Constraint Satisfaction Problems and friends: symmetries and algorithm design

by

Akbar Rafiey

M.Sc., Simon Fraser University, 2016
B.Sc., University of Tehran, 2013

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Summer 2022

Declaration of Committee

Name:	**Akbar Rafiey**
Degree:	**Doctor of Philosophy**
Thesis title:	**Constraint Satisfaction Problems and friends: symmetries and algorithm design**
Committee:	**Chair:** Eugenia Ternovska Associate Professor, School of Computing Science
	Andrei A. Bulatov Supervisor Professor, School of Computing Science
	Igor Shinkar Committee Member Assistant Professor, School of Computing Science
	Valentine Kabanets Examiner Professor, School of Computing Science
	Monaldo Mastrolilli External Examiner Professor, Algorithms and Complexity Group IDSIA-Istituto Dalle Molle di Studi sull'Intelligenza Artificiale

Abstract

The Constraint Satisfaction Problem (CSP) provides a general framework for a wide range of combinatorial problems dealing with mappings and assignments, including satisfiability, graph colorability, and systems of polynomial equations. Followed by seminal work of Feder and Vardi 1993, universal algebraic techniques have been developed and quite successfully employed in studies of CSPs from different perspectives. In this dissertation, we consider two significant generalizations of CSPs, namely the Ideal Membership Problem (IMP) and Valued CSP (VCSP), and study them through the lens of universal algebra.

IMP is a fundamental algorithmic problem in which we are given a real polynomial f and an ideal I and the question is to decide whether f belongs to the ideal I. We consider a systematic study of IMPs arising from CSPs where the type of constraints is limited to relations from a constraint language. We show that many CSP techniques can be translated to IMPs thus allowing us to significantly improve the methods of studying the complexity of the IMP. We also develop universal algebraic techniques for the IMP that have been so useful in the study of the CSP. This allows us to prove a general necessary condition for the tractability of the IMP, and several sufficient ones. We furthermore introduce a variant of the IMP and study its complexity. We prove several algorithmic consequences of it such as a unifying framework to design polynomial-time algorithm to construct Gröbner Bases for many combinatorial problems. Finally, we study applications of our results in automatability of Sum-of-Squares (SoS) proofs and construction of Theta Bodies of combinatorial problems.

We then turn our attention to the most general optimization variant of the CSP namely, Valued Constraint Satisfaction Problem (VCSP), which deals with both feasibility and optimization. We consider the Minimum Cost H-coloring problem which is an important type of VCSP and is a natural optimization version of the classical H-coloring problem. We give a complete classification of graphs, in terms of their polymorphisms, for which the Minimum Cost H-coloring is approximable within a constant factor, and present several positive results regarding digraphs. From a more practical point of view, VCSPs are at the core of many machine learning and data mining tasks and in numerous number of such applications the underlying VCSPs satisfy the submodularity property. We study central topics in machine learning, namely differential privacy and sparsification, in the context of submodularity, and present several algorithmic results.

Keywords: (Valued) Constraint Satisfaction Problems; Ideal Membership Problems; Polymorphisms; Gröbner Basis; Approximation Algorithms; Submodularity; Sparsification; Differential Privacy

Dedication

To Mohammadesmaeil and Zohreh, my parents.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my advisor Andrei A. Bulatov for his guidance, caring, engagement, patience, and support. Andrei and I spent countless hours meeting for the past few years on topics ranging from highly technical discussions on our joint projects to interesting research directions, and then to broader career and life advice. Andrei's utter commitment to my professional and personal development and countless hours of devotion to our work on one hand, and the freedom he gave me in my research on the other impacted me significantly. I am grateful for all his invaluable contributions of time and insight to make my PhD research productive and exciting. Thank you, Andrei!

I am immensely grateful to Yuichi Yoshida and Arash Rafiey for additional mentoring over the last few years. Working with Yuichi and learning from him had a great impact on me. I had a lot of fun throughout our collaborations and during my internship at NII. Arash has always been an endless source of support, and encouragement since the early days of my academic life at SFU.

I want to further express my gratitude to Monaldo Mastrolilli, Valentine Kabanets, Igor Shinkar, and Eugenia Ternovska for devoting their time and serving on my thesis committee.

I feel deeply indebted to the selfless love and endless support of my lovely parents, sisters, and my brother for their consistent encouragement and understanding throughout all my life. I am thankful to my friends, my second family, at SFU and in Vancouver who certainly made their marks on my life and made my studies more pleasant and enjoyable.

Last but not the least, my heartfelt thanks go to my partner, my best friend and my wife Nazanin for being the better half of me, for her unbelievable amount of love and support, for always encouraging me, for tolerating an always-busy, always-in-a-deadline partner, and for many fruitful discussions on my research. Thank you, Nazanin!

I would like to acknowledge the administrative and technical staff in the School of Computing Science for their help over the past few years. I also wish to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for their financial support in the course of this research.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Computational problems from many different areas involve finding an assignment of values to a set of variables, where that assignment must satisfy some specified feasibility conditions and perhaps optimize some specified objective function. This includes classical problems such as graph coloring problems and integer programming, to name a few . In this thesis we focus on a generic framework for such problems that captures their general form. Bringing all such problems into a common framework draws attention to common aspects that they all share, and allows a very general algebraic approach for analysing their complexity to be developed. The primary motivation for this line of research is to understand the general picture of complexity within this general framework, rather than to develop specialized techniques for specific applications.

This thesis revolves around the Constraint Satisfaction Problems (CSPs). In a CSP we are given a set of variables and a collection of constraints, and we have to decide whether the variables can simultaneously be assigned values so that all the constraints are satisfied. This provides a general framework for a wide range of combinatorial problems. CSPs are at the core of many combinatorial optimization problems arising in machine learning, graph theory, economics, game theory, to name a few. For instance, many fundamental problems such as Minimum/Maximum Cut, Minimum Vertex Cover, Maximum Clique, and etc, are examples of various optimization versions of CSPs.

As a consequence of these applications, CSPs and their (optimization) variants have been the subject of extensive body of research. In particular, following the seminal works of Schaefer 1978 [196] and Feder and Vardi 1993 [78], universal algebraic techniques have been developed and quite successfully employed in studies of CSPs from different perspectives, such as their approximability and complexity classifications [18]. The heart of the universal algebraic technique is to study the high level symmetry of the solution sets. As Jeavons et al. [121] discovered in the mid 90s, symmetries or lack thereof of combinatorial structures in many cases determine the complexity of the corresponding computational problems. This is a very intuitive and natural property with an elegant theory evolved around it, and over the course of 20 years has been instrumental in resolving a number of long standing

open problems, most important of which is the CSP Dichotomy Conjecture by Feder and Vardi [78] which was recently confirmed by Bulatov and Zhuk in [35, 222]. The main focus of this thesis is to investigate and advance these techniques and theories for fundamental problems in mathematics and theoretical computer science, namely *polynomial ideal membership problem*, *graph coloring problems*, and central topics in machine learning, namely *differential privacy* and *sparsification*.

1.1 Constraint Satisfaction Problems

Let D be a finite set, it will often be referred to as a domain. An *n-ary relation* on D is a set of n-tuples of elements from D; we use $\mathbf{R}_D$ to denote the set of all finitary relations on D. A *constraint language* is a subset of $\mathbf{R}_D$, and may be finite or infinite. Note that if we order (or just name) relations in a constraint language Γ with domain D, then Γ can be viewed as a *relational structure* $(D; R_1, R_2, \dots)$, or equivalently as a relational database, with universe D.

A *constraint* over a constraint language $\Gamma \subseteq \mathbf{R}_D$ is a pair $\langle \mathbf{s}, R \rangle$ with $\mathbf{s} = (x_1, \dots, x_k)$ a list of variables of length k (not necessarily distinct), called the *constraint scope*, and R a k-ary relation on D, belonging to Γ, called the *constraint relation*. Another common way to denote a constraint $\langle \mathbf{s}, R \rangle$ is by $R(\mathbf{s})$, that is, to treat R as a predicate, and we will use both notations interchangeably. A constraint is satisfied by a mapping $\varphi : \{x_1, \dots, x_k\} \to D$ if $(\varphi(x_1), \dots, \varphi(x_k)) \in R$.

Definition 1.1.1 (Constraint Satisfaction Problem)**.** *The constraint satisfaction problem over a constraint language $\Gamma \subseteq \mathbf{R}_D$, denoted CSP($\Gamma$), is defined to be the decision problem with instance $\mathcal{P} = (X, D, C)$, where X is a finite set of variables, D is the domain, and C is a set of constraints over Γ with variables from X. The goal is to decide whether or not there exists a solution, i.e. a mapping $\varphi : X \to D$ satisfying all of the constraints. We will use $\mathsf{Sol}(\mathcal{P})$ to denote the (possibly empty) set of solutions of $\mathcal{P}$.*

Throughout this thesis we work with CSPs arising from a fixed constraint language e.g., constraint languages over a fixed domain and with relations of fixed arities. The CSP over a fixed language can also be formulated as the *homomorphism problem* between relational structures with a fixed target structure [65, 78]. Assume that we have two relational structures $\Delta = (E; S_1, S_2, \dots)$ and $\Gamma = (D; R_1, R_2, \dots)$ which are similar, i.e. they have the same number of relations and the corresponding relations have the same arity. A homomorphism from Δ to Γ is a mapping $\varphi : E \to D$ such that, for all i, if $\mathbf{a} = (a_1, a_2, \dots) \in S_i$ then $\varphi(\mathbf{a}) = (\varphi(a_1), \varphi(a_2), \dots) \in R_i$. Then CSP($\Gamma$) is equivalent to the problem of deciding whether a given relational structure Δ similar to Γ has a homomorphism to Γ.

Example 1.1.2. In the k-Coloring problem we need to decide the existence of a proper k-coloring of a given graph G. It can be stated as a CSP by treating the vertices of G

as variables that need to be assigned one of the k colors, and edges of G as constraints requiring that if $uv \in E(G)$ then the values of u, v satisfy the predicate $\neq_k (u, v)$, which is the disequality relation on the set of colors. Note that k-Coloring can also be naturally represented in the homomorphic form: Any proper k-coloring of G is a homomorphism from G to the complete graph on k vertices. Accordingly, one can generalize k-Coloring to H-Coloring, where H is a fixed graph. The goal in this problem is to decide the existence of a homomorphism from a given graph G to H.

Example 1.1.3. A constraint language for the 3-SAT problem is as follows

$$\Gamma_{\text{3-SAT}} = \{R_{ijk} \mid i, j, k \in \{0, 1\}\}, \quad \text{where} \quad R_{ijk} = \{0, 1\}^3 \setminus \{(i, j, k)\}.$$

For instance, the formula,

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_4 \vee x_5 \vee x_1) \wedge (\neg x_1 \vee \neg x_4 \vee \neg x_3)$$

corresponds to the following instance of $\mathrm{CSP}(\Gamma_{\text{3-SAT}})$

$$R_{010}(x_1, x_2, x_3), R_{100}(x_4, x_5, x_1), R_{111}(x_1, x_4, x_3).$$

It is known [35, 222] that for any constraint language Γ (finite or infinite) on a finite set the problem $\mathrm{CSP}(\Gamma)$ is either solvable in polynomial time or is **NP**-complete.

1.2 Ideal Membership Problem and CSPs

The Ideal Membership Problem. The study of polynomial ideals and algorithmic problems related to them goes back to David Hilbert [114]. In spite of such a heritage, methods developed in this area till these days keep finding a wide range of applications in mathematics and computer science. In this thesis we consider the Ideal Membership Problem (IMP for short), in which the goal is to decide whether a given polynomial belongs to a given ideal. It underlies such proof systems as Nullstellensatz, Polynomial Calculus, and Ideal Proof System [92].

To introduce the problem more formally, let $\mathbb{F}$ be a field and $\mathbb{F}[x_1, x_2, \ldots, x_n]$ denote the ring of polynomials over $\mathbb{F}$ with indeterminates $x_1, \ldots, x_n$. In this thesis $\mathbb{F}$ is always the field of real or complex numbers. A set of polynomials $\mathtt{I} \subseteq \mathbb{F}[x_1, x_2, \ldots, x_n]$ is said to be an *ideal* if it is closed under addition and multiplications by elements from $\mathbb{F}[x_1, x_2, \ldots, x_n]$. By the Hilbert Basis Theorem every ideal $\mathtt{I}$ has a finite generating set [113], that is, there exists $P = \{f_1, f_2, \ldots, f_r\} \subseteq \mathbb{F}[x_1, x_2, \ldots, x_n]$ such that for every $f_0 \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ the polynomial f_0 belongs to $\mathtt{I}$ if and only if there exists a *proof*, that is, polynomials $h_1, \ldots, h_r$ such that the identity $f_0 = h_1 f_1 + \cdots + h_r f_r$ holds. Such proofs will also be referred to as *ideal membership* proofs. We then write $\mathtt{I} = \langle P \rangle$. The Hilbert Basis Theorem allows one

to state the IMP as follows: given polynomials $f_0, f_1, \ldots, f_r$ decide whether there exists a proof that $f_0 \in \langle f_1, \ldots, f_r \rangle$.

In many cases combinatorial or optimization problems can be encoded as collections of polynomials, and the problem is then reduced to proving or refuting that some polynomial vanishes at specified points or is nonnegative at those points. Polynomial proof systems can then be applied to find a proof or a refutation of these facts. Polynomial Calculus, Nullstellensatz, and the Ideal Proof System are some of the standard techniques to check for zeroes of a polynomial, and Sum-of-Squares (SOS) allows to prove or refute the nonnegativity of a polynomial. We may be interested in the length or degree of a proof in one of those systems. Sometimes such proofs can also be efficiently found — such proof systems are referred to as *automatable* — and in those cases we are also concerned with the complexity of finding a proof.

The general IMP is a difficult problem and it is not even obvious whether or not it is decidable. The decidability was established in [111, 194, 197]. Then Mayr and Meyer [164] were the first to study the complexity of the IMP. They proved an exponential space lower bound for the membership problem for ideals generated by polynomials with integer and rational coefficients. Mayer [163] went on establishing an exponential space upper bound for the IMP for ideals over $\mathbb{Q}$, thus proving that such IMPs are **EXPSPACE**-complete. The source of hardness here is that a proof that $f_0 \in \langle P \rangle$ may require polynomials of exponential degree. In the cases when the degree of a proof has a linear bound in the degree of f_0, the IMP can be solved more efficiently. (There is also the issue of exponentially long coefficients that we will mention later.)

Combinatorial ideals. By Hilbert's Nullstellensatz, polynomial ideals can often be characterized by the set of common zeroes of all the polynomials in the ideal. Such sets are known as *affine varieties* and provide a link between ideals and combinatorial problems, where an affine variety corresponds to the set of feasible solutions of a problem. Combinatorial problems give rise to a fairly narrow class of ideals known as *combinatorial* ideals. The corresponding varieties are finite, and therefore the ideals itself are zero-dimensional and radical. The former implies that the IMP can be decided in single-exponential time [62], while the latter will be important later for IMP algorithms. Indeed, if the IMP is restricted to radical ideals, it is equivalent to (negation of) the question: given $f_0, f_1, \ldots, f_r$ does there exists a zero of $f_1, \ldots, f_r$ that is not a zero of f_0.

To illustrate the connection we consider the following simple example. We claim that the graph in Figure 1.1 is 2-colorable if and only if polynomials

$$x(1-x), y(1-y), z(1-z), x+y-1, x+z-1, y+z-1$$

have a common zero. Indeed, denoting the two possible colors 0 and 1, the first three polynomials guarantee that the only zeroes this collection of polynomials can have are such that $x, y, z \in \{0, 1\}$. Then the last three polynomials make sure that in every common zero the values of x, y, z are pairwise different, and so correspond to a proper coloring of the graph. Of course, the graph in the picture is not 2-colorable, and by the Weak Nullstellensatz this is so if and only if the constant polynomial 1 belongs to the ideal generated by the polynomials above. A proof of that can be easily found

$$1 = (-4)\,[x(x-1)] + (2x-1)\,([x+y-1] - [y+z-1] + [x+z-1])\,.$$

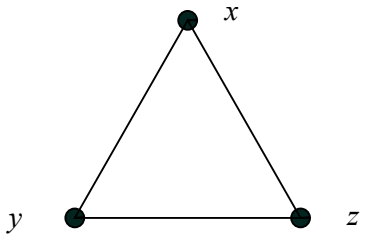


Figure 1.1: Graph 2-colorability

The special case of the IMP with $f_0 = 1$ has been studied for combinatorial problems in the context of lower bounds on Polynomial Calculus, Nullstellensatz proofs, and the Ideal Proof System, see e.g. [20, 46, 91, 92]. A broader approach of using polynomials to represent finite-domain constraints has been explored in [53, 122]. Clegg et al., [53], discuss a propositional proof system based on a bounded degree version of Buchberger's algorithm [30] for finding proofs of unsatisfiability. Jefferson et al., [122] use a modified form of Buchberger's algorithm that can be used to achieve the same benefits as the local-consistency algorithms which are widely used in constraint processing.

Complexity of the IMP and its applications in other proof systems. Whenever the degree of a proof $h_1, \ldots, h_r$ is bounded, that is, the degree of each h_i is bounded by a constant, there is an LP or SDP program of polynomial size whose solutions are the coefficients of the proof. If in addition the solution of the LP or SDP program can be represented by a polynomial number of bits (thus having low *bit complexity*), a proof can be efficiently found. This property also applies to SOS and provides one of the most powerful algorithmic methods for optimization problems.

It was recently observed by O'Donnell [178] that low degree of proofs does not necessarily implies its low bit complexity. More precisely, he presented a collection of polynomials and a polynomial such that there are low degree proofs of nonnegativity for these polynomials, that is, there also exists a polynomial size SDP whose solutions represent a SOS proof of that. However, the size of those solutions are always exponential (or the proof has high bit

complexity), and therefore the Ellipsoid method will take exponential time to find them. It therefore is possible that every low degree SOS proof has high bit complexity. Raghavendra and Weitz [192] also demonstrated an example showing that this is the case even if all the constraints in the instance are Boolean, that is, on a 2-element set.

The examples of O'Donnell and Raghavendra-Weitz indicate that it is important to identify conditions under which low degree proofs (Nullstellensatz, Polynomial Calculus, or SOS) exist and also have low bit complexity. Raghavendra and Weitz [192] suggested some sufficient conditions of this kind for SOS proofs that are satisfied for a number of well studied problems such as Matching, TSP, and others. More precisely, they formulate three conditions that a polynomial system ought to satisfy to yield for a low bit complexity SOS proof. Two of these conditions hold for the majority of combinatorial ideals, while the third, the low degree of Nullstellensatz, is the only nontrivial one. Noting that Nullstellensatz proofs are basically witnesses of ideal membership, the IMP is at the core of all the three proof systems.

IMP and CSP. Following [122, 212, 161] every CSP can be associated with a polynomial ideal. Let CSP $\mathcal{P}$ be given on variables $x_1, \ldots, x_n$ that can take values from a set $D = \{0, \ldots, t-1\}$. The ideal $\mathcal{I}(\mathcal{P})$ of $\mathbb{F}[x_1, \ldots, x_n]$ whose corresponding variety equals the set of solutions of $\mathcal{P}$ is constructed as follows. First, for every x_i the ideal $\mathcal{I}(\mathcal{P})$ contains a *domain* polynomial $f_D(x_i)$ whose zeroes are precisely the elements of D. Then for every constraint $R(x_{i_1}, \ldots, x_{i_k})$, where R is a predicate on D, the ideal $\mathcal{I}(\mathcal{P})$ contains a polynomial $f_R(x_{i_1}, \ldots, x_{i_k})$ that interpolates R, that is, for $(x_{i_1}, \ldots, x_{i_k}) \in D^k$ it holds $f_R(x_{i_1}, \ldots, x_{i_k}) = 0$ if and only if $R(x_{i_1}, \ldots, x_{i_k})$ is true. Note that for a fixed constraint language of fixed arity the polynomials f_R have constant degree. This model generalizes a number of constructions used in the literature to apply Nullstellensatz or SOS proof systems to combinatorial problems, see, e.g., [20, 46, 91, 192].

The construction above also provides useful connections between the IMP and the CSP. For instance, the CSP $\mathcal{P}$ is unsatisfiable if and only if the variety associated with $\mathcal{I}(\mathcal{P})$ is empty, or equivalently, if and only if $1 \in \mathcal{I}(\mathcal{P})$ (1 here denotes the polynomial of degree 0). In this sense it is related to the standard decision version of the CSP. However, since $\mathcal{I}(\mathcal{P})$ is radical for any instance $\mathcal{P}$, the IMP reduces to verifying whether every point in the variety of $\mathcal{I}(\mathcal{P})$ is a zero of f_0. Thus, it is probably closer to the CSP Containment problem (given two CSP instances over the same variables, decide if every solution of the first one is also a solution of the second one), which has mainly been studied in the context of Database theory and Conjunctive Query Containment, see, e.g., [135].

Mastrolilli in [161] initiated a study of the IMP parameterized by constraint languages. Let IMP(Γ) be the IMP restricted to ideals produced by instances from CSP(Γ). As was observed above, the complement of CSP(Γ) reduces to IMP(Γ), and so IMP(Γ) is **coNP**-complete whenever CSP(Γ) is **NP**-hard. Therefore the question posed in [161] is:

Problem 1.2.1. *For which constraint languages* Γ *is it possible to efficiently find a generating set for the ideal* $\mathtt{I}(\mathcal{P})$*,* $\mathcal{P} \in CSP(\Gamma)$*, that allows for a low bit complexity proof of ideal membership?*

Mastrolilli [161] (along with [25]) resolved this question in the case when Γ is a Boolean language, that is, over the set $\{0, 1\}$. He proved that in this case $\mathrm{IMP}(\Gamma)$ is polynomial time solvable, and moreover ideal membership proofs can be efficiently found, too, for any Boolean Γ for which $\mathrm{CSP}(\Gamma)$ is polynomial time solvable. However, $\mathrm{IMP}(\Gamma)$ in [161] satisfies two restrictions. First, the result is obtained under the assumption that Γ contains the constant relations that allows one to fix a value of a variable. Mastrolilli called such languages *idempotent*. We will show that this suffices to obtain a more general result. Second, in the majority of cases a bound on the degree of the input polynomial f_0 has to be introduced. The IMP where the input polynomial has degree at most d will be denoted by $\mathrm{IMP}_d, \mathrm{IMP}_d(\Gamma)$. The exact result is that for any idempotent Boolean language Γ the problem $\mathrm{IMP}_d(\Gamma)$ is polynomial time solvable for any d when $\mathrm{CSP}(\Gamma)$ is polynomial time solvable, and an ideal membership proof can be efficiently found, otherwise there is a constant $d \in \{0, 1, 2\}$ such that $\mathrm{IMP}_d(\Gamma)$ is **coNP**-complete. We will reflect on the distinction between IMP and IMP_d later in the thesis, but do not go deeply into that. The case when $\mathrm{CSP}(\Gamma)$ is equivalent to solving systems of linear equations modulo 2 was also considered in [25] fixing a gap in [161]. There has been very little work done on $\mathrm{IMP}(\Gamma)$ beyond 2-element domains. The only results we are aware of are [24, 26]. In [24], authors prove $\mathrm{IMP}_d(\Gamma)$ is polynomial time when Γ is on a 3-element domain and is invariant under the so-called dual-discriminator operation. This is extended to the case where Γ is on any finite domain and is invariant under the dual-discriminator operation [26]. Being invariant under the dual-discriminator operation imposes very strong restrictions on the relations from Γ; we will discuss this case in greater details in Section 4.1.

The main tool for proving the tractability of $\mathrm{IMP}(\Gamma)$ is constructing a Gröbner Basis of the corresponding ideal. It is not hard to see that the degree of polynomials in a Gröbner Basis of an ideal of $\mathbb{F}[x_1, \ldots, x_n]$ that can occur in $\mathrm{IMP}(\Gamma)$ is only bounded by $n|D|$, where Γ is over a set D. Therefore the basis and polynomials themselves can be exponentially large in general. In fact, this is the main reason why considering $\mathrm{IMP}_d(\Gamma)$ instead makes the problem easier. For solving this problem it suffices to find a *d-truncated* Gröbner Basis, in which the degree of polynomials is bounded by d, and so such a basis always has polynomial size. Thus, the (possible) hardness of $\mathrm{IMP}_d(\Gamma)$ is due to the hardness of constructing a Gröbner Basis.

Our contributions. In this thesis we expand on [161] and [24, 25] in several ways. We consider $\mathrm{IMP}(\Gamma)$ for languages Γ over arbitrary finite set and attempt to obtain general results about such problems. However, we mainly focus on a slightly different problem than Problem 1.2.1.

$\mathrm{IMP}_d(\Gamma)$

Input: An instance $\mathcal{P}$ of $\mathrm{CSP}(\Gamma)$ and polynomial $f \in \mathbb{F}[X]$ of degree at most d,
Goal: Decide if $f \in \mathcal{I}(\mathcal{P})$.

Note that answering whether f_0 belongs to a certain ideal does not necessarily mean finding an ideal membership proof of that. However, we will argue that, firstly, in many applications this is the problem we need to solve and therefore our results apply. Secondly, in Section 5.1.2 we will show that in all the known cases if the existence of an ideal membership proof can be efficiently decided, such a proof can also be efficiently found.

In Chapter 2 we provide the necessary definitions and notion and give a fairly detailed overview of our results. Our results are based on papers [41, 42, 43] and can be separated into the following groups.

Expanding the constraint language. In Section 3.1 we study reductions between IMP's when the language Γ is enlarged in certain ways. The methods we apply are standard in the CSP research and include adding *constant* relations, *primitive-positive (pp-) definable* and *pp-interpretable* relations. We prove that in each case the IMP over the resulting constraint language is polynomial time reducible to the IMP over the original language.

Theorem 1.2.2. *Let Γ be a constraint language over D and Γ^* denote Γ with added constant relations. The problem $IMP(\Gamma^*)$ is polynomial time reducible to $IMP(\Gamma)$, and for any d the problem $IMP_d(\Gamma^*)$ is polynomial time reducible to $IMP_{d+|D|(|D|-1)}(\Gamma)$.*

Expanding a constraint language by means of pp-definitions and pp-interpretations is at the core of the so-called algebraic approach to the CSP. This approach was first applied to various proof systems in [6], although that work is mostly concerned with proof complexity rather than computational complexity. Mastrolilli [161] ventured into pp-definability without proving any reductions. In particular, our first reductions techniques from [161] for projections of ideals. It will later allow us to develop further universal algebra techniques for the IMP. The second part of the following theorem will also work towards more powerful universal algebra methods.

Theorem 1.2.3. *(1) Let Γ, Δ be constraint languages over the same set D, Δ is finite, and every relation from Δ is pp-definable in Γ. Then $IMP(\Delta)$ is polynomial time reducible to $IMP(\Gamma)$ and $IMP_d(\Delta)$ is polynomial time reducible to $IMP_d(\Gamma)$ for any d.*

(2) Let Γ, Δ be constraint languages, Δ is finite, and Δ is pp-interpretable in Γ. Then there is a constant k such that $IMP_d(\Delta)$ is polynomial time reducible to $IMP_{kd}(\Gamma)$ for any d.

In Section 3.3 we introduce a technique new to the IMP research, although it has been extensively used for the CSP. This technique is *multi-sorted* problems in which every variable has its own domain of values. This framework is standard for the CSP, and also works very well for the IMP, as long as the domain of each variable can be embedded into the field of real or complex numbers. However, many concepts such as pp-definitions, pp-interpretations, polymorphisms have to be significantly adjusted, and several existing results have to be reproved in this more general setting. However, in spite of this extra work, the multi-sorted IMP may become the standard framework in this line of research.

Polymorphisms and algebras. A *polymorphism* of a constraint language Γ over a set D is a multi-ary operation on D that can be viewed as a multi-dimensional symmetry of relations from Γ. As in the case of the CSP, our reductions imply polymorphisms of Γ is what determines the complexity of $\mathrm{IMP}(\Gamma)$. This allows us to represent IMPs through polymorphisms and classify the complexity of IMPs according to the corresponding polymorphisms.

Theorem 1.2.4. *Let Γ, Δ be constraint languages on a finite set D and Δ finite and let $\mathsf{Pol}(\Gamma), \mathsf{Pol}(\Delta)$ denote the set of all polymorphisms of Γ and Δ, respectively. If $\mathsf{Pol}(\Gamma) \subseteq \mathsf{Pol}(\Delta)$ then $IMP(\Delta)$ [$IMP_d(\Delta)$] is polynomial time reducible to $IMP(\Gamma)$ [$IMP_d(\Gamma)$].*

In Section 3.2.2, we shall open the way to the use of a further set of powerful analytical tools by making the final translation step, from sets of operations (e.g., polymorphisms) to *algebras*. We prove that the standard features of the universal algebraic approach to the CSP work for IMP as well. These include reductions for standard algebraic constructions such as *subalgebras*, *direct powers*, and *homomorphic images*. One implication of these results is a necessary condition for tractability of $\mathrm{IMP}(\Gamma)$ that follows from a similar one for the CSP.

Sufficient conditions for tractability. The method of classifying the complexity of IMPs through polymorphisms has been initiated by Mastrolilli and Bharathi [24, 25, 161], although mainly for 2-element sets and one case of finite domain. In Chapter 4, we apply this approach to obtain several sufficient conditions for tractability of the $\mathrm{IMP}(\Gamma)$: when Γ has a semilattice or the dual-discriminator polymorphism, or the affine polymorphism modulo a prime number. The latter case covers all problems $\mathrm{IMP}(\Gamma)$, in which every relation from Γ can be represented by a system of linear equations modulo a prime number. We furthermore prove tractability of $\mathrm{IMP}(\Gamma)$ when Γ has the affine operation of an Abelian group as its polymorphism; these types of constraint languages are considered one of the most important types of tractable CSPs.

Theorem 1.2.5. *Let Γ be a constraint language over a set D. Then if one of the following conditions holds, $IMP_d(\Gamma)$ is decidable in polynomial time for any d.*

1. Γ *has the* dual-discriminator *polymorphism (i.e. a ternary operation g such that $g(x,y,z) = x$ unless $y = z$, in which case $g(x,y,z) = y$);*

2. Γ *has a* semilattice *polymorphism (i.e. a binary operation f such that $f(x,x) = x$, $f(x,y) = f(y,x)$, and $f(f(x,y),z) = f(x,f(y,z))$);*

3. *$|D| = p$, p prime, and Γ has an* affine *polymorphism modulo p (i.e. a ternary operation $h(x,y,z) = x \ominus y \oplus z$, where $\oplus, \ominus$ are addition and subtraction modulo p, or, equivalently, of the field $\mathsf{GF}(p)$). In this case every CSP can be represented as a system of linear equations over $\mathsf{GF}(p)$.*

4. *D is an Abelian group and the affine operation $x - y + z$ of D is a polymorphism of Γ.*

The three polymorphisms mentioned above have played an important role in the CSP research. For one reason they completely cover the tractable cases when $|D| = 2$ and therefore the results of [161, 25], although we used some results (on semilattice polymorphisms) from [161]. Second, it has been observed that there are two main algorithmic approaches to solving the CSP. The first one is based on the local consistency of the problem. CSPs that can be solved solely by establishing some kind of local consistency are said to have *bounded width* [40, 17]. The property to have bounded width is related to a rather surprising number of other seemingly unrelated properties, see e.g. [6, 208]. CSP algorithms of the second type are based on the *few subalgebras* property and achieve results similar to those of Gaussian elimination: they construct a concise representation of the set of all solutions of a CSP [36, 118]. Problems CSP(Γ) where Γ has an affine polymorphism were pivotal in the development of few subpowers algorithms, and, in a sense, constitute the main nontrivial case of them. Among our results on the IMP, IMP(Γ) for Γ invariant under a semilattice or dual-discriminator polymorphism (a special kind of majority polymorphisms) belong to the local consistency part of the algorithmic spectrum, while those for Γ invariant with respect to an affine operation are on the 'few subalgebras' part of it. It is therefore important to observe differences in approaches to the IMP in these two cases.

The IMP with indeterminate coefficients. Chapter 5 consists of a number of applications of the techniques developed in Chapters 3 and 4. The key to those applications is an extension of the IMP defined as follows. Given an ideal $\mathtt{I} \subseteq \mathbb{F}[x_1, \dots, x_n]$ and a vector of ℓ polynomials $M = (g_1, \dots, g_\ell)$, the χIMP asks if there exist coefficients $\mathbf{c} = (c_1, \dots, c_\ell) \in \mathbb{F}^\ell$ such that $\mathbf{c}M = \sum_{i=1}^{\ell} c_i g_i$ belongs to the ideal $\mathtt{I}$.

As with the regular IMP, χIMP can be parametrized by specifying a constraint language Γ, in which case the resulting problem χIMP(Γ) (or $\chi\mathrm{IMP}_d(\Gamma)$ if the degree of input polynomials is bounded) only allows ideal produced by instances of CSP(Γ).

We prove that $\chi\text{IMP}_d(\Gamma)$ can be solved in polynomial time when a (d-truncated) Gröbner Basis can be efficiently generated, and also admits the same reductions as the IMP. This theorem allows us to show that for every Γ for which $\text{IMP}_d(\Gamma)$ is polynomial time solvable, so is $\chi\text{IMP}_d(\Gamma)$. This includes constraint languages invariant under dual-discriminator, semilattice, and affine polymorphisms.

In Section 5.1.2 we use χIMP along with the factor ring $\mathbb{F}[X]/\mathtt{I}$ modulo an ideal $\mathtt{I}$ to generate a basis for the factor ring consisting of monomials of degree at most d, and then use it to construct a d-truncated Gröbner Basis for $\mathtt{I}$.

Theorem 1.2.6. *Let $\mathcal{H}$ be a class of ideals for which χIMP_d is polynomial time solvable. Then there exists a polynomial time algorithm that constructs a degree d Gröbner Basis (with respect to a grlex) of an ideal $\mathtt{I} \in \mathcal{H}$, $\mathtt{I} \subseteq \mathbb{F}[x_1, \ldots, x_n]$, in time $O(n^d)$.*

This makes it possible to construct d-truncated Gröbner Bases in all cases $\text{IMP}(\Gamma)$ is known to be polynomial time. Thus, it basically eliminates the gap between deciding the existence of an ideal membership proof and finding such a proof.

Applications to SOS and Theta bodies. The problem χIMP and the results mentioned above can also be used to study some semialgebraic proof systems such as the SOS. It also finds applications in the study of theta-bodies. These and other applications are presented in Sections 5.3 and 5.4.

1.3 Valued Constraint Satisfaction Problems

There are several natural optimization versions of the CSP. Let Γ be a constraint language over D, then one optimization version of $\text{CSP}(\Gamma)$ is, for every instance $\mathcal{P} = (X, D, C)$ of $\text{CSP}(\Gamma)$, to find a mapping $\varphi : X \to D$ that maximizes (minimizes) the number of satisfied (unsatisfied) constraints. This problem is known under the name of Max CSP (Min CSP). For example, the most basic Boolean Max 2-CSP problem is Max Cut where Γ is the graph containing only one edge. This line of research has received a lot of attention in the literature and there are very strong results concerning various aspects of approximability of Max 2-CSP and Min 2-CSP [7, 86, 100, 131, 148]. See [159] for a recent survey on this and approximation of Max k-CSP and Min k-CSP. Another natural optimization version of the $\text{CSP}(\Gamma)$ is where not only we are interested in the existence of a satisfying assignment, but want to find the "best satisfying assignmen". Examples of such setting include Minimum Cost Homomorphism [103, 187], (Weighted) Min Ones [4, 60, 129], Min Sol [124, 210], a large class of bounded integer linear programs, retraction problems [77], Minimum Sum Coloring [13, 85, 141], and various optimum cost chromatic partition problems [99, 120, 123, 140].

The most general optimization variant of the CSP is the Valued Constraint Satisfaction Problem, or VCSP for short, which deals with both feasibility and optimization. A valued

constraint language Γ is a set of functions on a fixed domain and a VCSP instance over Γ is given by a sum of functions from Γ with the goal to minimize the sum.

We define the VCSP formally as follows. As before, let D be a fixed finite set and $\overline{\mathbb{Q}} = \mathbb{Q} \cup \{\infty\}$ denote the set of rational numbers with (positive) infinity. An r-ary weighted relation over D is a mapping $\sigma : D^r \to \overline{\mathbb{Q}}$. We write $ar(\sigma) = r$ for the arity of σ. A *valued constraint language*, or just a constraint language, over D is a set of weighted relations over D. We denote by VCSP(Γ) the class of all VCSP instances in which the weighted relations are all contained in Γ.

A valued constraint over a valued constraint language Γ and variable X is an expression of the form $\sigma(\mathbf{x})$ where σ is a weighted relation in Γ and $\mathbf{x} \in X^{ar(\sigma)}$ is a list of variables, called the scope of the constraint. A valued constraint is satisfied by a mapping $\varphi : X \to D$ if $\sigma(\varphi(\mathbf{x})) < \infty$.

Definition 1.3.1 (Valued Constraint Satisfaction Problem)**.** *An instance $\mathcal{P}$ of the valued constraint satisfaction problem (VCSP) over a valued constraint language Γ is specified by a finite set $X = \{x_1, \ldots, x_n\}$ of variables, a finite set D, and an objective function $\Sigma_{\mathcal{P}}$ expressed as follows:*

$$\Sigma_{\mathcal{P}}(x_1, \ldots, x_n) = \sum_{i=1}^{q} \sigma_i(\mathbf{x}_i), \tag{1.1}$$

where each $\sigma_i(\mathbf{x}_i)$, $1 \leq i \leq q$, is a valued constraint from Γ. Each constraint may appear multiple times in $\mathcal{P}$. An assignment to $\mathcal{P}$ is a map $\varphi : X \to D$. The goal is to find an assignment that minimizes the objective function.

