertices of K_r have been split in k groups. The partition $(S, \overline{S})$ thus induces a partition $(S_i, \overline{S}_i)$ within each of these groups.

Proposition 7.2.7. *For every pair of natural numbers (k, n) and $r = k\lfloor \log n \rfloor$ it holds that $\mathrm{CUT}^{\square}(K_r)$ is a projection of $\mathrm{STAB}_{k^2}(\mathrm{PLC}(k, n))$.*

Proof. See [30], Lemma 3.4 (Appendix F). □

A lower bound on the extension complexity of $\mathrm{STAB}_{k^2}(\mathrm{PLC}(k, n))$ immediately follows.

Proposition 7.2.8. *There exists a constant $c' > 0$ such that for $k, n \in \mathbb{N}$,*

$$\mathrm{xc}\left(\mathrm{STAB}_{k^2}(\mathrm{PLC}(k, n))\right) \geqslant n^{c'k}.$$

Proof. By Proposition 7.2.7, $\mathrm{STAB}_{k^2}(\mathrm{PLC}(k, n))$ is an extended formulation of $\mathrm{CUT}^{\square}(K_r)$ with $r = k\lfloor \log n \rfloor$. So any extended formulation of $\mathrm{STAB}_{k^2}(\mathrm{PLC}(k, n))$ is also an extended formulation of $\mathrm{CUT}^{\square}(K_r)$. By proposition 3.3.6, $\mathrm{xc}\left(\mathrm{CUT}^{\square}(K_r)\right) \geqslant 2^{\Omega(r)}$. Therefore,

$$\mathrm{xc}\left(\mathrm{STAB}_{k^2}(\mathrm{PLC}(k, n))\right) \geqslant \mathrm{xc}\left(\mathrm{CUT}^{\square}(K_r)\right) \geqslant 2^{\Omega(r)} \geqslant n^{c'k}$$

for some constant $c' > 0$. □

We can now easily conclude that the parameterized extension complexity of $\mathbb{STAB}$ parameterized by the size of the independent sets is not FPT.

Proposition 7.2.9. *There does not exist any function $f : \mathbb{N} \to \mathbb{R}$ such that $\mathrm{xc}(\mathrm{STAB}_k(G)) \leqslant f(k) \cdot n^{O(1)}$ for all natural numbers k and all graphs G on n vertices.*

Proof. Suppose, on the contrary, that such a function f exists. That is, there is a constant c such that for every pair of natural numbers (ℓ, m) and for all m-vertex graphs G it holds that $\mathrm{xc}(\mathrm{STAB}_\ell(G)) \leqslant f(\ell) \cdot m^c$.

Given a pair (k, n) of natural numbers consider the graph $\mathrm{PLC}(k, n)$. By Proposition 7.2.8, we have that $\mathrm{xc}(\mathrm{STAB}_{k^2}(\mathrm{PLC}(k, n))) \geqslant n^{c'k}$ for some constant $c' > 0$. On the other hand, from our assumption for $\ell = k^2$ and $m \leqslant (kn)^2$ we have that $\mathrm{xc}(\mathrm{STAB}_{k^2}(G)) \leqslant f(k^2) \cdot (kn)^{2c}$. Therefore, $n^{c'k} \leqslant f(k^2) \cdot (kn)^{2c}$ and so $c'k \log n \leqslant \log f(k^2) + 2c(\log k + \log n)$. This in turn implies that $\log n \leqslant \frac{\log f(k^2) + 2c \log k}{c'k - 2c}$ which clearly cannot be true for any fixed k and arbitrary n, and hence no such function f exists. □

7.3 FPT UPPER BOUNDS

Now we will see that FPT upper bounds exist for a large number of interesting polytopes.

7.3.1 *MSO Polytopes parametrized by Treewidth*

In most cases, we stick to standard notation as given by Libkin [47] and by Downey and Fellows [22]. We use the standard approach and view every graph $G = (V, E)$ as a labeled graph $I(G) = (V_I, E_I, L_V, L_E)$, called the *incidence graph* of G, where $V_I = V \cup E$, $E_I = \{\{v, e\} \mid v \in e, e \in E\}$, $L_V = V$ and $L_E = E$; this way, every MSO_2 formula about the original graph G can be turned into an MSO formula about $I(G)$. Since the treewidth of the incidence graph $I(G)$ is at most $tw(G) + 1$ [43], this does not pose any limitation.

Also, for simplicity, we will work with a version of MSO that has only set variables and a special predicate s of arity one to emulate element variables (for every graph $G = (V, E)$ and every $X \subseteq V \cup E$, $s(X)$ is true in G if and only if $|X| = 1$); it is easy to see that this syntactical restriction does not mean any restriction in the expressive power. All results can be extended to general finite structures where the restriction on treewidth applies to the treewidth of their Gaifman graph [22].

Formally, the set of MSO formulae is defined recursively as follows. We assume an infinite supply of set variables $X, Y, X_1, \ldots$. For every two variables X and Y, $s(X)$, $ver(X)$, $edg(X)$, $inc(X, Y)$, $X \subseteq Y$ and $X = Y$ are formulae, namely *atomic* formulae. For a given graph G, $ver(X)$ or $edg(X)$ is true, if $X \subseteq L_V$ or $X \subseteq L_E$, resp.; $inc(X, Y)$ is true if and only if $s(X)$, $s(Y)$ are true and $\{x, y\} \in E_I$ where x is the only element in X and y is the only element in Y. If φ, ψ_1 and ψ_2 are formulae then $\neg\varphi$, $\psi_1 \wedge \psi_2$ and $\exists X \varphi(X)$ are formulae.

A variable X is *free* in φ if it does not appear in any quantification in φ. If $\vec{X}$ is the tuple of all free variables in φ, we write $\varphi(\vec{X})$. A variable X is *bound* in φ if it is not free. By $qr(\varphi)$ we denote the *quantifier rank* of φ which is the number of quantifiers of φ when transformed into the prenex form (i.e., all quantifiers are in the front of the formula).

For a given MSO formula $\varphi(\vec{X})$ with m free set variables $X_1, \ldots, X_m$, we define a polytope of satisfying assignments on a given graph G with n vertices in a natural way. We encode any assignment of vertices of G to the sets $X_1, \ldots, X_m$ as follows. For each X_i in φ and each

v in G, we introduce a binary variable y_i^v. We set y_i^v to be one if $v \in X_i$ and zero otherwise. For a given 0/1 vector y, we say that y *satisfies* φ if interpreting the coordinates of y as described above yields a satisfying assignment for φ. The polytope of satisfying assignments, also called the *MSO polytope*, is defined as

$$\mathrm{MSO}_\varphi(G) = \mathrm{conv}\left(\{y \in \{0,1\}^{nm} \mid y \text{ satisfies } \varphi\}\right) .$$

Proposition 7.3.1. *For every graph G on n vertices with $\tau(G) = \tau$ and for every $\varphi \in MSO$, $\mathrm{xc}(MSO_\varphi(G)) = f(|\varphi|, \tau) \cdot n$ where f is some computable function.*

Proof. See [44], Theorem 2 (Appendix G). □

In fact the extended formulation can be efficiently constructed.

Proposition 7.3.2. *Let G be a graph of treewidth τ and let φ be an MSO formula. An extended formulation for $MSO_\varphi(G)$ (with size bounded as mentioned in Proposition 7.3.1) can be constructed in time $f'(|\varphi|, \tau) \cdot n$, for some computable function f'.*

Proof. See [44], Theorem 3 (Appendix G). □

In the language that we have adopted so far it means that the extension complexity of MSO polytopes, when parameterized by the treewidth of the underlying graph and the size of the MSO formula, is FPT.

7.3.2 *FO Polytopes parameterized by Expansion*

The *first-order logic of graphs* (abbreviated as FO) applies the standard language of first-order logic to a graph G viewed as a relational structure with the domain $V(G)$ and the single binary (symmetric) relation $E(G)$. For example, the formula $\iota(x_1, \ldots, x_k) \equiv \bigwedge_{i \neq j} \left(\neg \mathrm{edge}(x_i, x_j) \wedge x_i \neq x_j\right)$ asserts that $\{x_1, \ldots, x_k\}$ is an independent set of size exactly k. A slightly more involved example describes a vertex cover tuple as $\gamma(x_1, \ldots, x_k) \equiv \forall y,z\big(\mathrm{edge}(y,z) \to \bigvee_{i=1}^{k}(y = x_i \vee z = x_i)\big)$.

To any FO formula $\phi(x_1, \ldots, x_k)$ and a graph, one can assign a polytope in the following way. For an ordered k-tuple of vertices $W = (w_1, \ldots, w_k) \in V(G)^k$ we thus define its characteristic vector χ^W of length $k|V(G)|$ by

$$\chi^W_{v,i} = \begin{cases} 1 & \text{if } v = w_i, \\ 0 & \text{otherwise.} \end{cases}$$

Note that χ^W always satisfies $\sum_{v \in V(G)} \chi^W_{v,i} = 1$ for each $i = 1, \ldots, k$, by the definition.

If $W = (w_1, \ldots, w_k) \in V(G)^k$ is such that $\phi(w_1, \ldots, w_k)$ holds true in G, we write $G \models \phi(w_1, \ldots, w_k)$. We can now give the following definition:

Definition 7.3.3 (FO polytope)**.** Let $\phi(x_1, \ldots, x_k)$ be an FO formula with k free variables. The *(first-order) ϕ-polytope of* G, denoted by

$\mathrm{FOP}_\phi(G)$, is defined to be the convex hull of the characteristic vectors of every k-tuple of vertices of G such that $\phi(w_1,\dots,w_k)$ holds true in G. That is,

$$\mathrm{FOP}_\phi(G) = \mathrm{conv}\left(\left\{\chi^W \in \{0,1\}^n \;\middle|\; \begin{array}{rcl} W & = & (w_1,\dots,w_k) \in V(G)^k, \\ G & \models & \phi(w_1,\dots,w_k) \end{array}\right\}\right).$$

FO polytopes are quite general. For example, the polytopes $\mathrm{STAB}_k(G)$ defined earlier is easily seen to be an instance of an FO polytope.

Proposition 7.3.4. *Let* $\iota(x_1,\dots,x_k) \equiv \bigwedge_{i\neq j}\left(\neg \mathtt{edge}(x_i,x_j) \wedge x_i \neq x_j\right)$ *(the* k*-independent set formula). For every graph* G*, the* ι*-polytope* $\mathrm{FOP}_\iota(G)$ *is an extension of* $\mathrm{STAB}_k(G)$*.*

Proof. If G has n vertices then

$$\mathrm{STAB}_k(G) = \left\{ y \in \mathbb{R}^n \;\middle|\; y_v = \sum_{i=1}^{k} \chi^W_{v,i},\ \chi^W \in \mathrm{FOP}_\iota(G) \right\}.$$

Therefore, $\mathrm{STAB}_k(G)$ is a projection of $\mathrm{FOP}_\iota(G)$ given by the projection map described by $y_v = \sum_{i=1}^{k} \chi^W_{v,i}$ for all vertices v of G. □

We say that an FO formula $\phi(x_1,\dots,x_k)$ is *existential FO* if it can be written as $\phi(x_1,\dots,x_k) \equiv \exists y_1 \dots y_\ell\, \psi(x_1,\dots,x_k,y_1,\dots,y_\ell)$, where ψ is quantifier-free. The number ℓ of quantified variables in ϕ is called the *quantifier rank of* ϕ.

Proposition 7.3.5. *Let* $\phi(x_1,\dots,x_k)$ *be an existential FO formula with* k *free variables and quantifier rank* ℓ*. Also, let* $\mathcal{G}$ *be any graph class of bounded expansion. Then there exists a computable function* $f : \mathbb{N} \to \mathbb{N}$*, depending on the expansion function of* $\mathcal{G}$*, such that*

$$\mathrm{xc}\big(\mathrm{FOP}_\phi(G)\big) \leqslant f(k+\ell)\cdot n$$

holds for every integer n *and every* n*-vertex graph* $G \in \mathcal{G}$*. Furthermore, an explicit extension of* $\mathrm{FOP}_\phi(G)$ *of size at most* $f(k+\ell)\cdot n$ *can be found in linear time for fixed* k, ℓ *and* $\mathcal{G}$*.*

Proof. See [30], Theorem 20 (Appendix F). □

8

$\mathcal{H}$-FREE EXTENDED FORMULATIONS

Since linear programming is in P we will not be able to solve an NP-hard problem X in polynomial time by linear programming unless $P = NP$. On the other hand, since linear programming is P-complete, we will not be able to prove a super-polynomial lower bound on solving X by a linear program (LP) without showing that $P \neq NP$. One way to make progress on this problem is to consider restricted versions of linear programming which have two properties:

PROPERTY (1): Problems in P will still be solvable in polynomial time even in the restricted version of linear programming.

PROPERTY (2): Known NP-hard problems with natural LP formulations will have provable super-polynomial lower bounds under the restricted version of linear programming.

Note that results of type (1) and (2) will still be true, independently of whether or not $P = NP$.

Here we propose a stronger version of extension complexity which satisfies property (1). We also exhibit some NP-hard problems that satisfy property (2). In the proposed model we concentrate on the separation problem rather than the polynomial time equivalent optimization problem.

8.1 $\mathcal{H}$-FREE EXTENSIONS

Definition 8.1.1. Let $P = P(\mathbf{A}, \mathbf{b})$ be a polytope and let H be a set of valid inequalities for P. We delete from $\mathbf{Ax} \leqslant \mathbf{b}$ all inequalities that are redundant with respect to H and call the resulting (possibly empty) polyhedron P_H. The *$\mathcal{H}$-free extension complexity of* P *with respect to the inequalities* H is defined to be the extension complexity of P_H.

Let X be some computational problem that can be solved by an LP over a polytope Q. For the applications considered in this chapter, it is convenient to consider the case where Q is given by an implicit description of its vertices. So for the matching problem, Q is the convex hull of all 0/1 matching vectors, and for the TSP problem it is the convex hull of all 0/1 incidence vectors of Hamiltonian circuits.

Let $H = H(Q)$ be a possibly super-polynomial size set of valid inequalities for Q equipped with an H-separation oracle. We can solve the separation problem for Q for a point x by first solving it for H and then, if necessary, for Q_H. Suppose x is not in Q. If x is not in H we get a violated inequality by the oracle. Otherwise x must violate a facet of Q_H. We will allow separation for Q_H to be performed using any extension Q'_H of Q_H by explicitly checking the facets of Q'_H for

the lifting of x. We call Q'_H an *$\mathcal{H}$-free EF for Q*. Using this separation algorithm and the ellipsoid method we have a way to solve LPs over Q. We call such a restricted method of solving LPs an *$\mathcal{H}$-free LP for Q*.

Definition 8.1.2. Let Q be a polytope and H be a set of inequalities valid for Q. We say that an *$\mathcal{H}$-free EF for Q has polynomial size* if:

(a) The H-separation oracle runs in polynomial time and

(b) $xc(Q_H)$ is polynomial in the input size of X.

In this case we also have an $\mathcal{H}$-free LP for Q that runs in polynomial time. On the other hand, if for given H, $xc(Q_H)$ is super-polynomial in the size of X then we say that all $\mathcal{H}$-free LPs for X run in super-polynomial time. Note that this statement is independent of whether or not $P = NP$. When H is empty all of the above definitions reduce to standard definitions for EFs and extension complexity.

Example 8.1.3. For the matching problem if H is the set of odd-set inequalities then Q_H is empty. In this case we have an $\mathcal{H}$-free EF for matching of poly-size even though matching has exponential extension complexity.

This example generalizes to show that every problem X in P has a poly-size $\mathcal{H}$-free EF for some H. Indeed, since LP is P-complete, X can be solved by optimizing over a polytope Q. Let H be the entire facet list $F(Q)$ so that Q_H is again empty. Optimization over Q can be performed in polynomial time so, by the equivalence of optimization and separation, separation over H can be performed in polynomial time also. Therefore (a) and (b) are satisfied as required.

For the TSP, let H be the sub-tour constraints. In this case Q_H is non-empty and in fact one can show (cf. next section) that it has exponential extension complexity. Therefore $\mathcal{H}$-free LPs for the TSP require exponential time, extending the existing extension complexity result for this problem.

We remark that H is an essential parameter here. Matching, for example, has poly-size $\mathcal{H}$-free extension complexity when H are the odd set inequalities, but not when H is empty. Nevertheless, any problem with poly-size $\mathcal{H}$-free extension complexity for some H can of course be solved in polynomial time. For a given hard problem, one gets stronger hardness results by letting H be larger and larger sets of poly-size separable inequalities, as long as one can still prove that Q_H has super-polynomial extension complexity. We give some examples to illustrate this in subsequent sections.

8.2 MATCHING PROBLEMS

Recall that Edmonds' polytope has the following halfspace representation[23]:

$$\sum_{e \in S} x_e \leqslant (|S|-1)/2, \quad S \subseteq V, \ |S| \text{ is odd} \tag{7}$$

$$0 \leqslant x_e \leqslant 1, \quad e \in E. \tag{8}$$

Let H be this half-space representation of Q. Since optimization over Q can be performed in polynomial time by Edmonds' algorithm there is a polynomial time separation algorithm for H. It follows that the matching problem has a poly-size $\mathcal{H}$-free EF.

In the next three subsections we give NP-hard generalizations of the matching problem which have super-polynomial lower bounds on their $\mathcal{H}$-free extension complexity, where H are the odd set inequalities (7).

8.2.1 *Induced matchings*

Let Q be the convex hull of the incidence vectors of all induced matchings in G. Let H be the odd set inequalities (7). Clearly H are valid for Q, and as remarked above, they admit a polynomial time separation oracle. It can be shown that $\text{xc}_H(Q)$ is super-polynomial.

Proposition 8.2.1. $\text{xc}_H(Q)$ *is super-polynomial.*

Proof. Since Proposition 4.2.15 applies to bipartite graphs G, each of the odd set inequalities (7) is redundant for the induced matching polytope of G. Therefore the $\mathcal{H}$-free extension complexity of the induced matching polytope is super polynomial in the worst case. □

Although this example offers an example of $\mathcal{H}$-free extension complexity, it suffers from one obvious weakness. For every graph, all of the inequalities in H are redundant with respect to Q even for non-bipartite graphs! A graph is called *hypomatchable* if the deletion of any vertex yields a graph with a perfect matching. Pulleyblank proved in 1973 (see [48]) that facet-inducing inequalities in (7) correspond to subsets S that span 2-connected hypomatchable subgraphs of G. Let x be the incidence vector for any matching M in G that satisfies such an inequality as an equation. Since S spans a 2-connected subgraph, M cannot be an induced matching.

In order to avoid such trivial cases it is desirable that most, if not all, inequalities of H define facets for at least one polytope Q that corresponds to some instance of the given problem.

8.2.2 *Maximal matchings*

Again, since the graphs G in Proposition 4.2.20 are bipartite, each of the odd set inequalities (7) is redundant for the maximal matching polytope of G. Therefore the $\mathcal{H}$-free extension complexity of the induced matching polytope is super polynomial in the worst case.

This example differs from the example in the previous subsection in that (7) are facet defining for maximum matching polytopes of non-bipartite graphs. To see this, fix a graph G and odd-set S of its vertices. Pulleyblanks's characterisation [48] states that (7) is facet defining for the matching polytope of G whenever S spans a 2-connected hypomatchable subgraph. The only matchings in G that lie on this facet have precisely $(|S|-1)/2$ edges from the set S and are therefore maximal on S. Each of these matchings can be extended to a maximal

matching in G which appears as a vertex of $MM(G)$. Therefore, provided these extensions do not lie in a lower dimensional subspace and $MM(G)$ is full dimensional, (7) is also facet inducing for $MM(G)$ for the given set S. For example, the odd cycles C_{2k+1}, $k \geqslant 3$ with the addition of a chord cutting off a triangle are a family of such graphs.