Note that CSPs are a special case of VCSPs with $\{0, \infty\}$-valued relations with the goal to determine the existence of a satisfying assignment. Similar to the CSPs, a valued constraint language Γ is called tractable if VCSP(Γ') can be solved (to optimality) in polynomial time for every finite subset $\Gamma' \subseteq \Gamma$, and Γ is called **NP**-hard if VCSP(Γ') is **NP**-hard for some finite $\Gamma' \subseteq \Gamma$.

Example 1.3.2 (Digraph Homomorphism)**.** Given two digraphs $G = (V(G), E(G))$ and $H = (V(H), E(H))$, a mapping $f : V(G) \to V(H)$ is a homomorphism from G to H if f preserves edges, that is, $(u, v) \in E(G)$ implies $(f(u), f(v)) \in E(H)$. The problem whether an input digraph G admits a homomorphism to a fixed digraph H is also known as the H-Coloring problem and has been actively studied in graph theory [105], see also [144]. For any digraph H, let Γ_H be the language that contains just the single binary cost function $\sigma_H : V(H)^2 \to \overline{\mathbb{Q}}$ defined by

$$\sigma_H(x, y) = \begin{cases} 0 & \text{if } (u, v) \in E(H), \\ \infty & \text{otherwise.} \end{cases}$$

If we add all *unary crisp*[1] functions to Γ_H then the resulting VCSP is known as List H-Coloring or List Homomorphism [105].

Fractional polymorphisms. Analogous to constraint languages, multi-dimensional symmetry of valued constraint languages can be associated with a set of operations, known as the polymorphisms. The complexity of *exact minimization* of VCSPs is well understood [136, 207], and they are formulated in terms of the existences of particular types of *fractional* polymorphisms.

A function $\psi : D^k \to D$ is called a k-ary operation on D. For a weighted relation $\sigma : D^r \to \overline{\mathbb{Q}}$, we denote by $\mathsf{Feas}(\sigma) = \{\mathbf{x} \in D^m \mid \sigma(\mathbf{x}) \text{ is finite}\}$ the feasibility relation of σ. We will view $\mathsf{Feas}(\sigma)$ both as a relation and as a $\{0, \infty\}$-valued cost function.

Definition 1.3.3 (Polymorphism)**.** *Let $\sigma : D^r \to \overline{\mathbb{Q}}$ be a weighted relation. We say that a k-ary operation $\psi : D^k \to D$ is a polymorphism of σ if, for any $\mathbf{a}^1, \ldots, \mathbf{a}^k \in \mathsf{Feas}(\sigma)$ we have that $\psi(\mathbf{a}^1, \ldots, \mathbf{a}^k) \in \mathsf{Feas}(\sigma)$. Here, by $\psi(\mathbf{a}^1, \ldots, \mathbf{a}^k)$ we understand the component-wise action of ψ, that is, if $\mathbf{a}^i = (a^i_1, \ldots, a^i_r)$ then*

$$\psi(\mathbf{a}^1, \ldots, \mathbf{a}^k) = (\psi(a^1_1, \ldots, a^k_1), \ldots, \psi(a^1_r, \ldots, a^k_r))$$

For a valued constraint language Γ, we denote by $\mathsf{Pol}(\Gamma)$ the set of all operations which are polymorphisms of all $\sigma \in \Gamma$. We write $\mathsf{Pol}(\sigma)$ for $\mathsf{Pol}(\{\sigma\})$.

A probability distribution ω over the set of k-ary operations on D is called an k-ary fractional operation. We define $\mathsf{supp}(\omega)$ to be the set of operations assigned positive probability by ω. We call ω a fractional polymorphism of σ if $\mathsf{supp}(\omega) \subseteq \mathsf{Pol}(\sigma)$ and for any $\mathbf{a}^1, \ldots, \mathbf{a}^k \in \mathsf{Feas}(\sigma)$, we have

$$\mathbb{E}_{\psi \sim \omega}[\sigma(\psi(\mathbf{a}^1, \ldots, \mathbf{a}^k))] \leq \mathsf{avg}\{\sigma(\mathbf{a}^1), \ldots, \sigma(\mathbf{a}^k)\}.$$

In this thesis, we make advancement in applications of (fractional) polymorphisms in different algorithmic aspects of VCSPs. We first turn our attention to approximability of VCSPs where our ultimate goal is to completely characterize valued constraint languages for which efficient approximation algorithms are feasible. Secondly, we consider central topics in the intersection of machine learning and optimization. VCSPs are at the core many machine learning and data mining tasks and an intriguing question is to what extend are the effects of universal algebraic techniques and polymorphisms in designing algorithms in such areas. More specifically, our focus is on sparsification and differential privacy. As we will discuss, these two have been the subject of intense studies both in machine learning and theoretical computer science communities. In this thesis, we focus on valued constraint

[1] Crisp functions take only the values 0 or ∞.

languages that admit the submodularity property. In such case, we first focus on designing efficient sparsification algorithms for submodular functions and study applications and consequences of our results. In the last chapter of this thesis, we consider designing optimization algorithms that preserve differential privacy and discuss several applications of our algorithms. We hope our studies here help to better understand and advance the notion of polymorphisms and universal algebraic concepts in algorithm design both in theory and practice, and points towards uniform approaches for general VCSPs.

1.3.1 Approximation of VCSPs and Minimum Cost H-coloring

For a minimization problem, an *α-approximation* algorithm is a (randomized) polynomial time algorithm that finds an approximate solution of cost at most α times the minimum cost. A constant ratio approximation algorithm is an α-approximation algorithm for some constant α. For *finite* VCSPs where weighted relations only takes finite values, Raghavendra [190] showed how to use the basic SDP relaxation to obtain a constant factor approximation. Moreover, he proved that the approximation ratio cannot be improved under the Unique Game Conjecture (UGC). This constant is not explicit, but there is an algorithm that can compute it with any given accuracy in doubly exponential time with respect to the domain size, arity, and the accuracy factor [191]. In another line of research, the power of so-called *basic linear program (BLP)* concerning constant factor approximation of finite VCSPs has been recently studied in [61, 74]. However, the approximability of VCSPs for constraint languages that are not finite-valued remains poorly understood, and [103, 124, 162] are the only results on approximation of VCSP for languages that have cost functions that can take infinite values. The results of [103, 162] deal with the Minimum Cost Homomorphism problem also known as Minimum Cost H-coloring problem to a digraph which is a generalization and optimization version of Digraph Homomorphism problem in Example 1.3.2.

Minimum Cost Homomorphism. For a digraph G, let $V(G)$ denote the vertex set of G, and let $A(G)$ denote the arcs of G. We denote the number of vertices of G by $|G|$. Instead of $(u, v) \in A(G)$, we use the shorthand $uv \in A(G)$ or simply $uv \in G$. A graph G is a symmetric digraph, that is, $xy \in A(G)$ if and only if $yx \in A(G)$. An edge is just a symmetric arc.

Recall that a homomorphism of a digraph G to a digraph H (a.k.a H-Coloring) is a mapping $f : V(G) \to V(H)$ such that for each arc xy of G, $f(x)f(y)$ is an arc of H. We say mapping f does not satisfy arc xy, if $f(x)f(y)$ is not an arc of H. The homomorphism problem for a fixed target digraph H, HOM(H), takes a digraph G as input and asks whether there is a homomorphism from G to H. Therefore, by fixing the digraph H we obtain a class of problems, one for each digraph H. For example, HOM(H), when H is an edge, is exactly the problem of determining whether the input graph G is bipartite (i.e., the 2-Coloring problem). Similarly, if H is the complete graph on 3 vertices K_3, then HOM(H)

is exactly the classical 3-Coloring problem. More generally, if H is a clique on k vertices, then HOM(H) is the k-Coloring problem. Of course, HOM(H) is a special case of CSP in which the constraint language is a binary relation. A celebrated result due to Hell and Nesetril [104], states that, for graph H, HOM(H) is in **P** if H is bipartite or contains a looped vertex, and that it is **NP**-complete for all other graphs H. See [32] for an algebraic proof of the same result.

The optimization version of HOM(H) where the goal is to find a mapping $f : V(G) \to V(H)$ that maximizes (minimizes) the number of satisfied (unsatisfied) arcs in G is known as Max 2-CSP (Min 2-Csp). For instance, the most basic Boolean Max 2-CSP problem is Max Cut where the target graph H is an edge. This line of research has attracted a great deal of attention, see [159] and references therein. Our focus is another optimization variant of the HOM(H) problem, i.e., we are not only interested in the existence of a homomorphism, but want to find the "best homomorphism". The *minimum cost homomorphism problem* to H, denoted by MinHOM(H), for a given input digraph G, and a cost function $c(x, i), x \in V(G), i \in V(H)$, seeks a homomorphism f of G to H that minimizes the total cost $\sum_{x \in V(G)} c(x, f(x))$. The cost function c can take non-negative rational values and positive infinity, that is $c : V(G) \times V(H) \to \mathbb{Q}_{\geq 0} \cup \{+\infty\}$. The MinHOM was introduced in [97], where it was motivated by a real-world problem in defence logistics. The MinHOM problem offers a natural and practical way to model and generalizes many optimization problems.

Example 1.3.4 ((Weighted) Minimum Vertex Cover)**.** This problem can be formulated as MinHOM(H) where $V(H) = \{0, 1\}, E(H) = \{11, 01\}$ and $c(u, 0) = 0$, $c(u, 1) > 0$ for every $u \in V(G)$. Note that G and H are graphs in this example.

Example 1.3.5 (Chromatic Sum)**.** In this problem, we are given a graph G, and the objective is to find a proper coloring of G with colors $\{1, \ldots, k\}$ with *minimum color sum*. This can be seen as MinHOM(H) where H is a clique of size k with $V(H) = \{1, \ldots, k\}$ and the cost function is defined as $c(u, i) = i$. The Chromatic Sum problem appears in many applications such as resource allocation problems [13] and scheduling problems [160].

Example 1.3.6 (Min Cut in graphs)**.** Let G be a graph. Each edge e of G has a weight $w(e)$, and the goal is to partition the vertices of G into $P \subset V(G)$ and $V(G) \setminus P$ so that sum of weights of edges between P and $V(G) \setminus P$ is minimum. This problem can be formulated as MinHOM as follows. Add two extra vertices s, t and connect s, t to every vertex in G and set the edge weights zero. Construct G' by replacing each edge $e = uv$ of G by a path u, x_e, v. Let H be a graph with vertices α, β, γ and edge set $\{\alpha\alpha, \alpha\beta, \beta\gamma, \gamma\gamma\}$. For every vertex $u \in V(G') \setminus \{s, t\}$, set $c(u, \alpha) = 0$, $c(u, \gamma) = 0$, and $c(x_e, \beta) = w(e)$. Set $c(s, \alpha) = 0$ and $c(s, \beta) = c(s, \gamma) = |G|$. Finally, set $c(t, \alpha) = c(t, \beta) = |G|$, and $c(t, \gamma) = 0$. Now, finding a minimum cut in G is equivalent to finding a minimum cost homomorphism from graph G' to H.

Example 1.3.7 (List Homomorphism (LHOM))**.** LHOM(H), seeks, for a given input digraph D and lists $L(x) \subseteq V(H), x \in V(D)$, a homomorphism f from D to H such that $f(x) \in L(x)$ for all $x \in V(D)$. This is equivalent to MinHOM(H) with $c(u,i) = 0$ if $i \in L(u)$, otherwise $c(u,i) = +\infty$. This problem is also known as List H-Coloring and its complexity is fully understood due to series of results [14, 31, 33, 34, 76, 108].

Note that in the MinHOM problem the cost function is a part of the input. That is the corresponding valued constraint language contains infinitely many possible unary cost functions. A special case of MinHOM problem is where the cost function c is chosen from a fixed set Δ. This problem is denoted by MinHOM(H, Δ) [55, 106, 210, 211]. Interestingly, a recent work by Cohen *et al.* [55] proved that VCSPs over a fixed valued constraint language are polynomial-time equivalent to MinHOM(H, Δ) over a fixed digraph and a proper choice of Δ.

Our contributions. Most of the minimum cost homomorphism problems are **NP**-hard, therefore we investigate the approximation of MinHOM(H).

Approximating MinHOM(H)

Input: A digraph G and a vertex-mapping costs $c(x,u), x \in V(G), u \in V(H)$,
Output: A homomorphism f of G to H with the total cost of $\sum_{x \in V(G)} c(x, f(x)) \leq \alpha \cdot \mathsf{Opt}$, where α is a constant.

Here, Opt denotes the cost of a minimum cost homomorphism of G to H. Moreover, we assume size of H is constant. Recall that we approximate the cost over real homomorphisms, rather than approximating the maximum weight of satisfied constraints, as in, say, Max CSP.

Ultimately, our goal is to fully characterize digraphs for which MinHOM(H) admits a polynomial time constant factor approximation algorithm. We take important steps towards this goal by providing constant factor approximation algorithms for MinHOM(H) where H belongs to these two important cases of digraphs, namely

Theorem 1.3.8. *Let H be a digraph. MinHom(H) admits a constant factor approximation algorithm in the following cases*

1. *H is a* bi-arc *digraph i.e., it admits a* conservative semilattice *polymorphism a.k.a min-ordering,*

2. *H is a* k-arc *digraph i.e., it admits a k-min-ordering.*

Furthermore, we obtain a full characterization of graphs which admit a constant factor approximation algorithm.

Theorem 1.3.9 (Dichotomy for graphs)**.** *Let H be a graph. There exists a constant factor approximation algorithm for MinHOM(H) if H is a bi-arc graph i.e. admits a conservative majority polymorphism, otherwise, MinHOM(H) is inapproximable unless $\boldsymbol{P} = \boldsymbol{NP}$.*

Finally we conjecture a criteria for approximability of MinHOM(H) when H is a digraph. Our results are presented in Chapter 6, and are based on the paper [187].

Conjecture 1.3.10. *Let H be a digraph. MinHOM(H) admits a constant factor approximation algorithm when H is a DAT-free digraph, otherwise, MinHOM(H) is not approximable unless $\boldsymbol{P} = \boldsymbol{NP}$.*

1.3.2 Sparsification of VCSPs and submodularity

VCSPs are at the core of many machine learning and data mining tasks. In many data intensive applications, however, the number of underlying cost functions in the original function $\Sigma(x_1, \ldots, x_n) = \sum_{i=1}^{q} \sigma_i(\mathbf{x}_i)$ is so large, q is too large, that we need prohibitively large amount of time to process it and/or it does not even fit in the main memory. To overcome this issue, we study the notion of sparsification for VCSPs whose objective is to obtain an accurate approximation of the original function that is a (weighted) sum of only a few cost functions. Sparsification is an algorithmic paradigm where a dense object is replaced by a sparse one with similar "features", which often leads to significant improvements in efficiency of algorithms, including running time, space complexity, and communication.

Formally, let Γ be a valued constraint language and $\mathcal{P}$ be an instance of VCSP(Γ) with objective function

$$\Sigma_{\mathcal{P}}(x_1, \ldots, x_n) = \sum_{i=1}^{q} \sigma_i(\mathbf{x}_i).$$

For $0 < \epsilon < 1$, an ϵ-sparsification of $\mathcal{P}$ is a *re-weighted* instance $\mathcal{P}_\epsilon$

$$\Sigma_{\mathcal{P}_\epsilon}(x_1, \ldots, x_n) = \sum_{i=1}^{q} w_i \sigma_i(\mathbf{x}_i),$$

such that for any assignment $\varphi : X \to D$ we have $\mathsf{Val}(\mathcal{P}_\epsilon, \varphi) \in (1 \pm \epsilon)\mathsf{Val}(\mathcal{P}, \varphi)$. The goal here is to find a sparsifier with minimum number of constraints. Depending on the application one can define a threshold function $\tau(n, \epsilon)$ for sparsification. Then a valued constraint language Γ is called *sparsifiable* if for every instance $\mathcal{P}$ of VCSP(Γ) on n variables and for every $0 < \epsilon < 1$ there is an ϵ-sparsification for $\mathcal{P}$ with $\tau(n, \epsilon)$ many cost functions. The goal is to characterize valued constraint languages that are sparsifiable with respect to a threshold function $\tau(n, \epsilon)$. Such a characterization is known for Boolean binary VCSPs and threshold function $\tau(n, \epsilon) = O(\frac{n}{\epsilon^2})$ due to Filtser and Krauthgamer [82], this was extended to binary VCSPs over finite domain by Butti and Zivný[47]. We extend upon these results by considering decomposable submodular functions i.e., the corresponding valued constraint language admits a nice 2-ary fractional polymorphism.

Sparsification and submodularity. Submodularity of a set function is an intuitive diminishing returns property, stating that adding an element to a smaller set helps gaining more return than adding it to a larger set. Formally, a set function $f\colon 2^E \to \mathbb{R}$ is *submodular* if for any $S \subseteq T \subseteq E$ and $e \in E \setminus T$ it holds that

$$f(T \cup \{e\}) - f(T) \leq f(S \cup \{e\}) - f(S)$$

Equivalently, a set function $f\colon 2^E \to \mathbb{R}$ is called submodular if for all subsets S and T of E

$$f(S \cap T) + f(S \cup T) \leq f(S) + f(T). \tag{1.2}$$

This can be expressed in terms of fractional polymorphisms as well. If we set $D = \{0, 1\}$, then any set function f on E can be associated with a ($|E|$-ary) weighted relation σ defined on the characteristic vectors of subsets of E. The intersection and union operations on subsets correspond to the Min and Max operations on the associated characteristic vectors. Hence, f is submodular if and only if the associated cost function σ satisfies the following inequality:

$$\sigma(\mathsf{Max}(\mathbf{x}_1, \mathbf{x}_2)) + \sigma(\mathsf{Min}(\mathbf{x}_1, \mathbf{x}_2)) \leq \sigma(\mathbf{x}_1) + \sigma(\mathbf{x}_2).$$

This means that σ admits the 2-ary fractional polymorphism ω_{sub}, defined by $\omega_{sub}(\mathsf{Min}) = \omega_{sub}(\mathsf{Max}) = \frac{1}{2}$.

Submodularity is a fundamental structure that has emerged as a very beneficial property in many combinatorial optimization problems arising in machine learning, graph theory, economics, game theory, to name a few. The theory of *submodular maximization* provides a general and unified framework for various combinatorial optimization problems including the Maximum Coverage, Maximum Cut, and Facility Location problems. Furthermore, it also appears in a wide variety of applications such as viral marketing [128], information gathering [139], feature selection for classification [138], influence maximization in social networks [128], document summarization [152], and speeding up satisfiability solvers [204]. For a survey, see [137]. As a consequence of these applications and importance, a wide range of efficient approximation algorithms have been developed for maximizing submodular functions subject to different constraints [48, 175, 177, 216]. However, the need for efficient optimization methods that can be used in data-intensive tasks is wide-spread. In this thesis we address these issues by studying sparsification of *decomposable* submodular functions. A submodular function is called decomposable if it can be written as a sum of several submodular functions i.e.,

$$F(S) = \sum_{i=1}^{N} f_i(S) \qquad \forall S \subseteq E,$$

where each $f_i : 2^E \to \mathbb{R}$ is a submodular function on the ground set E.

Submodular sparsification

Input: A decomposable submodular function $F(S) = \sum_{i=1}^{N} f_i(S)$ and ϵ,
Goal: Find a $\boldsymbol{w} \in \mathbb{R}^N$ with minimum number of non-zero entries such that for $F'(S) = \sum_{i=1}^{N} \boldsymbol{w}_i f_i(S)$ we have

$$(1-\epsilon)F'(S) \leq F(S) \leq (1+\epsilon)F'(S),$$

for all subsets $S \subseteq E$.

Our contributions. Given a decomposable submodular function $F = \sum_{i=1}^{N} f_i$, we consider the following problem and present a randomized algorithm that yields a sparse representation that approximates F. We prove our algorithm yields a sparsifier of small size (***independent of*** N) with a very good approximation of F.

Theorem 1.3.11. *Let $F = \sum_{i=1}^{N} f_i$ be a decomposable submodular function. For any $\epsilon > 0$, there exists a vector $\boldsymbol{w} \in \mathbb{R}^N$ with at most $O(\frac{B \cdot n^2}{\epsilon^2})$ non-zero entries such that for the submodular function $F' = \sum_{i=1}^{N} \boldsymbol{w}_i f_i$ we have*

$$(1-\epsilon)F'(S) \leq F(S) \leq (1+\epsilon)F'(S) \quad \forall S \subseteq E.$$

Moreover, if all f_i's are monotone, then there exists a polynomial-time randomized algorithm that outputs a vector $\boldsymbol{w} \in \mathbb{R}^N$ with at most $O(\frac{B \cdot n^{2.5} \log n}{\epsilon^2})$ non-zero entries in expectation such that for the submodular function $F' = \sum_{i=1}^{N} \boldsymbol{w}_i f_i$, with high probability, we have

$$(1-\epsilon)F'(S) \leq F(S) \leq (1+\epsilon)F'(S) \quad \forall S \subseteq E.$$

We furthermore study saprsification under various constraints such as matroid constraint, and provide a randomized algorithm that obtains a sparsifier of a smaller size in comparison to the general setting. We then discuss applications of our results in speeding up optimization algorithms for submodular maximization/minimization. Finally, we empirically examine our algorithm and demonstrate that it constructs a concise sparsifier on which we can efficiently perform algorithms. These results are presented in Chapter 7, and are based on the paper [189].

1.3.3 VCSPs under differential privacy

The need for efficient optimization methods that guarantee the privacy of individuals is widespread across many applications concerning sensitive data about individuals, e.g., medical data, web search query data, salary data, social networks. Consider the problem of assigning people using a social network to one of two servers so that most pairs of friends are assigned to the same server which can be modeled as an instance of the Min Cut problem; or the

problem of opening a small number of drop-off centers for undercover agents so that each agent is able to visit some site convenient to her (each providing a list of acceptable sites) which can be modeled as an instance of the Set Cover problem. In these scenarios and in many others, the input data (friendship relations, medical history, agents locations) represent sensitive information about individuals, and it is preferable to use a private algorithm that gives somewhat suboptimal solutions to a non-private optimal algorithm.

Differential privacy is a rigorous notion of privacy that allows statistical analysis of sensitive data while providing strong privacy guarantees. Basically, differential privacy requires that computations be insensitive to changes in any particular individual's record. A dataset is a collection of records from some domain, and two datasets are *neighboring* if they differ in a single record. Simply put, the requirement for differential privacy is that the computation behaves nearly identically on two neighboring datasets.

Various combinatorial problems have been considered in the differential privacy framework and efficient algorithms have been developed for them, for instance see [94] and references therein. However, these approaches are ad hoc and studying differential privacy in a general and unifying framework for combinatorial problems such as the VCSPs is absent in the literature. We therefore consider the problem of characterizing of valued constraint languages Γ that admit efficient and accurate algorithms under differential privacy. In this thesis, we focus on valued constraint languages with submodular property and present several positive results.

Submodular optimization under differential privacy. In this thesis we consider designing a differentially private algorithm for maximizing nonnegative and *monotone* submodular functions in *low-sensitivity* regime. In this regime, informally speaking, the value of the function is not very sensitive to changes in the dataset, yet this factor must be taken into consideration. Whilst, a *cardinality* constraint is a natural one to place on a submodular maximization problem (e.g., finding a set of size k that maximizes the function), many other problems, e.g., personalized data summarization [170], require the use of more general types of constraints, i.e., *matroid* constraints. The problem of maximizing a submodular function under a matroid constraint is a classical problem [70], with many important special cases, e.g., uniform matroid (the subset selection problem, see Example 8.1.1), partition matroid (submodular welfare/partition problem). We consider the following.

Submodular maximization under differential privacy

Input: A sensitive dataset D associated to a monotone submodular function $F_D \colon 2^E \to \mathbb{R}_+$ and a matroid $\mathcal{M} = (E, \mathcal{I})$,

Goal: Find a subset $S \in \mathcal{I}$ that approximately maximizes F_D in a manner that guarantees differential privacy with respect to the input dataset D.

Furthermore, we consider a natural generalization of submodular functions, namely, k-submodular functions. k-submodular function maximization allows for a richer problem structure than submodular maximization. For instance, coupled feature selection [199], sensor placement with k kinds of measures [179], and influence maximization with k topics can be expressed as k-submodular function maximization problems.

The property of k-submodularity can be formulated in terms of polymorphisms as follows. Let $D = \{0, \dots, k\}$. Consider the following two operations on D:

$$\mathsf{Min}_0(s,t) = \begin{cases} 0 & \text{if } s \neq 0, t \neq 0, s \neq t \\ \min(s,t) & \text{otherwise.} \end{cases}$$

and

$$\mathsf{Max}_0(s,t) = \begin{cases} 0 & \text{if } s \neq 0, t \neq 0, s \neq t \\ \max(s,t) & \text{otherwise.} \end{cases}$$

where $\min(s,t)$ (respectively, $\max(s,t)$) returns the smaller (respectively, the larger) of s and t with respect to the usual order on the integers. As usual, for vectors $\mathbf{s}$ and $\mathbf{t}$ in $\{0, \dots, k\}^E$ we let $\mathsf{Min}_0(\mathbf{s}, \mathbf{t})$ (respectively, $\mathsf{Max}_0(\mathbf{s}, \mathbf{t})$) denote the vector obtained from applying Min_0 (respectively, Max_0) to $\mathbf{s}$ and $\mathbf{t}$ coordinate-wise. Using these operations we can define the general class of k-submodular function. Given a natural number $k \geq 1$ and a finite nonempty set E, a function $f : \{0, \dots, k\}^E \to \mathbb{R}_+$ is called *k-submodular* if for all $\mathbf{s}$ and $\mathbf{t}$ in $\{0, \dots, k\}^E$,

$$f(\mathsf{Min}_0(\mathbf{s}, \mathbf{t})) + f(\mathsf{Max}_0(\mathbf{s}, \mathbf{t})) \leq f(\mathbf{s}) + f(\mathbf{t}).$$

We consider the following problem.

---

k-submodular maximization under differential privacy

Input:	A sensitive dataset D associated to a monotone k-submodular function $F_D \colon (k+1)^E \to \mathbb{R}_+$ and a matroid $\mathcal{M} = (E, \mathcal{I})$,
Goal:	Find $S = (S_1, \dots, S_k)$ with $\bigcup_{i \in [k]} S_i \in \mathcal{I}$ that approximately maximizes F_D in a manner that guarantees differential privacy with respect to the input dataset D.

Our contributions. In Chapter 8, we study the problem of maximizing monotone submodular functions subject to matroid constraints in the framework of differential privacy. We provide a $(1 - \frac{1}{\mathrm{e}})$-approximation algorithm which improves upon the previous results in

terms of approximation guarantee. This is done with an almost cubic number of function evaluations in our algorithm.

Theorem 1.3.12. *Suppose F_D is monotone with sensitivity Δ and $\mathcal{M} = (E, \mathcal{I})$ is a matroid with rank $r(\mathcal{M})$. For every $\epsilon > 0$, there is an $(\epsilon r(\mathcal{M})^2)$-differentailly private algorithm that, with high probability, returns $S \in \mathcal{I}$ with quality at least $(1 - \frac{1}{\mathrm{e}})OPT - O\left(\sqrt{\epsilon} + \frac{\Delta r(\mathcal{M})|E| \ln |E|}{\epsilon^3}\right)$.*

Moreover, we study k-submodularity, a natural generalization of submodularity. We give the first $\frac{1}{2}$-approximation algorithm that preserves differential privacy for maximizing monotone k-submodular functions subject to matroid constraints. The approximation ratio is asymptotically tight and is obtained with an almost linear number of function evaluations.

Theorem 1.3.13. *Suppose $F_D : (k+1)^E \to \mathbb{R}_+$ is monotone with sensitivity Δ and $\mathcal{M} = (E, \mathcal{I})$ is a matroid with rank $r(\mathcal{M})$. For any $\epsilon > 0$, there is an $O(\epsilon r(\mathcal{M}))$-differentially private algorithm that, with high probability, returns a solution $X = (X_1, \ldots, X_k) \in (k+1)^E$ with $\bigcup_{i \in [k]} X_i \in \mathcal{I}$ and quality at least $\frac{1}{2}\mathrm{OPT} - O(\frac{\Delta r(\mathcal{M}) \ln |E|}{\epsilon})$.*

Chapter 2

Ideal Membership Problem and CSPs

2.1 Preliminaries

2.1.1 Ideals, varieties and the Ideal Membership Problem

We follow the same notation and terminology as [59, 161] and the presentation of this section closely follows [161].

Let $\mathbb{F}$ denote an arbitrary field. Let $\mathbb{F}[x_1, \ldots, x_n]$ be the ring of polynomials over a field $\mathbb{F}$ and indeterminates $x_1, \ldots, x_n$. Sometimes it will be convenient not to assume any specific ordering or names of the indeterminates. In such cases we use $\mathbb{F}[X]$ instead, where X is a set of indeterminates, and treat points in $\mathbb{F}^X$ as mappings $\varphi : X \to \mathbb{F}$. The value of a polynomial $f \in \mathbb{F}[X]$ is then written as $f(\varphi)$. Let $\mathbb{F}[x_1, \ldots, x_n]_d$ denote the subset of polynomials of degree at most d. An *ideal* of $\mathbb{F}[x_1, \ldots, x_n]$ is a set of polynomials from $\mathbb{F}[x_1, \ldots, x_n]$ closed under addition and multiplication by a polynomial from $\mathbb{F}[x_1, \ldots, x_n]$. We will need ideals represented by a generating set.

Definition 2.1.1. *The ideal (of $\mathbb{F}[x_1, \ldots, x_n]$) generated by a finite set of polynomials $\{f_1, \ldots, f_m\}$ in $\mathbb{F}[x_1, \ldots, x_n]$ is defined as*

$$\mathbf{I}(f_1, \ldots, f_m) \stackrel{\text{def}}{=} \Big\{ \sum_{i=1}^{m} t_i f_i \mid t_i \in \mathbb{F}[x_1, \ldots, x_n] \Big\}.$$

Definition 2.1.2. *The set of polynomials that vanish in a given set $S \subset \mathbb{F}^n$ is called the* vanishing ideal *of S and denoted*

$$\mathbf{I}(S) \stackrel{\text{def}}{=} \{f \in \mathbb{F}[x_1, \ldots, x_n] : f(a_1, \ldots, a_n) = 0 \ \ \forall (a_1, \ldots, a_n) \in S\}.$$

Definition 2.1.3. *An ideal $\mathtt{I}$ is* radical *if $f^m \in \mathtt{I}$ for some integer $m \geq 1$ implies that $f \in \mathtt{I}$. For an arbitrary ideal $\mathtt{I}$ the smallest radical ideal containing $\mathtt{I}$ is denoted $\sqrt{\mathtt{I}}$. In other words $\sqrt{\mathtt{I}} = \{f \in \mathbb{F}[x_1, \ldots, x_n] \mid f^m \in \mathtt{I}$ for some $m\}$.*

Another common way to denote $\mathbf{I}(f_1, \ldots, f_m)$ is by $\langle f_1, \ldots, f_m \rangle$ and we will use both notations interchangeably.

Definition 2.1.4. *Let $\{f_1, \ldots, f_m\}$ be a finite set of polynomials in $\mathbb{F}[x_1, \ldots, x_n]$. We call*

$$\boldsymbol{V}(f_1, \ldots, f_m) \stackrel{\text{def}}{=} \{(a_1, \ldots, a_n) \in \mathbb{F}^n \mid f_i(a_1, \ldots, a_n) = 0 \quad 1 \leq i \leq m\}$$

the affine variety *defined by $f_1, \ldots, f_m$.*

Similarly, for an ideal $\mathtt{I} \subseteq \mathbb{F}[x_1, \ldots, x_n]$ we denote by $\boldsymbol{V}(\mathtt{I})$ the set $\boldsymbol{V}(\mathtt{I}) = \{(a_1, \ldots, a_n) \in \mathbb{F}^n \mid f(a_1, \ldots, a_n) = 0 \quad \forall f \in \mathtt{I}\}$.

The Weak Nullstellensatz states that in any polynomial ring, algebraic closure is enough to guarantee that the only ideal which represents the empty variety is the entire polynomial ring itself. This is the basis of one of the most celebrated mathematical results, Hilbert's Nullstellensatz.

Theorem 2.1.5 (The Weak Nullstellensatz)**.** *Let $\mathbb{F}$ be an algebraically closed field and let $\mathtt{I} \subseteq \mathbb{F}[x_1, \ldots, x_n]$ be an ideal satisfying $\boldsymbol{V}(\mathtt{I}) = \emptyset$. Then $\mathtt{I} = \mathbb{F}[x_1, \ldots, x_n]$.*

One might hope that the correspondence between ideals and varieties is one-to-one provided only that one restricts to algebraically closed fields. Unfortunately, this is not the case[1]. Indeed, the reason that the map $\mathbf{V}$ fails to be one-to-one is that a power of a polynomial vanishes on the same set as the original polynomial.

Theorem 2.1.6 (Hilbert's Nullstellensatz)**.** *Let $\mathbb{F}$ be an algebraically closed field. If f_0, $f_1, \ldots, f_s \in \mathbb{F}[x_1, \ldots, x_n]$, then $f_0 \in \mathbf{I}(\boldsymbol{V}(f_1, \ldots, f_s))$ if and only if $f_0^m \in \langle f_1, \ldots, f_s \rangle$ for some integer $m \geq 1$.*

By definition, radical ideals consist of all polynomials which vanish on some variety V. This together with Theorem 2.1.6 suggests that there is a one-to-one correspondence between affine varieties and radical ideals.

Theorem 2.1.7 (The Strong Nullstellensatz)**.** *Let $\mathbb{F}$ be an algebraically closed field. If $\mathtt{I}$ is an ideal in $\mathbb{F}[x_1, \ldots, x_n]$, then $\mathbf{I}(\boldsymbol{V}(\mathtt{I})) = \sqrt{\mathtt{I}}$.*

The following theorem is a useful tool for finding unions and intersections of varieties. We will use it in the following subsections where we construct ideals corresponding to CSP instances.

Theorem 2.1.8 ([59])**.** *If I and J are ideals in $\mathbb{F}[x_1, \ldots, x_n]$, then*

i. $\boldsymbol{V}(I \cap J) = \boldsymbol{V}(I) \cup \boldsymbol{V}(J)$,

ii. $\boldsymbol{V}(I + J) = \boldsymbol{V}(I) \cap \boldsymbol{V}(J)$.

[1]For example $\mathbf{V}(x) = \mathbf{V}\left(x^2\right) = \{0\}$ works over any field.

2.1.2 The ideal-CSP correspondence

Here, we explain how to construct an ideal corresponding to a given instance of CSP. Constraints are in essence varieties, see e.g. [122, 212]. Following [161, 212], we shall translate CSPs to polynomial ideals and back. Let $\mathcal{P} = (X, D, C)$ be an instance of CSP(Γ) where, throughout this thesis, Γ is a fixed constraint language with relations of fixed arities. Without loss of generality, we assume that $D \subset \mathbb{F}$[2]. Let $\mathsf{Sol}(\mathcal{P})$ be the (possibly empty) set of all solutions of $\mathcal{P}$. We wish to map $\mathsf{Sol}(\mathcal{P})$ to an ideal $\mathcal{I}(\mathcal{P}) \subseteq \mathbb{F}[X]$ such that $\mathsf{Sol}(\mathcal{P}) = \mathbf{V}(\mathcal{I}(\mathcal{P}))$. First, for every x_i the ideal $\mathcal{I}(\mathcal{P})$ contains a *domain* polynomial $f_D(x_i)$ whose zeroes are precisely the elements of D. Then for every constraint $R(x_{i_1}, \ldots, x_{i_k})$, where R is a predicate on D, the ideal $\mathcal{I}(\mathcal{P})$ contains a polynomial $f_R(x_{i_1}, \ldots, x_{i_k})$ that interpolates R, that is, for $(x_{i_1}, \ldots, x_{i_k}) \in D^k$ it holds $f_R(x_{i_1}, \ldots, x_{i_k}) = 0$ if and only if $R(x_{i_1}, \ldots, x_{i_k})$ is true. Note that each f_R has bounded degree, this is because D and k are fixed.

Including a domain polynomial for each variable has the advantage that it ensures that the ideals generated by our systems of polynomials are radical (see Lemma 8.19 of [21]). Hence, by Theorem 2.1.5 and Theorem 2.1.7, we have the following properties.

Theorem 2.1.9. *Let $\mathcal{P}$ be an instance of the CSP(Γ) and $\mathtt{I}(\mathcal{P})$ constructed as above. Then*

$$\boldsymbol{V}(\mathtt{I}(\mathcal{P})) = \emptyset \Leftrightarrow 1 \in \mathtt{I}(\mathcal{P}) \Leftrightarrow \mathtt{I}(\mathcal{P}) = \mathbb{F}[X], \qquad \text{(Weak Nullstellensatz)}$$

$$\mathtt{I}(\boldsymbol{V}(\mathtt{I}(\mathcal{P}))) = \sqrt{\mathtt{I}(\mathcal{P})}, \qquad \text{(Strong Nullstellensatz)}$$

$$\sqrt{\mathtt{I}(\mathcal{P})} = \mathtt{I}(\mathcal{P}). \qquad \text{(Radical Ideal)}$$

2.1.3 The Ideal Membership Problem

In the general Ideal Membership Problem we are given an ideal $\mathcal{I} \subseteq \mathbb{F}[x_1, \ldots, x_n]$, usually by some finite generating set, and a polynomial f_0. The question then is to decide whether or not $f_0 \in \mathcal{I}$. If $\mathcal{I}$ is given through a CSP instance, we can be more specific.