8.2.3 *Edge disjoint matching and perfect matching*

Note that for every pair of odd subsets S_1, S_2 of G two odd set inequalities can be written: one corresponding to the odd set inequalities for perfect matching polytope on variables x_i, and the other corresponding to the odd set inequalities for matching polytope on variables y_i. For a subset of vertices S, let $\delta(S)$ denote the subset of edges with exactly one endpoint in S. The two sets of inequalities are:

$$\sum_{e \in \delta(S_1)} x_e \geqslant 1, \qquad S_1 \subseteq V, \ |S_1| \text{ is odd} \tag{9}$$

$$\sum_{e \in S_2} y_e \leqslant \frac{|S_2|-1}{2}, \quad S_2 \subseteq V, \ |S_2| \text{ is odd} \tag{10}$$

Again the graphs G in the Proposition 4.2.24 are bipartite so each of the odd set inequalities (9,10) is redundant for $MPM(G)$. Therefore taking H to be the set of these inequalities we have that the $\mathcal{H}$-free extension complexity of the these polytopes is super polynomial in the worst case.

8.3 THE TSP POLYTOPE

Recall that an undirected TSP instance X is defined by a set of integer weights $w_{ij}, 1 \leqslant i < j \leqslant n$, for each edge of the complete graph K_n. A tour is a Hamiltonian cycle in K_n defined by a permutation of its vertices. It is required to compute a tour of minimum weight. We define the polytope Q to be the convex hull of the 0/1 incidence vectors $x = (x_{ij} : 1 \leqslant i < j \leqslant n\}$ of the tours. It is known that $xc(Q) = 2^{\Omega(n)}$ [55].

We define H to be the set of *subtour elimination* constraints:

$$\sum_{i,j \in S, i \neq j} x_{ij} \leqslant |S|-1, \quad S \subseteq \{1,2,...,n-1\}, \ |S| \geqslant 2. \tag{11}$$

$$x_{ij} \geqslant 0, \quad 1 \leqslant i < j \leqslant n \tag{12}$$

It is well known that the subtour elimination constraints can be polynomial time separated by using network flows. These constraints by themselves define the convex hull of all forests in K_{n-1} and Martin [49] has given an EF for them that has size $O(n^3)$.

Therefore, $xc(Q_H) = 2^{\Omega(n)}$, otherwise together with Martin's result and Proposition 3.2.6), it would imply an upper bound of $2^{o(n)}$ for the travelling salesman polytope. It follows that every $\mathcal{H}$-free LP for the TSP runs in exponential time, where H are the subtour inequalities.

8.3.1 *Comb inequalities for TSP*

Definition 8.3.1. For a graph $G = (V, E)$, a comb is defined by a subset of vertices H called the handle and a set of subsets of vertices $T_i, 1 \leqslant i \leqslant k$ where k is an odd number at least three. The sets T_i are called the teeth. The handle and the teeth satisfy the following properties:

$$H \cap T_i \neq \emptyset, \tag{13}$$

$$T_i \cap T_j = \emptyset, \qquad \forall i \neq j \tag{14}$$

$$H \setminus \bigcup_{i=1}^{k} T_i \neq \emptyset \tag{15}$$

The following inequality is valid for the TSP polytope of G and is called the comb inequality for the comb defined by handle H and teeth T_i as above.

$$x(\delta(H)) + \sum_{i=1}^{k} x(\delta(T_i)) \geqslant 3k + 1$$

Grötschel and Padberg [34] showed that every comb inequality defines a facet of TSP_n for each $n \geqslant 6$. It is not known whether separating over comb inequalities is NP-hard, neither is a polynomial time algorithm known.

For a given comb C and a TSP tour T of G, the slack between the corresponding comb inequality and T is denoted by $sl_{comb}(C, T)$.

8.3.2 *2-matching inequalities*

Definition 8.3.2. A comb inequality corresponding to a handle H and k teeth T_i is called a 2-matching inequality if each tooth T_i has size exactly two.

In particular this means that $|H \cap T_i| = 1$ and $|T_i \setminus H| = 1$ for each $1 \leqslant i \leqslant k$. These inequalities are sometimes also referred to as blossom inequalities. Padberg and Rao [51] gave a polynomial time algorithm to separate over the 2-matching inequalities.

8.3.2.1 *Simple comb inequalities*

Definition 8.3.3. A comb inequality corresponding to a handle H and k teeth T_i is called a simple comb inequality if $|H \cap T_i| = 1$ or $|T_i \setminus H| = 1$ for each $1 \leqslant i \leqslant k$.

Simple comb inequalities contain all the 2-matching inequalities. It is not known whether one can separate over them in polynomial time.

8.3.2.2 (h, t)-*uniform comb inequalities*

Let us define a subclass of comb inequalities called (h, t)-*uniform comb inequalities* associated with what we will call (h, t)-*uniform combs* for arbitrary $1 \leqslant h < t$.

Definition 8.3.4. A comb, with handle H and k teeth T_i, is said be (h,t)-uniform if $|T_i| = t$ and $H \cap T_i = h$, for all $1 \leqslant i \leqslant k$.

8.3.3 *Odd set inequalities for perfect matching*

Definition 8.3.5. Let V denote the vertex set of K_n. For every odd set $U \subseteq V$ the following inequality is valid for the perfect matching polytope PM_n and is called an odd set inequality.

$$x(\delta(U)) \geqslant 1$$

For a given odd set S and a perfect matching M of K_n, the slack between the corresponding odd set inequality and M is denoted by $sl_{odd}(S, M)$.

8.3.4 *t-subdivided prisms of a graph*

Definition 8.3.6. A prism over a graph G is obtained by taking two copies of G and connecting corresponding vertices.

It is helpful to visualise this as stacking the two copies one over the other and then connecting corresponding vertices in the two copies by a vertical edge. A t-subdivided prism is then obtained by subdividing the vertical edges by putting $t-2$ extra vertices on them. See Figure 6 for an example.

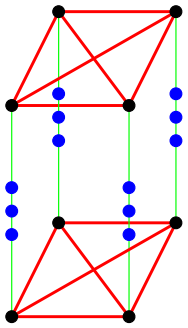


Figure 6: A 5-subdivided prism over K_4.

Let G be the t-subdivided prism of K_n. Let the vertices of the two copies be labeled $u_1^1, \ldots, u_n^1$ and $u_1^t, \ldots, u_n^t$. As a shorthand we will denote the path $u_i^1, u_i^2, \ldots, u_i^t$ as $u_i^1 \rightsquigarrow u_i^t$. Similarily, $u_i^t \rightsquigarrow u_i^1$ will denote $u_i^t, \ldots, u_i^2, u_i^1$.

The graph G has path $u_i^1 \rightsquigarrow u_i^t$ for all $i \in [n]$ and $(u_i^1, u_j^1), (u_i^t, u_j^t)$ for all $i \neq j, i, j \in [n]$. Thus G has tn vertices and $2\binom{n}{2} + (t-1)n$ edges.

8.3.5 *Motivation*

The motivation for looking at t-subdivided prisms stems from a simple observation which we state in the form of a proof of the following proposition:

Proposition 8.3.7. *Let* $2\mathrm{MP}(n)$ *be the convex hull of the incidence vectors of all 2-matchings of the complete graph* K_n*. Then,* $\mathrm{xc}(2\mathrm{MP}(n)) \geqslant 2^{\Omega(n)}$.

Proof. Let G be a graph with n vertices and m edges and let G' be the 3-subdivided prism of G. G' has $3n$ vertices and $2m + 2n$ edges. Any 2-matching in G' contains all the vertical edges and thus when restricted to a single copy – say the bottom one – of G gives a matching in G. Conversely, any matching in G can be extended to a (not necessarily unique) 2-matching in G'.

Taking G as K_n we obtain a G' that is a subgraph of K_{3n}. The 2-matching polytope of G' lies on a face of the 2-matching polytope of the complete graph on $3n$ vertices (corresponding to all missing edges having value 0). Therefore, the extension complexity of the 2-matching polytope $2\mathrm{MP}(n)$ is at least as large as that of the perfect matching polytope. That is, $\mathrm{xc}(2\mathrm{MP}(n)) \geqslant 2^{\Omega(n)}$. □

The above generalizes to P-matching polytopes for arbitrary P in the obvious way, and is probably part of folklore[1].

The generalization of the 3-subdivided prism to larger subdivisions allows us to be able to argue not only about the 2-matching inequalities – which are the facet-defining inequalities for the 2-matching polytope – but also about comb inequalities by using the vertical paths as teeth for constructing combs.

8.3.6 *Uniform combs of odd sets*

Let n and t be positive integers. In the following we will assume that n is a multiple of t. Since we are interested in asymptotic statements only, this does not result in any loss of generality. Let G be the t-subdivided prism of $K_{n/t}$ for some $t \geqslant 2$. Given an odd set S and a perfect matching M in $K_{n/t}$, and arbitrary $1 \leqslant h < t$, we are interested in constructing a comb C and a TSP tour T in K_n such that the following conditions hold:

(C1): C is a (h, t)-uniform comb.

(C2): C depends only on S and 2 edges of M.

(C3): T depends only on M.

(C4): $\mathrm{sl}_{\mathrm{comb}}(C, T) = \mathrm{sl}_{\mathrm{odd}}(S, M)$.