Definition 2.1.10. *The* Ideal Membership Problem *associated with a constraint language Γ over a set D is the problem IMP(Γ) in which the input is a pair $(f_0, \mathcal{P})$ where $\mathcal{P} = (X, D, C)$ is a CSP(Γ) instance and f_0 is a polynomial from $\mathbb{F}[X]$. The goal is to decide whether f_0 lies in the ideal $\mathtt{I}(\mathcal{P})$. We use IMP$_d$(Γ) to denote IMP(Γ) when the input polynomial f_0 has degree at most d.*

As $\mathcal{I}(\mathcal{P})$ is radical, by the Strong Nullstellensatz an equivalent way to solve the membership problem $f_0 \in \mathcal{I}(\mathcal{P})$ is to answer the following question:

Does there exist an $\mathbf{a} \in \mathbf{V}(\mathcal{I}(\mathcal{P}))$ such that $f_0(\mathbf{a}) \neq 0$?

[2] In fact, we will mainly assume $\mathbb{F} = \mathbb{R}$ and $D = \{0, 1, \ldots, |D| - 1\}$

In the **yes** case, such an **a** exists if and only if $f_0 \notin \mathbf{I}(\mathbf{V}(\mathcal{I}(\mathcal{P})))$ and therefore f_0 is **not** in the ideal $\mathcal{I}(\mathcal{P})$. This observation also implies

Lemma 2.1.11 ([161])**.** *For any constraint language* Γ *the problem* $IMP_0(\Gamma)$ *is equivalent to* not-$CSP(\Gamma)$.

As was observed in the Introduction, IMP(Γ) belongs to **coNP** for any Γ over a finite domain. We say that IMP(Γ) is *tractable* if it can be solved in polynomial time. We say that IMP(Γ) is *d-tractable* if $\mathrm{IMP}_d(\Gamma)$ can be solved in polynomial time for every d. Since IMP(Γ) is in **coNP**, throughout this thesis by the search IMP we understand the following problem.

> **Search version of IMP.** Let $(f_0, \mathcal{P})$ be an instance of IMP(Γ) such that $f_0 \notin \mathcal{I}(\mathcal{P})$, the problem is to find an assignment $\varphi \in \mathbf{V}(\mathcal{I}(\mathcal{P}))$ such that $f_0(\varphi) \neq 0$.

2.1.4 IMP and Gröbner Bases

In this section we present a very light introduction to Gröbner Bases and lay down some notation and background. We follow the standard notation in [59, 161].

A possible way to solve the IMP is via polynomial division. Informally, if a remainder of division of f_0 by generating polynomials of $\mathcal{I}(\mathcal{P})$ is zero then $f_0 \in \mathcal{I}(\mathcal{P})$. Let us recall some standard notations from algebraic geometry that are needed to present a division algorithm and the notion of Gröbner Bases.

A monomial ordering $\succ$ on $\mathbb{F}[x_1, \ldots, x_n]$ is a relation $\succ$ on $\mathbb{Z}^n_{\geq 0}$, or equivalently, a relation on the set of monomials $\mathbf{x}^\alpha$, $\alpha \in \mathbb{Z}^n_{\geq 0}$ (see [59], Definition 1, p.55). Each monomial $\mathbf{x}^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ corresponds to an n-tuple of exponents $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{Z}^n_{\geq 0}$. This establishes a one-to-one correspondence between the monomials in $\mathbb{F}[x_1, \ldots, x_n]$ and $\mathbb{Z}^n_{\geq 0}$. Any ordering $\succ$ we establish on the space $\mathbb{Z}^n_{\geq 0}$ will give us an ordering on monomials: if $\alpha \succ \beta$ according to this ordering, we will also say that $\mathbf{x}^\alpha \succ \mathbf{x}^\beta$.

Definition 2.1.12. *Let* $\alpha = (\alpha_1, \ldots, \alpha_n), \beta = (\beta_1, \ldots, \beta_n) \in \mathbb{Z}^n_{\geq 0}$ *and* $|\alpha| = \sum_{i=1}^n \alpha_i$, $|\beta| = \sum_{i=1}^n \beta_i$. Lexicographic order *and* graded lexicographic order *are defined as follows.*

1. *We say* $\alpha \succ_{\mathsf{lex}} \beta$ *if the leftmost nonzero entry of the vector difference* $\alpha - \beta \in \mathbb{Z}^n$ *is positive. We will write* $\mathbf{x}^\alpha \succ_{\mathsf{lex}} \mathbf{x}^\beta$ *if* $\alpha \succ_{\mathsf{lex}} \beta$.

2. *We say* $\alpha \succ_{\mathsf{grlex}} \beta$ *if* $|\alpha| > |\beta|$, *or* $|\alpha| = |\beta|$ *and* $\alpha \succ_{\mathsf{lex}} \beta$.

Definition 2.1.13. *For any* $\alpha = (\alpha_1, \cdots, \alpha_n) \in \mathbb{Z}^n_{\geq 0}$ *let* $\mathbf{x}^\alpha \stackrel{\text{def}}{=} \prod_{i=1}^n x_i^{\alpha_i}$. *Let* $f = \sum_\alpha a_\alpha \mathbf{x}^\alpha$ *be a nonzero polynomial in* $\mathbb{F}[x_1, \ldots, x_n]$ *and let* $\succ$ *be a monomial order.*

1. *The* multidegree *of* f *is* $\mathrm{multideg}(f) \stackrel{\text{def}}{=} \max(\alpha \in \mathbb{Z}^n_{\geq 0} : a_\alpha \neq 0)$.

2. *The* degree *of* f *is* $deg(f) = |\mathrm{multideg}(f)|$ *where* $|\alpha| = \sum_{i=1}^n \alpha_i$. *In this thesis, this is always according to* grlex *order.*

3. *The* leading coefficient *of* f *is* $\mathrm{LC}(f) \stackrel{\text{def}}{=} a_{\mathrm{multideg}(f)} \in \mathbb{F}$.

4. *The* leading monomial *of* f *is* $\mathrm{LM}(f) \stackrel{\text{def}}{=} \mathbf{x}^{\mathrm{multideg}(f)}$ *(with coefficient 1).*

5. *The* leading term *of* f *is* $\mathrm{LT}(f) \stackrel{\text{def}}{=} \mathrm{LC}(f) \cdot \mathrm{LM}(f)$.

Definition 2.1.14 (A division algorithm)**.** *Let* $\succ$ *be a monomial order on* $\mathbb{Z}^n_{\geq 0}$*, and let* $F = \{f_1, \ldots, f_s\} \subset \mathbb{F}[x_1, \ldots, x_n]$*. Then every* $f \in \mathbb{F}[x_1, \ldots, x_n]$ *can be written as* $f = q_1 f_1 + \cdots + q_s f_s + r$*, where* $q_i, r \in \mathbb{F}[x_1, \ldots, x_n]$*, and either* $r = 0$ *or* r *is a linear combination, with coefficients in* $\mathbb{F}$*, of monomials, none of which is divisible by any of* $\mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_s)$*. Furthermore, if* $q_i f_i \neq 0$*, then* $\mathrm{multideg}(f) \succeq \mathrm{multideg}(q_i f_i)$*. We call* r *a* remainder *of* f *on division by* F*. Also, we say that* f reduces to r modulo F*, written* $f \to_F r$*.*

The above definition suggests the following procedure to compute a remainder. Repeatedly, choose an $f_i \in F$ such that $\mathrm{LT}(f_i)$ divides some term t of f and replace f with $f - \frac{t}{\mathrm{LT}(f_i)} f_i$, until it cannot be further applied. Note that the order we choose the polynomial f_i is not specified. Unfortunately, depending on the generating polynomials of the ideal a remainder of division may not be unique. Moreover, such a remainder depends on the order we do division.

Example 2.1.15. Let $f = x^2y - xy^2 + y$ and $I = \langle f_1, f_2 \rangle$ with $f_1 = x^2$ and $f_2 = xy - 1$. Consider the grlex order with $x \succ_{\mathsf{lex}} y$. On one hand, $f = 0 \cdot f_1 + (x - y) \cdot f_2 - x$. On the other hand, $f = y \cdot f_1 - y \cdot f_2 + 0$.

The Hilbert Basis Theorem states that every ideal has a finite generating set (see, e.g., Theorem 4 on page 77 [59]). Fortunately, for every ideal there is a finite generating set that suits our purposes. That is, there is a generating set so that the remainder of division by that set is uniquely defined, no matter in which order we do the division.

Definition 2.1.16. *Fix a monomial order on the polynomial ring* $\mathbb{F}[x_1, \ldots, x_n]$*. A finite subset* $G = \{g_1, \ldots, g_t\}$ *of an ideal* $\mathtt{I} \subseteq \mathbb{F}[x_1, \ldots, x_n]$ *different from* $\{0\}$ *is said to be a Gröbner Basis (or* standard basis*) if*

$$\langle \mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_t) \rangle = \langle \mathrm{LT}(I) \rangle$$

where $\langle \mathrm{LT}(I) \rangle$ *denotes the ideal generated by the leading terms of elements of* $\mathtt{I}$*.*

Proposition 2.1.17 ([59], Proposition 1, p.83)**.** *Let* $\mathtt{I} \subseteq \mathbb{F}[x_1, \ldots, x_n]$ *be an ideal and let* $G = \{g_1, \ldots, g_t\}$ *be a Gröbner Basis for* $\mathtt{I}$*. Then given* $f \in \mathbb{F}[x_1, \ldots, x_n]$*, there is a unique* $r \in \mathbb{F}[x_1, \ldots, x_n]$ *with the following two properties:*

1. *No term of* r *is divisible by any of* $\mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_t)$*,*

2. *There is* $g \in \mathtt{I}$ *such that* $f = g + r$*.*

In particular, r is the remainder on division of f by G no matter how the elements of G are listed when using the division algorithm.

The remainder r is called the *normal form of f by G*, denoted by $f|_G$. Note that, although the remainder r is unique, even for a Gröbner Basis, the "quotients" q_i produced by the division algorithm $f = q_1g_1 + \cdots + q_tg_t + r$ can change if we list the generators in a different order. As a corollary of Proposition 2.1.17, we get the following criterion for when a given polynomial lies in an ideal.

Corollary 2.1.18 ([59], Corollary 2, p.84)**.** *Let $G = \{g_1, \ldots, g_t\}$ be a Gröbner Basis for an ideal $\mathtt{I} \subseteq \mathbb{F}[x_1, \ldots, x_n]$ and let $f \in \mathbb{F}[x_1, \ldots, x_n]$. Then $f \in \mathtt{I}$ if and only if the remainder on division of f by G is zero.*

There is a criterion, known as Buchberger's criterion, that tells us whether a given generating set of an ideal is a Gröbner Basis. In order to formally express this criterion, we need to define the notion of S-polynomials.

Definition 2.1.19 (S-polynomial)**.** *Let $f, g \in \mathbb{F}[x_1, \ldots, x_n]$ be nonzero polynomials. If* multideg$(f) = \alpha$ *and* multideg$(g) = \beta$*, then let $\gamma = (\gamma_1, \ldots, \gamma_n)$, where $\gamma_i = \max(\alpha_i, \beta_i)$ for each i. We call x^γ the* least common multiple *of* LM(f) *and* LM(g)*, written $x^\gamma =$* lcm(LM(f), LM(g))*. The S-*polynomial *of f and g is the combination*

$$S(f, g) = \frac{x^\gamma}{\mathrm{LT}(f)} \cdot f - \frac{x^\gamma}{\mathrm{LT}(g)} \cdot g.$$

Theorem 2.1.20 (Buchberger's Criterion [59], Theorem 3, p.105)**.** *Let $\mathtt{I}$ be a polynomial ideal. Then a basis $G = \{g_1, \ldots, g_t\}$ of $\mathtt{I}$ is a Gröbner Basis of $\mathtt{I}$ if and only if for all pairs $i \neq j$, the remainder on division of $S(g_i, g_j)$ by G (listed in some order) is zero.*

Proposition 2.1.21 ([59], Proposition 4, p.106)**.** *We say the leading monomials of two polynomials f, g are relatively prime if* lcm(LM(f), LM(g)) = LM$(f) \cdot$ LM(g)*. Given a finite set $G \subseteq \mathbb{F}[x_1, \ldots, x_n]$, suppose that we have $f, g \in G$ such that the leading monomials of f and g are relatively prime. Then $S(f, g) \to_G 0$.*

If we restrict ourselves to the polynomials of degree at most d then we obtain a *d-truncated Gröbner Basis.*

Definition 2.1.22 (d-truncated Gröbner Basis)**.** *If G is a Gröbner Basis of an ideal, the d-truncated Gröbner Basis G' of G is defined as*

$$G' = G \cap \mathbb{F}[x_1, \ldots, x_n]_d,$$

where $\mathbb{F}[x_1, \ldots, x_n]_d$ is the set of polynomials of degree less than or equal to d.

Solving IMP$_d$ and Gröbner Bases. Note that having a Gröbner Basis does not guarantee a polynomial time membership test unless the input polynomial has bounded degree. For instance, consider the following example. Let $\mathcal{I} \subseteq \mathbb{F}[x_1, \ldots, x_n, y_1, \ldots, y_n]$ be the ideal generated by set of polynomials $G = \{x_1 - y_1 - 1, \ldots, x_n - y_n - 1\}$ which is indeed a Gröbner Basis with respect to the grlex ordering $x_1 \succ_{\mathsf{lex}} \ldots \succ_{\mathsf{lex}} x_n \succ_{\mathsf{lex}} y_1 \succ_{\mathsf{lex}} \ldots \succ_{\mathsf{lex}} y_n$. Now let the input polynomial be $x_1 \cdots x_n$. If we apply the division algorithm, we obtain the expansion of the polynomial $(y_1 + 1) \cdots (y_n + 1)$, which contains exponentially many monomials. Hence, while the division algorithm solves the problem correctly, it produces exponentially long intermediate results, and therefore is exponential time.

To solve IMP$_d$ or even find a proof of membership it suffices to compute a d-truncated Gröbner Basis with respect to a grlex order. This is because, for the input polynomial f_0 of degree d, the only polynomials from G that can possibly divide f_0 are those from G_d. Moreover, the remainders of such divisions have degree at most d.

2.2 Overview of our contributions

In this thesis we expand on [161] and [24, 25] in several ways. We consider IMP(Γ) for languages Γ over arbitrary finite set and attempt to obtain general results about such problems. However, we mainly focus on a slightly different problem than Problem 1.2.1.

Problem 2.2.1. *For which constraint languages Γ the problem IMP(Γ) [or IMP$_d$(Γ)] can be solved in polynomial time?*

Note that answering whether f_0 belongs to a certain ideal does not necessarily mean finding an ideal membership proof of that. However, we will argue that, firstly, in many applications this is the problem we need to solve and therefore our results apply. Secondly, in Section 5.1.2 we will show that in the majority of cases if the existence of an ideal membership proof can be efficiently decided, such a proof can also be efficiently found.

Expanding the constraint language. Firstly, in Section 3.1 we study reductions between IMP's when the language Γ is enlarged in certain ways. Let Γ be a constraint language over a set D. By Γ^* we denote Γ with added *constant* relations, that is, relations of the form $\{(a)\}$, $a \in D$. Imposing such a constraint on a variable x essentially fixes the allowed values of x to be a. First, we prove that adding constant relations does not change the complexity of the IMP. (The case of $d = 0$ and $|D| = 2$ of the following theorem was considered in [161] under the name of singleton expansion.)

Theorem 2.2.2. *For any Γ over D the problem IMP(Γ^*) is polynomial time reducible to IMP(Γ), and for any d the problem IMP$_d$(Γ^*) is polynomial time reducible to IMP$_{d+|D|(|D|-1)}$(Γ).*

Theorem 2.2.2 has two immediate consequences. Since IMP(Γ) is in **co-NP**, for any CSP(Γ) instance $\mathcal{P}$ there is always a proof that the input polynomial f_0 does not belong to

the ideal $\mathcal{I}(\mathcal{P})$. Any solution of $\mathcal{P}$ that is not a zero of f_0 will do. Finding such a proof may be treated as a search version of IMP(Γ). Through self-reducibility, Theorem 2.2.2 allows us to solve the search problem.

Theorem 2.2.3. *Let Γ be such that IMP(Γ) [IMP$_{d+|D|(|D|-1)}(\Gamma)$] is solvable in polynomial time. Then for any instance $(f_0, \mathcal{P})$ of IMP(Γ) [IMP$_d(\Gamma)$] such that $f_0 \notin \mathtt{I}(\mathcal{P})$, a solution $\mathbf{a}$ of $\mathcal{P}$ such that $f_0(\mathbf{a}) \neq 0$ can also be found in polynomial time.*

Theorem 2.2.2 also provides a hint at a more plausible conjecture for which languages Γ the problem IMP(Γ) or IMP$_d(\Gamma)$ is polynomial time. In particular, it allows to find an example of Γ such that CSP(Γ) is tractable while IMP$_d(\Gamma)$ is not, even for a Γ on a 2-element set and $d = 1$. Later we state some results that might indicate that IMP$_d(\Gamma)$ is polynomial time for every Γ such that CSP(Γ^*) is polynomial time. Note that the structure of such CSPs is now very well understood.

Another way of expanding a constraint language is by means of *primitive-positive (pp-) definitions* and *pp-interpretations*, and it is at the core of the so-called algebraic approach to the CSP. A relation R is said to be pp-definable in Γ if there is a first order formula Φ using only conjunctions, existential quantifiers, equality relation, and relations from Γ that is equivalent to R. Pp-interpretations are more complicated (see Section 3.1.3) and allow for certain encodings of R.

Theorem 2.2.4. *(1) Let Γ, Δ be constraint languages over the same set D, Δ is finite, and every relation from Δ is pp-definable in Γ. Then IMP(Δ) is polynomial time reducible to IMP(Γ) and IMP$_d(\Delta)$ is polynomial time reducible to IMP$_d(\Gamma)$ for any d.*

(2) Let Γ, Δ be constraint languages, Δ is finite, and Δ is pp-interpretable in Γ. Then there is a constant k such that IMP$_d(\Delta)$ is polynomial time reducible to IMP$_{kd}(\Gamma)$ for any d.

The approach of Theorem 2.2.4 was first applied to various proof systems in [6], although that work is mostly concerned with proof complexity rather than computational complexity. Mastrolilli [161] ventured into pp-definability without proving any reductions. In particular, the first part of Theorem 2.2.4 uses techniques from [161] for projections of ideals. It will later allow us to develop further universal algebra techniques for the IMP. The second part of that theorem will also work towards more powerful universal algebra methods. We extend the reductions of Theorem 2.2.4 to the multi-sorted case where each variable has its own domain. The multi-sorted framework is standard for CSPs and is pivotal in proofs of the dichotomy theorem of CSPs [35, 222] and we anticipate the multi-sorted IMP becomes the standard framework in this line of research.

Recall that according to [192] in order to find an SOS proof one needs to be able to find Nullstellensatz proofs efficiently. The reductions from Theorems 2.2.2, 2.2.4 do not always

allow to recover such a proof efficiently only confirming such a proof exists, and therefore cannot be directly used in the conditions from [192]. However, in Section 5.3 we address this issue.

Polymorphisms and sufficient conditions for tractability. Recall that a *polymorphism* of a constraint language Γ over a set D is a multi-ary operation on D that can be viewed as a multi-dimensional symmetry of relations from Γ. By $\mathsf{Pol}(\Gamma)$ we denote the set of all polymorphisms of Γ. As in the case of the CSP, Theorem 2.2.4 implies that polymorphisms of Γ is what determines the complexity of $\mathrm{IMP}(\Gamma)$. In Section 3.2.1 we show the following.

Corollary 2.2.5. *Let Γ, Δ be constraint languages over the same set D, Δ is finite. If $\mathsf{Pol}(\Gamma) \subseteq \mathsf{Pol}(\Delta)$, then $IMP(\Delta)$ is polynomial time reducible to $IMP(\Gamma)$ and $IMP_d(\Delta)$ is polynomial time reducible to $IMP_d(\Gamma)$ for any d.*

Corollary 2.2.5 allows us to represent IMPs through polymorphisms and classify the complexity of IMPs according to the corresponding polymorphisms. The method has been initiated by Mastrolilli and Bharathi [24, 25, 161], although mainly for 2-element set and one case of finite domain. We apply this approach to obtain several sufficient conditions for tractability of the IMP.

Theorem 2.2.6. *Let Γ be a constraint language over a set D. Then if one of the following conditions holds, $IMP_d(\Gamma)$ is decidable in polynomial time for any d.*

1. *Γ has the* dual-discriminator *polymorphism (i.e. a ternary operation g such that $g(x,y,z) = x$ unless $y = z$, in which case $g(x,y,z) = y$);*
2. *Γ has a* semilattice *polymorphism (i.e. a binary operation f such that $f(x,x) = x$, $f(x,y) = f(y,x)$, and $f(f(x,y),z) = f(x,f(y,z))$);*
3. *$|D| = p$, p prime, and Γ has an* affine *polymorphism modulo p (i.e. a ternary operation $h(x,y,z) = x \ominus y \oplus z$, where $\oplus, \ominus$ are addition and subtraction modulo p, or, equivalently, of the field $\mathsf{GF}(p)$). In this case every CSP can be represented as a system of linear equations over $\mathsf{GF}(p)$.*
4. *D is an Abelian group and the affine operation $x - y + z$ of D is a polymorphism of Γ.*

For the first part of Theorem 2.2.6 we come up with a technique of preprocessing the input polynomial f_0 that allows us to get rid of permutation constraints and greatly simplify the proof for the dual-discriminator polymorphism, including the special case $|D| = 3$ considered in [24]. After our results, [26] gave an algorithm to compute a bounded degree Gröbner Basis for this case.

In the second part of Theorem 2.2.6 we use the fact, see [181], that any language with a semilattice polymorphism is pp-interpretable in a language on a 2-element set also having a semilattice polymorphism. Then Theorem 2.2.6(2) follows from Theorem 2.2.4(2) and the results of [161].

As is mentioned, the third part of Theorem 2.2.6 is in fact about systems of linear equations over $\mathsf{GF}(p)$, since every instance of CSP(Γ), where Γ has an affine polymorphism, is equivalent to a system of linear equations over $\mathsf{GF}(p)$. Bharathi and Mastrolilli solved this case for $p = 2$ showing that $\mathrm{IMP}_d(\Gamma)$ is polynomial time for any d. Their approach is based on FGML algorithm [75] to construct a bounded degree Gröbner Basis. Instead, we map an instance of IMP(Γ) on a different domain consisting of p-th roots of unity rather than $\{0, \dots, p-1\}$. This transforms the generators of the ideal into very simple polynomials that form a Gröbner Basis without any further modifications. Such a transformation makes it difficult to find an ideal membership proof, but this is resolved in Section 5.1.2.

For the fourth part of Theorem 2.2.6 we need to use a completely different approach than the one used in the third part; system of linear equations over $\mathsf{GF}(p)$. The reasons are that, in the case of Abelian groups, an instance generally cannot be represented as a system of linear equations, Gaussian elimination does not work on systems of linear equations over an arbitrary Abelian group, and a reduced row-echelon form cannot be converted into a Gröbner basis. Given an instance $(f_0, \mathcal{P})$ of IMP(Γ) we use the Fundamental Theorem of Abelian groups and a generalized version of pp-interpretations for the IMP [41] to reduce $(f_0, \mathcal{P})$ to an instance $(f_0', \mathcal{P}')$ of multi-sorted IMP(Δ), in which every variable takes values from a set of the form $\mathbb{Z}_{p^\ell}$, p prime. Then we replace the domains $\mathbb{Z}_{p^\ell}$ of $(f_0', \mathcal{P}')$ by sets of roots of unity that allows for a more concise representation of polynomials. Finally, we show that a Gröbner Basis for the resulting problem can be efficiently constructed.

Algebras. In Section 3.2.2 we prove that the standard features of the universal algebraic approach to the CSP work for IMP as well. These include reductions for standard algebraic constructions such as *subalgebras*, *direct powers*, and *homomorphic images*. They easily follow from Theorem 2.2.4(2). A more general construction of direct product requires a more general version of CSP(Γ), and therefore of IMP(Γ), the multi-sorted one, in which every variable can have its own domain of values. This is discussed in details in Section 3.3. One implication of these results is a necessary condition for tractability of IMP(Γ) that follows from a similar one for the CSP.

The IMP with indeterminate coefficients. In Chapter 5 we consider a number of applications of the techniques developed in the first part. The key to those applications is an extension of the IMP defined as follows. Given an ideal $\mathtt{I} \subseteq \mathbb{F}[x_1, \dots, x_n]$ and a vector of ℓ polynomials $M = (g_1, \dots, g_\ell)$, the χIMP asks if there exist coefficients $\mathbf{c} = (c_1, \dots, c_\ell) \in \mathbb{F}^\ell$ such that $\mathbf{c}M = \sum_{i=1}^{\ell} c_i g_i$ belongs to the ideal $\mathtt{I}$.

As with the regular IMP, χIMP can be parametrized by specifying a constraint language Γ, in which case the resulting problem $\chi\mathrm{IMP}(\Gamma)$ (or $\chi\mathrm{IMP}_d(\Gamma)$ if the degree of input polynomials is bounded) only allows ideal produced by instances of $\mathrm{CSP}(\Gamma)$.

We prove that $\chi\mathrm{IMP}_d$ can be solved in polynomial time when a (d-truncated) Gröbner Basis can be efficiently generated, and also admits the same reductions as the IMP.

Theorem 2.2.7. *Let Γ, Δ be constraint languages and Δ is finite. Then*

(1) If every relation from Δ is pp-definable in Γ, then $\chi IMP(\Delta)$ is polynomial time reducible to $\chi IMP(\Gamma)$ and $\chi IMP_d(\Delta)$ is polynomial time reducible to $\chi IMP_d(\Gamma)$ for any d.

(2) If Δ is pp-interpretable in Γ, then there is a constant k such that $\chi IMP_d(\Delta)$ is polynomial time reducible to $\chi IMP_{kd}(\Gamma)$ for any d.

The theorem above allows us to show that for every Γ for which $\mathrm{IMP}_d(\Gamma)$ is polynomial time solvable, so is $\chi\mathrm{IMP}_d(\Gamma)$. This includes constraint languages invariant under dual-discriminator, semilattice, and affine polymorphisms.

In Section 5.1.2 we use χIMP along with the factor ring $\mathbb{F}[x_1, \ldots, x_n]/\mathtt{I}$ modulo an ideal $\mathtt{I}$ to generate a basis for the factor ring consisting of monomials of degree at most d, and then use it to construct a d-truncated Gröbner Basis for $\mathtt{I}$. More precisely, we prove the following.

Theorem 2.2.8. *Let $\mathcal{H}$ be a class of ideals for which χIMP_d is polynomial time solvable. Then there exists a polynomial time algorithm that constructs a degree d Gröbner Basis (with respect to a grlex) of an ideal $\mathtt{I} \in \mathcal{H}$, $\mathtt{I} \subseteq \mathbb{F}[x_1, \ldots, x_n]$, in time $O(n^d)$.*

Theorem 2.2.8 makes it possible to construct d-truncated Gröbner Bases (with respect to a grlex) in all cases $\mathrm{IMP}_d(\Gamma)$ is known to be polynomial time. Thus, it basically eliminates the gap between deciding the existence of an ideal membership proof and finding such a proof.

While our results for tractable cases of $\chi\mathrm{IMP}_d$ are proved in an ad hoc manner, they all share the same scheme. That is, to solve the $\chi\mathrm{IMP}_d$ one might reduce the problem at hand to a problem for which a Gröbner Basis can be constructed in a relatively simple way. We formally formulate this powerful idea and in Section 5.2 develop a unifying construction based on *substitution reductions* that covers all the useful cases so far. We believe our substitution techniques will find further applications in the study of IMP.

IMP and SOS proofs. In Section 5.3, we apply χIMP and algebraic techniques to finding Sum-of-Squares (SoS) proofs. For variables $x_1, \ldots, x_n$ a semialgebraic set is given by a collection of polynomial equalities and inequalities such as

$$S = \{\mathbf{x} \in \mathbb{R}^n \mid p_1(\mathbf{x}) = 0, \ldots, p_m(\mathbf{x}) = 0, q_1(\mathbf{x}) \geq 0, \ldots, q_\ell(\mathbf{x}) \geq 0\}.$$

The goal is to prove that some polynomial $r(\mathbf{x})$ is nonnegative on S. An SOS proof of $r(\mathbf{x}) \geq 0$ is given by a polynomial identity of the form

$$r(\mathbf{x}) = \sum_{i=1}^{t_0} h_i^2(\mathbf{x}) + \sum_{k=1}^{\ell} (\sum_{j=1}^{t_k} s_j^2(\mathbf{x})) q_k(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i(\mathbf{x}) p_i(\mathbf{x}). \tag{2.1}$$

The degree of an SOS proof is often defined to be the maximum degree of the polynomials involved in the proofs i.e., $\max\{deg(h_i^2), deg(s_j^2 q_k), deg(\lambda_i p_i)\}$. If the degree is bounded such a proof can *often* be found using an SDP program.

O'Donnell [178] discovered that in some cases although an SOS proof of low degree exists, it may involve exponentially long coefficients, and therefore it may be impossible to find such a proof efficiently. The example given by O'Donnell essentially contains the following system

$$\mathcal{P} = \{x_1^2 - x_2, x_2^2 - x_3, \ldots, x_{n-1}^2 - x_n, x_n^2\}.$$

Note that $\mathcal{P}$ is indeed a Gröbner Basis however the ideal generated by $\mathcal{P}$, $\mathtt{I}(\mathcal{P})$, *is not radical*. The polynomial $\epsilon - x_1$ is nonnegative on $S = \{(0, \ldots, 0)\}$ and every SOS proof of nonnegativity of bounded degree must involve polynomials with exponentially large coefficients [218].

It turns out even adding domain polynomials which makes the ideal radical does not guarantee low bit complexity of coefficients in (2.1), for instance see the Knapsack example in [192]. Raghavendra and Weitz [192] suggested three conditions such that if the set S satisfies them, the existence of a low degree SOS proof of the form (2.1) implies the existence of a low bit complexity one of the form (2.1). Two of these conditions hold for the majority of combinatorial problems, and the third one is so called kd-completeness of the IMP part of the proof. This means the ideal generated by $p_1, \ldots, p_m$ is radical, and furthermore $p_1, \ldots, p_m$ are k-effective meaning every degree d polynomial in the ideal has a degree kd derivation from $p_1, \ldots, p_m$. Ideally, we would like to have $k = O(d)$, this is a very strong condition and there is no known universal strategy to verify k-effectiveness; even if we are given a Gröbner Basis of the ideal[3]. We point out the strategy suggested and used by Raghavandra and Weitz "is by no means universally applicable, and it had to be applied on a case-by-case basis", as marked in [218].

The situation is different if we are interested in SOS proofs of nonnegativity *modulo* the ideal $\mathcal{I}(\mathcal{P}) = \langle p_1, \ldots, p_m \rangle$ rather than modulo $\{p_1, \ldots, p_m\}$ — which is sufficient to show $r(\mathbf{x})$ is nonnegative on S. A polynomial $r(\mathbf{x})$ is called SOS modulo S (or modulo $\mathcal{I}(\mathcal{P})$, when

[3]Obviously, if $p_1, \ldots, p_m$ form a Gröbner Basis then they are 1-effective i.e. every degree d polynomial in the ideal has a degree d derivation from $p_1, \ldots, p_m$.

the ideal is radical) if there are h_i, s_j, and $g \in \mathcal{I}(\mathcal{P})$ such that

$$r(\mathbf{x}) = \sum_{i=1}^{t_0} h_i^2(\mathbf{x}) + \sum_{k=1}^{\ell} (\sum_{j=1}^{t_k} s_j^2(\mathbf{x}))q_k(\mathbf{x}) + g \tag{2.2}$$

Note that polynomial g might be of the form $g = \sum_{i=1}^{m} \lambda_i(\mathbf{x})p_i(\mathbf{x}) + f$ with $f \in \mathcal{I}(\mathcal{P})$ and some nonzero $\lambda_i(\mathbf{x})$. It is known that every nonnegative polynomial on S is of the form (2.2), provided that the ideal $\mathcal{I}(\mathcal{P})$ is radical which is a crucial requirement, for example see [145]. Adapting the proof from [192], it is immediate to prove that any nonnegative polynomial over S that admits a degree d SOS proof of form (2.2) also admits a degree d SOS proof of form (2.2) with low bit complexity coefficients.

Theorem 2.2.9. *Let $\mathcal{P}$ be an instance of CSP(Γ) with domain D, and $\mathtt{I}(\mathcal{P}) = \langle p_1 \dots, p_m \rangle$ be the corresponding ideal to $\mathcal{P}$. Assume that $\forall q_i, \forall \mathbf{a} \in S$ we have $q_i(\mathbf{a}) > \varepsilon$.*

Then if r has a degree d SOS proof of nonnegativity modulo $\mathtt{I}(\mathcal{P})$, it also has a degree d SOS proof of nonnegativity modulo $\mathtt{I}(\mathcal{P})$ with coefficients bounded by $2^{poly(n^d, n^{|D|}, \log \frac{1}{\varepsilon})}$. In particular, if there are no polynomial inequalities then every coefficient can be written down with only $poly(n^d, n^{|D|})$ bits.

The above theorem does not tell us how to decide in polynomial time if a polynomial r is SOS modulo S as opposed to the SOS proofs of the form (2.1) where the problem reduces to solving an SDP. One approach suggested by Raghavendra and Weitz and Mastrolilli, is to express the polynomial g in (2.2) in terms of "nice" generating sets of polynomials such as Gröbner Bases and then use an SDP formulation, see Problem 1.2.1. Here, we present a different perspective through the lens of χIMP to decide the existence of an SOS proof modulo S. The following theorem puts forward the idea that the χIMP is the main player in SOS proofs automatability and allows us to use a much larger tool box than the usual Gröbner Basis.

Theorem 2.2.10. *Let Γ be a constraint language such that $\chi IMP_d(\Gamma)$ is polynomial time solvable. Let $\mathcal{P}$ be an instance of CSP(Γ) and $\mathtt{I}(\mathcal{P}) = \langle p_1 \dots, p_m \rangle$ be the corresponding ideal to $\mathcal{P}$. Assume that $\forall q_i, \forall \mathbf{a} \in S$ we have $q_i(\mathbf{a}) > \varepsilon$.*

Then for a polynomial r, the existence of an SOS proof of form (2.2) with

$$\max\{deg(h_i^2), deg(s_j^2 q_k)\} \leq d$$

is polynomial time decidable. Furthermore, such a proof can be found in polynomial time, if one exists.

Theorem 2.2.10 requires $\chi\text{IMP}_d(\Gamma)$ to be polynomial time solvable. However, in many cases e.g., Boolean CSPs such as Vertex Cover, Clique, Stable Set, and 2-SAT, the corresponding χIMP is polynomial time solvable and a Gröbner Basis can be computed in

polynomial time. On one hand, in such cases, it is straightforward to impose restrictions on SOS polynomials, for instance having domain polynomials $x_i^2 - x_i$ allows us to restrict to multilinear SOS proofs. However, on the other hand, in such cases the degree restriction is less demanding particularly on the "ideal part". It would be interesting to see whether this lead to some improvements on the existing SOS SDP relaxations for problems where the corresponding χIMP is polynomial time solvable.

IMP and theta bodies. In Section 5.4, we apply our techniques to show that the *theta bodies* [90] arising from certain combinatorial problems can be constructed in polynomial time. One of the core problems in optimization is to understand the $\texttt{conv}(S)$ or a relaxation of $\texttt{conv}(S)$, where S the set of feasible solutions to a given problem and $\texttt{conv}(S)$ denotes the convex hull of S. Here, we consider combinatorial ideals arising from CSPs where S is a finite subset of $\mathbb{R}^n$. Theta body relaxations, introduced by Gouveia, Parrilo and Thomas [90], obtain a hierarchy of relaxations to $\texttt{conv}(S)$. They are strong relaxations, for instance, they achieve the best approximation among all symmetric SDPs of a comparable size [218] and are known to have nice properties [90].

Let $TH_k(\mathcal{I}(\mathcal{P}))$, $\mathcal{P} \in \mathrm{CSP}(\Gamma)$, denote the k-th theta body of $\mathcal{I}(\mathcal{P})$. Theta bodies create a nested sequence of closed convex relaxations i.e., $TH_1(\mathcal{I}(\mathcal{P})) \supseteq TH_2(\mathcal{I}(\mathcal{P})) \supseteq \cdots \supseteq \texttt{conv}(S)$. We say a constraint language Γ is TH_k-exact if for any instance $\mathcal{P}$ of $\mathrm{CSP}(\Gamma)$ the ideal $\mathcal{I}(\mathcal{P})$ is TH_k-*exact*. The ideal $\mathcal{I}(\mathcal{P})$ is TH_k-exact if $TH_k(\mathcal{I}(\mathcal{P})) = \texttt{conv}(S)$. Zero-dimensional ideals are TH_k-exact for some finite k [145]. An intriguing question is characterizing TH_k-exact constraint languages, for constant k: *Which constraint languages are TH_k-exact, for some constant k?*. This is analogous to Lovász's question [155] where he asked: *Which ideals in $\mathbb{R}[x_1, \ldots, x_n]$ are TH_k-exact* [4]. This question is partially answered in [90] for the case $k = 1$ from a completely different perspective. From algorithmic point of view, a natural question to ask is that for which constraint languages constructing theta bodies is a polynomial time task.