If such a pair (C, T) of a comb and a TSP tour is shown to exist for every pair (S, M) of an odd set and a perfect matching, then we can show that any EF-protocol for computing the slack $\mathrm{sl}_{\mathrm{comb}}(C, T)$ can be used to construct an EF-protocol for computing $\mathrm{sl}_{\mathrm{odd}}(S, M)$ due to condition (C4). Furthermore, due to conditions (C2) and (C3) the number of bits required for the later protocol will not be much larger than the number of bits required for the former, as C can be locally constructed from S after an exchange of two edges, and T can be locally constructed from M.

1 W. Cook (private communication) attributes the same argument to T. Rothvoß

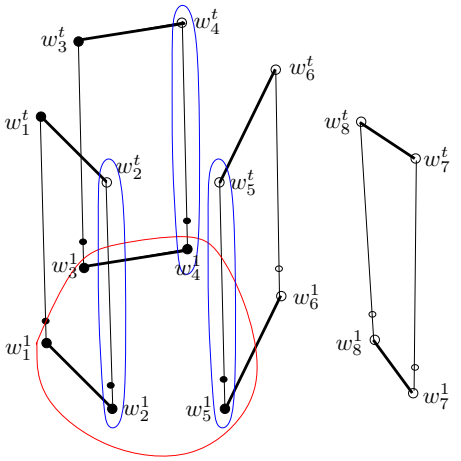


Figure 7: Construction of a comb from given odd set: The odd set consists of 5 vertices displayed as big filled circles in the bottom copy. The corresponding handle consists of all vertices represented by filled circles. The teeth are represented by the vertical ellipsoidal enclosures.
The big circles represent vertices of the original graph and their top copies. The small circles represent the h-th copy, while the other copies have been omitted here. Bold edges at the bottom are matching edges. All other edges displayed are just for illustration of the relationship of various copies of vertices.

Now we show that such a pair does exist if at least two edges of M are contained in S and $|S| \geqslant 5$.

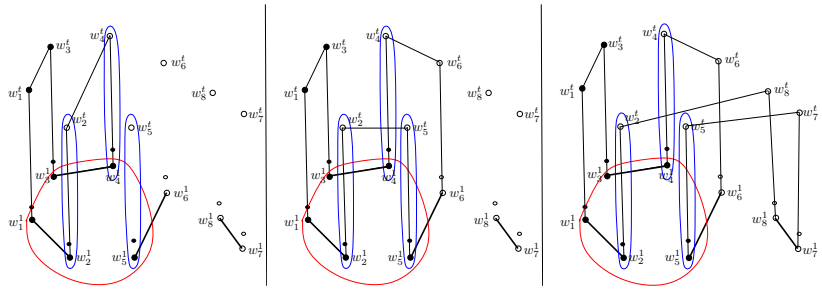
Proposition 8.3.8. *Let (S, M) be a pair of an odd set and a perfect matching in $K_{n/t}$, and let $1 \leqslant h < t$. Suppose that $|S| \geqslant 5$, and let $w_1, w_2, w_3, w_4 \in S$ be distinct with (w_1, w_2) and (w_3, w_4) in M. Then, there exists a pair (C, T) of a comb C and a TSP tour T in K_n satisfying the four conditions (C1)–(C4).*

Proof. Let $|S| = s$. For simplicity of exposition, we assume that the vertices of S are labeled $w_1, \ldots, w_s$. By w_i^j, we denote the copy of w_i in the j-th layer of the t-subdivided prism over $K_{n/t}$.

The comb C is constructed as follows. The handle H is obtained by taking all vertices in S and the copies $w_1^2, \ldots, w_1^t$ and $w_3^2, \ldots, w_3^t$. For every other vertex $w \in S$ the vertices $w^2, \ldots, w^h$ are also added to H. The teeth T_i are formed by pairing each vertex v in $S \setminus \{w_1, w_3\}$ with its copies $v^2, \ldots, v^t$ producing $s-2$ teeth. See Figure 7 for an illustration. Since $s \geqslant 5$ is odd, the number of teeth is odd and at least 3. Thus, the constructed comb is (h, t)-uniform satisfying conditions (C1) and (C2), and the corresponding comb inequality is

$$x(\delta(H)) + \sum_{i=1}^{s-2} x(\delta(T_i)) \geqslant 3(s-2) + 1. \tag{16}$$

To construct a tour T from the given perfect matching M such that conditions (C3) and (C4) are satisfied, we start with a subtour $(w_1^1 \rightsquigarrow w_1^t, w_3^t \rightsquigarrow w_3^1, w_4^1 \rightsquigarrow w_4^t, w_2^t \rightsquigarrow w_2^1, w_1^1)$. At each stage we maintain a subtour that contains all matching edges on the induced vertices in the lower copy, the edge (w_1^t, w_3^t), and at least one top edge different from (w_1^t, w_3^t). Clearly the starting subtour satisfies these requirements. As long as we have some matching edges in M that are not in our subtour, we pick an arbitrary edge (w_a, w_b) in M and extend our subtour as follows. Select a top edge (w_q^t, w_r^t) different from (w_1^t, w_3^t), remove the edge and add the path $(w_q^t, w_a^t \rightsquigarrow w_a^1, w_b^1 \rightsquigarrow w_b^t, w_r^t)$. The new subtour contains the selected perfect matching edge (w_a^1, w_b^1), the paths $w_a^1 \rightsquigarrow w_a^t$ and $w_b^1 \rightsquigarrow w_b^t$ and has one more top edge distinct from (w_1^t, w_3^t) than in the previous subtour. See Figure 8 for an example.



(a) The initial tour going through $w_1^1, w_1^t, w_3^t, w_3^1, w_4^1, w_4^t, w_2^t, w_2^1, w_1^t$

(b) Adding a new matching edge (w_5, w_6) by removing (w_2^t, w_4^t)

(c) The final tour

Figure 8: Constructing a TSP tour from a perfect matching.

At the completion of the procedure, we have a TSP tour that satisfies the following properties:

1. Each edge of M is used in the tour.
2. Each vertical path $w_i^1 \rightsquigarrow w_i^t$ for all $i \in [n]$ is used in the tour.
3. Edge (w_1^t, w_3^t) is used in the tour.

From the construction, edges in $|\delta(H) \cap T|$ are precisely the edges in $|\delta(S) \cap M|$ together with $s-2$ other edges exiting the comb: one through each of the $s-2$ teeth. Therefore, $|\delta(H) \cap T| = |\delta(S) \cap M| + s - 2$. Also, the tour T enters and exits each teeth precisely once so $|\delta(T_i) \cap T| = 2$ for each of the $s-2$ teeth. Substituting these values in the inequality 16, we obtain the slack $\mathrm{sl}_{\mathrm{comb}}(C, T) = |\delta(S) \cap M| + (s-2) + 2(s-2) - 3(s-2) - 1 = \mathrm{sl}_{\mathrm{odd}}(S, M)$. This completes the proof because the pair (C, T) satisfies conditions (C1)–(C4).

□

Using the existence of the pair (C, T) as described earlier and the fact that any EF-protocol for the perfect matching polytope requires an exchange of a linear number of bits, we will lower bound the number of bit exchanged by any EF-protocol computing the slack of (h, t)-uniform comb inequalities with respect to TSP tours. In the next section we will use the following proposition multiple times by fixing different values for the parameters h and t.

Proposition 8.3.9. *Any EF-protocol computing the slack of (h, t)-uniform comb inequalities with respect to the TSP tours of K_n, requires an exchange of $\Omega(n/t)$ bits. Equivalently, the extension complexity of the polytope of (h, t)-uniform comb inequalities is $2^{\Omega(n/t)}$.*

Proof. Due to Proposition 3.1.13 and 3.1.25, it suffices to show if such a protocol uses r bits, then an EF-protocol for the perfect matching polytope for $K_{n/t}$ can be constructed, that uses $r + \mathcal{O}(\log(n/t))$ bits. The protocol for computing the slack of an odd set inequality with respect to a perfect matching in $K_{n/t}$ works as follows.

Suppose Alice has an odd set S in $K_{n/t}$, with $|S| = s$, and Bob has a matching M in $K_{n/t}$. The slack of the odd-set inequality corresponding to S with respect to matching M in the perfect matching polytope for $K_{n/t}$ is $|\delta(S) \cap M| - 1$.

We assume that $s \geqslant 5$. Otherwise, Alice can send the identity of the entire set S with at most $4 \log(n/t)$ bits and Bob can output the slack exactly.

Alice first sends an arbitrary vertex $w_1 \in S$, to Bob. Bob replies with the matching vertex of w_1, say w_2. Alice then sends another arbitrary vertex $w_3 \in S, w_3 \neq w_2$ to Bob who again replies with the matching vertex for w_3, say w_4. So far the number of bits exchanged is $4 \lceil \log(n/t) \rceil$.

Now there are two possibilities: either at least one of the vertices w_2, w_4 is not in S, or both w_2, w_4 are in S. Alice sends one bit to communicate which of the possibilities has occurred and accordingly they switch to one of the two protocols as described next.

In the former case, Alice has identified an edge, say e, in $\delta(S) \cap M$. Now Bob selects an edge e' of his matching uniformly at random (i.e. with probability $2/n$) and sends it to Alice. If e' is in $\delta(S) \setminus \{e\}$, Alice outputs $n/2$. Otherwise, Alice outputs zero. The expected contribution by edges in $(\delta(S) \cap M) \setminus \{e\}$ is then exactly one while the expected contribution of all other edges is zero. Therefore the expected output is $|\delta(S) \cap M| - 1$, and the number of bits exchanged for this step is $\lceil \log m \rceil$ where m is the number of edges in $K_{n/t}$. Thus the total cost in this case is $O(\log(n/t))$ bits.

In the latter case, the matching edges (w_1, w_2) and (w_3, w_4) lie inside S. Alice constructs a comb C in the t-subdivided prism of $K_{n/t}$, and Bob a TSP tour T in the t-subdivided prism of $K_{n/t}$ such that (C, T) satisfies conditions (C1)–(C4). By Proposition 8.3.8 they can do this without exchanging any more bits. Since $\mathrm{sl}_{\mathrm{comb}}(C, T) = \mathrm{sl}_{\mathrm{odd}}(S, M)$, they proceed to compute the corresponding slack with the new inequality and tour, exchanging r bits. The total number of bits exchanged in this case is $r + 4 \lceil \log(n/t) \rceil + 1 = r + O(\log(n/t))$. □

8.3.7 *lower bounds*

In this section we consider the extension complexity of the polytope of comb inequalities and $\mathcal{H}$-free extension complexity of the TSP polytope when $\mathcal{H}$ is the set of simple comb inequalities. As we will see, the results in this section are obtained by instantiating Proposition 8.3.9 with different values of the parameters h and t.

8.3.8 *Extension complexity of Comb inequalities*

We show that the polytope defined by the Comb inequalities has high extension complexity.

Proposition 8.3.10. *Let $COMB(n)$ be the polytope defined by the intersection of all comb inequalities for TSP_n. Then $\mathrm{xc}(COMB(n)) \geqslant 2^{\Omega(n)}$.*

Proof. Suppose there exists an EF-protocol that computes the slack of $\mathrm{COMB}(n)$ that uses r bits. Since $(1,2)$-uniform comb inequalities are valid for TSP_n we can use the given protocol to compute the slack of these inequalities with respect to the TSP tours of K_n using r bits. Then, using Proposition 8.3.9, the slack matrix of the perfect matching polytope for $K_{n/2}$ can be computed using $r + \mathcal{O}(\log n)$ bits. By Proposition 3.1.13 and 3.1.25, this must be $\Omega(n)$. Finally, by Proposition 3.1.25 this implies that $\mathrm{xc}(\mathrm{COMB}(n)) \geqslant 2^{\Omega(n)}$. □

8.3.9 $\mathcal{H}$-free extension complexity

Let $\mathcal{C}_{h,t}$ be the set of (h,t)-uniform comb inequalities for fixed values of h and t. Observe that, since at least three teeth are required to define a comb and the handle must contain some vertex not in any teeth, for (h,t)-uniform combs on n vertices we must have $t \leqslant \left\lfloor \frac{n-1}{3} \right\rfloor$. So for any values of $1 \leqslant h < t \leqslant \left\lfloor \frac{n-1}{3} \right\rfloor$, the set $\mathcal{C}_{h,t}$ is a nonempty set of facet-defining inequalities for TSP_n, and for any other values of h and t the set $\mathcal{C}_{h,t}$ is empty.

Proposition 8.3.11. *If $\mathcal{H}$ is a set of inequalities valid for the polytope TSP_n, such that $\mathcal{H} \cap \mathcal{C}_{h,t} = \emptyset$ for some nonempty $\mathcal{C}_{h,t}$, then the $\mathcal{H}$-free extension complexity of TSP_n is at least $2^{\Omega(n/t)}$.*

Proof. Let $1 \leqslant h < t$ be integers such that $\mathcal{H} \cap \mathcal{C}_{h,t} = \emptyset$. That is, the set $\mathcal{H}$ does not contain any (h,t)-uniform comb inequalities. Let P be the polytope formed from TSP_n by throwing away any facet-defining inequalities that are in $\mathcal{H}$. Then, any EF-protocol computing the slack matrix of P correctly must use $\Omega(n/t)$ bits due to Proposition 8.3.9. The claim then follows from Proposition 3.1.25. □

The previous proposition shows that for every set $\mathcal{H}$ of valid inequalities of TSP_n, if the extension complexity of the TSP polytope becomes polynomial after removing the inequalities in $\mathcal{H}$, then $\mathcal{H}$ must contain some inequalities from every (h,t)-uniform comb inequality class, for all $t = o(n/\log n)$.

Exercise 8.3.12. Show that the statement can be made stronger by replacing the requirement $\mathcal{H} \cap \mathcal{C}_{h,t} = \emptyset$ with $|\mathcal{H} \cap \mathcal{C}_{h,t}| \leqslant \mathrm{poly}(n)$. (**Hint:** See the discussion in Section 8.3).

We can use the previous proposition to give lower bounds for $\mathcal{H}$-free extension complexity of the TSP polytope with respect to important classes of valid inequalities by simply demonstrating some class of (h,t)-uniform comb inequalities that has been missed.

2-matching inequalities

Proposition 8.3.13. *Let P be the polytope obtained by removing the 2-matching inequalities from the TSP polytope. Then, $\mathrm{xc}(P) = 2^{\Omega(n)}$.*

Proof. The 2-matching inequalities are defined by combs for which each tooth has size exactly two. Therefore the set of $(1,3)$-uniform combs are not 2-matching inequalities, and Proposition 8.3.11 applies. □

Simple comb inequalities

Proposition 8.3.14. *Let* P *is the polytope obtained by removing the set of simple comb inequalities from the TSP polytope. Then,* $\mathrm{xc}(P) = 2^{\Omega(n)}$.

Proof. Recall that a comb is called simple if $|H \cap T_i| = 1$ or $|T_i \setminus H| = 1$ for all $1 \leqslant i \leqslant k$ where k is the (odd) number of teeth in the comb and H is the handle. Clearly, $(2,4)$-uniform combs are not simple and Proposition 8.3.11 applies. □

As mentioned before, simple comb inequalities define a superclass of 2-matching inequalities and a polynomial time separation algorithm is known for 2-matching inequalities. Althought a similar result was claimed for simple comb inequalities, the proof was apparently incorrect, as pointed out by Fleischer et al. [29]. This latter paper includes a polynomial time separation algorithm for the wider class of simple domino-parity inequalities that we do not consider here.

It remains unknown whether there exists a polynomial time separation algorithm for the (h,t)-uniform comb inequalities.

9

WEAK EXTENDED FORMULATIONS

9.1 P-COMPLETENESS OF LINEAR PROGRAMMING

It is well established that Linear Programming is P-complete with respect to logspace reductions. That is, any problem in P can be reduced to a Linear Programming problem using a logspace reduction. On the other hand it is now also known that the perfect matching polytope has exponential extension complexity. That is any polytope that projects to the perfect matching polytope of the complete graph K_{2n} requires at least $2^{\Omega(n)}$ inequalities to describe. These two facts may appear contradictory at first sight. How come the perfect matching problem is solvable in polynomial time and yet the polytope requires exponential size?

The previous conundrum is easily resolved if one notices subtle differences between decision and optimization problems, and between "reduction to a Linear Program" and "extension complexity of the perfect matching polytope". Extended formulations require the feasible region of the LP formulations to project exactly to the perfect matching polytope, while a reduction to a Linear Programming problem may produce other polytopes as a feasible region. Based on the particular objective function (that is, a particular instance) the reduction may produce different polytopes. At the heart of this issue is the fact that logspace reductions can do fairly non-trivial computation with the objective function. Indeed, it is not even known whether LOGSPACE $\neq$ NP. So for all we know, the reduction may as well solve the perfect matching problem and produce a trivial LP instance.

One may still obtain reasonable and interesting statements if one were to ask the following: Can we obtain a small Linear Program for perfect matching even if the input instance is allowed to be modified only in very restricted ways? The answer obviously depeneds on how this question is formulated in a precise way. We will give one interpretation and obtain small Linear Programs for problems in P/poly. Before we formalize anything we start with an example.

Definition 9.1.1. Let n be an even integer and let x be a binary vector of length $\binom{n}{2}$. We let $G(x) = (V, E)$ denote the graph with edge incidence vector given by x, let n be the number of its vertices and $m = \mathbf{1}^\top x$ the number of its edges. Furthermore, let $w_x = 1$ if $G(x)$ has a perfect matching and zero otherwise. We define the polytope PM_n as:

$$PM_n = \operatorname{conv}\left\{ \begin{pmatrix} x \\ w_x \end{pmatrix} \middle| \, x \in \{0,1\}^{\binom{n}{2}} \right\} \tag{17}$$

PM_n may be visualized by starting with a hypercube in dimension $\binom{n}{2}$ and embedding it in one higher dimension with extra coordi-

nate w. For vertices of the cube corresponding to graphs with perfect matchings $w = 1$ else $w = 0$. It is easy to see that PM_n has precisely $2^{\binom{n}{2}}$ vertices. EP_n is closely related to PM_n, in fact it forms a face.

Proposition 9.1.2. *EP_n is a face of PM_n and can be defined by*

$$EP_n = \left\{ x \;\middle|\; \begin{pmatrix} x \\ w \end{pmatrix} \in PM_n \cap \left\{ \mathbf{1}^\top x + (1-w)n^2 = \frac{n}{2} \right\} \right\} \tag{18}$$

Proof. We first show that the inequality

$$\mathbf{1}^\top x + (1-w)n^2 \geqslant \frac{n}{2} \tag{19}$$

is valid for PM_n. We need only verify it for the extreme points (x, w_x) given in (17). If $w_x = 0$, (19) holds since $\mathbf{1}^\top x + n^2 \geqslant \frac{n}{2}$. Otherwise $w_x = 1$, x is the incidence vector of graph containing a perfect matching, so $\mathbf{1}^\top x \geqslant n/2$. The vectors x with $w_x = 1$ and $\mathbf{1}^\top x = n/2$ are the incidence vectors of perfect matchings of K_n and are precisely those used to define EP_n. □

For a given input graph $G(\bar{x}) = (V, E)$ we define the vector $\mathbf{c}$ by:

$$\mathbf{c}_{ij} = 1 \quad ij \in E \qquad \mathbf{c}_{ij} = -1 \quad ij \notin E \qquad 1 \leqslant i < j \leqslant n \tag{20}$$

and let d be a constant such that $0 < d \leqslant 1/2$. We construct the LP:

$$\begin{aligned} z^* = \max\ z &= \mathbf{c}^\top x + dw \\ \begin{pmatrix} x \\ w \end{pmatrix} &\in PM_n \end{aligned} \tag{21}$$