Problem 2.2.11. *For which constraint languages Γ the k-th theta body $TH_k(\mathtt{I}(\mathcal{P}))$ is computable in polynomial time where $\mathcal{P}$ is an instance of CSP(Γ)?*

In Section 5.4, we provide strong evidence that the polymorphisms of constraint languages might be the right notion that one should consider to address Problem 2.2.11. To construct the k-th theta bodies it is sufficient to obtain a basis for the factor ring $\mathbb{R}[x_1, \ldots, x_n]_k/\mathcal{I}$ modulo ideal $\mathcal{I}$ [145, 184]. Our result in Theorem 2.2.8 gives us such a luxury. We point out that the connection between theta bodies and the IMP was first reported in [161] and efficient constructions of theta bodies of Boolean problems have been addressed

[4] In fact, Lovász's asked which ideals in $\mathbb{R}[x_1, \ldots, x_n]$ are $(1, k)$-SOS. An ideal is $(1, k)$-SOS if every nonnegative linear polynomial on S is k-SOS mod $\mathcal{I}$. However, it turns out that a radical ideal in $\mathbb{R}[x_1, \ldots, x_n]$ is $(1, k)$-SOS if and only if it is TH_k-exact [90].

by Mastrolilli. While there are only very few problems for which efficient construction of theta bodies are known, we provide a unifying framework to study the computational aspects of theta bodies and, ultimately, making progress towards answering Problem 2.2.11. In particular, we present (rediscover) several positive results for problems such as Stable Set, Binary Matroids, H-Coloring, Min/Max Ones, and Strict CSPs.

Chapter 3

Algebraic approach to IMP

3.1 Expanding the constraint language

In this section we discuss constructions on relations that allow us to reduce one IMP with a fixed constraint language to another. First we show that adding so-called *constant* relations does not change the complexity of the problem. Second, we will consider languages on the same domain, and prove that *primitive positive* (*pp*-, for short) *definitions* between constraint languages provides a reduction between the corresponding IMPs. Third, we will turn our attention to the case where two languages are defined on different domains. In this case, we study a stronger notion called *primitive positive interpretability*. We prove that if a language Γ pp-interprets a language Δ, then $\mathrm{IMP}(\Delta)$ is reducible to $\mathrm{IMP}(\Gamma)$. Finally, we discuss how ideal membership proofs can (or cannot) be recovered under these reductions.

3.1.1 Constant relations and the search problem

We start with expansion of a constraint language Γ on a set D by constant relations. A *constant* relation R_a, $a \in D$, is the unary relation, that is, a subset of D, that contains just one element a. Using it in a CSP is equivalent to *pinning* a variable to a fixed value a. Expansion by constant relations is very important for CSPs. It preserves the complexity of the decision version of the problem when Γ is a *core*, see [39], and it preserves the complexity of the counting version of the problem for any Γ, see [37]. For $A \subseteq D$ let Γ^A denote $\Gamma \cup \{R_a \mid a \in A\}$. We also call constraints of the form $\langle x, R_a \rangle$ *pinning* constraints.

Proposition 3.1.1. *Let Γ be a constraint language on a set D and $A \subseteq D$. Then $IMP_d(\Gamma^A)$ is polynomial time reducible to $IMP_{d+|A|(|D|-1)}(\Gamma)$.*

Proof. Let $\mathcal{P} = (X, D, C)$ be an instance of $\mathrm{CSP}(\Gamma^A)$ and let $\mathcal{I}(\mathcal{P})$ be the ideal corresponding to $\mathcal{P}$. Suppose $(f_0, \mathcal{P})$ is an instance of $\mathrm{IMP}_d(\Gamma^A)$ where we want to decide if $f_0 \in \mathcal{I}(\mathcal{P})$. First, we perform some preprocessing of $(f_0, \mathcal{P})$. Note that if $\mathcal{P}$ contains constraints $\langle x, R_a \rangle$, $\langle x, R_b \rangle$, $a \neq b$, then $\mathcal{P}$ has no solution and so $1 \in \mathcal{I}(\mathcal{P})$ implying $f_0 \in \mathcal{I}(\mathcal{P})$. Let X_a denote the set of variables x, for which there is a constraint $\langle x, R_a \rangle \in C$. Introduce new variable

x_a for each $a \in A$ and replace every $x \in X_a$, with x_a in both f_0 and $\mathcal{P}$. In particular, let

$$X' = \left(X \setminus \bigcup_{a \in A} X_a \right) \cup \{x_a \mid a \in A\}.$$

The resulting instance $(f_0', \mathcal{P}')$ has the following properties:

- The solutions of $\mathcal{P}$ and $\mathcal{P}'$ are in one-to-one correspondence, since for every solution φ of $\mathcal{P}$ we have $\varphi(x) = a$ for each $x \in X_a$, and so the mapping $\varphi' : X' \to D$ such that $\varphi(x_a) = a$ for $a \in A$ and $\varphi'(x) = \varphi(x)$ otherwise is a solution of $\mathcal{P}'$ and vice versa.
- $f_0(\varphi) = 0$ if and only if $f_0'(\varphi') = 0$.

Now let $\mathcal{P}^* = (X', D, C^*)$ be an instance of CSP(Γ) where C^* consists of all constraint from C' except the ones of the form $\langle x, R_a \rangle$, $a \in A$. We define a new polynomial f_0^* as follows.

$$f_0^* = \left(\prod_{a \in A} \prod_{b \in D \setminus \{a\}} (x_a - b) \right) \cdot f_0'.$$

Observe that, for any $a \in A$ and $\varphi^* : X' \to D$, if $\varphi^*(x_a) \neq a$ then $f_0^*(\varphi^*) = 0$. Suppose $\varphi' \in \mathbf{V}(\mathcal{I}(\mathcal{P}'))$. As φ' satisfies all the pinning constraints in C', we have $f_0'(\varphi') \neq 0$ if and only if $f_0^*(\varphi') \neq 0$. Moreover, suppose $\varphi^* \in \mathbf{V}(\mathcal{I}(\mathcal{P}^*)$ and $f_0^*(\varphi^*) \neq 0$. This implies that

1. $\prod_{a \in A} \prod_{b \in D \setminus \{a\}} (\varphi^*(x_a) - b) \neq 0$, which means φ^* satisfies all the pinning constraints in C, and hence $\varphi^* \in \mathbf{V}(\mathcal{I}(\mathcal{P}'))$, and
2. $f_0'(\varphi') \neq 0$.

Combining the preprocessing step with the second one there exists $\varphi \in \mathbf{V}(\mathcal{I}(\mathcal{P}))$ such that $f_0(\varphi) \neq 0$ if and only if there exists $\varphi^* \in \mathbf{V}(\mathcal{I}(\mathcal{P}^*)$ such that $f_0^*(\varphi^*) \neq 0$. This completes the proof of the proposition. □

Proposition 3.1.1 together with the fact that IMP(Γ) is a subproblem of IMP(Γ^A) implies a close connection between the complexity of IMP(Γ) and IMP(Γ^A).

Corollary 3.1.2. *For any constraint language Γ on D and any $A \subseteq D$, the problem IMP(Γ^A) is tractable (d-tractable) if and only if IMP(Γ) is tractable (d-tractable), and IMP(Γ^A) is **coNP**-complete if and only if IMP(Γ) is **coNP**-complete.*

Recall that $\Gamma^* = \Gamma^D$. Then

Theorem 3.1.3. *For any Γ over D the problem IMP(Γ^*) is polynomial time reducible to IMP(Γ), and for any d the problem IMP$_d$(Γ^*) is polynomial time reducible to IMP$_{d+|D|(|D|-1)}$(Γ).*

However, Proposition 3.1.1 leaves some room for possible complexity of IMP$_d$(Γ) for small d, less than $|D|(|D|-1)$.

Example 3.1.4. Fix D, $\ell \leq |D|$. Let NEQ_s, $s \leq |D|$ denote the s-ary disequality relation on D given by

$$\mathsf{NEQ}_s = \{(a_1, \dots, a_s) \mid |\{a_1, \dots, a_s\}| = s\}.$$

In particular, CSP(NEQ_2) is equivalent to $|D|$-Coloring. Now let a $(\ell+2)$-ary relation R be defined as follows

$$R = (\mathsf{NEQ}_2 \times \mathsf{NEQ}_\ell) \cup \{(a_1, \dots, a_{2+\ell}) \mid |\{a_1, \dots, a_{2+\ell}\}| < \ell\},$$

and let $\Gamma = \{R\}$. It is easy to see that CSP(Γ) is polynomial time, as assigning the same value to all variables always provides a solution. As we observed in Lemma 2.1.11 this implies that IMP_0 is also easy. Actually, f_0 of degree 0 never belongs to the ideal except $f_0 = 0$.

It can also be shown that for any $A \subseteq D$ with $|A| < \ell$ assigning a constant $a \in A$ to all variables except those bound by the pinning constraints is also a solution of CSP(Γ^A). Therefore, $\mathrm{IMP}_0(\Gamma^A)$ is easy for any such set. On the other hand, if $|A| = \ell$, say, $A = \{a_1, \dots, a_\ell\}$, then CSP($\Gamma^A$) can simulate $|D|$-Coloring by using $R(x, y, a_1, \dots, a_\ell)$. (This will be made more precise in Section 3.1.2.) Therefore, $\mathrm{IMP}_0(\Gamma^A)$ is **coNP**-complete in this case. Clearly that playing with the exact definition of R one can construct a language Γ such that $\mathrm{IMP}_0(\Gamma^A)$ becomes **coNP**-complete for any specified collection of subsets A while remains easy for the rest of the subsets.

In the case of a 2-element D we can show a more definitive result.

Proposition 3.1.5 (see also [161])**.** *Let Γ be a constraint language on the set $\{0, 1\}$. Then*

(1) $\mathrm{IMP}_d(\Gamma^*)$ is polynomial time equivalent to $\mathrm{IMP}_{d+2}(\Gamma)$.

*(2) $\mathrm{IMP}_0(\Gamma)$ is polynomial time [**coNP**-complete] if and only if CSP(Γ) is polynomial time [**NP**-complete].*

*(3) If CSP($\Gamma^{\{0\}}$) or CSP($\Gamma^{\{1\}}$) is **NP**-complete then $\mathrm{IMP}_1(\Gamma)$ is **coNP**-complete.*

Items (1),(3) follow from Proposition 3.1.1 and item (2) follows from Lemma 2.1.11. Moreover, replacing the relation NEQ_2 in Example 3.1.4 with the Not-All-Equal relation, one can construct constraint languages Γ such that the borderline between easiness and hardness in the sequence $\mathrm{IMP}_0(\Gamma), \mathrm{IMP}_1(\Gamma), \mathrm{IMP}_2(\Gamma)$ lies in any desirable place.

Proposition 3.1.1 also provides a connection between the decision version of the IMP and its search version. Since IMP(Γ) is in **coNP**, here by the search IMP we understand the following problem. Let $(f_0, \mathcal{P})$ be an instance of IMP(Γ) such that $f_0 \notin \mathcal{I}(\mathcal{P})$, the problem is to find an assignment $\varphi \in \mathbf{V}(\mathcal{I}(\mathcal{P}))$ such that $f_0(\varphi) \neq 0$.

Corollary 3.1.6. *A decision problem IMP(Γ) is tractable [d-tractable] if and only if the corresponding search problem is tractable [d-tractable].*

Proof. One direction is trivial as the tractability of the search problem implies the tractability of the corresponding decision problem.

For the converse, let Γ be a constraint language over a finite set D such that IMP(Γ) is (d-) tractable. Consider $(f_0, \mathcal{P})$, an instance of IMP(Γ), where $\mathcal{P} = (X, D, C)$ is an instance of CSP(Γ). By the choice of Γ, we can decide in polynomial time whether there exists φ such that $\varphi \in \mathbf{V}(\mathcal{I}(\mathcal{P}))$ but $f_0(\varphi) \neq 0$. Suppose such φ exists and hence $f_0 \notin \mathcal{I}(\mathcal{P})$. Then for each $x \in X$ there must be some $a \in D$, for which in the following instance $(f_0', \mathcal{P}')$ of the IMP we have $f_0' \notin \mathcal{I}(\mathcal{P}')$:

1. define f_0' to be the polynomial obtained from f_0 by substituting a for x.
2. define $\mathcal{P}'$ with $\mathcal{P}' = (X, D, C' = C \cup \{\langle x, \{a\}\rangle\})$.

Checking whether $f_0' \in \mathcal{I}(\mathcal{P}')$ is an instance of IMP(Γ^*) and therefore can be done in polynomial time. Hence, by considering each possible value $a \in D$ we can find a value for x that is a part of $\varphi \in \mathbf{V}(\mathcal{I}(\mathcal{P}))$ such that $f_0(\varphi) \neq 0$. Having found such a value a for x we retain $x = a$ and move to the smaller problem. Repeating the process for each variable in turn we can find a required φ. The algorithm requires solving at most $|X| \cdot |D|$ instances of IMP(Γ^*), each of which can be solved in polynomial time. □

3.1.2 Primitive positive definability

One of the most useful reductions between CSPs is by means of primitive-positive definitions.

Definition 3.1.7 (pp-definability)**.** *Let Γ, Δ be constraint languages on the same set D. We say that Γ pp-defines Δ (or Δ is pp-definable from Γ) if for each relation (predicate) $R \subseteq D^k$ in Δ there exists a first order formula L over variables $\{x_1, \ldots, x_m, x_{m+1}, \ldots, x_{m+k}\}$ that uses predicates from Γ, equality relations, and conjunctions such that*

$$R(x_{m+1}, \ldots, x_{m+k}) = \exists x_1 \ldots \exists x_m L$$

Such an expression is often called a primitive positive (pp-) formula*.*

Mastrolilli showed that there is an analogue of existential quantification on the IMP side.

Definition 3.1.8. *Given $\mathtt{I} = \langle f_1, \ldots, f_s\rangle \subseteq \mathbb{F}[X]$, for $Y \subseteq X$, the Y-elimination ideal $\mathtt{I}_{X\setminus Y}$ is the ideal of $\mathbb{F}[X \setminus Y]$ defined by*

$$\mathtt{I}_{X\setminus Y} = \mathtt{I} \cap \mathbb{F}[X \setminus Y]$$

In other words, $\mathtt{I}_{X\setminus Y}$ consists of all consequences of $f_1 = \cdots = f_s = 0$ that do not depend on variables from Y.

Theorem 3.1.9 ([161]). *Let $\mathcal{P} = (X, D, C)$ be an instance of the CSP(Γ), and let $\mathtt{I}(\mathcal{P})$ be the corresponding ideal. For any $Y \subseteq X$ let $\mathtt{I}_Y$ be the $(X \setminus Y)$-elimination ideal. Then, for any partial solution $\varphi_Y \in \mathbf{V}(\mathtt{I}_Y)$ there exists an extension $\psi : X \setminus Y$ such that $(\varphi, \psi) \in \mathbf{V}(\mathtt{I}(\mathcal{P}))$.*

Let Γ, Δ be constraint languages on the same domain D such that Γ pp-defines Δ. This means that for every relation R from Δ there is a pp-definition in Γ

$$R(x_{m_R+1}, \ldots, x_{m_R+k_R}) = \exists x_1 \ldots \exists x_{m_R} L_R.$$

Suppose $\mathcal{P}_\Delta = (X, D, C)$ is an instance of CSP(Δ). This instance can be converted into an instance $\mathcal{P}_\Gamma = (X', D, C')$ of CSP(Γ), see Theorem 2.16 in [39], in such a way that $X \subseteq X'$ and the instance $\mathcal{P}_\Delta$ has a solution if and only if $\mathcal{P}_\Gamma$ does. Moreover, it can be shown that $\mathcal{P}_\Gamma, \mathcal{P}_\Delta$ satisfy the following condition.

> **The Extension Condition.** Every solution of $\mathcal{P}_\Delta$ can be extended to a solution of $\mathcal{P}_\Gamma$, and, vice versa, the restriction of every solution of $\mathcal{P}_\Gamma$ onto variables from X is a solution of $\mathcal{P}_\Delta$.

As usual, let $\mathit{I}(\mathcal{P}_\Delta)$ be the ideal of $\mathbb{F}[X]$ corresponding to $\mathcal{P}_\Delta$ and $\mathit{I}(\mathcal{P}_\Gamma)$ the ideal of $\mathbb{F}[X']$ corresponding to $\mathcal{P}_\Gamma$. We would like to relate the set of solutions of $\mathcal{P}_\Delta$ to the variety of the $X' \setminus X$-elimination ideal of $\mathit{I}(\mathcal{P}_\Gamma)$. The next lemma states that the variety of the $X' \setminus X$-elimination ideal of $\mathit{I}(\mathcal{P}_\Gamma)$ is equal to the the variety of $\mathit{I}(\mathcal{P}_\Delta)$.

Lemma 3.1.10 ([161], Lemma 6.1, paraphrased). *Let $\mathtt{I}_X = \mathtt{I}(\mathcal{P}_\Gamma) \cap \mathbb{F}[X]$ be the $X' \setminus X$-elimination ideal of $\mathtt{I}(\mathcal{P}_\Gamma)$. Then*

$$\mathbf{V}(\mathtt{I}(\mathcal{P}_\Delta)) = \mathbf{V}(\mathtt{I}_X).$$

We can now prove a reduction for pp-definable constraint languages.

Theorem 3.1.11. *If Γ pp-defines Δ, then IMP(Δ) [IMP$_d$(Δ)] is polynomial time reducible to IMP(Γ) [respectively, to IMP$_d$(Γ)].*

Proof. Let $(f_0, \mathcal{P}_\Delta)$, $\mathcal{P}_\Delta = (X, D, C_\Delta)$, be an instance of IMP($\Delta$) where $X = \{x_{m+1}, \ldots, x_{m+k}\}$, $f_0 \in \mathbb{F}[x_{m+1}, \ldots, x_{m+k}]$, $k = |X|$, and m will be defined later, and $\mathit{I}(\mathcal{P}_\Delta) \subseteq \mathbb{F}[x_{m+1}, \ldots, x_{m+k}]$. From this we construct an instance $(f_0', \mathcal{P}_\Gamma)$ of IMP(Γ) where $f_0' \in \mathbb{F}[x_1, \ldots, x_{m+k}]$ and $\mathit{I}(\mathcal{P}_\Gamma) \subseteq \mathbb{F}[x_1, \ldots, x_{m+k}]$ such that $f_0 \in \mathit{I}(\mathcal{P}_\Delta)$ if and only if $f_0' \in \mathit{I}(\mathcal{P}_\Gamma)$.

Using pp-definitions of relations from Δ we convert the instance $\mathcal{P}_\Delta$ into an instance $\mathcal{P}_\Gamma = (\{x_1, \ldots, x_{m+k}\}, D, C_\Gamma)$ of CSP(Γ) such that every solution of $\mathcal{P}_\Delta, \mathcal{P}_\Gamma$ satisfy the Extension Condition above. Such an instance $\mathcal{P}_\Gamma$ can be constructed in polynomial time as follows.

By the assumption each $S \in \Delta$, say, t_S-ary, is pp-definable in Γ by a pp-formula involving relations from Γ and the equality relation, $=_D$. Thus,

$$S(y_{q_S+1}, \ldots, y_{q_S+t_S}) = \exists y_1, \ldots, y_{q_S}(R_1(w_1^1, \ldots, w_{l_1}^1) \wedge \cdots \wedge R_r(w_1^r, \ldots, w_{l_r}^r)),$$

where $w_1^1, \ldots, w_{l_1}^1, \ldots, w_1^k, \ldots, w_{l_k}^k \in \{y_1, \ldots, y_{q_S+t_S}\}$ and $R_1, \ldots, R_r \subseteq \Gamma \cup \{=_D\}$.

Now, for every constraint $B = \langle \mathbf{s}, S \rangle \in C_\Delta$, where $\mathbf{s} = (x_{i_1}, \ldots, x_{i_t})$ create a fresh copy of $\{y_1, \ldots, y_{q_S}\}$ denoted by Y_B, and add the following constraints to C_Γ

$$\langle (w_1^1, \ldots, w_{l_1}^1), R_1 \rangle, \ldots, \langle (w_1^r, \ldots, w_{l_r}^r), R_r \rangle.$$

We then set $m = \sum_{B \in C} |Y_B|$ and assume that $\cup_{B \in C} Y_B = \{x_1, \ldots, x_m\}$. Note that the problem instance obtained by this procedure belongs to $\mathrm{CSP}(\Gamma \cup \{=_D\})$. All constraints of the form $\langle (x_i, x_j), =_D \rangle$ can be eliminated by replacing all occurrences of the variable x_i with x_j. Moreover, it can be checked (see also Theorem 2.16 in [39]) that $\mathcal{P}_\Delta, \mathcal{P}_\Gamma$ satisfy the Extension Condition.

Let $\mathcal{I}(\mathcal{P}_\Gamma) \subseteq \mathbb{F}[x_1, \ldots, x_{m+k}]$ be the ideal corresponding to $\mathcal{P}_\Gamma$ and set $f_0' = f_0$. Since $f_0 \in \mathbb{F}[x_{m+1}, \ldots, x_{m+k}]$ we also have $f_0 \in \mathbb{F}[x_1, \ldots, x_{m+k}]$. Hence, $(f_0, \mathcal{P}_\Gamma)$ is an instance of $\mathrm{IMP}(\Gamma)$. We prove that $f_0 \in \mathcal{I}(\mathcal{P}_\Delta)$ if and only if $f_0 \in \mathcal{I}(\mathcal{P}_\Gamma)$.

Suppose $f_0 \notin \mathcal{I}(\mathcal{P}_\Delta)$, this means there exists $\varphi \in \mathbf{V}(\mathcal{I}(\mathcal{P}_\Delta))$ such that $f(\varphi) \neq 0$. By Theorem 3.1.9, φ can be extended to a point $\varphi' \in \mathbf{V}(\mathcal{I}(\mathcal{P}_\Gamma))$. This in turn implies that $f_0 \notin \mathcal{I}(\mathcal{P}_\Gamma)$. Conversely, suppose $f_0 \notin \mathcal{I}(\mathcal{P}_\Gamma)$. Hence, there exists $\varphi' \in \mathbf{V}(\mathcal{I}(\mathcal{P}_\Gamma))$ such that $f_0(\varphi') \neq 0$. Projection of φ' to its last k coordinates gives a point $\varphi \in \mathbf{V}(\mathcal{I}_X)$. By Lemma 3.1.10, $\varphi \in \mathbf{V}(\mathcal{I}(\mathcal{P}_\Delta))$ which implies $f_0 \notin \mathcal{I}(\mathcal{P}_\Delta)$. □

Remark 3.1.12. *The smallest set of all relations pp-defined from a constraint language $\Gamma \subseteq \mathbf{R}_A$ is called the* relational clone *of Γ, denoted by $\langle \Gamma \rangle$. Hence, as a corollary to Theorem 3.1.11, for a finite set of relations Γ, IMP(Γ) is tractable [d-tractable] if and only if IMP(Δ) is tractable [d-tractable] for any finite $\Delta \subseteq \langle \Gamma \rangle$. Similarly, IMP($\Gamma$) is **coNP**-complete if and only if IMP(Δ) is **coNP**-complete for some finite $\Delta \subseteq \langle \Gamma \rangle$.*

3.1.3 Primitive positive interpretability

Pp-definability is a useful technique that tells us what additional relations can be added to a constraint language without changing the complexity of the corresponding problem class, and provides a tool for comparing different languages on the same domain. Next, we discuss a more powerful tool that can be used to compare the complexity of the IMP for languages over different domains.

Definition 3.1.13 (pp-interpretability)**.** *Let Γ, Δ be constraint languages over finite domains D, E, respectively, and Δ is finite. We say that Γ pp-interprets Δ if there exists a*

natural number ℓ, a set $F \subseteq D^\ell$, and an onto mapping $\pi : F \to E$ such that Γ pp-defines the following relations

1. *the relation F,*
2. *the π-preimage of the equality relation on E, and*
3. *the π-preimage of every relation in Δ,*

where by the π-preimage of a k-ary relation S on E we mean the ℓk-ary relation $\pi^{-1}(S)$ on D defined by

$$\pi^{-1}(S)(x_{11}, \ldots, x_{1k}, x_{21}, \ldots, x_{2k}, \ldots, x_{\ell 1}, \ldots, x_{\ell k}) \qquad \textit{is true}$$

if and only if

$$S(\pi(x_{11}, \ldots, x_{\ell 1}), \ldots, \pi(x_{1k}, \ldots, x_{\ell k})) \qquad \textit{is true.}$$

Example 3.1.14. Suppose $D = \{0, 1\}$ and $E = \{0, 1, 2\}$ and define relations $R_D = \{(0, 0), (0, 1), (1, 1)\}$ and $R_E = \{(0, 0), (0, 1), (0, 2), (1, 1), (1, 2), (2, 2)\}$. Note that relations R_D, R_E are orders $0 \leq 1$ and $0 \leq 1 \leq 2$ on D, E, respectively. Set $\Gamma = \{R_D\}$ and $\Delta = \{R_E\}$.

Let $n = 2$ and define $F = \{(0, 0), (0, 1), (1, 1)\} \subseteq D^2$. The language Γ pp-defines F i.e. $F = \{(x, y) \mid x \leq y \text{ and } x, y \in \{0, 1\}\}$. Now define mapping $\pi : F \to E$ as follows $\pi((0, 0)) = 0, \pi((0, 1)) = 1, \pi((1, 1)) = 2$. The π-preimage of the relation R_E is the relation

$$R_F = \{(0, 0, 0, 0), (0, 0, 0, 1), (0, 0, 1, 1), (0, 1, 0, 1), (0, 1, 1, 1), (1, 1, 1, 1)\}.$$

The language Γ pp-defines $\Gamma' = \{R_F\}$ through the following pp-formula

$$R_F = \{(x_1, x_2, y_1, y_2) \mid (x_1 \leq y_1) \wedge (x_2 \leq y_2) \wedge (x_1 \leq x_2) \wedge (y_1 \leq y_2), \text{ and } (x_1, x_2, y_1, y_2) \in \{0, 1\}^4\}.$$

Consider instance $(\{x, y, z\}, E, C)$ of CSP(Δ) where the set of constraints is $C = \{\langle (x, y), R_E\rangle, \langle (y, z), R_E\rangle\}$. This basically means the requirements $(x \leq y) \wedge (y \leq z)$. This instance is equivalent to the following instance of CSP(Γ'):

$$\langle (x_1, x_2, y_1, y_2), R_F\rangle \wedge \langle (y_1, y_2, z_1, z_2), R_F\rangle \tag{3.1}$$

As was pointed out, Γ pp-defines F as well as R_F. Hence, we define $(\{x_1, x_2, y_1, y_2, z_1, z_2\}, D, C')$, an instance of CSP($\Gamma$), with the constraints

$$\langle (x_1, x_2), R_D\rangle \wedge \langle (y_1, y_2), R_D\rangle \wedge \langle (z_1, z_2), R_D\rangle \tag{3.2}$$

$$\wedge \langle (x_1, y_1), R_D\rangle \wedge \langle (x_2, y_2), R_D\rangle \tag{3.3}$$

$$\wedge \langle (y_1, z_1), R_D\rangle \wedge \langle (y_2, z_2), R_D\rangle \tag{3.4}$$

Note that (3.2) is a pp-definition of relation F and forces $(x_1x_2), (y_1y_2), (z_1z_2) \in F$. Moreover, equations (3.3) and (3.4) are equivalent to

$$(x_1 \leq y_1) \wedge (x_2 \leq y_2) \wedge (y_1 \leq z_1) \wedge (y_2 \leq z_2) \tag{3.5}$$

Applying the mapping π, every solution of the instance $(\{x_1, x_2, y_1, y_2, z_1, z_2\}, D, C')$ can be transformed to a solution of instance $(\{x, y, z\}, E, C)$ and back.

One can interpolate mapping π in Definition 3.1.13 by a polynomial of low degree. It is a known fact that given $N+1$ distinct $\mathbf{x}_0, \ldots, \mathbf{x}_N \in \mathbb{R}^\ell$ and corresponding values $y_0, \ldots, y_N$, there exists a polynomial p of degree at most ℓN that interpolates the data i.e. $p(\mathbf{x}_j) = y_j$ for each $j \in \{0, \ldots, N\}$ (such a polynomial can be obtained by a straightforward generalization of the Lagrange interpolating polynomial, see, e.g., [185]). Hence, we can interpolate the mapping π by a polynomial of total degree at most $\ell|E|$.

Theorem 3.1.15. *Let Γ, Δ be constraint languages on sets D, E, respectively, and let Γ pp-interprets Δ. Then $IMP_d(\Delta)$ is polynomial time reducible to $IMP_{\ell|E|}(\Gamma)$.*

Proof. Let $(f_0, \mathcal{P}_\Delta)$ be an instance of $\mathrm{IMP}_d(\Delta)$ where $f_0 \in \mathbb{F}[x_1, \ldots, x_n]$, $\mathcal{P}_\Delta = (\{x_1, \ldots, x_n\}, E, C_\Delta)$, an instance of $\mathrm{CSP}(\Delta)$, and $\mathcal{I}(\mathcal{P}_\Delta) \subseteq \mathbb{F}[x_1, \ldots, x_n]$.

The properties of the mapping π from Definition 3.1.13 allow us to rewrite an instance of $\mathrm{CSP}(\Delta)$ to an instance of $\mathrm{CSP}(\Gamma')$ over the constraint language Γ'. Recall that, by Definition 3.1.13, Γ' contains all the ℓk-ary relations $\pi^{-1}(S)$ on D where $S \in \Delta$ is k-ary relation.

Note that Γ' is pp-definable from Γ. By Theorem 3.1.11, $\mathrm{IMP}(\Gamma')$ is reducible to $\mathrm{IMP}(\Gamma)$. It remains to show $\mathrm{IMP}_d(\Delta)$ is reducible to $\mathrm{IMP}_d(\Gamma')$. To do so, from instance $(f_0, \mathcal{P}_\Delta)$ of $\mathrm{IMP}_d(\Delta)$ we construct an instance $(f_0', \mathcal{P}_{\Gamma'})$ of $\mathrm{IMP}_d(\Gamma')$ such that $f_0 \in \mathcal{I}(\mathcal{P}_\Delta)$ if and only if $f_0' \in \mathcal{I}(\mathcal{P}_{\Gamma'})$.

Let p be a polynomial of total degree at most $\ell|E|$ that interpolates mapping π. For every $f_0 \in \mathbb{F}[x_1, \ldots, x_n]$, let $f_0' \in \mathbb{F}[x_{11}, \ldots, x_{\ell 1}, \ldots, x_{1n}, \ldots, x_{\ell n}]$ be the polynomial that is obtained from f_0 by replacing each indeterminate x_i with $p(x_{1i}, \ldots, x_{\ell i})$. Clearly, for any assignment $\varphi : \{x_1, \ldots, x_n\} \to E$, $f_0(\varphi) = 0$ if and only if $f_0'(\psi) = 0$ for every $\psi : \{x_{11}, \ldots, x_{\ell n}\} \to D$ such that

$$\varphi(x_i) = \pi(\psi(x_{1i}), \ldots, \psi(x_{\ell i}))$$

for every $i \leq n$. Moreover, for any such φ, ψ it holds $\varphi \in \mathbf{V}(\mathcal{I}(\mathcal{P}_\Delta))$ if and only if $\psi \in \mathbf{V}(\mathcal{I}(\mathcal{P}_{\Gamma'}))$. This yields that

$$(\exists \varphi \in \mathbf{V}(\mathcal{I}(\mathcal{P}_\Delta)) \wedge f_0(\varphi) \neq 0) \iff (\exists \psi \in \mathbf{V}((\mathcal{P}_{\Gamma'})) \wedge f_0'(\psi) \neq 0)$$

Note that the condition that f_0 has bounded degree is important here, because otherwise f'_0 may have exponentially more monomials than f_0. This completes the proof of the theorem. □

3.2 Polymorphisms and algebras

3.2.1 Polymorphisms and a necessary condition for tractability

Sets of relations closed under pp-definitions allow for a succinct representation through polymorphisms. Let R be a k-ary relation on a set D and ψ an n-ary operation on the same set. Operation ψ is said to be a *polymorphism* of R if for any $\mathbf{a}^1, \dots, \mathbf{a}^n \in R$ the tuple $\psi(\mathbf{a}^1, \dots, \mathbf{a}^n)$ belongs to R. Here by $\psi(\mathbf{a}^1, \dots, \mathbf{a}^n)$ we understand the component-wise action of ψ, that is, if $\mathbf{a}^i = (a_1^i, \dots, a_k^i)$ then

$$\psi(\mathbf{a}^1, \dots, \mathbf{a}^n) = (\psi(a_1^1, \dots, a_1^n), \dots, \psi(a_k^1, \dots, a_k^n)).$$

For more background on polymorphisms, their properties, and links to the CSP the reader is referred to a relatively recent survey [18]. Most of the standard results we use below can be found in this survey.

A polymorphism of a constraint language Γ is an operation that is a polymorphism of every relation in Γ. The set of all polymorphisms of the language Γ is denoted by $\mathsf{Pol}(\Gamma)$. For a set Ψ of operations by $\mathsf{Inv}(\Psi)$ we denote the set of relations R such that every operation from Ψ is a polymorphism of R. The operators Pol and Inv induces so called *Galois correspondence* between sets of operations and constraint languages. There is a rich theory of this correspondence, however, for the sake of this thesis we only need one fact.

Theorem 3.2.1. *Let Γ, Δ be constraint languages on a finite set D. Then $\mathsf{Pol}(\Gamma) \subseteq \mathsf{Pol}(\Delta)$ if and only if Γ pp-defines Δ. In particular, $\mathsf{Inv}(\mathsf{Pol}(\Gamma))$ is the set of all relations pp-definable in Γ.*

Combining Theorem 3.2.1 and Theorem 3.1.11, polymorphisms of constraint languages provide reductions between IMPs.

Corollary 3.2.2. *Let Γ, Δ be constraint languages on a finite set D and Δ finite. If $\mathsf{Pol}(\Gamma) \subseteq \mathsf{Pol}(\Delta)$ then $\mathrm{IMP}(\Delta)$ [$\mathrm{IMP}_d(\Delta)$] is polynomial time reducible to $\mathrm{IMP}(\Gamma)$ [$\mathrm{IMP}_d(\Gamma)$].*

Corollary 3.2.2 amounts to saying that similar to $\mathrm{CSP}(\Gamma)$ the complexity of $\mathrm{IMP}(\Gamma)$ is determined by the polymorphisms of Γ.

Next we use the known necessary condition for CSP tractability [39] to obtain some necessary conditions for tractability of $\mathrm{IMP}(\Gamma)$.

A *projection* is an operation $\psi : D^k \to D$ such that there is $i \in [k]$ with $\psi(x_1, \dots, x_k) = x_i$ for any $x_1, \dots, x_k \in D$. If the only polymorphisms of a constraint language are projections, every relation is pp-definable in Γ implying that $\mathrm{IMP}(\Gamma)$ is **coNP**-complete.

Theorem 3.1.3 is another ingredient for our necessary condition. Recall that for a language Γ by Γ^* we denote the language with added *constant relations* R_a for all $a \in D$. It is known that every polymorphism ψ of all the constant relations is *idempotent*, that is, satisfies the condition $\psi(x, \dots, x) = x$. Therefore, by Theorem 3.1.3 it suffices to focus on idempotent polymorphisms.

Proposition 3.2.3. *Let Γ be a constraint language over a finite set D. If the only idempotent polymorphisms of Γ are projections then $IMP_{|D|(|D|-1)}(\Gamma)$ is **coNP**-complete.*

Example 3.2.4. Consider the relation $N = \{0,1\}^3 \setminus \{(0,0,0),(1,1,1)\}$. This relation corresponds to Not-All-Equal Satisfiability problem and it is known that the idempotent operations from $\mathsf{Pol}(\{N\})$ are projections [186]. $\mathrm{CSP}(\{N\})$ was shown to be **NP**-complete by Schaefer [196] and $\mathrm{IMP}(\{N\})$ is shown to be **coNP**-complete in [161].

3.2.2 Algebras and a better necessary condition

In this section we briefly review the basics of structural properties of (universal) algebra in application to the IMP. Universal algebras have been instrumental in the study of CSPs, and, although we do not go deeper into this theory in this thesis, we expect they should be useful for IMPs as well. We follow textbooks [45, 165] and texts on the algebraic theory of the CSP, see, e.g., [15, 16, 39, 37].

Algebras. An *algebra* is a pair $\mathcal{D} = (D, \Psi)$ where D is a set (always finite in this thesis) and Ψ is a set of operations on D (perhaps multi-ary). The operations from Ψ are called *basic*, and any operation that can be obtained from operations in Ψ by means of composition is called a *term* operation. The set of all term operations will be denoted by $\mathsf{Term}(\mathcal{D})$. For example, Ψ can be the set $\mathsf{Pol}(\Gamma)$ for some constraint language Γ on D, in which case $\mathcal{D}$ is called the *algebra of polymorphisms* of Γ and will be denoted $\mathsf{Alg}(\Gamma)$. Thus, $\mathsf{Alg}(\Gamma) = (D, \mathsf{Pol}(\Gamma))$.

By Corollary 3.2.2 the algebra $\mathsf{Alg}(\Gamma)$ determines the complexity of $\mathrm{IMP}(\Gamma)$ and $\mathrm{IMP}_d(\Gamma)$ for sufficiently large d. The advantage of using algebras rather than just polymorphisms is that it unlocks a variety of structural methods that cannot be easily applied if we only use polymorphisms. Algebra $\mathcal{D} = (D, \Psi)$ is said to be *tractable* [*d-tractable*] if for any finite constraint language Γ such that $\Psi \subseteq \mathsf{Pol}(\Gamma)$ the problem $\mathrm{IMP}(\Gamma)$ is tractable [d-tractable]. Algebra $\mathcal{D}$ is said to be ***coNP**-complete* if for some finite language Γ with $\Psi \subseteq \mathsf{Pol}(\Gamma)$ the problem $\mathrm{IMP}(\Gamma)$ is **coNP**-complete. In the rest of this section apart from another necessary condition of tractability we prove several results that deduce the tractability [d-tractability, **coNP**-completeness] of a certain algebra derivative from $\mathcal{D}$ from a similar property of $\mathcal{D}$.

Idempotent algebras. The first step will be to reduce the kind of algebras we have to study. By Theorem 3.1.3 idempotent polymorphisms determine the complexity of $\mathrm{IMP}(\Gamma)$.

On the algebraic side, an algebra $\mathcal{D}$ is said to be *idempotent* if each of its basic operations (and therefore each of its term operations) is idempotent. Every algebra $\mathcal{D} = (D, \Psi)$ can be converted into an idempotent algebra simply by throwing out all the non-idempotent term operations. Let $\mathsf{Term}_{id}(\mathcal{D})$ denote the set of all idempotent operations from $\mathsf{Term}(\mathcal{D})$. Then the *full idempotent reduct* of $\mathcal{D} = (D, \Psi)$ is the algebra $\mathsf{Id}(\mathcal{D}) = (D, \mathsf{Term}_{id}(\mathcal{D}))$.

Proposition 3.2.5. *For any finite algebra $\mathcal{D} = (D, \Psi)$, $\mathcal{D}$ is tractable [d-tractable] if and only if $\mathsf{Id}(\mathcal{D})$ is tractable [d-tractable]. Also $\mathsf{Id}(\mathcal{D})$ is **coNP**-complete if and only if $\mathcal{D}$ is **coNP**-complete.*

Proof. Note that an operation ψ on a set D is idempotent if and only if it preserves all the relations in the set $\Gamma_{\mathrm{CON}} = \{R_a \mid a \in D\}$, consisting of all constant relations R_a on D. Hence, $\mathsf{Inv}(\mathsf{Term}_{id}(\mathcal{D}))$ is the relational clone generated by $\mathsf{Inv}(\Psi) \cup \Gamma_{\mathrm{CON}}$, or, in other words, every relation R such that $\mathsf{Term}_{id}(\mathcal{D}) \subseteq \mathsf{Pol}(R)$ is pp-definable in $\mathsf{Inv}(\Psi) \cup \Gamma_{\mathrm{CON}}$.

Let Δ be a finite set from $\mathsf{Inv}(\mathsf{Term}_{id}(\mathcal{D}))$. By the observation above there is a finite $\Gamma \subseteq \mathsf{Inv}(\Psi) \cup \Gamma_{\mathrm{CON}}$ such that Γ pp-defines Δ. By Theorem 3.1.3 for any d the problem $\mathrm{IMP}_d(\Delta)$ can be reduced in polynomial time to $\mathrm{IMP}_{d+|D|(|D|-1)}(\Gamma)$, and the result follows. □

Subalgebras, homomorphisms, and direct powers. The following standard algebraic constructions have been very useful in the study of the CSP.

Definition 3.2.6. *Let $\mathcal{D} = (D, \Psi)$ be an algebra.*

- **(Subalgebra)** *Let $E \subseteq D$ such that, for any $\psi \in \Psi$ and for any $b_1, \dots, b_k \in E$, where k is the arity of ψ, we have $\psi(b_1, \dots, b_k) \in E$. In other words, ψ is a polymorphism of E or $E \in \mathsf{Inv}(\Psi)$. The algebra $\mathcal{E} = (E, \Psi|_E)$, where $\Psi|_E$ consists of the restrictions of all operations in Ψ to E, is called a* subalgebra *of $\mathcal{D}$.*

- **(Direct power)** *For a natural number k the k-th* direct power *$\mathcal{D}^k$ of $\mathcal{D}$ is the algebra $\mathcal{D}^k = (D^k, \Psi^k)$, where Ψ^k consists of all the operations from Ψ acting on D^k component-wise (see the definition of polymorphism).*

- **(Homomorphic image)** *Let E be a set and $\chi : D \to E$ a mapping such that for any (say, k-ary) $\psi \in \Psi$ and any $a_1, \dots, a_k, b_1, \dots, b_k \in D$, if $\chi(a_i) = \chi(b_i)$, $i \in [k]$, then $\chi(\psi(a_1, \dots, a_k)) = \chi(\psi(b_1, \dots, b_k))$. The algebra $\mathcal{E} = (E, \Psi_\chi)$ is called a* homomorphic image *of $\mathcal{D}$, where for every $\psi \in \Psi$ the set Ψ_χ contains $\psi/_\chi$ given by $\psi/_\chi(c_1, \dots, c_k) = \chi(\psi(a_1, \dots, a_k))$ and $a_1, \dots, a_k \in D$ are such that $c_i = \chi(a_i)$, $i \in [k]$.*

If an algebra is the algebra of polymorphisms of some constraint language, the concepts above are related to pp-definitions and pp-interpretations. We will use the following easy observation.

Lemma 3.2.7. *Let $\mathcal{D} = (D, \Psi) = \mathsf{Alg}(\Gamma)$ for a constraint language Γ over D.*

- *If $\mathcal{E} = (E, \Psi|_E)$ is a subalgebra of $\mathcal{D}$ then E is pp-definable in Γ.*
- *Every relation from $\mathsf{Inv}(\Psi^k)$ is pp-interpretable in Γ.*
- *Let $\mathcal{E} = (E, \Psi_\chi)$ be a homomorphic image of $\mathcal{D}$. Then every relation from $\mathsf{Inv}(\Psi_\chi)$ is pp-interpretable in Γ.*

The standard algebraic constructions also include direct products of different algebras. Direct products also have a strong connection to the CSP and therefore IMP. However, they require a more general framework, multi-sorted CSPs and IMPs. These are beyond the scope of this thesis.

We are now ready to prove the reductions induced by subalgebras, direct powers, and homomorphic images.

Theorem 3.2.8. *Let $\mathcal{D}$ be an algebra and $\mathcal{E}$ its subalgebra [direct power, homomorphic image]. If $\mathcal{D}$ is d-tractable, then so is $\mathcal{E}$. If $\mathcal{E}$ is **coNP**-complete, then $\mathcal{D}$ is also **coNP**-complete. Moreover, if $\mathcal{E}$ is a subalgebra of $\mathcal{D}$, then $\mathcal{E}$ is tractable whenever $\mathcal{D}$ is.*

Proof. The theorem is almost straightforward from Lemma 3.2.7 and Theorems 3.1.11, 3.1.15. Let $\mathcal{D} = (D, \Psi)$, $\mathcal{E} = (E, \Psi')$ and $\Delta \subseteq \mathsf{Inv}(\Psi')$, a finite set. Note that $\mathcal{D} = \mathsf{Alg}(\Gamma)$ for $\Gamma = \mathsf{Inv}(\Psi)$.

If $\mathcal{E}$ is a subalgebra of $\mathcal{D}$, that is, $\Psi' = \Psi|_E$ then by Lemma 3.2.7 E is pp-definable in Γ and therefore $\Delta \subseteq \mathsf{Inv}(\Psi|_E) \subseteq \mathsf{Inv}(\Psi)$. The result follows.

If $\mathcal{E}$ is a direct power, say, $\mathcal{E} = (D^k, \Psi^k)$, then since every relation from $\mathsf{Inv}(\Psi^k)$ is pp-interpretable in Γ, there is a finite set $\Gamma' \subseteq \Gamma$ that pp-interprets Δ. Then by Theorem 3.1.15 $\mathrm{IMP}_d(\Delta)$ can be reduced to $\mathrm{IMP}_d(\Gamma')$ in polynomial time. The result follows.

In the case when $\mathcal{E}$ is a homomorphic image of Γ, the proof is identical to the previous case due to Lemma 3.2.7. □

Stronger necessary condition for tractability. Subalgebras, direct powers, and homomorphic images allow us to state a stronger condition for tractability of constraint languages. In the case of the CSP, when a constraint language Γ contains all the constant relations, $\mathrm{CSP}(\Gamma)$ is **NP**-complete if and only if $\mathsf{Alg}(\Gamma)$ has a homomorphic image $\mathcal{D}$ of a subalgebra such that all the term operations of $\mathcal{D}$ are projections. Using Theorem 3.1.3 we can make this condition even stronger (although only necessary).

Theorem 3.2.9. *Let Γ be a constraint language with the property that there exists a homomorphic image $\mathcal{E}$ of a subalgebra of a direct power of $\mathsf{Id}(\mathsf{Alg}(\Gamma))$ such that all the term operations of $\mathcal{E}$ are projections. Then $IMP_{|D|(|D|-1)}(\Gamma)$ is **coNP**-complete.*

Proof. Let $\mathcal{E} = (E, \Psi)$. Since the term operations of $\mathcal{E}$ are only projections, by Proposition 3.2.3 there is a finite set Δ such that $\mathrm{IMP}_0(\Delta)$ is **coNP**-complete. Then, by Lemma 3.2.7, Γ^* pp-interprets Δ, and we obtain the result by Theorems 3.1.3 and 3.1.15. □

3.3 Multi-sorted CSPs and IMP

3.3.1 Multi-sorted problems

In most theoretical studies of the CSP all variables are assumed to have the same domain, this type of CSPs are known as *one-sorted* CSPs. However, for various purposes, mainly for more involved algorithms such as in [35, 222] one might consider CSPs where different variables of a CSP have different domains, this type of CSPs are known as *multi-sorted* CSPs [38]. We study this notion in the context of the IMP and provide a reduction for multi-sorted languages that are pp-interpretable. This in particular is useful in this thesis as it provides a reduction between languages that are invariant under an affine polymorphism over an arbitrary Abelian group and languages over several cyclic p-groups. Definitions below are from [38].

Definition 3.3.1. *For any finite collection of finite domains $\mathcal{D} = \{D_t \mid t \in T\}$, and any list of indices $(t_1, t_2, \ldots, t_m) \in T^m$, a subset R of $D_{t_1} \times D_{t_2} \times \cdots \times D_{t_m}$, together with the list $(t_1, t_2, \ldots, t_m)$, is called a* multi-sorted *relation over $\mathcal{D}$ with arity m and signature $(t_1, t_2, \ldots, t_m)$. For any such relation R, the signature of R is denoted $\sigma(R)$.*

As an example consider $\mathcal{D} = \{D_1, D_2\}$ with $D_1 = \{0, 1\}$, $D_2 = \{0, 1, 2\}$. Then $\mathbb{Z}_6$, which is the direct sum of $\mathbb{Z}_2$ and $\mathbb{Z}_3$, $\mathbb{Z}_2 \oplus \mathbb{Z}_3$, can be viewed as a multi-sorted relation over $\mathcal{D}$ of arity 2 with signature $(1, 2)$.

Similar to regular one-sorted CSPs, given any set of multi-sorted relations, we can define a corresponding class of multi-sorted CSPs. Let Γ be a set of multi-sorted relations over a collection of sets $\mathcal{D} = \{D_t \mid t \in T\}$. The multi-sorted constraint satisfaction problem over Γ, denoted MCSP(Γ), is defined to be the decision problem with instance $\mathcal{P} = (X, \mathcal{D}, \delta, \mathcal{C})$, where X is a finite set of variables, $\delta : X \to T$, and $\mathcal{C}$ is a set of constraints where each constraint $C \in \mathcal{C}$ is a pair $\langle \mathbf{s}, R \rangle$, such that

- $\mathbf{s} = (x_1, \ldots, x_{m_C})$ is a tuple of variables of length m_C, called the constraint scope;
- R is an element of Γ with arity m_C and signature $(\delta(x_1), \ldots, \delta(x_{m_c}))$, called the constraint relation.

The goal is to decide whether or not there exists a solution, i.e. a mapping $\varphi : X \to \cup_{D \in \mathcal{D}} D$, with $\varphi(x) \in D_{\delta(x)}$, satisfying all of the constraints. We will use $\mathsf{Sol}(\mathcal{P})$ to denote the (possibly empty) set of solutions of the instance $\mathcal{P}$.

The multi-sorted IMP, that is, IMP(Γ) for a multi-sorted constraint language Γ is largely defined in the same way as the regular one. The ideal corresponding to an instance $\mathcal{P}$ of MCSP(Γ) is constructed similar to the one-sorted case, the only difference is that for an instance $\mathcal{P} = (X, \mathcal{D}, \delta, \mathcal{C})$ the corresponding ideal $\mathcal{I}(\mathcal{P})$ contains domain polynomials $\prod_{a \in D_{\delta(x_i)}} (x_i - a)$ for each variable x_i. As with one-sorted CSPs, the IMP associated with a multi-sorted constraint language Γ over a set $\mathcal{D}$ is the problem IMP(Γ) in which the input

is a pair $(f, \mathcal{P})$ where $\mathcal{P} = (X, \mathcal{D}, \delta, \mathcal{C})$ is a MCSP(Γ) instance and f is a polynomial from $\mathbb{F}[X]$. The goal is to decide whether f lies in the ideal $\mathcal{I}(\mathcal{P})$. We use $\mathrm{IMP}_d(\Gamma)$ to denote IMP(Γ) when the input polynomial f has degree at most d.

3.3.2 Multi-sorted languages, pp-definability and interpretability

Here we introduce the definition of pp-definitions and the more powerful construction, pp-interpretations, in the multi-sorted case, and prove that, similar to the one-sorted case, they give rise to reductions between IMPs.

Definition 3.3.2 (pp-definability)**.** *Let Γ be a multi-sorted constraint language on a collection of sets $\mathcal{D} = \{D_t \mid t \in T\}$. A* primitive-positive (pp-) formula *in the language Γ is a first order formula L over variables X that uses predicates from Γ, equality relations, existential quantifier, and conjunctions, and satisfies the condition:*

> *if $R_1(x_1, \dots, x_k), R_2(y_1, \dots, y_\ell)$ are atomic formulas in L with signatures σ_1, σ_2 and such that x_i, y_j is the same variable, then $\sigma_1(i) = \sigma_2(j)$.*

The condition above determines the signature $\sigma : X \to T$ of L.

Let Δ be another multi-sorted language over $\mathcal{D}$. We say that Γ pp-defines Δ (or Δ is pp-definable from Γ) if for each (k-ary) relation (predicate) $R \in \Delta$ there exists a pp-formula L over variables $\{x_1, \dots, x_m, x_{m+1}, \dots, x_{m+k}\}$ such that

$$R(x_{m+1}, \dots, x_{m+k}) = \exists x_1 \dots \exists x_m L,$$

and if σ, σ' are the signatures of L and R, respectively, then $\sigma' = \sigma_{|\{m+1,\dots,m+k\}}$.

An analog of the following result for one-sorted CSPs is proved in Section 3.1.2.

Theorem 3.3.3. *If multi-sorted constraint language Γ pp-defines multi-sorted constraint language Δ, then IMP(Δ) [$\mathrm{IMP}_d(\Delta)$] is polynomial time reducible to IMP(Γ) [respectively, to $\mathrm{IMP}_d(\Gamma)$].*

The proof of Theorem 3.3.3 is very similar to the proof of Theorem 3.1.11 and we defer the proof to Section 3.3.4.

Multi-sorted pp-interpretations are also similar to the one-sorted case, but require a bit more care.

Definition 3.3.4 (pp-interpretability)**.** *Let Γ, Δ be multi-sorted constraint languages over finite collections of sets $\mathcal{D} = \{D_t \mid t \in T\}, \mathcal{E} = \{E_s \mid s \in S\}$, respectively, and Δ is finite. We say that Γ pp-interprets Δ if for every $s \in S$ there exist $i_{s,1}, \dots, i_{s,\ell_s} \in T$, a set $F_s \subseteq D_{i_{s,1}} \times \dots \times D_{i_{s,\ell_s}}$, and an onto mapping $\pi_s : F_s \to E_s$ such that Γ pp-defines the following relations*

1. *the relations F_s, $s \in S$,*

2. the π_s-preimage of the equality relations on E_s, $s \in S$, and

3. the π-preimage of every relation in Δ,

where by the π-preimage of a k-ary relation $Q \subseteq E_{s_1} \times \cdots \times E_{s_k}$ over $\mathcal{E}$ we mean the m-ary relation $\pi^{-1}(Q)$ over $\mathcal{D}$, with $m = \sum_{i=1}^{k} \ell_{s_i}$, defined by

$$\pi^{-1}(Q)(x_{1,1}, \ldots, x_{1,\ell_{s_1}}, x_{2,1}, \ldots, x_{2,\ell_{s_2}}, \ldots, x_{k,1}, \ldots, x_{k,\ell_{s_k}}) \qquad \textit{is true}$$

if and only if

$$Q(\pi_{s_1}(x_{1,1}, \ldots, x_{1,\ell_{s_1}}), \ldots, \pi_{s_k}(x_{k,1}, \ldots, x_{k,\ell_{s_k}})) \qquad \textit{is true.}$$

Example 3.3.5. Suppose $\mathcal{D} = \{\mathbb{Z}_2, \mathbb{Z}_3\}$ and $\mathcal{E} = \{\mathbb{Z}_6\}$. Now, any relation on $\mathcal{E}$ is pp-interpretable in a language in $\mathcal{D}$ via $F = \mathbb{Z}_2 \times \mathbb{Z}_3$ and $\pi : F \to \mathbb{Z}_6$ as

$$\begin{array}{lll} \pi(0,0) = 0 & \pi(1,2) = 1 & \pi(0,1) = 2 \\ \pi(1,0) = 3 & \pi(0,2) = 4 & \pi(1,1) = 5. \end{array}$$

Example 3.3.6. Here we present a very simple example of pp-interpretation that will be useful later. Suppose $\mathcal{D} = \{D = \mathbb{Z}_2\}$ and $\mathcal{E} = \{E_1, E_2\}$ with $E_1 = E_2 = \mathbb{Z}_2 \times \mathbb{Z}_2$. Define relations $R_D = \{(0,0), (1,1)\}$ and

$$R_E = \begin{pmatrix} (0,0) & (1,0) & (0,1) & (1,1) \\ (0,0) & (0,1) & (1,0) & (1,1) \end{pmatrix} \begin{array}{l} \leftarrow x \\ \leftarrow y \end{array}$$

Note that the relation R_E contains all pairs $(x, y) \in E_1 \times E_2$ with $x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} y$ and the relation R_D is the equality. Set $\Gamma = \{R_D\}$ and $\Delta = \{R_E\}$.

Set $F = D \times D$ and define mapping $\pi : F \to \mathcal{E}$ as follows $\pi(x_1, x_2) = (x_1, x_2)$. The π-preimage of the relation R_E is the relation

$$R_F = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix} \begin{array}{l} \leftarrow x_1 \\ \leftarrow x_2 \\ \leftarrow y_1 \\ \leftarrow y_2 \end{array}$$

The language Γ pp-defines $\Gamma' = \{R_F\}$ through the following pp-formula

$$R_F = \{(x_1, x_2, y_1, y_2) \mid (x_1 = y_2) \wedge (x_2 = y_1), \text{ and } (x_1, x_2, y_1, y_2) \in \{0,1\}^4\}.$$

Consider instance $\mathcal{P} = (\{x, y, z\}, \mathcal{E}, \delta, C)$ of MCSP(Δ) where the set of constraints is

$$C = \{\langle (x, y), R_E \rangle, \langle (y, z), R_E \rangle\}$$

and δ maps x to 1 and y, z to 2. This basically means the requirements $x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} y$ and $y = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} z$. This instance is equivalent to the following instance $\mathcal{P}'$ of CSP(Γ'):

$$\langle (x_1, x_2, y_1, y_2), R_F \rangle \wedge \langle (y_1, y_2, z_1, z_2), R_F \rangle$$

Applying the mapping π, every solution of the instance $\mathcal{P}'$ can be transformed to a solution of instance $\mathcal{P}$ and back. This in turn is equivalent to the following instance of CSP(Γ)

$$\langle (x_1, y_2), R_D \rangle \wedge \langle (x_2, y_1), R_D \rangle \wedge \langle (y_1, z_2), R_D \rangle \wedge \langle (x_2, z_1), R_D \rangle.$$

As in the one-sorted case, pp-interpretations give rise to reductions between IMPs.

Theorem 3.3.7. *Let Γ, Δ be multi-sorted constraint languages over collections of sets $\mathcal{D} = \{D_t \mid t \in T\}, \mathcal{E} = \{E_s \mid s \in S\}$, respectively, and let Γ pp-interprets Δ. Then IMP$_d(\Delta)$ is polynomial time reducible to IMP$_{O(d)}(\Gamma)$.*

Again, the proof of Theorem 3.3.7 is similar to that of Theorem 3.1.15 and is moved to Section 3.3.5.

3.3.3 Multi-sorted polymorphisms

Polymorphisms provide a link between constraint languages and relations pp-definable in those languages.

Proposition 3.3.8 ([28, 84])**.** *Let Γ be a constraint language on set A and R a relation on the same set. The relation R is pp-definable in Γ if and only if $\mathsf{Pol}(\Gamma) \subseteq \mathsf{Pol}(R)$.*

Corollary 3.3.9 ([121, 43])**.** *Let Γ, Δ be constraint languages on a set D, Δ finite, and $\mathsf{Pol}(\Gamma) \subseteq \mathsf{Pol}(\Delta)$. Then CSP($\Delta$) is polynomial time reducible to CSP(Γ), and IMP$_d(\Delta)$ is polynomial time reducible to IMP$_d(\Gamma)$, for any d.*

We will need a version of polymorphisms adapted to multi-sorted relations. Let $\mathcal{D} = \{D_t \mid t \in T\}$ be a collection of sets. A multi-sorted operation on $\mathcal{D}$ is a *functional symbol* f with associated *arity* k along with an interpretation f^{D_t} of f on every set $D_t \in \mathcal{D}$, which is a k-ary operation on D_t. A multi-sorted operation f is said to be a *(multi-sorted) polymorphism* of a multi-sorted relation $R \subseteq D_{t_1} \times \cdots \times D_{t_n}$, $t_1, \ldots, t_n \in T$, if for any $\mathbf{a}_1, \ldots, \mathbf{a}_k \in R$ the tuple

$$f(\mathbf{a}_1, \ldots, \mathbf{a}_k) = (f^{D_{t_1}}(a_{1,1}, \ldots, a_{1,k}), \ldots, f^{D_{t_n}}(a_{n,1}, \ldots, a_{n,k}))$$

belongs to R.

Example 3.3.10. Note that for the sake of defining a multi-sorted operation, the collection $\mathcal{D}$ does not have to be finite. Let $\mathcal{A}$ be the class of all finite Abelian groups and f a ternary functional symbol that is interpreted as the affine operation $f^{\mathbb{A}}(x, y, z) = x - y + z$ on every $\mathbb{A} \in \mathcal{A}$, where $+, -$ are operations of $\mathbb{A}$.

Consider the multi-sorted binary relation $R \subseteq \mathbb{Z}_2 \times \mathbb{Z}_4$ over $\mathcal{D} = \{\mathbb{Z}_2, \mathbb{Z}_4\}$ given by

$$R = \{(0, 1), (0, 3), (1, 0), (1, 2)\}.$$

It is straightforward to verify that f is a polymorphism of R. For instance,

$$f\left(\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\2\end{pmatrix}\right) = \begin{pmatrix}0-1+1\\1-0+2\end{pmatrix} = \begin{pmatrix}0\\3\end{pmatrix} \in R.$$

To make sure f is a polymorphism of R we of course have to check every combination of pairs from R.

The connection between multi-sorted polymorphisms and pp-definitions is more complicated than that in the one-sorted case [38], and we do not need it here. However, we will need the following well known fact.

Lemma 3.3.11. *Let $R \subseteq D_1 \times \cdots \times D_n$, f a k-ary polymorphism of R, and let $f_1, \ldots, f_k$ be m-ary polymorphisms of R. Then the* composition *of f and $f_1, \ldots, f_k$ given by*

$$g^{D_i}(x_1, \ldots, x_m) = f^{D_i}(f_1^{D_i}(x_1, \ldots, x_m), \ldots, f_k^{D_i}(x_1, \ldots, x_m))$$

for $i \in [n]$ is a polymorphism of R.

3.3.4 Proof of Theorem 3.3.3

In this section we provide a proof for Theorem 3.3.3. Our proof is a slight modification of the one given for one-sorted CSPs. In order to establish the result, we first analyze the relationship between pp-definability and the notion of elimination ideal from algebraic geometry. This has been explored in case of one-sorted CSPs in [41, 161]. Here we establish a relationship in the case of multi-sorted CSPs. The proofs and ideas used here are almost identical to the ones in [161].

Theorem 3.3.12. *Let $\mathcal{P} = (X, \mathcal{D}, \delta, C)$ be an instance of the MCSP(Γ), and let $\mathtt{I}(\mathcal{P})$ be its corresponding ideal. For any $Y \subseteq X$ let $\mathtt{I}_Y$ be the $(X \setminus Y)$-elimination ideal. Then, for any partial solution $\varphi_Y \in \mathbf{V}(\mathtt{I}_Y)$ there exists an extension $\psi : X \setminus Y \to \cup_{t \in T} D_t$ such that $(\varphi, \psi) \in \mathbf{V}(\mathtt{I}(\mathcal{P}))$.*

Proof. Suppose $\mathcal{D} = \{D_t \mid t \in T\}$, $\delta : X \to T$ and set $\mathcal{I} = \mathcal{I}(\mathcal{P})$. We may assume $\mathbf{V}(\mathcal{I}) \neq \emptyset$. This is because if $\mathbf{V}(\mathcal{I}) = \emptyset$ then $1 \in \mathcal{I}$ which implies $1 \in \mathcal{I}_Y$. If the latter holds then the claim is vacuously true. We proceed by assuming $\mathbf{V}(\mathcal{I}) \neq \emptyset$ and $\mathbf{V}(\mathcal{I}_Y) \neq \emptyset$.

The proof is by contradiction. Suppose there exists $\mathbf{a} = (a_1, \ldots, a_m) \in \mathbf{V}(\mathcal{I}_Y)$, with $m = |Y|$, that does not extend to a feasible solution from $\mathbf{V}(\mathcal{I})$. Assume $Y = \{y_1, \ldots, y_m\} \subseteq X$, and define the polynomial

$$q(y_1, \ldots, y_m) = \prod_{i=1}^{m} \prod_{j \in D_{\delta(y_i)} \setminus \{a_i\}} (y_i - j).$$

Observe that $q(a_1, \ldots, a_m) \neq 0$ however for every $\mathbf{b} = (b_1, \ldots, b_m)$ that can be extended to a feasible solution of $\mathbf{V}(\mathcal{I})$ we have $q(\mathbf{b}) = 0$. This implies

$$q(y_1, \ldots, y_m) \in \mathbf{I}(\mathbf{V}(\mathcal{I})) \cap \mathbb{F}[y_1, \ldots, y_m] = \mathcal{I} \cap \mathbb{F}[y_1, \ldots, y_m] = \mathcal{I}_Y$$

where the first equality follows from the Strong Nullstellensat and $\mathcal{I}$ being radical. Having $q(y_1, \ldots, y_m) \in \mathcal{I}_Y$ implies that $\mathbf{a} = (a_1, \ldots, a_m) \notin \mathbf{V}(\mathcal{I}_Y)$, a contradiction. □

Let Δ and Γ be multi-sorted constraint languages over $\mathcal{D}$ where Γ pp-defines Δ. That is for each (k-ary) relation (predicate) $R \in \Delta$ there exists a pp-formula L over variables $\{x_1, \ldots, x_m, x_{m+1}, \ldots, x_{m+k}\}$ such that

$$R(x_{m+1}, \ldots, x_{m+k}) = \exists x_1 \ldots \exists x_m L, \tag{3.6}$$

and if σ, σ' are the signatures of L and R, respectively, then $\sigma' = \sigma_{|\{m+1,\ldots,m+k\}}$. Let $S = \mathsf{Sol}(L)$ be the set of satisfying assignments for L and let $\mathbf{I}(S)$ be its corresponding vanishing ideal. Note that $\mathbf{I}(S) \subseteq \mathbb{F}[x_1, \ldots, x_{m+k}]$.

Lemma 3.3.13. *$R = \boldsymbol{V}(\mathtt{I}_X)$ where $\mathtt{I}_X = \mathbf{I}(S) \cap \mathbb{F}[x_{m+1}, \ldots, x_m]$.*

Proof. Define the mapping $\pi_X : \mathbb{F}^{m+k} \to \mathbb{F}^k$ to be the projection

$$\pi(a_1, \ldots, a_{m+k}) = (a_{m+1}, \ldots, a_{m+k}).$$

If we apply π_X to S we get $\pi_X(S) \subseteq \mathbb{F}^k$. Now it is easy to see $\pi(S) \subseteq \mathbf{V}(\mathcal{I}_X)$. This is because every polynomial in $\mathcal{I}_X$ vanishes on all the points in $\pi(S)$. Provided that $\pi(S) \subseteq \mathbf{V}(\mathcal{I}_X)$ we can write R as follows

$$R = \pi_X(S) = \{(a_{m+1}, \ldots, a_{m+k}) \in \mathbf{V}(\mathcal{I}_X) \mid \exists a_1, \ldots, a_m \in \mathbb{F} \; s.t. \; (a_1, \ldots, a_{m+k}) \in S\}$$

This is exactly the set of points in $\mathcal{I}_X$ that can be extended to a solution in S. However, by Theorem 3.3.12 all the points in $\mathbf{V}(\mathcal{I}_X)$ can be extended to a solution in S. This means $R = \mathbf{V}(\mathcal{I}_X)$ as desired. □

We now have all the required ingredients to prove Theorem 3.3.3.

Proof of Theorem 3.3.3. Let $(f, \mathcal{P}_\Delta)$, $\mathcal{P}_\Delta = (X, \mathcal{D}, \delta_\Delta, C_\Delta)$, be an instance of IMP($\Delta$) where $X = \{x_{m+1}, \ldots, x_{m+k}\}$, $f \in \mathbb{F}[x_{m+1}, \ldots, x_{m+k}]$, $k = |X|$, and m will be defined later, and $\mathcal{I}(\mathcal{P}_\Delta) \subseteq \mathbb{F}[x_{m+1}, \ldots, x_{m+k}]$. From this we construct an instance $(f', \mathcal{P}_\Gamma)$ of IMP(Γ) where $f' \in \mathbb{F}[x_1, \ldots, x_{m+k}]$ and $\mathcal{I}(\mathcal{P}_\Gamma) \subseteq \mathbb{F}[x_1, \ldots, x_{m+k}]$ such that $f \in \mathcal{I}(\mathcal{P}_\Delta)$ if and only if $f' \in \mathcal{I}(\mathcal{P}_\Gamma)$.

From $\mathcal{P}_\Delta$ we construct an instance $\mathcal{P}_\Gamma = (\{x_1, \ldots, x_{m+k}\}, \mathcal{D}, , \delta_\Gamma, C_\Gamma)$ of CSP(Γ) as follows. By the assumption each $S \in \Delta$, say, t_S-ary, is pp-definable in Γ. Thus,

$$S(y_{q_S+1}, \ldots, y_{q_S+t_S}) = \exists y_1, \ldots, y_{q_S} \overbrace{(R_1(w_1^1, \ldots, w_{l_1}^1) \wedge \cdots \wedge R_r(w_1^r, \ldots, w_{l_r}^r))}^{L},$$

where $w_1^1, \ldots, w_{l_1}^1, \ldots, w_1^k, \ldots, w_{l_k}^k \in \{y_1, \ldots, y_{q_S+t_S}\}$ and $R_1, \ldots, R_r \subseteq \Gamma \cup \{=_\mathcal{D}\}$. Moreover, for σ and σ_S, the signatures of L and S respectively, we have $\sigma_S = \sigma_{|\{q_S+1,\ldots,q_S+t_S\}}$. Now, for every constraint $B = \langle \mathbf{s}, S \rangle \in C_\Delta$, where $\mathbf{s} = (x_{i_1}, \ldots, x_{i_t})$ create a fresh copy of $\{y_1, \ldots, y_{q_S}\}$ denoted by Y_B, and add the following constraints to C_Γ

$$\langle (w_1^1, \ldots, w_{l_1}^1), R_1 \rangle, \ldots, \langle (w_1^r, \ldots, w_{l_r}^r), R_r \rangle.$$

where for each R_i, R_j we have if $w_k^i, w_{k'}^j$ are the same variable, then $\sigma_{R_i}(w_k^i) = \sigma_{R_j}(w_{k'}^j)$. Set $m = \sum_{B \in C} |Y_B|$ and assume that $\cup_{B \in C} Y_B = \{x_1, \ldots, x_m\}$. All constraints of the form $\langle (x_i, x_j), =_\mathcal{D} \rangle$ can be eliminated by replacing all occurrences of the variable x_i with x_j.

Let $\mathcal{I}(\mathcal{P}_\Gamma) \subseteq \mathbb{F}[x_1, \ldots, x_{m+k}]$ be the ideal corresponding to $\mathcal{P}_\Gamma$ and set $f' = f$. Since $f \in \mathbb{F}[x_{m+1}, \ldots, x_{m+k}]$ we also have $f \in \mathbb{F}[x_1, \ldots, x_{m+k}]$. Hence, $(f, \mathcal{P}_\Gamma)$ is an instance of IMP(Γ). We prove that $f \in \mathcal{I}(\mathcal{P}_\Delta)$ if and only if $f \in \mathcal{I}(\mathcal{P}_\Gamma)$.

Suppose $f \notin \mathcal{I}(\mathcal{P}_\Delta)$, this means there exists $\varphi \in \mathbf{V}(\mathcal{I}(\mathcal{P}_\Delta))$ such that $f(\varphi) \neq 0$. By Theorem 3.3.12, φ can be extended to a point $\varphi' \in \mathbf{V}(\mathcal{I}(\mathcal{P}_\Gamma))$. This in turn implies that $f \notin \mathcal{I}(\mathcal{P}_\Gamma)$. Conversely, suppose $f \notin \mathcal{I}(\mathcal{P}_\Gamma)$. Hence, there exists $\varphi' \in \mathbf{V}(\mathcal{I}(\mathcal{P}_\Gamma))$ such that $f(\varphi') \neq 0$. Projection of φ' to its last k coordinates gives a point $\varphi \in \mathbf{V}(\mathcal{I}_X)$. By Lemma 3.3.13, $\varphi \in \mathbf{V}(\mathcal{I}(\mathcal{P}_\Delta))$ which implies $f \notin \mathcal{I}(\mathcal{P}_\Delta)$. □

3.3.5 Proof of Theorem 3.3.7

In this section we provide a proof for Theorem 3.3.7.

Proof of Theorem 3.3.7. Recall that Γ, Δ are multi-sorted constraint languages over collections of sets $\mathcal{D} = \{D_t \mid t \in T\}, \mathcal{E} = \{E_s \mid s \in S\}$, respectively, and Γ pp-interprets Δ.

Let $(f, \mathcal{P}_\Delta)$ be an instance of IMP$_d$(Δ) where $f \in \mathbb{F}[x_1, \ldots, x_n]$, $\mathcal{P}_\Delta = (\{x_1, \ldots, x_n\}, \mathcal{E}$, $\delta_\Delta, C_\Delta)$, an instance of CSP(Δ), and $\mathcal{I}(\mathcal{P}_\Delta) \subseteq \mathbb{F}[x_1, \ldots, x_n]$.

As Γ pp-interprets Δ then for every $s \in S$ there exist $i_{s,1}, \ldots, i_{s,\ell_s} \in T$, a set $F_s \subseteq D_{i_{s,1}} \times \cdots \times D_{i_{s,\ell_s}}$, and an onto mapping $\pi_s : F_s \to E_s$ such that Γ pp-defines the following relations. By Definition 3.3.4, according to the properties of $\pi_1, \ldots, \pi_{|S|}$ we can rewrite an instance of CSP(Δ) to an instance of CSP(Γ') over the constraint language Γ'. Γ' contains

1. the relations F_s, $s \in S$,
2. the π_s-preimage of the equality relations on E_s, $s \in S$, and
3. the π-preimage of every relation in Δ.

Note that if $\delta_\Delta(x) = s$ then $\delta_{\Gamma'}(x_{s,j}) = i_{s,j}$, for all $1 \leq j \leq \ell_s$. Moreover, note that Γ' is pp-definable from Γ. By Theorem 3.3.3, IMP(Γ') is reducible to IMP(Γ). It remains to show $\mathrm{IMP}_d(\Delta)$ is reducible to $\mathrm{IMP}_d(\Gamma')$. To do so, from instance $(f, \mathcal{P}_\Delta)$ of $\mathrm{IMP}_d(\Delta)$ we construct an instance $(f', \mathcal{P}_{\Gamma'})$ of $\mathrm{IMP}_d(\Gamma')$ such that $f \in \mathcal{I}(\mathcal{P}_\Delta)$ if and only if $f' \in \mathcal{I}(\mathcal{P}_{\Gamma'})$.