Proposition 9.1.3. *For any edge incidence vector $\bar{x} \in [0,1]^{\binom{n}{2}}$ let $m = \mathbf{1}^\top \bar{x}$. The optimum solution to (21) is unique, $z^* = m + d$ if $G(\bar{x})$ has a perfect matching, and $z^* = m$ otherwise.*

Proof. See [4], Proposition 2 (Appendix J). □

9.2 WEAK EXTENDED FORMULATIONS

Let X denote a poly-time decision problem defined on binary input vectors $x = (x_1, \ldots, x_q)$, and an additional bit w_x, where $w_x = 1$ if x results in a "yes" answer and $w_x = 0$ otherwise. We define the polytope P as:

$$P = \operatorname{conv}\left\{ \begin{pmatrix} x \\ w_x \end{pmatrix} \middle| \, x \in \{0,1\}^q \right\} \tag{22}$$

For a given binary input vector $\bar{x}$ we define the vector $\mathbf{c}$ by:

$$\mathbf{c}_j = 1 \quad \bar{x}_j = 1 \quad \text{and} \quad \mathbf{c}_j = -1 \quad \bar{x}_j = 0 \qquad 1 \leqslant j \leqslant q \tag{23}$$

and let d be a constant such that $0 < d \leqslant 1/2$. As before we construct an LP:

$$\begin{aligned} z^* = \max\ z &= \mathbf{c}^\top x + dw \\ \begin{pmatrix} x \\ w \end{pmatrix} &\in P \end{aligned} \tag{24}$$

The following proposition can be proved in an identical way to Proposition 9.1.3.

Proposition 9.2.1. *For any $\bar{x} \in [0,1]^q$ let $m = \mathbf{1}^\top \bar{x}$. The optimum solution to (24) is unique, $z^* = m + d$ if $\bar{x}$ has a "yes" answer and $z^* = m$ otherwise.*

Definition 9.2.2. Let Q be a polytope which is a subset of the $(q+t)$-cube with variables labeled $x_1, ..., x_q, y_1, ..., y_t$. We say that Q has the *x-o/1 property* if each of the 2^q ways of assigning $0/1$ to the x variables uniquely extends to a vertex $(x^\top, y^\top)^\top$ of Q and, furthermore, y is $0/1$ valued. Q may have additional fractional vertices.

In polyhedral terms, this says that the intersection of Q with the hyperplanes $x_j = e_j, j = 1, ..., q$ is a $0/1$ vertex, for each assignment of zero or one to the e_j's. We can show that we can solve a polytime decision problem X by replacing P in (21) by a polytope Q of polynomial size, while maintaining the same objective functions. We call Q a *weak extended formulation* as it does not necessarily project onto P.

Definition 9.2.3. A polytope

$$Q = \left\{ \begin{pmatrix} x \\ w \\ s \end{pmatrix} \in [0,1]^{q+1+r} \;\middle|\; Ax + bw + Cs \leqslant h \right\}$$

is a *weak extended formulation (WEF)* of P if

- Q has the x-$0/1$ property.
- For any binary vector $\bar{x} \in [0,1]^q$ let $m = \mathbf{1}^\top \bar{x}$. Let c be defined by (23) and let $0 < d \leqslant 1/2$. The optimum solution

$$z^* = \max \left\{ c^\top x + dw \;\middle|\; \begin{pmatrix} x \\ w \\ s \end{pmatrix} \in Q \right\}$$

  is unique and takes the value $z^* = m + d$ if $\bar{x}$ has a "yes" answer. Otherwise $z^* < m + d$ and for all sufficiently small d, $z^* = m$ and is unique.

For example, let X be the perfect matching problem so that $P = PM_n$. Let $Q = Q_n$ be a WEF as given by this definition. It follows from Proposition 9.1.3 that we can determine whether an input graph G has a perfect matching by solving an LP over either PM_n or Q_n using the same objective function which is derived directly from the edge adjacency vector of G.

Example 9.2.4. Consider $n = 2$ giving $PM_2 = \text{conv}\{(0,0)^\top, (1,1)^\top\}$. A WEF, for example, is given by:

$$Q_2 = \text{conv}\{(0,0,0)^\top, (1,1,1)^\top, (1/4,1,1/2)^\top\}$$

Initially let $d = 1/2$. When $G(\bar{x})$ is an edge, $m = 1$, $c_{12} = 1$ and $z = c^\top x + dw$ obtains the same optimum solution of $z^* = 3/2 = m + d$

over both PM_2 and Q_2. When $G(\bar{x})$ is a non-edge, $m = 0$, $c_{12} = -1$ and $z = c^\top x + dw$ obtains the optimum solution of $z^* = 0 = m$ over PM_2 and $z^* = 1/4 < 1/2 = m + d$ over Q_2, at the fractional vertex $(1/4, 1, 1/2)^\top$. However, if $0 < d < 1/4$ then $z = \mathbf{c}^\top \mathbf{x} + dw$ obtains the unique optimum solution of $z^* = 0 = m$ over both PM_2 and Q_2. We see that Q_2 projects onto a triangle in the (x, w)-space, whereas PM_2 is a line segment.

9.3 WEAK EXTENSION FOR P/poly

In order to show that Linear Programming is **P**-complete, Valiant [61] gave a construction to transform boolean circuits into a linear sized set of linear inequalities with the x-o/1 property (where x_i are the variables corresponding to the inputs of the circuit); a similar construction was used by Yannakakis [63] in the context of the Hamiltonian Circuit problem. One can show that Valiant's construction implies the following.

Proposition 9.3.1. *Every decision problem* X *in* P/poly *admits a weak extended formulation* Q *of polynomial size.*

Valiant's point of view is slightly different from ours in that he explicitly fixes the values of the input variables before solving an LP-feasibility problem (as opposed to using different objective functions with a fixed set of inequalities). Showing that the result of this fixing is a 0/1-vertex is precisely our x-o/1 property.

We begin with a standard definition[1]:

Definition 9.3.2. A *(boolean) circuit* with q inputs $x = (x_1, x_2, \ldots, x_q)$ is a directed acyclic graph in which each of its t nodes, called *gates*, is either an AND($\wedge$) gate, an OR($\vee$) or a NOT($\neg$) gate. We label each gate by its output bit. One of these gates is designated as the *output gate* and gives output bit w. The *size* of a circuit is the number of gates it contains and its *depth* is the maximal length of a path from an input gate to the output gate.

For example, the circuit shown in Figure 9 can be used to compute whether or not a graph on 4 nodes has a perfect matching. The input is the binary edge-vector of the graph and the output is $w = 1$ if the graph has a matching (e.g. G_1) or $w = 0$ if it does not (e.g. G_2). If the graph has a perfect matching, exactly one of y_{12}, y_{13} or y_{14} is one, defining the matching. For each gate we have labeled the output bit by a new variable. We will construct a polytope from the circuit by constructing a system of inequalities on the same variables.

From an AND gate, say $y_{12} = x_{12} \wedge x_{34}$, we generate the inequalities:

$$\begin{aligned} x_{12} + x_{34} - y_{12} &\leqslant 1 \\ -x_{12} \qquad + y_{12} &\leqslant 0 \\ -x_{34} + y_{12} &\leqslant 0 \\ y_{12} &\geqslant 0 \end{aligned} \tag{25}$$

1 See, e.g., the text by Savage [56]

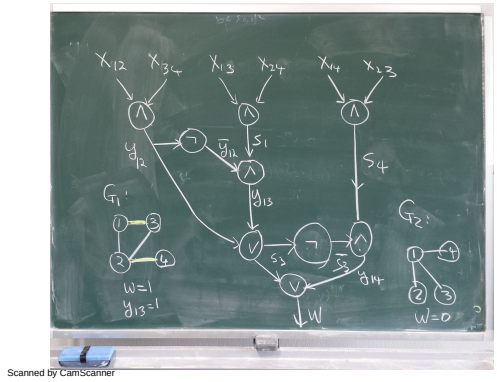


Figure 9: A circuit to compute whether a 4 node graph has a perfect matching

The system (25) defines a polytope in three variables whose 4 vertices represent the truth table for the AND gate:

x_{12}	x_{34}	y_{12}
0	0	0
0	1	0
1	0	0
1	1	1

Note that the variables x_{12}, x_{34} define a 2-cube and so the polytope is an extension of the 2-cube. In the terminology of the last section, it has the $\{x_{12}, x_{34}\}$-0/1 property.

From an OR gate, say $s_3 = y_{12} \vee y_{13}$, we generate the inequalities:

$$\begin{aligned} -y_{12} - y_{13} + s_3 &\leqslant 0 \\ y_{12} - s_3 &\leqslant 0 \\ y_{13} - s_3 &\leqslant 0 \\ s_3 &\leqslant 1 \end{aligned} \tag{26}$$

The system (26) defines a polytope in three variables whose 4 vertices represent the truth table for the OR gate, as can easily be checked. Indeed, this polytope has the $\{y_{12}, y_{13}\}$-0/1 property.

From a NOT gate, say $\bar{y}_{12} = \neg y_{12}$, we could generate the equation

$$\bar{y}_{12} = 1 - y_{12} \tag{27}$$

However it is equivalent to just replace all instances of $\bar{y}_{12}$ by $1 - y_{12}$ in the inequality system, and this is what we will do in the sequel.

The circuit in Figure 9 contains 5 AND gates and 2 OR gates. By suitably replacing variables in (25) and (26) we obtain a system of 28 inequalities in 13 variables. As just mentioned, the NOT gates are handled by variable substitution rather than explicit equations. Let Q_4 denote the corresponding polytope. It will follow by the general argument below that Q_4 is a weak extended formulation (WEF) of PM_4.

It can be shown that the above construction can be applied to any boolean circuit C to obtain a polytope Q which has the 0/1 property with respect to the inputs of C.

Proposition 9.3.3 ([61]). *Let* C *be a boolean circuit with* q *input bits* $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_q)$, t *gates labeled by their output bits* $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_t)$ *and with circuit output bit* $w = \mathbf{y}_t$. *Construct the polytope* Q *with* $4t$ *inequalities and* $q + t$ *variables using the systems (25) and (26) respectively.* Q *has the the* $\mathbf{x}$*-0/1 property and for every input* $\mathbf{x}$ *the value of* w *computed by* C *corresponds to the value of* $\mathbf{y}_t$ *in the unique extension* $(\mathbf{x}^\top, \mathbf{y}^\top)^\top \in Q$ *of* $\mathbf{x}$.

Proof. See [4], Lemma 1 (Appendix J). □

This allows one to construct a WEF for any problem in P/poly.

Proposition 9.3.4. *Let* C *be a circuit that solves a decision problem* X *with* q *input bits* $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_q)$ *and has associated polytope* P *as defined in (22). The polytope* Q *constructed in Proposition 9.3.3 is a WEF for* P.

Proof. See [4], Lemma 2 (Appendix J). □

Since each gate in the circuit gives rise to 4 inequalities and one new variable, we have the following.

Proposition 9.3.5. *Let* X *be a decision problem with corresponding polytope* P *defined by (22). A set of circuits for* X *with size* $p(n)$ *generate a WEF* Q *for* P *with* $4p(n)$ *inequalities and variables.*

BIBLIOGRAPHY

[1] David Avis and Hans Raj Tiwary. "A generalization of extension complexity that captures P." In: *Information Processing Letters* 115.6-8 (2015), pp. 588–593. DOI: 10.1016/j.ipl.2015.02.005.

[2] David Avis and Hans Raj Tiwary. "On the extension complexity of combinatorial polytopes." In: *Mathematical Programming* 153.1 (2015), pp. 95–115. DOI: 10.1007/s10107-014-0764-2.

[3] David Avis and Hans Raj Tiwary. "On the $\mathcal{H}$-free extension complexity of the TSP." In: *Optimization Letters* (2016), pp. 1–11. ISSN: 1862-4480. DOI: 10.1007/s11590-016-1029-1.

[4] David Avis, David Bremner, Hans Raj Tiwary, and Osamu Watanabe. "Polynomial size linear programs for non-bipartite matching problems and other problems in P." In: *CoRR* abs/1408.0807 (2014). eprint: arXiv:1408.0807.

[5] David Avis, Hiroshi Imai, Tsuyoshi Ito, and Yuuya Sasaki. "Two-party Bell inequalities derived from combinatorics via triangular elimination." In: *Journal of Physics A: Mathematical and General* 38.50 (2005), p. 10971. DOI: 10.1088/0305-4470/38/50/007.

[6] Ajesh Babu, Nutan Limaye, Jaikumar Radhakrishnan, and Girish Varma. "Streaming algorithms for language recognition problems." In: *Theoretical Computer Science* 494 (2013), pp. 13–23. DOI: 10.1016/j.tcs.2012.12.028.

[7] Francisco Barahona. "On cuts and matchings in planar graphs." In: *Mathematical Programming* 60 (1993), pp. 53–68. DOI: 10.1007/BF01580600.

[8] Gábor Braun and Sebastian Pokutta. "Common information and unique disjointness." In: *Proc. FOCS*. 2013. DOI: 10.1109/FOCS.2013.79.

[9] Gábor Braun, Samuel Fiorini, Sebastian Pokutta, and David Steurer. "Approximation Limits of Linear Programs (Beyond Hierarchies)." In: *Mathematics of Operations Research* 40.3 (2015), pp. 756–772. DOI: 10.1287/moor.2014.0694.

[10] Gábor Braun, Rahul Jain, Troy Lee, and Sebastian Pokutta. "Information theoretic approximations of the nonnegative rank." In: *Electronic Colloquium on Computational Complexity (ECCC)* 20 (2013), p. 158. URL: http://eccc.hpi-web.de/report/2013/158.

[11] Mark Braverman and Ankur Moitra. "An information complexity approach to extended formulations." In: *Proc. STOC*. 2013, pp. 161–170. DOI: 10.1145/2488608.2488629.

[12] Jop Briët, Daniel Dadush, and Sebastian Pokutta. "On the existence of 0/1 polytopes with high semidefinite extension complexity." In: *Mathematical Programming* 153.1 (2015), pp. 179–199. DOI: 10.1007/s10107-014-0785-x.

[13] Austin Buchanan and Segiy Butenko. *Tight extended formulations for independent set*. Available on Optimization Online. 2014. URL: http://www.optimization-online.org/DB_HTML/2014/09/4540.html.

[14] Michael R. Bussieck and Marco E. Lübbecke. "The vertex set of a 0/1-polytope is strongly P-enumerable." In: *Computational Geometry* 11.2 (1998), pp. 103–109. DOI: 10.1016/S0925-7721(98)00021-2.

[15] Kathie Cameron. "Induced matchings." In: *Discrete Applied Mathematics* 24.1-3 (1989), pp. 97–102. DOI: 10.1016/0166-218X(92)90275-F.

[16] Robert D. Carr and Goran Konjevod. "Polyhedral Combinatorics." In: *Tutorials on Emerging Methodologies and Applications in Operations Research*. Vol. 76. International Series in Operations Research & Management Science. 2005, pp. 2–1–2–46. ISBN: 978-0-387-22826-6.

[17] Lewis Carroll. *Through the Looking Glass*. Penguin Books, 1871.

[18] Siu On Chan, James R. Lee, Prasad Raghavendra, and David Steurer. "Approximate Constraint Satisfaction Requires Large LP Relaxations." In: *Proc. FOCS*. 2013, pp. 350–359. DOI: 10.1109/FOCS.2013.45.

[19] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. "Extended formulations in combinatorial optimization." In: *4OR* 8.1 (2010), pp. 1–48. DOI: 10.1007/s10288-010-0122-z.

[20] Michele Conforti and Kanstantsin Pashkovich. "The projected faces property and polyhedral relations." In: *Mathematical Programming* 156.1-2 (2016), pp. 331–342. DOI: 10.1007/s10107-015-0882-5.

[21] Michel Marie Deza and Monique Laurent. *Geometry of cuts and metrics*. Vol. 15. Algorithms and Combinatorics. Springer-Verlag, 1997, pp. xii, 587. DOI: 10.1007/978-3-642-04295-9.

[22] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013, I–SSS, 3–707. ISBN: 978-1-4471-5558-4; 978-1-4471-5559-1.

[23] Jack Edmonds. "Maximum Matching and a Polyhedron with 0, 1 Vertices." In: *Journal of Research of the National Bureau of Standards* 69 B (1965), pp. 125–130.

[24] Yuri Faenza, Samuel Fiorini, Roland Grappe, and Hans Raj Tiwary. "Extended formulations, nonnegative factorizations, and randomized communication protocols." In: *Mathematical Programming* 153.1 (2015), pp. 75–94. DOI: 10.1007/s10107-014-0755-3.

[25] Hamza Fawzi and Pablo A. Parrilo. "Exponential lower bounds on fixed-size psd rank and semidefinite extension complexity." In: *CoRR* abs/1311.2571 (2013). URL: http://arxiv.org/abs/1311.2571.

[26] Samuel Fiorini, Thomas Rothvoß, and Hans Raj Tiwary. "Extended Formulations for Polygons." In: *Discrete & Computational Geometry* 48.3 (2012), pp. 658–668. DOI: 10.1007/s00454-012-9421-9.

[27] Samuel Fiorini, Serge Massar, Sebastian Pokutta, Hans Raj Tiwary, and Ronald de Wolf. "Exponential Lower Bounds for Polytopes in Combinatorial Optimization." In: *Journal of the ACM* 62.2 (2015), p. 17. DOI: 10.1145/2716307.

[28] Samuel Fiorini, Serge Massar, Manas K Patra, and Hans Raj Tiwary. "Generalized probabilistic theories and conic extensions of polytopes." In: *Journal of Physics A: Mathematical and Theoretical* 48.2 (2015), p. 025302. DOI: 10.1145/2716307.

[29] Lisa Fleischer, Adam N. Letchford, and Andrea Lodi. "Polynomial-Time Separation of a Superclass of Simple Comb Inequalities." In: *Mathematics of Operations Research* 31.4 (2006), pp. 696–713. DOI: 10.1287/moor.1060.0214.

[30] Jakub Gajarský, Petr Hliněný, and Hans Raj Tiwary. "Parameterized Extension Complexity of Independent Set and Related Problems." In: *CoRR* abs/1511.08841 (2015). eprint: arXiv:1511.08841.

[31] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990. ISBN: 0716710455.

[32] Albertus M. H. Gerards. "Compact Systems for T-join and Perfect Matching Polyhedra of Graphs with Bounded Genus." In: *Operations Research Letters* 10.7 (Oct. 1991), pp. 377–382. ISSN: 0167-6377. DOI: 10.1016/0167-6377(91)90038-Q.

[33] João Gouveia, Pablo A. Parrilo, and Rekha R. Thomas. "Lifts of Convex Sets and Cone Factorizations." In: *Mathematics of Operations Research* 38.2 (2013), pp. 248–264. DOI: 10.1287/moor.1120.0575.

[34] Martin Grötschel and Manfred Padberg. "On the symmetric travelling salesman problem II: Lifting theorems and facets." In: *Mathematical Programming* 16.1 (1979), pp. 281–302. DOI: 10.1007/BF01582117.

[35] Branko Grünbaum. *Convex polytopes*. Graduate texts in mathematics. Springer, 2003. ISBN: 0-387-00424-6.

[36] Juris Hartmanis, Neil Immerman, and Stephen R. Mahaney. "One-Way Log-Tape Reductions." In: *Proc. FOCS*. 1978, pp. 65–72. DOI: 10.1109/SFCS.1978.31.