Let p_s be a polynomial of total degree at most $|E_s|(|D_{i_{s,1}}| + \cdots + |D_{i_{s,\ell_s}}|)$ that interpolates mapping π_s, $s \in S$. For every $f \in \mathbb{F}[x_1, \ldots, x_n]$, let

$$f' \in \mathbb{F}[x_{1,1}, \ldots, x_{1,\ell_{s_1}}, x_{2,1}, \ldots, x_{2,\ell_{s_2}}, \ldots, x_{k,1}, \ldots, x_{k,\ell_{s_k}}]$$

be the polynomial that is obtained from f by replacing each indeterminate x_i, with $\delta_\Delta(x_i) = s$, by $p_s(x_{s,1}, \ldots, x_{s,\ell_s})$.

Now, for any assignment $\varphi : \{x_1, \ldots, x_n\} \to \cup_{s \in S} E_s$, with the condition $\varphi(x_i) \in E_{\delta_\Delta(x_i)}$ for all x_i, $f(\varphi) = 0$ if and only if $f'(\psi) = 0$ for every

$$\psi : \{x_{1,1}, \ldots, x_{1,\ell_{s_1}}, x_{2,1}, \ldots, x_{2,\ell_{s_2}}, \ldots, x_{k,1}, \ldots, x_{k,\ell_{s_k}}\} \to \cup_{t \in T} D_t$$

such that for each x_i with $\delta_\Delta(x_i) = s$ we have

$$\varphi(x_i) = \pi(\psi(x_{s,1}), \ldots, \psi(x_{s,\ell_s}))$$

Moreover, for any such φ, ψ it holds $\varphi \in \mathbf{V}(\mathcal{I}(\mathcal{P}_\Delta))$ if and only if $\psi \in \mathbf{V}(\mathcal{I}(\mathcal{P}_{\Gamma'}))$. This yields that

$$(\exists \varphi \in \mathbf{V}(\mathcal{I}(\mathcal{P}_\Delta)) \wedge f(\varphi) \neq 0) \iff (\exists \psi \in \mathbf{V}((\mathcal{P}_{\Gamma'})) \wedge f'(\psi) \neq 0)$$

Note that the condition that f has bounded degree is important here, because otherwise f' may have exponentially more monomials than f. This completes the proof of the theorem. □

Chapter 4

Sufficient conditions for tractability of IMP

4.1 The dual-discriminator

Here we deal with a *majority* operation. Over the Boolean domain there is only one majority operation, called the *dual-discriminator*. In the Boolean case, Mastrolilli [161] proved that the IMP(Γ) is tractable when the constraint language Γ is closed under the dual-discriminator operation. Later, Bharathi and Mastrolilli [24] expand this tractability result to constraint languages over the ternary domain. We establish tractability result for any finite domain. We point out that constraint languages closed under the dual-discriminator operation are also known as 0/1/all constraints. We start off with the definition of a majority polymorphism and explain an appealing structure of majority closed relations.

Definition 4.1.1. *Let μ be a 3-ary operation from D^3 to D. If for all $x, y \in D$ we have $\mu(x, x, y) = \mu(x, y, x) = \mu(y, x, x) = x$, then μ is called a* majority operation.

For a (n-ary) relation R and $T \subseteq [n]$ by $\mathrm{pr}_T R$ we denote the *projection* of R onto T, that is, the set of tuples $(a_i)_{i \in T}$ such that there is $(b_1, \ldots, b_n) \in R$ with $b_i = a_i$ for each $i \in T$.

Proposition 4.1.2 ([121])**.** *Let R be a relation of arity n with a majority polymorphism, and let $C = \langle S, R \rangle$ constraining the variables in S with relation R. For any problem $\mathcal{P}$ with constraint C, the problem $\mathcal{P}'$ which is obtained by replacing C by the set of constraints*

$$\{((S[i], S[j]), \mathrm{pr}_{i,j}(R)) \mid 1 \leq i \leq j \leq n\}$$

has exactly the same solutions as $\mathcal{P}$.

The above proposition suggests that, without loss of generality, we may assume that all the constraints are binary when a constraint language has a majority polymorphism μ. Let Γ be a language over a set D such that $\mu \in \mathsf{Pol}(\Gamma)$, and let $\mathcal{P} = (X, D, C)$ be an instance

of CSP(Γ). We assume each constraint in C is binary. That is

$$C = \{C_{ij} = \langle (x_i, x_j), R_{ij} \rangle \mid R_{ij} \subseteq D_i \times D_j \text{ where } D_i, D_j \subseteq D\}.$$

Relations closed under the dual-discriminator operation admit a much nicer structure that has been characterized, see [205]. Indeed, such a characterization states that constraints can only be of three types. Let us first define the dual-discriminator operation before formulating the characterization. The dual-discriminator operation is defined as follows.

$$\nabla(x, y, z) = \begin{cases} y & \text{if } y = z, \\ x & \text{otherwise.} \end{cases}$$

Lemma 4.1.3 ([58, 205])**.** *Suppose $\nabla \in \mathsf{Pol}(\Gamma)$. Then each constraint $C_{ij} = \langle (x_i, x_j), R_{ij} \rangle$ is one of the following three types.*

1. *A complete constraint: $R_{ij} = D_i \times D_j$ for some $D_i, D_j \subseteq D$,*
2. *A permutation constraint: $R_{ij} = \{(a, \pi(a)) \mid a \in D_i\}$ for some $D_i \subseteq D$ and some bijection $\pi : D_i \to D_j$, where $D_j \subseteq D$,*
3. *A two-fan constraint: $R_{ij} = \{(\{a\} \times D_j) \cup (D_i \times \{b\})\}$ for some $D_i, D_j \subseteq D$ and $a \in D_i, b \in D_j$.*

We introduce two preprocessing steps that we apply to a an instance of a CSP before solving the IMP. Let Γ be a constraint language such that $\nabla \in \mathsf{Pol}(\Gamma)$ and $\mathcal{P} = (X, D, C)$ be an instance of CSP(Γ). The first step is standard in the CSP research and is referred to as establishing certain local consistency, [5, 18]. In the case of dual discriminator we need $(2, 3)$-consistency. This procedure works as follows. First of all we set up a binary constraint $C_{uv} = \langle (u, v), R_{uv} \rangle$ for every pair u, v of variables from X. Initially, if there is a constraint on u, v, we set C_{uv} to be that constraint; for the remaining pairs we set $R_{uv} = D^2$. Then iterate the following until further improvements are impossible: pick $u, v, w \in X$ and for every $(a, b) \in R_{uv}$ check whether there is $c \in D$ such that $(a, c) \in R_{uw}$ and $(b, c) \in R_{vw}$. If such a c does not exist, remove (a, b) from R_{uv}. The resulting instance $\mathcal{P}' = (V, D, C')$ satisfies the following conditions. Every solution of $\mathcal{P}$ is a solution of $\mathcal{P}'$, ∇ is a polymorphism of every constraint relation from $\mathcal{P}'$, and it is impossible to combine a limited number of constraints to produce a constraint on u, v that is tighter than C_{uv}.

The second preprocessing step handles permutation constraints and transform the instance into an instance without any permutation constraints. Such a transformation simplifies the problem and yields to an efficient computation of a Gröbner Basis. Let $\mathcal{P} = (X, D, C)$ be an instance of CSP(Γ) such that $\nabla \in \mathsf{Pol}(\Gamma)$. We proceed by assuming $\mathcal{P}$ is $(2, 3)$-consistence. Suppose the set of constraints C contains a permutation constraint

$C_{ij} = \langle (x_i, x_j), R_{ij} \rangle$ with $R_{ij} = \{(a, \pi(a)) \mid a \in D_i, \pi(a) \in D_j\}$. Define instance $\mathcal{P}' = (X \setminus \{x_j\}, D, C')$ as follows.

1. $C'_{st} = C_{st}$ if $s \neq j$ and $t \neq j$,
2. replace each constraint $C_{sj} = \langle (x_s, x_j), R_{sj} \rangle$ by $C'_{si} = \langle (x_s, x_i), R'_{si} \rangle$ where $R'_{si} = \{(a, \pi^{-1}(b)) \mid (a, b) \in R_{sj}\}$,
3. replace each constraint $C_{js} = \langle (x_j, x_s), R_{js} \rangle$ by $C'_{is} = \langle (x_i, x_s), R'_{is} \rangle$ where $R'_{is} = \{(\pi^{-1}(a), b) \mid (a, b) \in R_{js}\}$.

Note that instance $\mathcal{P}$ has a solution if and only if instance $\mathcal{P}'$ has a solution. The next lemma states the above preprocessing step does not change the complexity of the IMP. In the next lemma we use a polynomial interpolation of permutation π. The permutation π can be interpolated by a polynomial $p : \mathbb{R} \to \mathbb{R}$ such that $p(a) = \pi(a)$ for all $a \in D_i$.

Lemma 4.1.4. *Let $\mathtt{I}(\mathcal{P})$ and $\mathtt{I}(\mathcal{P}')$ be the corresponding ideals to instances $\mathcal{P}$ and $\mathcal{P}'$, respectively. Given a polynomial f_0, define polynomial f'_0 to be the polynomial obtained from f_0 by replacing every occurrence of x_j with $p(x_i)$. Then $f_0 \in \mathtt{I}(\mathcal{P})$ if and only if $f'_0 \in \mathtt{I}(\mathcal{P}')$.*

Proof. Note that, by our construction, there is a one-to-one correspondence between the points in $\mathbf{V}(\mathcal{I}(\mathcal{P}))$ and the points in $\mathbf{V}(\mathcal{I}(\mathcal{P}'))$. That is, each $\mathbf{a} = (a_1, \ldots, a_i, \ldots, a_j = p(a_i), \ldots, a_n) \in \mathbf{V}(\mathcal{I}(\mathcal{P}))$ corresponds to $\mathbf{a}' = (a_1, \ldots, a_{j-1}, a_{j+1}, \ldots, a_n) \in \mathbf{V}(\mathcal{I}(\mathcal{P}'))$. Furthermore, for all $\mathbf{a} \in \mathbf{V}(\mathcal{I}(\mathcal{P}))$ and the corresponding $\mathbf{a}' \in \mathbf{V}(\mathcal{I}(\mathcal{P}'))$ we have $f_0(\mathbf{a}) = f'_0(\mathbf{a}')$. This yields that

$$(\exists \mathbf{a} \in \mathbf{V}(\mathcal{I}(\mathcal{P})) \wedge f_0(\mathbf{a}) \neq 0) \iff (\exists \mathbf{a}' \in \mathbf{V}(\mathcal{I}(\mathcal{P}')) \wedge f'_0(\mathbf{a}') \neq 0)$$

This finishes the proof of the lemma. □

The preprocessing steps and Lemma 4.1.4 suggest that we can massage any instance of IMP(Γ) and obtain an instance IMP(Γ) without permutation constraints. This can be carried out in polynomial time by processing permutation constraints one by one in turn, provided the original polynomial has bounded degree, as otherwise the number of monomials in the resulting polynomial may be exponentially greater than that of the original one.

Lemma 4.1.5. *Let $\mathcal{P} = (X, D, C)$ be an instance of CSP(Γ) such that $\nabla \in \mathsf{Pol}(\Gamma)$ and C contains no permutation constraint. Then a Gröbner Basis of the corresponding ideal $\mathtt{I}(\mathcal{P})$ can be computed in polynomial time.*

Proof. The satisfiability of the instance $\mathcal{P}$ can be decided in polynomial time [58] so we may assume that $1 \notin \mathcal{I}(\mathcal{P})$, else $G = \{1\}$ is a Gröbner Basis. Moreover, we assume for every $x_i, x_j, x_k \in X$ if $(a, b) \in R_{ij}$ then there exists c such that $(a, c) \in R_{ik}$ and $(c, b) \in R_{kj}$. This is because of the $(2, 3)$-consistency. Note that such an assumption does not change $\mathsf{Sol}(\mathcal{P})$.

Equivalently, it does not change $\mathbf{V}(\mathcal{I}(\mathcal{P}))$ which, by Theorem 2.1.9, means the ideal $\mathcal{I}(\mathcal{P})$ does not change. This arc consistency assumption has a consequence in terms of polynomials which enables us to prove our set of polynomials is indeed a Gröbner Basis.

First, let us give a set G of polynomials that represent binary constraints. Initially, G contains all the domain polynomials i.e. $G = \{\prod_{a\in D}(x_i - a) \mid x_i \in X\}$. We proceed as follows.

i) To each complete constraint $C_{ij} = \langle (x_i, x_j), R_{ij} = D_i \times D_j\rangle$ we associate two polynomials $g_i = \prod_{a\in D_i}(x_i - a)$ and $g_j = \prod_{b\in D_j}(x_j - b)$ and replace $\prod_{a\in D}(x_i - a)$ by g_i, and replace $\prod_{a\in D}(x_j - a)$ by g_j.

ii) To each two-fan constraint $C_{ij} = \langle (x_i, x_j), R_{i,j} = \{(\{a\} \times D_j) \cup (D_i \times \{b\})\}\rangle$ we associate polynomial $g_{ij} = (x_i - a)(x_j - b)$ and set $G = G \cup \{g_{ij}\}$. Furthermore, we replace domain polynomial $\prod_{a\in D}(x_i - a)$ by $\prod_{a\in D_i}(x_i - a)$, and replace domain polynomial $\prod_{a\in D}(x_j - a)$ by $\prod_{a\in D_j}(x_j - a)$.

Observe that, as a consequence of arc consistency, for every two polynomials $f = (x_i - a)(x_j - b)$ and $g = (x_i - c)(x_k - d)$ in G with $a \neq c$ polynomial $h = (x_j - b)(x_k - d)$ is also in G.

Consider grlex order with $x_1 \succ_{\mathsf{lex}} \cdots \succ_{\mathsf{lex}} x_n$. Now, we prove G is a Gröbner Basis with respect to the grlex by showing for any two polynomials $f, g \in G$ we have $S(f, g) \to_G 0$. We dismiss the cases where $\mathrm{LM}(f)$ and $\mathrm{LM}(g)$ are relatively prime. In these cases, by Proposition 2.1.21, we have $S(f, g) \to_G 0$. Hence, we focus on the cases where $\mathrm{LM}(f)$ and $\mathrm{LM}(g)$ are not relatively prime.

1. Suppose $f = (x_i - a)(x_j - b)$ and $g = (x_i - a)(x_k - d)$. Then

$$S(f, g) = x_k \cdot f - x_j \cdot g = d \cdot x_i \cdot x_j - b \cdot x_i \cdot x_k + a \cdot b \cdot x_k - a \cdot d \cdot x_j = d \cdot f - b \cdot g.$$

   Observe that $\mathrm{multideg}(S(f, g)) \succeq \mathrm{multideg}(d \cdot f)$ and $\mathrm{multideg}(S(f, g)) \succeq \mathrm{multideg}(b \cdot g)$. Hence, by Definition 2.1.14, we have $S(f, g) \to_{\{f,g\}} 0$.

2. Suppose $f = (x_i - a)(x_j - b)$ and $g = (x_i - c)(x_k - d)$ where $a \neq c$. Then $S(f, g) = x_k \cdot f - x_j \cdot g$ and $S(f, g) \to_{\{f,g\}} (c - a)(x_j - b)(x_k - d)$. However, $h = (x_j - b)(x_k - d)$ is in G and hence $S(f, g) \to_G 0$.

3. Suppose $f = \prod_{a\in D_i}(x_i - a)$ and $g = (x_i - c)(x_j - b)$ where $c \in D_i \subseteq D$. Observe that c must be in D_i due to the $(2, 3)$-consistency preprocessing. Define $f_1 = \prod_{a\in D_i\setminus\{c\}}(x_i - a)$

and $g_1 = (x_j - b)$. Hence, $f = (x_i - c) \cdot f_1$ and $g = (x_i - c) \cdot g_1$.

$$\begin{aligned}
S(f,g) &= x_j \cdot f - (x_i^{|D_i|-1}) \cdot g \\
&= [(x_j - b) + b] \cdot f - \left[\prod_{a \in D_i \setminus \{c\}} (x_i - a) - \left(\prod_{a \in D_i \setminus \{c\}} (x_i - a) - x_i^{|A_i|-1} \right) \right] \cdot g \\
&= (x_j - b) \cdot f - (b) \cdot f - \prod_{a \in D_i \setminus \{c\}} (x_i - a) \cdot g + \left(\prod_{a \in D_i \setminus \{c\}} (x_i - a) - x_i^{|D_i|-1} \right) \cdot g \\
&= (-b) \cdot f + \left(\prod_{a \in D_i \setminus \{c\}} (x_i - a) - x_i^{|D_i|-1} \right) \cdot g \\
&= (g_1 - \mathrm{LT}(g_1)) \cdot f + (f_1 - \mathrm{LT}(f_1)) \cdot g
\end{aligned}$$

We show that $\mathrm{multideg}(S(f,g)) \succeq \mathrm{multideg}((\mathrm{LT}(g_1) - g_1) \cdot f)$ and $\mathrm{multideg}(S(f,g)) \succeq \mathrm{multideg}((f_1 - \mathrm{LT}(f_1)) \cdot g)$. This follows by showing $\mathrm{LM}((\mathrm{LT}(g_1) - g_1) \cdot f) \neq \mathrm{LM}((f_1 - \mathrm{LT}(f_1)) \cdot g)$. By contradiction, if $\mathrm{LM}((\mathrm{LT}(g_1) - g_1) \cdot f) = \mathrm{LM}((f_1 - \mathrm{LT}(f_1)) \cdot g)$ then

$$\mathrm{LM}(\mathrm{LT}(g_1) - g_1) \cdot \mathrm{LM}(f) = \mathrm{LM}(f_1 - \mathrm{LT}(f_1)) \cdot \mathrm{LM}(g) \implies x_i^{|A_i|} = x_i^{|A_i|-2} \cdot x_i x_j$$

The latter is impossible, hence $\mathrm{multideg}(S(f,g)) \succeq \mathrm{multideg}((\mathrm{LT}(g_1) - g_1) \cdot f)$ and $\mathrm{multideg}(S(f,g)) \succeq \mathrm{multideg}((f_1 - \mathrm{LT}(f_1)) \cdot g)$. Therefore, we have $S(f,g) \to_{\{f,g\}} 0$ (recall Definition 2.1.14).

We have shown for any two polynomials $f, g \in G$ we have $S(f,g) \to_G 0$. Hence, by Buchberger's criterion (Theorem 2.1.20), G is a Gröbner Basis for $\mathcal{I}(\mathcal{P})$. □

Theorem 4.1.6. *Let Γ be a constraint language. If Γ has the dual-discriminator polymorphism, then $\mathrm{IMP}_d(\Gamma)$ is decidable in polynomial time.*

Note that unlike in the results of [161, 24], due to the preprocessing step, Theorem 4.1.6 does not always allow one to find a proof that a polynomial belongs to the ideal, but this is resolved in Chapter 5. Indeed we prove that we can find a proof of membership for $\mathrm{IMP}_d(\Gamma)$, if one exists, by constructing a d-truncated Gröbner Basis with respect to a grlex .

4.2 Semilattice polymorphisms

In this section we study the $\mathrm{IMP}(\Gamma)$ for languages Γ where $\mathsf{Pol}(\Gamma)$ contains a *semilattice* operation. A binary operation $\psi(x,y)$ satisfying the following three conditions is said to be a semilattice operation:

1. Associativity: $\psi(x, \psi(y,z)) = \psi(\psi(x,y), z)$
2. Commutativity: $\psi(x,y) = \psi(y,x)$

3. Idempotency: $\psi(x,x) = x$

Mastrolilli [161] considered this problem for languages over the Boolean domain i.e., $D = \{0,1\}$, and proved the following. We remark that a Boolean relation is closed under a semilattice operation if and only if it can be defined by a conjunction of *dual-Horn* clauses or can be defined by a conjunction of *Horn* clauses [121].

Theorem 4.2.1 ([161])**.** *Let Γ be a finite Boolean constraint language. If Γ has a semilattice polymorphism, then $IMP_d(\Gamma)$ can be solved in $n^{O(d)}$ time for $d \geq 1$.*

We extend this tractability result to languages over any finite domain D. That is, we prove that $\mathrm{IMP}_d(\Gamma)$ is polynomial time solvable when Γ is a language over D and it has a semilattice polymorphism. To do so, we use a well-known result in semilattice theory. A *semilattice* is an algebra $\mathcal{D} = (D, \{\psi\})$, where ψ is a semilattice operation. Informally speaking, every semilattice is a subalgebra of a direct power of a 2-element semilattice.

Theorem 4.2.2 ([181])**.** *Let $\mathcal{D} = (D, \psi)$ be a finite semilattice where ψ is a semilattice operation. Then there is k such that $\mathcal{D}$ is a subalgebra of the direct power $\mathcal{B}^k$ of $\mathcal{B} = (\{0,1\}, \varphi)$, where φ is a semilattice operation on $\{0,1\}$.*

Armed with Theorem 4.2.2 a proof of tractability of semilattice IMPs is straightforward.

Theorem 4.2.3. *Let Γ be a finite constraint language over domain D. If Γ has a semilattice polymorphism, then $IMP_d(\Gamma)$ is decidable in polynomial time.*

Proof. Let ψ be a semilattice polymorphism of Γ. Then $\mathcal{D} = (D, \psi)$ is a semilattice, and therefore is a subalgebra of $\mathcal{B}^k$, where $\mathcal{B} = (\{0,1\}, \varphi)$ is a 2-element semilattice. By Lemma 3.2.7, there is a finite constraint language Δ over $\{0,1\}$ such that φ is a polymorphism of Δ. By Theorem 3.1.15, $\mathrm{IMP}_d(\Gamma)$ reduces to $\mathrm{IMP}_{ud}(\Delta)$ in polynomial time for a constant u. By Theorem 4.2.1, we get the result. □

In Chapter 5 we prove that we can indeed find a proof of membership for $\mathrm{IMP}_d(\Gamma)$, if one exists, by constructing a d-truncated Gröbner Basis with respect to a grlex .

Example 4.2.4 (Totally Ordered Domain)**.** Let $D = \{1, \ldots, t\}$ be a finite domain and Γ be a language defined on D. A semilattice polymorphism ψ is *conservative* if $\psi(x,y) \in \{x,y\}$. Suppose Γ has a conservative semilattice polymorphism $\psi : D^2 \to D$. Note that ψ defines a total ordering on $\{1, \ldots, t\}$ so that $u \leq v$ if and only if $\psi(u,v) = u$. Define $\pi : D \to \{0,1\}^t$ to be the following mapping

$$\pi(i) = (0, \ldots, 0, \overbrace{1, \ldots, 1}^{i}).$$

Let $\mathcal{P} = (X, D, C)$ denote an instance of $\mathrm{CSP}(\Gamma)$ where $X = \{x_1, \ldots, x_n\}$. Construct CSP instance $\mathcal{P}' = (X', \{0,1\}, C')$ with $X' = \{x_{11}, \ldots, x_{1t}, \ldots, x_{n1}, \ldots, x_{nt}\}$ with the following set of constraints

1. $x_{ij} \leq x_{ik}$ for all $1 \leq i \leq n$ and $1 \leq k \leq j \leq t$,

2. if $R(x_{i_1}, \ldots, x_{i_k}) \in C$ then $\pi(R)(x_{i_1 1}, \ldots, x_{i_1 t}, \ldots, x_{i_k 1}, \ldots, x_{i_k t}) \in C'$.

Observe that $\mathbf{a} \in \mathbf{V}(\mathcal{I}(\mathcal{P}))$ if and only if $\pi(\mathbf{a}) \in \mathbf{V}(\mathcal{I}(\mathcal{P}'))$. Given $f_0 \in \mathbb{F}[x_1, \ldots, x_n]$, define $f_0' \in \mathbb{F}[x_{11}, \ldots, x_{1t}, \ldots, x_{n1}, \ldots, x_{nt}]$ to be the polynomial obtained from f_0 where we replace each indeterminate x_i with $x_{i1} + \cdots + x_{it}$. It is easy to check that, for $\mathbf{a} \in \mathbf{V}(\mathcal{I}(\mathcal{P}))$, we have $f_0(\mathbf{a}) = 0$ if and only if $\pi(\mathbf{a}) \in \mathbf{V}(\mathcal{I}(\mathcal{P}'))$ and $f_0'(\pi(\mathbf{a})) = 0$. Therefore, deciding if $f_0 \in \mathcal{I}(\mathcal{P})$ is equivalent to deciding if $f_0' \in \mathcal{I}(\mathcal{P}')$ where the later one is polynomial time solvable by Theorem 4.2.1.

4.3 Affine operations I: linear system in $\mathsf{GF}(p)$

In this section and the subsequent section we consider IMPs over languages invariant under affine operations of $\mathsf{GF}(p)$ and arbitrary finite Abelian groups, respectively. This type of constraint languages played an important role in the study of the CSP for three reasons. First, it captures a very natural class of problems. Problems CSP(Γ) where Γ is invariant under an affine operation of a finite field $\mathbb{F}$ can be expressed by systems of linear equations over $\mathbb{F}$ and therefore admit a classic solution algorithm such as Gaussian elimination or coset generation. In the case of a general Abelian group $\mathbb{A}$ the connection with systems of linear equations is more complicated, although it is still true that every instance of CSP(Γ) in this case can be thought of as a system of linear equations with coefficients from some ring — the ring of endomorphisms of $\mathbb{A}$.

Problems CSP(Γ) where Γ has an affine polymorphism were pivotal in the development of few subpowers algorithms, and, in a sense, constitute the main nontrivial case of them. The few subalgebras algorithms [36, 118] when applied to systems of linear equations serve as an alternative to Gaussian elimination that also work in a more general situation and are less sensitive to the algebraic structure behind the problem. There is, therefore, a hope that studying IMPs with an affine polymorphism may teach us about proof systems that use the IMP and do not quite work in the affine case.

In this section we focus on constraint languages that are expressible as a system of linear equations modulo a prime number. Let Γ be a constraint language over a set D with $|D| = p$, and p a prime number. Suppose Γ has an affine polymorphism modulo p (i.e. a ternary operation $\psi(x, y, z) = x \ominus y \oplus z$, where $\oplus, \ominus$ are addition and subtraction modulo p, or, equivalently, of the field $\mathsf{GF}(p)$). In this case every CSP can be represented as a system of linear equations over $\mathsf{GF}(p)$. Without loss of generality, we may assume that the system of linear equations at hand is already in the *reduced row echelon* form. Transforming system of linear equations mod p in its reduced row echelon form to a system of polynomials in $\mathbb{R}[X]$ that are a Gröbner Basis is not immediate and requires substantial work. This is the case even if we restrict ourselves to lexicographic order.

A proof based on the Gröbner Bases conversion technique is given in Section 4.5. That proof is of an independent interest as it relies on the existence of the so-called *independent p-expressions* and it is quite technical. Here, we present an alternative simple algorithm that checks the membership in polynomial time. In section 5.1.2, we will see how this algorithm can be used to construct a d-truncated Gröbner Basis.

Let $\mathcal{P}$ be an instance of CSP(Γ) that is expressed as a system of linear equations $\mathscr{S}$ over $\mathbb{Z}_p$ with variables $x_1, \ldots, x_n$. A system of linear equations over $\mathbb{Z}_p$ can be solved by Gaussian elimination (this immediately tells us if $1 \in \mathcal{I}(\mathcal{P})$ or not, and we proceed only if $1 \notin \mathcal{I}(\mathcal{P})$). We assume a lexicographic order $\succ_{\mathsf{lex}}$ with $x_1 \succ_{\mathsf{lex}} \cdots \succ_{\mathsf{lex}} x_n$. We also assume that the linear system has $r \leq n$ equations and it is already in its reduced row echelon form with x_i as the leading monomial of the i-th equation. Let $Supp_i \subset [n]$ such that $\{x_j : j \in Supp_i\}$ be the set of variables appearing in the i-th equation of the linear system except for x_i.

Fix a prime p and let $\oplus, \ominus, \odot$ denote addition, subtraction, and multiplication modulo p, respectively. We will call a linear polynomial over $\mathbb{Z}_p$ a *p-expression*. Let the i-th equation be $g_i = 0 \pmod p$ where $g_i := x_i \ominus f_i$, with $i \in [r]$ and f_i is the p-expression $(\bigoplus_{j \in Supp_i} \alpha_j x_j) \oplus \alpha_i$ and $\alpha_j, \alpha_i \in \mathbb{Z}_p$. We will assume that each variable x_i is associated with its p-expression f_i which comes from the mod p equations. This is clear for $i \leq r$; for $i > r$ the p-expression $f_i = x_i$ itself. Hence, we can write down the reduced Gröbner Basis in the lex order in an implicit form as follows.

$$G_1 = \{x_1 \ominus f_1, \ldots, x_r \ominus f_r, \prod_{i \in \mathbb{Z}_p} (x_{r+1} - i), \ldots, \prod_{i \in \mathbb{Z}_p} (x_n - i)\} \tag{4.1}$$

Let $U_p = \{\omega, \omega^2, \ldots, \omega^p = \omega^0 = 1\}$ be the set of p-th roots of unity where ω is a primitive p-th root of unity. For a primitive p-th root of unity ω we have $\omega^a = \omega^b$ if and only if $a \equiv b \pmod p$. From $\mathcal{P}$ we construct a new CSP instance $\mathcal{P}' = (V, U_p, \widetilde{C})$ where for each equation $x_i \ominus f_i = 0$ with $f_i = (\bigoplus_{j \in Supp_i} \alpha_j x_j) \oplus \alpha_i$ we add the constraint $x_i - f_i' = 0$ with

$$f_i' = \omega^{\alpha_i} (\prod_{j \in Supp_i} x_j^{\alpha_j}).$$

Moreover, the domain constraints are different. For each variable x_j, $r+1 \leq j \leq n$, the domain polynomial is $(x_j)^p - 1$. Therefore, we write G over complex number domain as follows.

$$G' = \{x_1 - f_1', \ldots, x_r - f_r', (x_{r+1})^p - 1, \ldots, (x_n)^p - 1\} \tag{4.2}$$

Define univariate polynomial $\phi \in \mathbb{C}[x]$ so that it interpolates points $(0, \omega^0), (1, \omega), \ldots, (p-1, \omega^{p-1})$. This polynomial provides a one-to-one mapping between solutions of instance $\mathcal{P}$

and instance $\mathcal{P}'$. That is, $(a_1, \ldots, a_n)$ is a solution of $\mathcal{P}$ if and only if $(\phi(a_1), \ldots, \phi(a_n))$ is a solution of $\mathcal{P}'$.

Lemma 4.3.1. *For a polynomial $f \in \mathbb{R}[x_1, \ldots, x_n]$ define polynomial $f' \in \mathbb{C}[x_1, \ldots, x_n]$ to be*

$$f'(x_1, \ldots, x_n) = f(\phi^{-1}(x_1), \ldots, \phi^{-1}(x_n)).$$

Then $f \in \mathtt{I}(\mathcal{P})$ if and only if $f' \in \mathtt{I}(\mathcal{P}')$.

Proof. As $\mathcal{I}(\mathcal{P})$ is radical, by the Strong Nullstellensatz we have $f \notin \mathcal{I}(\mathcal{P})$ if and only if there exists a point $\mathbf{a} \in \mathbb{Z}_p^n$ such that $\mathbf{a}$ is in $Sol(\mathcal{P})$ and $f(\mathbf{a}) \neq 0$. Note $\mathcal{I}(\mathcal{P}')$ is also radical since for all $x_i \in \{x_1, \ldots, x_r\}$ the domain polynomial $x_i^p - 1$ is in $\mathcal{I}(\mathcal{P}')$ because remainder of division of $x_i^p - 1$ by G' is $0 = (f_i')^p - 1$. Hence, for $\mathcal{I}(\mathcal{P}')$ we have $f' \notin \mathcal{I}(\mathcal{P}')$ if and only if there exists a point $\mathbf{a}' \in U_p^n$ such that $\mathbf{a}' \in Sol(\mathcal{P}')$ and $f'(\mathbf{a}') \neq 0$.

Suppose $f \notin \mathcal{I}(\mathcal{P})$ and consider a point $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{Z}_p^n$ so that $\mathbf{a} \in Sol(\mathcal{P})$ and $f(\mathbf{a}) \neq 0$. Recall that $\mathbf{a} \in Sol(\mathcal{P})$ if and only if $\mathbf{a}' = (\omega^{a_1}, \ldots, \omega^{a_n}) \in Sol(\mathcal{P}')$. Furthermore,

$$f'(\omega^{a_1}, \ldots, \omega^{a_n}) = f(a_1, \ldots, a_n) \neq 0$$

Therefore, for $\mathbf{a}' \in Sol(\mathcal{P}')$ we have $f'(\mathbf{a}') \neq 0$ which implies $f' \notin \mathcal{I}(\mathcal{P}')$. This finishes the proof. □

The next lemma states that $\mathrm{IMP}(\mathcal{I}(\mathcal{P}'))$ is polynomial time solvable by showing that the set of polynomials G' is in fact a Gröbner Basis for $\mathcal{I}(\mathcal{P}')$.

Lemma 4.3.2. *G' is a Gröbner Basis for $\mathtt{I}(\mathcal{P}')$ with respect to lex order $x_1 \succ_{\mathsf{lex}} \cdots \succ_{\mathsf{lex}} x_n$.*

Proof. First, we proceed to show G' is a Gröbner Basis for $\langle G' \rangle$. Note that for each $x_i - f_i'$, we have $\mathrm{LM}(x_i - f_i') = x_i$. Moreover, the leading monomial of $(x_j)^p - 1$, $r + 1 \leq j \leq n$, is $(x_j)^p$. Hence, for every pair of polynomials in G' the reduced S-polynomial is zero as the leading monomials of any two polynomials in G' are relatively prime. By Buchberger's Criterion it follows that G' is a Gröbner Basis for $\langle G' \rangle$ over $\mathbb{C}[x_1, \ldots, x_n]$ (according to the lex order).

It remains to show $\langle G' \rangle = \mathbf{I}(Sol(\mathcal{P}'))$. According to our construction we have $\mathbf{V}(\langle G' \rangle) = Sol(\mathcal{P}')$ which implies $\langle G' \rangle \subseteq \mathbf{I}(Sol(\mathcal{P}'))$. In what follows, we prove $\mathbf{I}(Sol(\mathcal{P}')) \subseteq \langle G' \rangle$. Consider a polynomial $f \in \mathbf{I}(Sol(\mathcal{P}'))$. Note that $f(\mathbf{a}) = 0$ for all $\mathbf{a} \in Sol(\mathcal{P}')$. Let $r = f|_{G'}$. r does not contain variables $x_1, \ldots, x_r$, and hence it is a polynomial in $x_{r+1}, \ldots, x_n$. Now note that any $\mathbf{b} = (b_1, \ldots, b_{n-r}) \in U_p^{n-r}$ extends to a unique point in $Sol(\mathcal{P}')$. Therefore, all the points in U_p^{n-r} are zeros of r, hence $r = f|_{G'}$ is the zero polynomial. Since $f \in \mathbf{I}(Sol(\mathcal{P}'))$ was arbitrary chosen, it follows that for every $f \in \mathbf{I}(Sol(\mathcal{P}'))$ we have $f|_{G'} = 0$. Hence G' is a Gröbner Basis of $\mathbf{I}(Sol(\mathcal{P}'))$. □

Provided that G' is a Gröbner Basis with respect to the lex order, we can test membership of any bounded degree polynomial in polynomial time. Note that dividing any polynomial

by $x_1 - f'_1, \ldots, x_r - f'_r$ results in a polynomial only in $x_{r+1}, \ldots, x_n$ where the membership can be tested using only $\{(x_{r+1})^p - 1, \ldots, (x_n)^p - 1\}$, a Gröbner Basis with respect to grlex in $\mathbb{C}[x_{r+1}, \ldots, x_n]$.

We also remark that the substitution technique used in Lemma 4.3.1 may result in a polynomial with exponentially many monomials. However, for polynomials of bounded degree d it yields a polynomial of degree at most pd and polynomially many monomials. Thus, we obtain the following.

Theorem 4.3.3. *Let Γ be a constraint language over domain $D = \mathsf{GF}(p)$. If Γ has an affine polymorphism, then $\mathrm{IMP}_d(\Gamma)$ is decidable in polynomial time.*

In Chapter 5 we prove that we can indeed find a proof of membership for $\mathrm{IMP}_d(\Gamma)$, if one exists, by constructing a d-truncated Gröbner Basis with respect to a grlex .

4.4 Affine operations II: CSPs over Abelian groups

CSPs over Abelian groups, or more precisely problems of the form $\mathrm{CSP}(\Gamma)$ where Γ is a constraint language closed under the affine polymorphism $x - y + z$ of an Abelian group, are well understood. However, they are usually considered as a special case of either arbitrary finite groups, in which case the coset generation algorithm applies [79], or as a special case of CSPs with a Mal'tsev polymorphism [36, 118]. In the IMP literature [161, 25, 41] such CSPs have been mainly considered from the point of view of systems of linear equations. Such a representation is necessary, because it is used to construct a Gröbner basis of the corresponding ideal. While it is true that every CSP given by a system of linear equations over some Abelian group can also be thought of as an instance of $\mathrm{CSP}(\Gamma)$ for an appropriate language Γ closed under the affine operation, the converse is not true in general. For instance, the relation R_E from Example 3.3.6 is invariant under the affine operation $x-y+z$ of $\mathbb{Z}_2 \times \mathbb{Z}_2$, but cannot be represented by a system of linear equations over this group.

Therefore our goal in this section is to show that a CSP over an Abelian group can always be converted into a (multi-sorted) CSP that admits a representation by a system of linear equations (with caveats that will be discussed later), and then to demonstrate how a row-echelon form of such a system can be constructed, ready to be transformed into a Gröbner basis.

4.4.1 Abelian groups

In this section we state the facts about Abelian groups we will need in this thesis. It is well known that every finitely generated Abelian group is isomorphic to a *direct sum* of primary cyclic groups and infinite cyclic groups. In the following, the notation $G \oplus H$ denotes the direct sum of two (Abelian) groups.

Proposition 4.4.1. *(1) (*The Fundamental Theorem of Abelian Groups.*) Let $\mathbb{A}$ be a finite Abelian group. Then $\mathbb{A} = \mathbb{A}_1 \oplus \cdots \oplus \mathbb{A}_n$, where $\mathbb{A}_1, \ldots, \mathbb{A}_n$ are cyclic groups.*

(2) There exists a decomposition from item (1), in which the order of each $\mathbb{A}_i$ is a prime power.

Let $\mathbb{A}$ be an Abelian group. By Proposition 4.4.1, $\mathbb{A}$ can be decomposed into $\mathbb{A} = \mathbb{A}_1 \oplus \cdots \oplus \mathbb{A}_n$ where $\mathbb{A}_i = \mathbb{Z}_{q_i^{\ell_i}}$. Without loss of generality assume that for some $k_1, \ldots, k_s$ it holds $q_1 = \cdots = q_{k_1} = p_1$, $q_{k_1+1} = \cdots = q_{k_1+k_2} = p_2, \ldots, q_{k_1+\cdots+k_{s-1}+1} = \cdots = q_{k_1+\cdots+k_s} = p_s$. We also change the notation for ℓ_i so that $\mathbb{A}$ can be represented as

$$\mathbb{A} = \mathbb{Z}_{p_1^{\ell_{1,1}}} \oplus \cdots \oplus \mathbb{Z}_{p_1^{\ell_{1,k_1}}} \oplus \mathbb{Z}_{p_2^{\ell_{2,1}}} \cdots \oplus \mathbb{Z}_{p_2^{\ell_{2,k_2}}} \oplus \cdots \oplus \mathbb{Z}_{p_s^{\ell_{s,k_s}}}.$$

Later it will also be convenient to assume that ℓ_{r,k_r} is maximal among $\ell_{r,1}, \ldots, \ell_{r,k_r}$. We will denote this value by m_r.

Next we describe relations invariant under an affine polymorphism of an Abelian group in group-theoretic form. Recall that for an Abelian group $\mathbb{A}$ and its subgroup $\mathbb{B}$, a *coset* of $\mathbb{A}$ modulo $\mathbb{B}$ is a set of the form $a_0 + \mathbb{B} = \{a_0 + a \mid a \in \mathbb{B}\}$. The following statement is folklore, but we give a proof for completeness.

Lemma 4.4.2. *Let R be a subset of the Cartesian product of Abelian groups $\mathbb{A}_1 \times \cdots \times \mathbb{A}_n$. Then R is invariant with respect to the (multi-sorted) affine operation $f(x, y, z) = x - y + z$ of the groups $\mathbb{A}_1, \ldots, \mathbb{A}_n$ if and only if R is a coset of $\mathbb{A} = \mathbb{A}_1 \times \cdots \times \mathbb{A}_n$, viewed as an Abelian group, modulo some subgroup $\mathbb{B}$ of $\mathbb{A}$.*

Proof. If R is a coset of $\mathbb{A}$ modulo a subgroup $\mathbb{B}$, fix $\mathbf{a}_0 \in R$. Then $R = \{\mathbf{a}_0 + \mathbf{a} \mid \mathbf{a} \in \mathbb{B}\}$. For any $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{B}$ we have

$$f(\mathbf{a} + \mathbf{a}_0, \mathbf{b} + \mathbf{a}_0, \mathbf{c} + \mathbf{a}_0) = (\mathbf{a} - \mathbf{b} + \mathbf{c}) + \mathbf{a}_0 \in R,$$

as $\mathbf{a} - \mathbf{b} + \mathbf{c} \in \mathbb{B}$.

Conversely, suppose R is invariant under f. Fix $\mathbf{a}_0 \in R$ and set $\mathbb{B} = \{\mathbf{a} - \mathbf{a}_0 \mid \mathbf{a} \in R\}$. We need to show that $\mathbb{B}$ is a subgroup of $\mathbb{A}$. Since $\mathbb{A}$ is finite it suffices to show that $\mathbb{B}$ is closed under addition. Let $\mathbf{a}, \mathbf{b} \in \mathbb{B}$, then $\mathbf{a} + \mathbf{a}_0, \mathbf{b} + \mathbf{a}_0 \in R$. As R is invariant under f,

$$f(\mathbf{a} + \mathbf{a}_0, \mathbf{a}_0, \mathbf{b} + \mathbf{a}_0) = (\mathbf{a} + \mathbf{a}_0) - \mathbf{a}_0 + (\mathbf{b} + \mathbf{a}_0) = (\mathbf{a} + \mathbf{b}) + \mathbf{a}_0 \in R,$$

implying $\mathbf{a} + \mathbf{b} \in \mathbb{B}$. □

4.4.2 PP-interpretations in Abelian groups

In this section we show that any constraint language invariant under an affine operation of some Abelian group can be pp-interpreted by a multi-sorted constraint language over very simple groups. We use the notation from Section 4.4.1.

Proposition 4.4.3. *Let Δ be a finite constraint language invariant under the affine operation of $\mathbb{A}$. Then there is a multi-sorted constraint language Γ over $\mathbb{Z}_{p_1^{m_1}}, \dots, \mathbb{Z}_{p_s^{m_s}}$ invariant under the affine operation of $\mathbb{Z}_{p_1^{m_1}}, \dots, \mathbb{Z}_{p_s^{m_s}}$ such that Γ pp-interprets Δ.*

Proof. For a natural number r and an Abelian group $\mathbb{B}$ by $r\mathbb{B}$ we denote the subgroup $r\mathbb{B} = \{ra \mid a \in \mathbb{B}\}$ of $\mathbb{B}$. Observing that $\mathbb{Z}_{p^\ell}$ is isomorphic to $p^{m-\ell}\mathbb{Z}_{p^m}$ let

$$F = p_1^{m_1-\ell_{1,1}}\mathbb{Z}_{p_1^{m_1}} \times \cdots \times p_1^{m_1-\ell_{1,k_1}}\mathbb{Z}_{p_1^{m_1}} \times \cdots \times p_1^{m_s-\ell_{s,k_{s-1}+1}}\mathbb{Z}_{p_s^{m_s}} \times \cdots \times p_1^{m_s-\ell_{s,k_s}}\mathbb{Z}_{p_s^{m_s}},$$

and define a mapping $\pi : F \to \mathbb{A}$ by

$$\pi(x_{1,1}, \dots, x_{1,k_1}, \dots, x_{s,1}, \dots, x_{s,k_s}) = \\ ((p_1^{m_1-\ell_{1,1}})^{-1}x_{1,1}, (p_1^{m_1-\ell_{1,2}})^{-1}x_{1,2}, \dots, x_{1,k_1}, \dots, (p_s^{m_s-\ell_{s,k_{s-1}+1}})^{-1}x_{s,1}, \dots, (p_s^{m_s-\ell_{s,k_s}})^{-1}x_{s,k_s}).$$

Note that the values of the form $(p_r^{m_r-\ell_{r,i}})^{-1}x_{r,i} \in \mathbb{Z}_{p_r^{\ell_{r,i}}}$ are well defined because $x_{r,i} \in p_r^{m_r-\ell_{r,i}}\mathbb{Z}_{p_r^{m_r}}$. Then we set $\Gamma = \{F\} \cup \{\pi^{-1}(=_{\mathbb{A}})\} \cup \{\pi^{-1}(R) \mid R \in \Delta\}$. The language Γ contains F, the preimage of the equality relation on $\mathbb{A}$, and the preimages of all the relations from Δ. Therefore, by the definition of pp-interpretability Γ pp-interprets Δ. It remains to show that Γ is invariant under the affine operation of $\mathbb{Z}_{p_1^{m_1}}, \dots, \mathbb{Z}_{p_s^{m_s}}$ as a multi-sorted polymorphism.

The set F is clearly invariant under any operation of the groups $\mathbb{Z}_{p_1^{m_1}}, \dots, \mathbb{Z}_{p_s^{m_s}}$, as it is a Cartesian product of subgroups of those groups. For $\mathbf{a}, \mathbf{b} \in F$ we have $(\mathbf{a}, \mathbf{b}) \in \pi^{-1}(=_{\mathbb{A}})$ if and only if $p_r^{m_r-\ell_{r,i}}a_{r,i} = p_r^{m_r-\ell_{r,i}}b_{r,i}$ in $\mathbb{Z}_{p_r^{m_r}}$, where $i \in [k_r]$. So, if $(\mathbf{a}^1, \mathbf{b}^1), (\mathbf{a}^2, \mathbf{b}^2), (\mathbf{a}^3, \mathbf{b}^3) \in \pi^{-1}(=_{\mathbb{A}})$, $\mathbf{c} = \mathbf{a}^1 - \mathbf{a}^2 + \mathbf{a}^3, \mathbf{d} = \mathbf{b}^1 - \mathbf{b}^2 + \mathbf{b}^3$ then for any $r \in [s]$ and $i \in [k_r]$ we have

$$\begin{aligned} p_r^{m_r-\ell_{r,i}}c_{r,i} &= p_r^{m_r-\ell_{r,i}}(a^1_{r,i} - a^2_{r,i} + a^3_{r,i}) \\ &= p_r^{m_r-\ell_{r,i}}a^1_{r,i} - p_r^{m_r-\ell_{r,i}}a^2_{r,i} + p_r^{m_r-\ell_{r,i}}a^3_{r,i} \\ &= p_r^{m_r-\ell_{r,i}}b^1_{r,i} - p_r^{m_r-\ell_{r,i}}b^2_{r,i} + p_r^{m_r-\ell_{r,i}}b^3_{r,i} \\ &= p_r^{m_r-\ell_{r,i}}(b^1_{r,i} - b^2_{r,i} + b^3_{r,i}) \\ &= p_r^{m_r-\ell_{r,i}}d_{r,i}. \end{aligned}$$

Now, let $R \in \Delta$ be a t-ary relation and $R' = \pi^{-1}(R)$. We show that $x - y + z$ is a polymorphism of R'. Let $\mathbf{a}', \mathbf{b}', \mathbf{c}' \in R'$, $\mathbf{a} = \pi(\mathbf{a}'), \mathbf{b} = \pi(\mathbf{b}'), \mathbf{c} = \pi(\mathbf{c}')$, $\mathbf{d}' = \mathbf{a}' - \mathbf{b}' + \mathbf{c}'$, and $\mathbf{d} = \mathbf{a} - \mathbf{b} + \mathbf{c}$. It suffices to show that $\pi(\mathbf{d}') = \mathbf{d}$, as it implies $\mathbf{d}' \in R'$, since $\mathbf{d} \in R$. Each of the tuples $\mathbf{a}', \mathbf{b}', \mathbf{c}', \mathbf{d}'$ is a $t \cdot n$-tuple. We will denote its components by $a'_{i,r,j}$ ($b'_{i,r,j}, c'_{i,r,j}, d'_{i,r,j}$), $i \in [t], r \in [s], j \in [k_r]$ so that $\pi(a'_{i,1,1}, \dots, a'_{i,s,k_s}) = a_i$ ($\pi(b'_{i,1,1}, \dots, b'_{i,s,k_s}) = b_i$, $\pi(c'_{i,1,1}, \dots, c'_{i,s,k_s}) = c_i$). For any $i \in [t]$, $r \in [s]$, and $j \in [k_r]$ we

have

$$(p_r^{m_r-\ell_{r,j}})^{-1}d'_{i,r,j} = (p_r^{m_r-\ell_{r,j}})^{-1}a'_{i,r,j} - (p_r^{m_r-\ell_{rj}})^{-1}b'_{i,r,j} + (p_r^{m_r-\ell_{r,j}})^{-1}c'_{i,r,j}.$$

Therefore,

$$\begin{aligned}
&\pi(d'_{i,1,1},\dots,d'_{i,s,k_s})\\
&= (p_1^{m_1-\ell_{1,1}})^{-1}d'_{i,1,1} \oplus \cdots \oplus (p_s^{m_s-\ell_{s,k_s}})^{-1}d'_{i,s,k_s}\\
&= (p_1^{m_1-\ell_{1,1}})^{-1}(a'_{i,1,1} - b'_{i,1,1} + c'_{i,1,1}) \oplus \cdots \oplus (p_s^{m_s-\ell_{s,k_s}})^{-1}(a'_{i,s,k_s} - b'_{i,s,k_s} + c'_{i,s,k_s})\\
&= \left((p_1^{m_1-\ell_{1,1}})^{-1}a'_{i,1,1} \oplus \cdots \oplus (p_s^{m_s-\ell_{s,k_s}})^{-1}a'_{i,s,k_s}\right)\\
&\quad - \left((p_1^{m_1-\ell_{1,1}})^{-1}b'_{i,1,1} \oplus \cdots \oplus (p_s^{m_s-\ell_{s,k_s}})^{-1}b'_{i,s,k_s}\right)\\
&\quad + \left((p_1^{m_1-\ell_{1,1}})^{-1}c'_{i,1,1} \oplus \cdots \oplus (p_s^{m_s-\ell_{s,k_s}})^{-1}c'_{i,s,k_s}\right)\\
&= a_i - b_i + c_i = d_i.
\end{aligned}$$

Note that $\oplus$ in the formulas above denote the representation of elements of $\mathbb{A}$ as a direct sum of elements $\mathbb{Z}_{p_r^{\ell_{r,j}}}$. □

A nice property of languages over $\mathbb{Z}_{p_1^{m_1}},\dots,\mathbb{Z}_{p_s^{m_s}}$, with $p_i \neq p_j$ when $i \neq j$, is that any (multi-sorted) relation can be decomposed into relations of the same sort. Let $R \subseteq D_1 \times \cdots \times D_n$ and $I = \{i_1,\dots,i_k\} \subseteq [n]$. For $\mathbf{a} \in R$ by $\mathrm{pr}_I\mathbf{a}$ we denote the tuple $(a_{i_1},\dots,a_{i_k})$, and $\mathrm{pr}_I R = \{\mathrm{pr}_I\mathbf{a} \mid \mathbf{a} \in R\}$.

For two relations $R_1(x_1,\dots,x_r)$ and $R_2(y_1,\dots,y_t)$ of arities r and t respectively, their Cartesian product is the relation R of arity $r+t$ defined as

$$\begin{aligned}
R(x_1,\dots,x_r,y_1,\dots,y_t) &= R_1(x_1,\dots,x_r) \times R_2(y_1,\dots,y_t)\\
&= \{(x_1,\dots,x_r,y_1,\dots,y_t) \mid (x_1,\dots,x_r) \in R_1 \wedge (y_1,\dots,y_t) \in R_2\}.
\end{aligned}$$

Lemma 4.4.4. *Let $R(x_{1,1},\dots,x_{1,k_1},\dots,x_{s,1},\dots,x_{s,k_s})$ be such that $x_{i,j}$ has domain $\mathbb{Z}_{p_i^{m_i}}$ and R is invariant under the affine operation. Then R is decomposable as follows*

$$\begin{aligned}
&R(x_{1,1},\dots,x_{1,k_1},\dots,x_{s,1},\dots,x_{s,k_s}) =\\
&\qquad (\mathrm{pr}_{(1,1),\dots,(1,k_1)}R)(x_{1,1},\dots,x_{1,k_1}) \times \cdots \times (\mathrm{pr}_{(s,1),\dots,(s,k_s)}R)(x_{s,1},\dots,x_{s,k_s}).
\end{aligned}$$

Proof. It suffices to show that

$$\begin{aligned}
&R(x_{1,1},\dots,x_{1,k_1},\dots,x_{s,1},\dots,x_{s,k_s}) =\\
&\qquad (\mathrm{pr}_{(1,1),\dots,(1,k_1)}R)(x_{1,1},\dots,x_{1,k_1}) \times (\mathrm{pr}_{(2,1),\dots,(s,k_s)}R)(x_{2,1},\dots,x_{s,k_s}).
\end{aligned}$$

Let $M = p_1^{m_1} \cdot p_2^{m_2} \dots \cdot p_s^{m_s}$, $M_1 = M/p_1^{m_1}$, and let $u, v \in [M]$ be such that $u \equiv 1 \pmod{p_1^{m_1}}, u \equiv 0 \pmod{M_1}$ and $v \equiv 0 \pmod{p_1^{m_1}}, v \equiv 1 \pmod{M_1}$. Then for any $\mathbf{a} \in R$, $\mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2)$, where $\mathbf{a}_1 \in \mathrm{pr}_{(1,1),\dots,(1,k_1)}R$ and $\mathbf{a}_2 \in \mathrm{pr}_{(2,1),\dots,(s,k_s)}R$, it holds $u \cdot \mathbf{a} = (\mathbf{a}_1, \overline{0})$ and $v \cdot \mathbf{a} = (\overline{0}, \mathbf{a}_2)$, and $\overline{0}$ denotes the zero vector of an appropriate length.

We prove that, as any composition of polymorphisms of R is a polymorphism of R the operation $g(x, y, z) = ux + vy + (1 - u - v)z$ is a polymorphism of R. More precisely, we prove by induction on u', v' that $u'x + v'y + (1 - u' - v')z$ can be obtained as a composition of $f(x, y, z) = x - y + z$ with itself. Indeed, for $u' = v' = 1$ the operation $f(x, z, y) = x + y - z$ is as required. Suppose the statement is proved for $g'(x, y, z) = u'x + v'y + (1 - u' - v')z$. Then

$$f(x, z, g'(x, y, z)) = x - z + u'x + v'y + (1 - u' - v')z = (u' + 1)x + v'y + (1 - (u' + 1) - v')z,$$
$$f(y, z, g'(x, y, z)) = y - z + u'x + v'y + (1 - u' - v')z = u'x + (v' + 1)y + (1 - u' - (v' + 1))z.$$

We need to prove that if $\mathbf{a}, \mathbf{b} \in R$, $\mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2), \mathbf{b} = (\mathbf{b}_1, \mathbf{b}_2)$ then $(\mathbf{a}_1, \mathbf{b}_2) \in R$. This is however straightforward:

$$\begin{aligned} g(\mathbf{a}, \mathbf{b}, \mathbf{a}) &= (1 - v)\mathbf{a} + v\mathbf{b} \\ &= (1 - v)(\mathbf{a}_1, \mathbf{a}_2) + v(\mathbf{b}_1, \mathbf{b}_2) \\ &= ((1 - v)\mathbf{a}_1, (1 - v)\mathbf{a}_2) + (v\mathbf{b}_1, v\mathbf{b}_2) \\ &= (\mathbf{a}_1, 0) + (0, \mathbf{b}_2) \\ &= (\mathbf{a}_1, \mathbf{b}_2). \end{aligned}$$

□

Lemma 4.4.4 allows us to decompose multi-sorted CSPs into instances, in which every constraint contains variables of only one sort.

Proposition 4.4.5. *Let $\mathcal{P}$ be an instance of CSP(Γ), where Γ is a multi-sorted constraint language over $\mathcal{D} = \{\mathbb{Z}_{p_1^{m_1}}, \dots, \mathbb{Z}_{p_s^{m_s}}\}$ invariant with respect to the affine polymorphism of $\mathbb{Z}_{p_1^{m_1}}, \dots, \mathbb{Z}_{p_s^{m_s}}$. Then $\mathcal{P}$ is equivalent to $\mathcal{P}'$ such that for every constraint $\langle \mathbf{s}, R \rangle$, the variables in $\mathbf{s}$ are of the same sort. Moreover, the set of variables X of $\mathcal{P}'$ is the same as that of $\mathcal{P}$ and for any $x \in X$ its sort is the same in both $\mathcal{P}$ and $\mathcal{P}'$.*

4.4.3 Constructing a system of linear equations

Having the decomposition result in Lemma 4.4.4 and Proposition 4.4.5, we proceed to show that any instance of CSP(Γ) where Γ is invariant under the affine operation of $\mathbb{Z}_{p^m}$ can be transformed into a system of linear equations over $\mathbb{Z}_{p^m}$ that is in the *reduced row-echelon* form i.e., there are free variables and the rest of variables are linear combinations

thereof. Note that unlike linear equations over $\mathbb{Z}_p$, such a transformation is not immediate. In particular, it will require introducing new variables that will serve as free variables.

Lemma 4.4.6. *Let $\mathcal{P}$ be an instance of CSP(Γ) where Γ is a constraint language over $\mathbb{A} = \mathbb{Z}_{p^m}$ invariant under the affine operation of the group. Let $X = \{x_1, \dots, x_n\}$ be the set of variables of $\mathcal{P}$. Then there are parameters $y_1, \dots, y_r$ such that for every $j \in [n]$ there are $\alpha_{1,j}, \dots, \alpha_{r,j}, c_j \in \mathbb{Z}_{p^m}$, for which $(x_1, \dots, x_n) \in R$ if and only if $x_j = \alpha_{1,j}y_1 + \dots + \alpha_{r,j}y_r + c_j$ for some values $y_1, \dots, y_r \in \mathbb{Z}_{p^m}$.*

Proof. We start with a claim that indicates what the existing algorithms allow us to do with respect to the instance $\mathcal{P}$. Recall that $\mathsf{Sol}(\mathcal{P})$ denotes the set of solutions of $\mathcal{P}$.

Claim 1. (1) A solution of $\mathcal{P}$, if one exists, can be found in polynomial time.

(2) For any $x \in X$ and any $a \in \mathbb{Z}_{p^m}$, a solution $\varphi \in \mathsf{Sol}(\mathcal{P})$ can be found in polynomial time such that $\varphi(x) = a$, if one exists.

(3) For any $x \in X$ the set $\mathsf{Sol}_x(\mathcal{P}) = \{\varphi(x) \mid \varphi \in \mathsf{Sol}(\mathcal{P})\}$ can be found in polynomial time.

Proof of Claim 1. (1) A ternary operation f on a set A is said to be *Mal'tsev* if $f(x, x, y) = f(y, x, x) = y$ for $x, y \in A$. The affine operation of any Abelian group including $\mathbb{A}$ is Mal'tsev. It was proved in [36] that for any Γ invariant under a Mal'tsev operation the problem CSP(Γ) can be solved in polynomial time. Since Γ in Lemma 4.4.6 is invariant under a Mal'tsev operation, it implies item (1).

(2) The constant relation $R_a = \{(a)\}$ is invariant under the affine operation of $\mathbb{A}$. This means that the problem $\mathcal{P}'$ obtained from $\mathcal{P}$ by adding the constraint $\langle (x), R_a \rangle$ can be solved in polynomial time using the algorithm from [36]. A mapping φ is a solution of $\mathcal{P}'$ if and only if $\varphi \in \mathsf{Sol}(\mathcal{P})$ and $\varphi(x) = a$.

(3) To find the set $\mathsf{Sol}_x(\mathcal{P})$ one just needs to apply item (2) to every element of $\mathbb{A}$. □

Pick an arbitrary solution $\varphi_0 \in R$ and let $S' = \{\varphi - \varphi_0 \mid \varphi \in \mathsf{Sol}(\mathcal{P})\}$. By Lemma 4.4.2 S' is a subgroup of $\mathbb{A}^n$. First, find $\mathsf{Sol}_x(\mathcal{P})$ for every $x \in X$ and set $S'_x = \mathsf{Sol}_x(\mathcal{P}) - \varphi_0(x)$ (subtraction in $\mathbb{A}$). For $a \in \mathbb{A}$ let $p(a)$ denote the maximal power of p that divides a. Find $x \in X$ such that S'_x contains an element $a \in \mathbb{A}$ for which $p(a)$ is minimal possible. Without loss of generality let x be x_1, and denote the value a by a_1 and set $o_1 = m - p(a_1)$. Let also φ_1 be a solution of $\mathcal{P}$ such that $\varphi_1(x_1) = a_1 + \varphi_0(x_1)$, and $\varphi'_1 = \varphi_1 - \varphi_0$. Observe that since $p(a_1)$ is minimal, for any $x_i \in X$ we have $\varphi'_1(x_i) = \alpha'_{1,i}\varphi'_1(x_1)$ for some $\alpha'_{1,i} \in \mathbb{Z}_{p^m}$. Let $\mathcal{P}^{(1)}$ be the instance $\mathcal{P}$ with the extra constraint $\langle (x_1), R_{\varphi_0(x_1)} \rangle$. Suppose that $\mathcal{P}^{(i)}$ is constructed, that is obtained from $\mathcal{P}$ by adding constraints $\langle (x_j), R_{\varphi_0(x_j)} \rangle$ for $j \in [i]$. Let also $S^{(i)} = \{\varphi - \varphi_0 \mid \varphi \in \mathsf{Sol}(\mathcal{P}^{(i)}\}$. Again, find $x \in X - \{x_1, \dots, x_i\}$ and an element $a \in S^{(i)}_x$ such that $p(a)$ is minimal possible. Assume that $x = x_{i+1}$, $a = a_{i+1}$, and $o_{i+1} = m - p(a_{i+1})$. Find a solution $\varphi_{i+1} \in \mathsf{Sol}(\mathcal{P}^{(i)})$ with $\varphi_{i+1}(x_{i_1}) = a_{i+1} + \varphi_0(x_{i+1})$ and let $\varphi'_{i+1} = \varphi_{i+1} - \varphi_0$.

The process ends at some point, suppose at step r, as φ_0 is the only solution of $\mathcal{P}^{(r+1)}$. By construction, $\{\varphi'_1, \dots, \varphi'_r\}$ is a generating set of S' and $\varphi'_j(x_i) = 0$ for $j > i$, $i, j \in [r]$.

As we observed, for any $j \in [n]$ there are $\alpha'_{1,j}, \dots, \alpha'_{r,j}$, $\alpha'_{i,j} \in \mathbb{Z}_{q^{o_i}}$ for $i \in [r]$, such that $\varphi'_1(x_j) = \alpha'_{1,j}\varphi'_1(x_1), \dots, \varphi'_r(x_j) = \alpha'_{r,j}\varphi'_r(x_r)$. We claim that coefficients $\alpha_{i,j} = \alpha'_{i,j}\varphi'_i(x_i)$, $c_j = \varphi_0(x_i)$ $i \in [r], j \in [n]$ are as required. To see this, observe that, as $\{\varphi'_1, \dots, \varphi'_r\}$ is a generating set of S', we have $\varphi \in \mathsf{Sol}(\mathcal{P})$ if and only if there are $y_1, \dots, y_r \in \mathbb{Z}_{p^m}$, such that $\varphi - \varphi_0 = y_1\varphi'_1 + \dots + y_r\varphi'_r$. Thus, for any $j \in [n]$ we have

$$\begin{aligned}\varphi(x_j) - c_j &= y_1\varphi'_1(x_j) + \dots + y_r\varphi'_r(x_j)\\ &= y_1\alpha'_{1,j}\varphi'_1(x_1) + \dots + y_r\alpha'_{r,j}\varphi'_r(x_r)\\ &= \alpha_{1,j}y_1 + \dots + \alpha_{r,j}y_r.\end{aligned}$$

Thus, solutions of $\mathcal{P}$ are exactly the mappings satisfying the equations

$$x_j = \alpha_{1,j}y_1 + \dots + \alpha_{r,j}y_r + c_j.$$

□

Putting everything together, we have proposed a reduction that transforms every instance of CSP(Δ), where Δ is constraint language invariant under the affine operation of an Abelian group, into a systems of linear equations over cyclic p-groups. Note that by the Decomposition Lemma (Lemma 4.4.4) and Proposition 4.4.5 we can assume that these systems of linear equations do not share variables. We summarize the result of this section as the following proposition.

Proposition 4.4.7. *Let Δ be a constraint language invariant under the affine operation of an Abelian group $\mathbb{A}$. There are distinct primes $p_1, \dots, p_s$, integers $m_1, \dots, m_s$ (not necessary distinct), and a multi-sorted constraint language Γ over $\mathbb{Z}_{p_1^{m_1}}, \dots, \mathbb{Z}_{p_s^{m_s}}$ such that Γ is invariant under the affine operation of these groups, and Γ pp-interprets Δ. Moreover, for every instance $\mathcal{P}$ of CSP(Γ) there are integers $k_1, \dots, k_s$ (not necessary distinct) such that $\mathcal{P}$ is on the set of variables $X = \{x_{1,1}, \dots, x_{1,k_1}, \dots, x_{s,1}, \dots, x_{s,k_s}\}$, and it can be expressed as s systems $\mathcal{L}_1, \dots, \mathcal{L}_s$ of linear equations where*

1. *each $\mathcal{L}_i$ is a system of linear equations over $\mathbb{Z}_{p_i^{m_i}}$ with variables $X(\mathcal{L}_i) \cup Y(\mathcal{L}_i)$, where $X(\mathcal{L}_i) = \{x_{i,1}, \dots, x_{i,k_i}\}$, $Y(\mathcal{L}_i) = \{y_{i,1}, \dots, y_{i,r_i}\}$;*
2. *each $\mathcal{L}_i$ is of the following form*

$$(\mathbf{1}_{k_i \times k_i} \quad M_i)(x_{i,1}, \dots, x_{i,k_i}, y_{i,1}, \dots, y_{i,r_i}, 1)^T = \mathbf{0};$$

3. *$X(\mathcal{L}_i) \cap X(\mathcal{L}_j) = \emptyset$, $Y(\mathcal{L}_i) \cap Y(\mathcal{L}_j) = \emptyset$, for all $1 \le i, j \le s$ and $i \ne j$;*

4. *an assignment φ to variables from X is a solution of $\mathcal{P}$ if and only if for every $i \in [s]$ there are values of variables from $Y(\mathcal{L}_i)$ that together with $\varphi_{|X(\mathcal{L}_i)}$ satisfy $\mathcal{L}_i$.*

4.4.4 Solving the IMP

In this section we focus on solving the IMP for constraint languages that are invariant under the affine operation of an Abelian group. In fact, we will prove that in such cases one can construct a d-truncated Gröbner Basis.

Theorem 4.4.8. *Let $\mathbb{A}$ be an Abelian group. Then $IMP_d(\Delta)$ is polynomial time decidable for any finite constraint language Δ which is invariant under the affine operation of $\mathbb{A}$.*

The rest of this section is devoted to proving Theorem 4.4.8. We use the notation from Section 4.4.1. Let $\mathbb{A}$ be an Abelian group and Δ a finite constraint language invariant under the affine operation of $\mathbb{A}$. We will provide a polynomial time algorithm that, for any instance $\mathcal{P} = (X, \mathbb{A}, \mathcal{C})$ of $\mathrm{CSP}(\Delta)$, decides if an input polynomial $f \in \mathbb{C}[X]$ belongs to $\mathtt{I}(\mathcal{P}) \subseteq \mathbb{C}[X]$. As before, we assume

$$\mathbb{A} = \mathbb{Z}_{p_1^{\ell_{1,1}}} \oplus \cdots \oplus \mathbb{Z}_{p_1^{\ell_{1,k_1}}} \oplus \mathbb{Z}_{p_2^{\ell_{2,1}}} \cdots \oplus \mathbb{Z}_{p_2^{\ell_{2,k_2}}} \oplus \cdots \oplus \mathbb{Z}_{p_s^{\ell_{s,k_s}}},$$

and m_r is maximal among $\ell_{r,1}, \ldots, \ell_{r,k_r}$. By Proposition 4.4.3, Δ is pp-interpretable in a multi-sorted constraint language Γ which is invariant under the affine operation of $\mathbb{Z}_{p_1^{m_1}}, \ldots, \mathbb{Z}_{p_s^{m_s}}$. By Theorem 3.3.7, since multi-sorted constraint language Γ pp-interprets Δ then $\mathrm{IMP}_d(\Delta)$ is polynomial time reducible to $\mathrm{IMP}_{O(d)}(\Gamma)$. Combined with Proposition 4.4.7 this yields the following statement.

Proposition 4.4.9. *Let Δ be a constraint language that is invariant under the affine operation of $\mathbb{A}$. Then $IMP_d(\Delta)$ is polynomial time reducible to $IMP_{O(d)}(\Gamma)$ with Γ being a constraint language invariant under the affine operation of $\mathbb{Z}_{p_1^{m_1}}, \ldots, \mathbb{Z}_{p_s^{m_s}}$. Moreover, every instance $(f, \mathcal{P})$ of $IMP_d(\Delta)$ is transformed to an instance $(f', \mathcal{P}')$ of $IMP_{O(d)}(\Gamma)$ satisfying the following conditions.*

(1) For every $i \in [s]$ there is a set $Y_i = \{y_{i,1}, \ldots, y_{i,r_i}\}$ of variables of $\mathcal{P}'$ and $Y_i \cap Y_j = \emptyset$ for $i \neq j$.

(2) For every constraint $\langle \mathbf{s}, R \rangle$ of $\mathcal{P}'$ the following conditions hold:

(a) there is $i \in [s]$ such that $\mathbb{Z}_{p_i^{m_i}}$ is the domain of every variable from $\mathbf{s}$;

(b) R is represented by a linear equation of the form

$$x_j = \alpha_1 y_{i,1} + \cdots + \alpha_{r_i} y_{i,r_i}$$

over $\mathbb{Z}_{p_i^{m_i}}$.

For an instance $(f, \mathcal{P})$ of $\mathrm{IMP}_d(\Gamma)$ we assume that

$$\mathcal{P} = (X \cup Y, \mathcal{D}, \delta, \mathcal{C}),$$

with $X = \{x_{1,1}, \ldots, x_{1,k_1}, \ldots, x_{s,1}, \ldots, x_{s,k_s}\}$, $Y = \{y_{1,1}, \ldots, y_{1,r_1}, \ldots, y_{s,1}, \ldots, y_{s,r_s}\}$ $\mathcal{D} = \{\mathcal{D}_i \mid \mathcal{D}_i = \mathbb{Z}_{p_i^{m_i}}, 1 \leq i \leq s\}$ and $\delta : X \cup Y \to [s]$ defined as $\delta(x_{i,j}) = \delta(y_{i,j}) = i$. Furthermore, the input polynomial f is from $\mathbb{C}[x_{1,1}, \ldots, x_{1,k_1}, \ldots, x_{s,1}, \ldots, x_{s,k_s}]$.

In the next sections, we present a reduction that transforms the problem to an equivalent problem over roots of unities and then compute a truncated Gröbner Basis.

Reduction to roots of unity

By Propositions 4.4.7 and 4.4.9 any instance of $\mathrm{CSP}(\Gamma)$ can be thought of as a system of linear equations.

Note that a system of linear equations over $\mathbb{Z}_{p_i^{m_i}}$ can be solved in polynomial time. This immediately tells us if $1 \in \mathcal{I}(\mathcal{P})$ or not, and we proceed only when $1 \notin \mathcal{I}(\mathcal{P})$. We assume the lexicographic order $\succ_{\mathsf{lex}}$ with

$$\begin{aligned} x_{1,1} \succ_{\mathsf{lex}} \cdots \succ_{\mathsf{lex}} x_{1,k_1} \succ_{\mathsf{lex}} \cdots \succ_{\mathsf{lex}} x_{s,1} \succ_{\mathsf{lex}} \cdots \succ_{\mathsf{lex}} x_{s,k_s} \\ \succ y_{1,1} \succ \cdots \succ y_{1,r_1} \succ y_{2,1} \succ \cdots \succ y_{2,r_2} \succ \cdots \succ y_{s,r_s}. \end{aligned} \tag{4.3}$$

Since these systems of linear equations do not share any variables we construct a truncated Gröbner Basis for each of them independently, and then will show that the union of all these Gröbner Bases is indeed a Gröbner Basis for $\mathcal{I}(\mathcal{P})$. We denote the corresponding ideal for each $\mathcal{L}_i$ by $\mathcal{I}(\mathcal{L}_i)$.