[37] Colin de la Higuera and José Oncina. "Inferring Deterministic Linear Languages." In: *Proc. COLT*. 2002, pp. 185–200. DOI: 10.1007/3-540-45435-7_13.

[38] Hui-kai, Yuan-wu, Huikai, Yuanwu, Katsuki Sekida, and A. V. Grimstone. *Two Zen classics : Mumonkan and Hekiganroku / translated with commentaries by Katsuki Sekida ; edited and introduced by A.V. Grimstone*. English. 1st ed. Weatherhill New York, 1977, 413 p. ; ISBN: 0834801310 0834801302.

[39] Volker Kaibel. *Extended Formulations in Combinatorial Optimization*. Optima 85. 14 pages. 2011. URL: http://www.mathopt.org/Optima-Issues/optima85.pdf.

[40] Volker Kaibel, Kanstantsin Pashkovich, and Dirk Oliver Theis. "Symmetry Matters for Sizes of Extended Formulations." In: *SIAM Journal on Discrete Mathematics* 26.3 (2012), pp. 1361–1382. DOI: 10.1137/110839813.

[41] Volker Kaibel and Stefan Weltge. "A Short Proof that the Extension Complexity of the Correlation Polytope Grows Exponentially." In: *Discrete & Computational Geometry* 53.2 (2015), pp. 397–401. DOI: 10.1007/s00454-014-9655-9.

[42] Richard M. Karp. "Reducibility Among Combinatorial Problems." In: *Proc. Symposium on the Complexity of Computer Computations*. 1972, pp. 85–103. URL: http://www.cs.berkeley.edu/~luca/cs172/karp.pdf.

[43] Phokion G. Kolaitis and Moshe Y. Vardi. "Conjunctive-Query Containment and Constraint Satisfaction." In: *Journal of Computer and System Sciences* 61.2 (2000), pp. 302–332. DOI: 10.1006/jcss.2000.1713.

[44] Petr Kolman, Martin Koutecký, and Hans Raj Tiwary. "Extension Complexity, MSO Logic, and Treewidth." In: *Proceedings of the 15th SWAT* To appear (2016). eprint: arXiv:1507.04907.

[45] Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, 1997.

[46] Troy Lee and Dirk Oliver Theis. *Support-based lower bounds for the positive semidefinite rank of a nonnegative matrix*. arXiv:1203.3961. 2012.

[47] Leonid Libkin. *Elements of Finite Model Theory*. Berlin: Springer-Verlag, 2004. ISBN: 3-540-21202-7.

[48] László Lovász and Michael D. Plummer. *Matching theory*. North-Holland mathematics studies. Includes indexes. Amsterdam, New York: North-Holland, 1986. ISBN: 0-444-87916-1.

[49] R. Kipp Martin. "Using Separation Algorithms to Generate Mixed Integer Model Reformulations." In: *Operations Research Letters* 10.3 (Apr. 1991), pp. 119–128.

[50] Shmuel Onn and Vladimir A. Shlyk. "Some efficiently solvable problems over integer partition polytopes." In: *Discrete Applied Mathematics* 180 (2015), pp. 135–140. DOI: 10.1016/j.dam.2014.08.015.

[51] Manfred W. Padberg and M. R. Rao. "Odd Minimum Cut-Sets and *b*-Matchings." In: *Mathematics of Operations Research* 7.1 (1982), pp. 67–80. DOI: 10.1287/moor.7.1.67.

[52] Dömötör Pálvölgyi. *Partitioning to three matchings of given size is NP-complete for bipartite graphs*. Tech. rep. QP-2013-01. Egerváry Research Group, Budapest, 2013.

[53] Eric Raymond. *The new hacker's dictionary*. MIT Press, 1991.

[54] Thomas Rothvoß. "Some 0/1 polytopes need exponential size extended formulations." In: *Mathematical Programming* 142.1-2 (2013), pp. 255–268. DOI: 10.1007/s10107-012-0574-3.

[55] Thomas Rothvoß. "The matching polytope has exponential extension complexity." In: *Proc. STOC*. 2014, pp. 263–272. DOI: 10.1145/2591796.2591834.

[56] John E. Savage. *Models of Computation: Exploring the Power of Computation*. http://cs.brown.edu/~jes/book, 2015.

[57] Richard Edwin Stearns, Juris Hartmanis, and Philip M. Lewis II. "Hierarchies of memory limited computations." In: *Proc. Symposium on Switching Circuit Theory and Logical Design*. 1965, pp. 179–190. DOI: 10.1109/FOCS.1965.11.

[58] Larry J. Stockmeyer and Vijay V. Vazirani. "NP-Completeness of Some Generalizations of the Maximum Matching Problem." In: *Information Processing Letters* 15.1 (1982), pp. 14–19. DOI: 10.1016/0020-0190(82)90077-1.

[59] Andrzej Szepietowski. "Weak and Strong One-Way Space Complexity Classes." In: *Information Processing Letters* 68.6 (1998), pp. 299–302. DOI: 10.1016/S0020-0190(98)00176-8.

[60] Hans Raj Tiwary. "Extension Complexity of Formal Languages." In: *ArXiv e-prints* (Mar. 2016). arXiv: 1603.07786 [cs.CC].

[61] Leslie G. Valiant. "Reducibility by algebraic projections." In: *Enseignement Mathématique (2)* 28.3-4 (1982), pp. 253–268. ISSN: 0013-8584.

[62] Laurence A. Wolsey. "Using extended formulations in practice." In: (2011). 14 pages. URL: http://www.mathopt.org/Optima-Issues/optima85.pdf.

[63] Mihalis Yannakakis. "Expressing Combinatorial Optimization Problems by Linear Programs." In: *Journal of Computer and System Sciences* 43.3 (1991), pp. 441–466. DOI: 10.1016/0022-0000(91)90024-Y.

[64] Mihalis Yannakakis and Fanica Gavril. "Edge dominating sets in graphs." In: *SIAM Journal of Applield Mathematics* 38 (1980), pp. 364–372. DOI: 10.1137/0138030.

[65] Günter M. Ziegler. *Lectures on polytopes*. Vol. 152. Graduate Texts in Mathematics. Springer-Verlag, 1995, pp. ix+370.

Part IV

APPENDIX

A

EXPONENTIAL LOWER BOUNDS FOR POLYTOPES IN COMBINATORIAL OPTIMIZATION

Samuel Fiorini, Serge Massar, Sebastian Pokutta, Hans Raj Tiwary, and Ronald de Wolf. "Exponential Lower Bounds for Polytopes in Combinatorial Optimization." In:*Journal of the ACM* 62.2 (2015), p. 17. DOI: 10.1145/2716307

B

EXTENDED FORMULATIONS, NONNEGATIVE FACTORIZATIONS, AND RANDOMIZED COMMUNICATION PROTOCOLS

Yuri Faenza, Samuel Fiorini, Roland Grappe, and Hans Raj Tiwary. "Extended formulations, nonnegative factorizations, and randomized communication protocols." In: *Mathematical Programming* 153.1 (2015), pp. 75–94. DOI: 10.1007/s10107-014-0755-3.

C

EXTENDED FORMULATIONS FOR POLYGONS

Samuel Fiorini, Thomas Rothvoß, and Hans Raj Tiwary. "Extended Formulations for Polygons." In: *Discrete & Computational Geometry* 48.3 (2012), pp. 658–668. DOI: 10.1007/s00454-012-9421-9.

D

ON THE EXTENSION COMPLEXITY OF COMBINATORIAL POLYTOPES

David Avis and Hans Raj Tiwary. "On the extension complexity of combinatorial polytopes." In: *Mathematical Programming* 153.1 (2015), pp. 95–115. DOI: 10.1007/s10107-014-0764-2.

E

EXTENSION COMPLEXITY OF FORMAL LANGUAGES

Hans Raj Tiwary. "Extension Complexity of Formal Languages." In: *ArXiv e-prints* (Mar. 2016). arXiv: 1603.07786 [cs.CC].

F

PARAMETERIZED EXTENSION COMPLEXITY OF INDEPENDENT SET AND RELATED PROBLEMS

Jakub Gajarský, Petr Hliněný, and Hans Raj Tiwary. "Parameterized Extension Complexity of Independent Set and Related Problems." In: *CoRR abs/1511.08841* (2015). eprint: arXiv:1511.08841.

G

EXTENSION COMPLEXITY, MSO LOGIC, AND TREEWIDTH

Petr Kolman, Martin Koutecký, and Hans Raj Tiwary. "Extension Complexity, MSO Logic, and Treewidth." In: *Proceedings of the 15th SWAT To appear (2016)*. eprint: arXiv:1507.04907.

H

A GENERALISATION OF EXTENSION COMPLEXITY THAT CAPTURES P

David Avis and Hans Raj Tiwary. "A generalization of extension complexity that captures P." In: *Information Processing Letters* 115.6-8 (2015), pp. 588–593. DOI: 10.1016/j.ipl.2015.02. 005.

I

ON THE $\mathcal{H}$-FREE EXTENSION COMPLEXITY OF THE TSP

David Avis and Hans Raj Tiwary. "On the $\mathcal{H}$-free extension complexity of the TSP." In: *Optimization Letters* (2016), pp. 1–11. ISSN: 1862-4480. DOI: 10.1007/s11590-016-1029-1.

J

POLYNOMIAL SIZE LINEAR PROGRAMS FOR PROBLEMS IN P

David Avis, David Bremner, Hans Raj Tiwary, and Osamu Watanabe. "Polynomial size linear programs for non-bipartite matching problems and other problems in P." In: *CoRR abs/1408.0807* (2014). eprint: arXiv:1408.0807.

DECLARATION

The content of this work are based primarily on articles that I have coauthored. In cases of works by others I have attempted to cite the appropriate source correctly.

Prague, 2016

Hans Raj Tiwary