Note that each linear system $\mathcal{L}_i$ is already in its reduced row-echelon form with $x_{i,j}$ as the leading monomial of the j-th equation, $1 \leq j \leq k_i$. Each linear equation can be written as $x_{i,j} + f_{i,j} = 0 \pmod{p_i^{m_i}}$ where $f_{i,j}$ is a linear polynomial over $\mathbb{Z}_{p_i^{m_i}}$. This is elaborated on as follows.

$$\mathcal{L}_i := \left\{ \begin{array}{rcl} x_{i,1} + \overbrace{\alpha_{1,1}\, y_{i,1} + \cdots + \alpha_{1,r_i}\, y_{i,r_i} + \alpha_1}^{f_{i,1}} & = & 0 \pmod{p_i^{m_i}} \\ x_{i,2} + \overbrace{\alpha_{2,1}\, y_{i,1} + \cdots + \alpha_{2,r_i}\, y_{i,r_i} + \alpha_2}^{f_{i,2}} & = & 0 \pmod{p_i^{m_i}} \\ \vdots & & \\ x_{i,k_i} + \overbrace{\alpha_{k_i,1}\, y_{i,1} + \cdots + \alpha_{k_i,r_i}\, y_{i,r_i} + \alpha_{r_i}}^{f_{i,k_i}} & = & 0 \pmod{p_i^{m_i}} \end{array} \right. \longrightarrow \mathcal{L}_i := \left\{ \begin{array}{rcl} x_{i,1} + f_{i,1} & = & 0 \pmod{p_i^{m_i}} \\ x_{i,2} + f_{i,2} & = & 0 \pmod{p_i^{m_i}} \\ & \vdots & \\ x_{i,k_i} + f_{i,k_i} & = & 0 \pmod{p_i^{m_i}} \end{array} \right.$$

Hence, we can write down a generating set for $\mathcal{I}(\mathcal{L}_i)$ in an implicit form as follows where the addition is modulo $\mathbb{Z}_{p_i^{m_i}}$,

$$G_i = \left\{ x_{i,1} + f_{i,1}, \ldots, x_{i,k_i} + f_{i,k_i}, \prod_{j \in \mathbb{Z}_{p_i^{m_i}}} (y_{i,1} - j), \ldots, \prod_{j \in \mathbb{Z}_{p_i^{m_i}}} (y_{i,r_i} - j) \right\} \tag{4.4}$$

Let $U_{p_i^{m_i}} = \{\omega_i, \omega_i^2, \ldots, \omega_i^{(p_i^{m_i})} = \omega_i^0 = 1\}$ be the set of $p_i^{m_i}$-th roots of unity where ω_i is a primitive $p_i^{m_i}$-th root of unity. For a primitive $p_i^{m_i}$-th root of unity ω_i we have $\omega_i^a = \omega_i^b$ if and only if $a \equiv b \pmod{p_i^{m_i}}$. From $\mathcal{L}_i$ we construct a new CSP instance $\mathcal{L}_i' = (V, U_{p_i^{m_i}}, \widetilde{C})$ where for each equation $x_{i,t} + f_{i,t} = 0 \pmod{p_i^{m_i}}$ we add the constraint $x_{i,t} - f_{i,t}' = 0$ with

$$f_{i,t}' = \omega_i^{\alpha_t} \cdot \left(y_{i,1}^{\alpha_{t,1}} \cdot \ldots \cdot y_{i,r_i}^{\alpha_{t,r_i}} \right).$$

Moreover, the domain constraints are different. For each variable $x_{i,j}$, $j \in [k_i]$, or $y_{i,j}$, $j \in [r_i]$ the domain polynomial is $(x_{i,j})^{(p_i^{m_i})} = 1$, $(y_{i,j})^{(p_i^{m_i})} = 1$. Therefore, we represent G_i from (4.4) over complex number domain as follows.

$$G_i' = \left\{ x_{i,1} - f_{i,1}', \ldots, x_{i,k_i} - f_{i,k_i}', (y_{i,1})^{(p_i^{m_i})} - 1, \ldots, (y_{i,r_i})^{(p_i^{m_i})} - 1 \right\} \tag{4.5}$$

Define univariate polynomial $\phi_i \in \mathbb{C}[X]$ so that it interpolates points

$$(0, \omega_i^0), (1, \omega_i), \ldots, (p_i^{m_i} - 1, \omega_i^{(p_i^{m_i - 1})}).$$

This polynomial provides a one-to-one mapping between solutions of $\mathcal{L}_i$ and $\mathcal{L}_i'$. That is, $(a_{i,1}, \ldots, a_{i,k_i}, b_{i,1}, \ldots, b_{i,r_i})$ is a solution of $\mathcal{L}_i$ if and only if

$$(\phi_i(a_{i,1}), \ldots, \phi_i(a_{i,k_i}), \phi_i(b_{i,1}), \ldots, \phi_i(b_{i,r_i}))$$

is a solution of $\mathcal{L}_i'$.

For an instance $\mathcal{P}$ of CSP(Γ), which is a collection of systems of linear equations $\mathcal{L}_1, \ldots, \mathcal{L}_s$, define the instance $\mathcal{P}'$ which is a collection of systems of linear equations $\mathcal{L}_1', \ldots, \mathcal{L}_s'$. In the next lemma we prove our transformation to roots of unity gives rise to an equivalent ideal membership problem.

Lemma 4.4.10. *For a polynomial $p \in \mathbb{C}[X]$ define polynomial $p' \in \mathbb{C}[X]$ to be*

$$\begin{aligned} &p'(x_{1,1}, \ldots, x_{1,k_1}, \ldots, x_{s,1}, \ldots, x_{s,k_s}) \\ &\qquad = p\left(\phi_1^{-1}(x_{1,1}), \ldots, \phi_1^{-1}(x_{1,k_1}), \ldots, \phi_s^{-1}(x_{s,1}), \ldots, \phi_s^{-1}(x_{s,k_s}) \right). \end{aligned}$$

Then $p \in \mathtt{I}(\mathcal{P})$ if and only if $p' \in \mathtt{I}(\mathcal{P}')$.

Proof. Recall that $\mathcal{I}(\mathcal{P})$ is a radical ideal, then by the Strong Nullstellensatz we have $p \notin \mathcal{I}(\mathcal{P})$ if and only if there exists a point

$$\mathbf{a} \in \mathbb{Z}_{p_1^{m_1}}^{k_1} \times \cdots \times \mathbb{Z}_{p_s^{m_s}}^{k_s}$$

such that $\mathbf{a}$ is in $\mathsf{Sol}(\mathcal{P})$ and $p(\mathbf{a}) \neq 0$. Similarly, as $\mathcal{I}(\mathcal{P}')$ is radical [1] then $p' \notin \mathcal{I}(\mathcal{P}')$ if and only if there exists a point $(\mathbf{a}', \mathbf{a}'')$,

$$\mathbf{a}' \in U_{p_1^{m_1}}^{k_1} \times \cdots \times U_{p_s^{m_s}}^{k_s}, \qquad \mathbf{a}'' \in U_{p_1^{m_1}}^{r_1} \times \cdots \times U_{p_s^{m_s}}^{r_s}$$

such that $(\mathbf{a}', \mathbf{a}'') \in \mathsf{Sol}(\mathcal{P}')$ and $p'(\mathbf{a}') \neq 0$.

Moreover, by our construction, $\mathbf{a} = (a_{1,1}, \ldots, a_{1,k_1}, a_{2,1}, \ldots, a_{2,k_2}, \ldots, a_{s,k_s})$ is a solution of $\mathcal{P}$ if and only if

$$\mathbf{a}' = (\phi_1(a_{1,1}), \ldots, \phi_1(a_{1,k_1}), \phi_2(a_{2,1}), \ldots, \phi_2(a_{2,k_2}), \ldots, \phi_s(a_{s,k_s}))$$

can be extended to a solution of $\mathcal{P}'$. Finally,

$$p'\left(\phi_1^{-1}(a_{1,1}), \ldots, \phi_1^{-1}(a_{1,k_1}), \phi_2^{-1}(a_{2,1}), \ldots, \phi_2^{-1}(a_{2,k_2}), \ldots, \phi_s^{-1}(a_{s,k_s})\right) = 0$$

if and only if $p(\mathbf{a}) = 0$. This finishes the proof. □

4.4.5 Gröbner Bases for the problem over roots of unity

Having transformed the problem to a problem over (multi-sorted) roots of unity has a huge advantage, namely these new generating sets corresponding to each $\mathcal{L}_i$ are indeed Gröbner Bases. We first verify this for each $\mathcal{L}_i$, then will show the union of all these generating sets gives a Gröbner Basis for the entire problem.

Lemma 4.4.11. *For each $1 \leq i \leq s$, the set of polynomials G_i' in (4.5) is a Gröbner Basis for $\mathtt{I}(\mathcal{L}_i') = \mathbf{I}(\mathsf{Sol}(\mathcal{L}_i'))$ with respect to lex order $x_{i,1} \succ \cdots \succ x_{i,k_i} \succ y_{i,1} \succ \cdots \succ y_{i,r_i}$.*

Proof. The proof has two parts. In the first part we show that G_i' is a Gröbner Basis by showing that it satisfies the Buchberger's Criterion, Theorem 2.1.20. In the second part we show the generating ideal by G_i' is equivalent to the vanishing ideal of $\mathsf{Sol}(\mathcal{L}_i')$.

Consider $\langle G_i' \rangle$. We show that G_i' is a Gröbner Basis for $\langle G_i' \rangle$ by verifying that the leading monomials of every pair of polynomials in G_i' are relatively prime. For each $x_{i,j} - f_{i,j}'$, we have $\mathrm{LM}(x_{i,j} - f_{i,j}') = x_{i,j}$. Moreover, the leading monomial of $(x_{i,j})^{(p_i^{m_i})} - 1$ or $(y_{i,j})^{(p_i^{m_i})} - 1$ is $(x_{i,j})^{(p_i^{m_i})}$ and $(y_{i,j})^{(p_i^{m_i})}$, respectively. Hence, for every pair of polynomials in G_i' the leading monomials are relatively prime which, by Proposition 2.1.21, implies their reduced S-polynomial is zero. By Buchberger's Criterion, Theorem 2.1.20, it follows that G_i' is a Gröbner Basis for $\langle G_i' \rangle$ (according to the lex order).

It remains to show $\langle G_i' \rangle = \mathbf{I}(\mathsf{Sol}(\mathcal{L}_i'))$. It is easy to see that $\langle G_i' \rangle \subseteq \mathbf{I}(\mathsf{Sol}(\mathcal{L}_i'))$. This is because, by our construction we have $\mathbf{V}(\langle G_i' \rangle) = \mathsf{Sol}(\mathcal{L}_i')$ and hence any polynomial

[1] For all $i \in [s]$ and for all $j \in \{1, \ldots, k_i\}$, the domain polynomial $(x_{i,j})^{(p_i^{m_i})} - 1$ belongs to the idea $\mathcal{I}(\mathcal{P}')$ i.e. remainder of division of $(x_{i,j})^{(p_i^{m_i})} - 1$ by $(x_{i,j}) - f_{i,j}'$ is $0 = 1 - (f_{i,j}')^{(p_i^{m_i})}$.

$p \in \langle G'_i \rangle$ is zero over all the points in $\mathsf{Sol}(\mathcal{L}'_i)$ which implies $p \in \mathbf{I}(\mathsf{Sol}(\mathcal{L}'_i))$. Next we prove $\mathbf{I}(\mathsf{Sol}(\mathcal{L}'_i)) \subseteq \langle G'_i \rangle$. Consider a polynomial $p \in \mathbf{I}(Sol(\mathcal{L}'_i))$. We prove $p|_{G'_i} = 0$. Note that $p(\mathbf{a}) = 0$ for all $\mathbf{a} \in \mathsf{Sol}(\mathcal{L}'_i)$. Let $q = p|_{G'_i}$. Because of $x_{i,1} - f'_{i,1}, \ldots, x_{i,k_i} - f'_{i,k_i}$ in G', the polynomial q does not contain variables $x_{i,1}, \ldots, x_{i,k_i}$, and hence it is a polynomial in $y_{i,1}, \ldots, y_{i,r_i}$. Now note that any $\mathbf{b} = (b_1, \ldots, b_{r_i}) \in U_{p_i^{m_i}}^{r_i}$ extends to a unique point in $\mathsf{Sol}(\mathcal{L}'_i)$, this is because in G'_i (similarly in G_i and $\mathcal{L}_i$) all the $x_{i,1}, \ldots, x_{i,k_i}$ have coefficients and exponent equal to 1. Therefore, all the points in $U_{p_i^{m_i}}^{r_i}$ are zeros of q, hence $q = p|_{G'_i}$ is the zero polynomial. Since $p \in \mathbf{I}(\mathsf{Sol}(\mathcal{L}'_i))$ was arbitrary chosen, it follows that for every $p \in \mathbf{I}(\mathsf{Sol}(\mathcal{L}'_i))$ we have $p|_{G'_i} = 0$. Hence, $\mathbf{I}(\mathsf{Sol}(\mathcal{L}'_i)) \subseteq \langle G'_i \rangle$. This finishes the proof. □

Given the above lemma we prove $G' = \cup_{1 \leq i \leq s} G'_i$ is a Gröbner Basis for $\mathcal{I}(\mathcal{P}') = \mathbf{I}(\mathsf{Sol}(\mathcal{P}'))$ with respect to the lex order (4.3).

Lemma 4.4.12. *The set of polynomials $G' = \cup_{1 \leq i \leq s} G'_i$ forms a Gröbner Basis for $\mathtt{I}(\mathcal{P}') = \mathbf{I}(\mathit{Sol}(\mathcal{P}'))$ with respect to the lex order $x_{1,1} \succ_{\mathsf{lex}} \cdots \succ_{\mathsf{lex}} x_{1,k_1} \succ_{\mathsf{lex}} \cdots \succ_{\mathsf{lex}} x_{s,1} \succ_{\mathsf{lex}} \cdots \succ_{\mathsf{lex}} x_{s,k_s} \succ y_{1,1} \succ_{\mathsf{lex}} \cdots \succ_{\mathsf{lex}} y_{1,r_1} \succ_{\mathsf{lex}} \cdots \succ_{\mathsf{lex}} y_{s,1} \succ_{\mathsf{lex}} \cdots \succ_{\mathsf{lex}} y_{s,r_s}$.*

Proof. Recall that

$$\begin{aligned}
\mathsf{Sol}(\mathcal{P}') &= \mathsf{Sol}(\mathcal{L}'_1) \cap \cdots \cap \mathsf{Sol}(\mathcal{L}'_s) \\
&= \mathbf{V}\left(\mathbf{I}(\mathsf{Sol}(\mathcal{L}'_1))\right) \cap \cdots \cap \mathbf{V}\left(\mathbf{I}(\mathsf{Sol}(\mathcal{L}'_s))\right) \\
&= \mathbf{V}\left(\langle G'_1 \rangle\right) \cap \cdots \cap \mathbf{V}\left(\langle G'_s \rangle\right) && \text{(by Lemma 4.4.11)} \\
&= \mathbf{V}\left(\langle G'_1 \rangle + \cdots + \langle G'_s \rangle\right) && \text{(by Theorem 2.1.8)}
\end{aligned}$$

This implies,

$$\mathcal{I}(\mathcal{P}') = \mathbf{I}(\mathsf{Sol}(\mathcal{P}')) = \langle G'_1 \rangle + \cdots + \langle G'_s \rangle .$$

Now by Lemma 4.4.11 each G'_i is a Gröbner Basis. Moreover, observe that for all distinct i and j the ideals $\langle G'_i \rangle$ and $\left\langle G'_j \right\rangle$ do not share variables. Hence, the set of polynomials $G'_1 \cup \cdots \cup G'_s$ is indeed a Gröbner Basis for $\mathbf{I}(\mathsf{Sol}(\mathcal{P}'))$, according to the lex order. □

Lemma 4.4.10 states that checking if a polynomial p is in $\mathcal{I}(\mathcal{P})$ is equivalent to checking if the polynomial p' is in $\mathcal{I}(\mathcal{P}')$. However, the substitution technique used in Lemma 4.4.10 may result in a polynomial with exponentially many monomials and hence we only consider polynomials of bounded degree. Provided that G' is a Gröbner Basis for $\mathcal{I}(\mathcal{P}')$ with respect to the lex order, see Lemma 4.4.12, we can test membership of any bounded degree polynomial in polynomial time. Note that division of any polynomial by polynomials $x_{i,1} - f'_{i,1}, \ldots, x_{i,k_i} - f'_{i,k_i}, i \in [s]$ results in a polynomial over variables $y_{i,1}, \ldots, y_{i,r_i}$, $i \in [s]$, where

$$\cup_{i \in [s]} \{(y_{i,1})^{(p_i^{m_i})} - 1, \ldots, (y_{i,r_i})^{(p_i^{m_i})} - 1\}$$

is a Gröbner Basis with respect to the grlex order in $\mathbb{C}[\cup_{i\in s}\{y_{i,1},\ldots,y_{i,r_i}\}]$. This gives the following theorem.

Theorem 4.4.13. *Let Γ be a finite multi-sorted constraint language which is invariant under the affine operation of $\mathbb{Z}_{p_1^{m_1}},\ldots,\mathbb{Z}_{p_s^{m_s}}$. Then $IMP_d(\Gamma)$ is decidable in polynomial time.*

The above theorem together with Proposition 4.4.9 imply that the $\mathrm{IMP}_d(\Delta)$ is polynomial time decidable for constraint language Δ that is invariant under the affine operation of an Abelian group.

Theorem 4.4.14. *Let Δ be a finite constraint language invarient under the affine operation of Abelian group $\mathbb{A}$. Then $IMP_d(\Delta)$ is decidable in polynomial time.*

Later in Chapter 5 we will prove that in fact we can find a proof of membership, if one exists, for constraint languages over Abelian groups. In fact, we provide a polynomial time algorithm to construct a d-truncated Gröbner Basis with respect to a grlex .

4.5 Gröbner Bases for linear system in $\mathsf{GF}(p)$ via conversion technique

In this section we focus on constraint languages that are expressible as a system of linear equations modulo a prime number. Let Γ be a constraint language over a set D with $|D| = p$, and p a prime number. Suppose Γ has an *affine* polymorphism modulo p (i.e. a ternary operation $\psi(x,y,z) = x \ominus y \oplus z$, where $\oplus, \ominus$ are addition and subtraction modulo p, or, equivalently, of the field $\mathsf{GF}(p)$). In this case every CSP can be represented as a system of linear equations over $\mathsf{GF}(p)$. Without loss of generality, we may assume that the system of linear equations at hand is already in the *reduced row echelon* form. Transforming system of linear equations mod p in its reduced row echelon form to a system of polynomials in $\mathbb{R}[X]$ that are a Gröbner Basis is not immediate and requires substantial work. This is the case even if we restrict ourselves to lexicographic order. Let us elaborate on this by an example considering linear equations over $GF(2)$.

Example 4.5.1. We assume a lexicographic order $\succ$ with $x_1 \succ_{\mathsf{lex}} \cdots \succ_{\mathsf{lex}} x_n$. We also assume that the linear system has $r \leq n$ equations and is already in its reduced row echelon form with x_i as the leading monomial of the i-th equation. Let $Supp_i \subset [n]$ such that $\{x_j : j \in Supp_i\}$ is the set of variables appearing in the i-th equation of the linear system except for x_i. Let the i-th equation be $g_i = 0(mod\ 2)$ where $g_i = x_i \oplus f_i$, with $i \in [r]$ and f_i is the Boolean function $(\bigoplus_{j\in Supp_i} x_j) \oplus \alpha_i$ and $\alpha_i \in \{0,1\}$. Define a polynomial $M(f_i) \in \mathbb{R}[x_1,\ldots,x_n]$ interpolating f_i, that is, such that, for every $\mathbf{a} \in \{0,1\}^n$, $f_i(\mathbf{a}) = 0$ if and only if $M(f_i)(\mathbf{a}) = 0$, and $f_i(\mathbf{a}) = 1$ if and only if $M(f_i)(\mathbf{a}) = 1$. Now, consider the

following set of polynomials.

$$G = \{x_1 - M(f_1), \ldots, x_r - M(f_r), (x_{r+1}^2 - x_{r+1}), \ldots, (x_n^2 - x_n)\}.$$

Set $G \subset \mathbb{R}[x_1, \ldots, x_n]$ is a Gröbner Basis with respect to lex order. This is because for every pair of polynomials in G the reduced S-polynomial is zero as the leading monomials of any two polynomials in G are relatively prime. By Buchberger's Criterion (see Theorem 2.1.20) it follows that G is a Gröbner Basis with respect to the lex ordering.

However, this construction may not be computationally efficient as the polynomials $M(f_i)$ may have exponentially many monomials. This case was overlooked in [161]. Bharathi and Mastrolilli [25] resolved this issue in an elegant way. Having $G' = \{x_1 - f_1, \ldots, x_r - f_r, (x_{r+1}^2 - x_{r+1}), \ldots, (x_n^2 - x_n)\}$, they convert G' to set of polynomials in $\mathbb{R}[x_1, \ldots, x_n]$ which is a d-truncated Gröbner Basis. Their conversion algorithm has time complexity $n^{O(d)}$ where $d = O(1)$. Their algorithm is a modification of the conversion algorithm by Faugère, Gianni, Lazard and Mora [75]. See [25] for more details.

We consider this problem for any fixed prime p and prove that a d-truncated Gröbner Basis can be computed in time $n^{O(d)}$. First, we give a very brief introduction to the FGLM conversion algorithm [75]. Next, we present our conversion algorithm that, given a system of linear equations mod p, produces a d-truncated Gröbner Basis in graded lexicographic order. The heart of our algorithm is finding linearly independent expressions mod p that helps us carry the conversion.

4.5.1 Gröbner Basis conversion

We say a Gröbner Basis $G = \{g_1, \ldots, g_t\}$ is *reduced* if $\mathrm{LC}(g_i) = 1$ for all $g_i \in G$, and if for all $g_i \in G$ no monomial of g_i lies in $\langle \mathrm{LT}(G \setminus \{g_i\}) \rangle$. We note that for an ideal and a given monomial ordering the reduced Gröbner Basis of $\mathtt{I}$ is unique (see, e.g., [59], Theorem 5, p.93). Given the reduced Gröbner Basis of a zero-dimensional ideal $\mathtt{I} \subset \mathbb{F}[X]$ with respect to a monomial order $\succ_1$, where $\mathbb{F}$ is a computable field, the FGLM algorithm computes a Gröbner Basis of $\mathtt{I}$ with respect to another monomial order $\succ_2$. The complexity of the FGLM algorithm depends on the dimension of $\mathbb{F}$-vector space $\mathbb{F}[X]/\mathtt{I}$. More precisely, let $\mathscr{D}(I)$ denote the dimension of $\mathbb{F}$-vector space $\mathbb{F}[X]/\mathtt{I}$, then we have the following proposition.

Proposition 4.5.2 (Proposition 4.1 in [75])**.** *Let* $\mathtt{I}$ *be a zero-dimensional ideal and* G_1 *be the reduce Gröbner Basis with respect to an ordering* $\succ_1$*. Given a different ordering* $\succ_2$*, there is an algorithm that constructs a Gröbner Basis* G_2 *with respect to ordering* $\succ_2$ *in time* $O(n\mathscr{D}(I)^3)$*.*

We cannot apply the FGLM algorithm directly as $\mathscr{D}(\mathtt{I})$ could be exponentially large in our setting. Note that $\mathscr{D}(\mathtt{I})$ is equal to the number of common zeros of the polynomials from $\langle G_1 \rangle$, which in the case of linear equations is $\mathscr{D}(\mathtt{I}) = O(p^{n-r})$ where r is the number of

Algorithm 1 Conversion algorithm

Require: G_1 as in (4.6) that corresponds to $\mathcal{I}(\mathcal{P})$, degree d.

1: Let Q be the list of all monomials of degree at most d arranged in increasing order with respect to grlex .
2: $G_2 = \emptyset, B(G_2) = \{1\}$ (we assume $1 \notin \mathcal{I}(\mathcal{P})$). Let b_i (arranged in increasing grlex order) be the elements of $B(G_2)$.
3: **for** $q \in Q$ **do**
4: **if** q is divisible by some LM in G_2 **then**
5: Discard it and go to Step 3,
6: **if** $q|_{G_1} = \sum_j k_j b_j|_{G_1}$ **then** ▷ where $k_j \in \mathbb{R}, b_j \in B(G_2)$
7: $G_2 = G_2 \cup \{q - \sum_j k_j b_j\}$
8: **else**
9: $B(G_2) = B(G_2) \cup \{q\}$
10: **return** G_2

equations in the reduced row echelon form. Furthermore, as we discussed, we are not given the explicit reduced Gröbner Basis G_1 with respect to lex ordering (the Gröbner Basis is presented to us as a system of linear equation mod p rather than polynomials in $\mathbb{R}[X]$). We shall present an algorithm that resolves these issues.

Let $\mathcal{P}$ be an instance of CSP(Γ) that is expressed as a system of linear equations $\mathscr{S}$ over $\mathbb{Z}_p$ with variables $x_1, \ldots, x_n$. A system of linear equations over $\mathbb{Z}_p$ can be solved by Gaussian elimination (this immediately tells us if $1 \in \mathcal{I}(\mathcal{P})$ or not, and we proceed only if $1 \notin \mathcal{I}(\mathcal{P})$). We assume a lexicographic order $\succ_{\mathsf{lex}}$ with $x_1 \succ_{\mathsf{lex}} \cdots \succ_{\mathsf{lex}} x_n$. We also assume that the linear system has $r \leq n$ equations and it is already in its reduced row echelon form with x_i as the leading monomial of the i-th equation. Let $Supp_i \subset [n]$ such that $\{x_j : j \in Supp_i\}$ be the set of variables appearing in the i-th equation of the linear system except for x_i.

Fix a prime p and $\oplus$, $\ominus$, $\odot$ denote addition, subtraction, and multiplication modulo p, respectively. We will call a linear polynomial over $\mathbb{Z}_p$ a *p-expression*. Let the i-th equation be $g_i = 0 \pmod p$ where $g_i := x_i \ominus f_i$, with $i \in [r]$ and f_i is the p-expression $(\bigoplus_{j \in Supp_i} \alpha_j x_j) \oplus \alpha_i$ and $\alpha_j, \alpha_i \in \mathbb{Z}_p$. We will assume that each variable x_i is associated with its p-expression f_i which comes from the mod p equations. This is clear for $i \leq r$; for $i > r$ the p-expression $f_i = x_i$ itself. Hence, we can write down the reduced Gröbner Basis in the lex order in an implicit form as follows.

$$G_1 = \{x_1 - f_1, \ldots, x_r - f_r, \prod_{i \in \mathbb{Z}_p} (x_{r+1} - i), \ldots, \prod_{i \in \mathbb{Z}_p} (x_n - i)\} \tag{4.6}$$

Given G_1, our conversion algorithm, Algorithm 1, constructs a d-truncated Gröbner Basis over $\mathbb{R}[x_1, \ldots, x_n]$ with respect to the grlex order. At the beginning of the algorithm, there will be two sets: G_2, which is initially empty but will become the new Gröbner Basis with respect to the grlex order, and $B(G_2)$, which initially contains 1 and will grow to be the

grlex monomial basis of the quotient ring $\mathbb{R}[x_1, \ldots, x_n]/\mathcal{I}(\mathcal{P})$ as a $\mathbb{R}$-vector space. In fact, $B(G_2)$ contains the reduced monomials (of degree at most d) with respect to G_2. Every $f \in \mathbb{R}[x_1, \ldots, x_n]$ is congruent modulo $\mathcal{I}(\mathcal{P})$ to a unique polynomial r which is a $\mathbb{R}$-linear combination of the monomials in the *complement* of $\langle \mathrm{LT}(\mathcal{I}(\mathcal{P}))\rangle$. Furthermore, the elements of $\{\mathbf{x}^\alpha \mid \mathbf{x}^\alpha \notin \langle \mathrm{LT}(\mathcal{I}(\mathcal{P}))\rangle\}$ are "linearly independent modulo $\mathcal{I}(\mathcal{P})$" (see Proposition 5.1.8). This suggests the following. In Algorithm 1, Q is the list of all monomials of degree at most d arranged in increasing order with respect to grlex ordering. The algorithm iterates over monomials in Q in increasing grlex order and at each iteration decides exactly one of the followings given the current sets G_2 and $B(G_2)$.

1. q should be discarded (if q is divisible by some LM in G_2), or
2. a polynomial with q as its leading monomial should be added to G_2 (if $q|_{G_1} = \sum_j k_j b_j|_{G_1}$; $b_j \in B(G_2)$), or
3. q should be added to $B(G_2)$.

The trickiest part is to decide if the current monomial $q|_{G_1}$ is a $\mathbb{R}$-linear combination of $b_j|_{G_1}$ with b_j being the current elements in $B(G_2)$. Provided this can be done correctly and in polynomial time, the correctness of Algorithm 1 follows by the analyses in [75] and it runs in polynomial time as there are at most $O(n^d)$ monomials in Q. The rest of this section is devoted to provide a polynomial time procedure that correctly decides if $q|_{G_1} = \sum_j k_j b_j|_{G_1}$ holds for the current monomial q and the current b_js in $B(G_2)$.

4.5.2 Expansion in a basis of p-expressions

For a monomial q, the normal form of q by G_1, $q|_{G_1}$, is the remainder on division of q by G_1 in the lex order. $q|_{G_1}$ is unique and it does not matter how the elements of G_1 are listed when using the division algorithm. Here, it suffices for us to write $q|_{G_1}$ in terms of product of p-expressions. We start with a simple observation. Recall that r denotes the number of linear equations in the reduced row echelon form.

Observation 4.5.3. *Let $q = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ be a monomial such that for all $r < i$ we have $\alpha_i \leq p-1$. Then, $q|_{G_1} = f_1^{\alpha_1} \cdots f_n^{\alpha_n}$ where each f_i is the p-expression associated to x_i.*

A keystone of our conversion algorithm is a relation between a product of p-expressions and a sum of p-expressions. Intuitively speaking, we will prove that a product of p-expressions can be written as a $\mathbb{R}$-linear combination of (linearly independent) p-expressions. Indeed, we provide a set of p-expressions and prove the p-expressions in this set are linearly independent and span the space of functions from $\mathbb{Z}_p^d$ to $\mathbb{C}$. Consider the set $\mathcal{V}_{n,p}$ of functions from $\mathbb{Z}_p^n$ to $\mathbb{C}$ as a p^n-dimensional vector space, whose components are values of the function

at the corresponding point of $\mathbb{Z}_p^n$. Let

$$F_n = \left\{ \bigoplus_{i=1}^{n} \alpha_i x_i \oplus x_{n+1} \oplus \beta \mid \alpha_i \in \{0, \ldots, p-1\}, i \in [n], \beta \in \{0, \ldots, p-2\} \right\}$$

be a collection of linear functions over $\mathbb{Z}_p$, and let

$$\mathcal{F}_n = F_n \cup \cdots \cup F_0 \cup \{1\}.$$

Theorem 4.5.4. *For any n, the collection $\mathcal{F}_n$ of p-expressions is linearly independent as a set of vectors from $\mathcal{V}_{n+1,p}$ and forms a basis of $\mathcal{V}_{n+1,p}$.*

As a first application of Theorem 4.5.4 we show that any p-expression can be written as a $\mathbb{R}$-linear combination of p-expressions in our basis $\mathcal{F}_n$.

Lemma 4.5.5. *Any p-expression $\alpha_1 x_1 \oplus \alpha_2 x_2 \oplus \cdots \oplus \alpha_n x_n \oplus \beta$ with $\alpha_n \neq 0$ can be represented by a $\mathbb{R}$-linear combination of the p-expression basis from $\mathcal{F}_{n-1}$.*

Proof. By Theorem 4.5.4, for any x and y, the expression $y \oplus \alpha x \oplus \beta$ can be written as a $\mathbb{R}$-linear combination of p-expressions from $\mathcal{F}_1$. Such a $\mathbb{R}$-linear combination can be found in constant time $p^{O(1)}$ as the number of functions in $\mathcal{F}_1$ is p^2. We continue by assuming such a $\mathbb{R}$-linear combination of any $y \oplus \alpha x \oplus \beta$ is provided to us.

Now consider $\alpha_1 x_1 \oplus \alpha_2 x_2 \oplus \cdots \oplus \alpha_n x_n \oplus \beta$. Introduce a new variable y and set $y = \alpha_1 x_1 \oplus \alpha_2 x_2 \oplus \cdots \oplus \alpha_{n-1} x_{n-1}$. Hence, $\alpha_1 x_1 \oplus \alpha_2 x_2 \oplus \cdots \oplus \alpha_n x_n \oplus \beta = y \oplus \alpha_n x_n \oplus \beta$. By the above discussion, $y \oplus \alpha_n x_n \oplus \beta$ can be written as a $\mathbb{R}$-linear combination of p-expressions from $\mathcal{F}_1$. Therefore, there exist $c_\gamma, c_{\alpha\gamma}, \kappa \in \mathbb{R}$ so that

$$y \oplus \alpha_n x_n \oplus \beta = \sum_{\gamma=0}^{p-2} c_\gamma (y \oplus \gamma) + \sum_{\alpha \in [p-1], \gamma \in [p-2]} c_{\alpha\gamma} (\alpha y \oplus x_n \oplus \gamma) + \kappa \tag{4.7}$$

Note that p-expressions $y \oplus \gamma$ are in F_0, p-expressions $\alpha y \oplus x_n \oplus \gamma$ are in F_1, and κ is a constant. Substituting back for y, we observe that the second sum on the right hand side is already a $\mathbb{R}$-linear combination of p-expressions from F_{n-1}. Consider the first sum.

$$\sum_{\gamma=0}^{p-2} c_\gamma (y \oplus \gamma) = \sum_{\gamma=0}^{p-2} c_\gamma (\alpha_1 x_1 \oplus \alpha_2 x_2 \oplus \cdots \oplus \alpha_{n-1} x_{n-1} \oplus \gamma) \tag{4.8}$$

Now set $y = \alpha_1 x_1 \oplus \alpha_2 x_2 \oplus \cdots \oplus \alpha_{n-2} x_{n-2}$. This gives

$$\sum_{\gamma=0}^{p-2} c_\gamma (\alpha_1 x_1 \oplus \alpha_2 x_2 \oplus \cdots \oplus \alpha_{n-1} x_{n-1} \oplus \gamma) = \sum_{\gamma=0}^{p-2} c_\gamma (y \oplus \alpha_{n-1} x_{n-1} \oplus \gamma) \tag{4.9}$$

Note that each term $y \oplus \alpha_{n-1} x_{n-1} \oplus \gamma$ is expressible as a $\mathbb{R}$-linear combination of p-expressions from $\mathcal{F}_1$. This leads us to the following.

$$\sum_{\gamma=0}^{p-2} c_\gamma(y \oplus \alpha_{n-1}x_{n-1} \oplus \gamma)$$

$$= \sum_{\gamma=0}^{p-2} c_\gamma \left(\sum_{\delta=0}^{p-2} c_\delta(y \oplus \delta) + \sum_{\alpha\in[p-1],\delta\in[p-2]} c_{\alpha\delta}(\alpha y \oplus x_{n-1} \oplus \delta) + \kappa' \right) \tag{4.10}$$

$$= \sum_{\gamma=0}^{p-2} c_\gamma \left(\sum_{\delta=0}^{p-2} c_\delta(y \oplus \delta) \right) + \sum_{\gamma=0}^{p-2} c_\gamma \left(\sum_{\substack{\alpha\in[p-1],\\ \delta\in[p-2]}} c_{\alpha\delta}(\alpha y \oplus x_{n-1} \oplus \delta) \right) + \sum_{\gamma=0}^{p-2} c_\gamma\kappa'$$

$$= \sum_{\gamma=0}^{p-2} c'_\gamma(y \oplus \gamma) + \sum_{\substack{\alpha\in[p-1],\\ \gamma\in[p-2]}} c'_{\alpha\gamma}(\alpha y \oplus x_{n-1} \oplus \gamma) + \kappa'' \tag{4.11}$$

Note that p-expressions $y \oplus \gamma$ are in F_0, p-expressions $\alpha y \oplus x_{n-1} \oplus \gamma$ are in F_1, and κ'' is a constant. Substituting back for y, we observe that the second sum on the right hand side is already a $\mathbb{R}$-linear combination of p-expressions from F_{n-2}. Hence, it suffices to continue with the first term of the sum (4.11) which we can handle similar to the above procedure.

All in all, it requires to repeat the above procedure n times where at the i-th iteration we deal with a sum of $p-2$ p-expressions of form $\bigoplus_{i=1}^{n-i} \alpha_i x_i \oplus \beta$. This results in $O(np^{O(1)})$ running time. □

Another application of Theorem 4.5.4 is transforming a multiplication of p-expressions to an equivalent $\mathbb{R}$-linear combination of the basis in $\mathcal{F}_n$. Suppose $x_1, \ldots, x_d$ are (not necessary distinct) variables that take values $0, \ldots, p-1$. Let $x_1 \cdot x_2 \cdots x_d$ be their multiplication. In general, we are interested in a multiplication of p-expressions however, let us first discuss the simpler case of $x_1 \cdot x_2 \cdots x_d$. Unfortunately, the trick we used in the proof of Lemma 4.5.5 is no longer effective here. However, assuming d is a constant makes the situation easier. $x_1 \cdot x_2 \cdots x_d$ is a p^d-dimensional vector. By Theorem 4.5.4, the set $\mathcal{F}_{d-1}$ of p-expressions spans the set $\mathcal{V}_{d,p}$ of functions from $\mathbb{Z}_p^d$ to $\mathbb{C}$ as a p^d-dimensional vector space. Hence, in constant time (depending on p and d), we can have a $\mathbb{R}$-linear combination of the basis in $\mathcal{F}_{d-1}$ that represents $x_1 \cdot x_2 \cdots x_d$. We continue by assuming such a $\mathbb{R}$-linear combination of any $x_1 \cdot x_2 \cdots x_d$ is provided to us. The next lemma states that we can have a $\mathbb{R}$-linear combination of the basis in $\mathcal{F}_n$ for any $h_1 \cdot h_2 \cdots h_d$ where each h_i is a p-expression over variables $x_1, \ldots, x_n$.

Lemma 4.5.6. *Let $h_1, h_2, \ldots, h_d$ be (not necessary distinct) p-expressions over variables $x_1, \ldots, x_n$. The product $\mathscr{M} = h_1 \cdot h_2 \cdots h_d$ viewed as a function from $\mathbb{Z}_p^n$ to $\mathbb{C}$ can be represented as a $\mathbb{R}$-linear combination of the basis in $\mathcal{F}_{n-1}$.*

Proof. Let us treat h_is as indeterminates. Define

$$H_t = \left\{ \bigoplus_{i=1}^{t} \alpha_i h_i \oplus h_{t+1} \oplus \beta \mid \alpha_i \in \{0, \dots, p-1\}, i \in [t], \beta \in \{0, \dots, p-2\} \right\}$$

to be a collection of linear functions over $\mathbb{Z}_p$, and let

$$\mathcal{H}_{d-1} = H_{d-1} \cup \cdots \cup H_0 \cup \{1\}.$$

By Theorem 4.5.4 and the above discussion, $\mathscr{M}$ can be written as a $\mathbb{R}$-linear combination of functions in $\mathcal{H}_{d-1}$. Therefore, there are coefficients $c_{\alpha_1 \dots \alpha_t \beta} \in \mathbb{R}$ and constant $\kappa \in \mathbb{R}$ so that

$$\mathscr{M} = \sum_{t=0}^{d-1} \sum_{\substack{\alpha_i \in [p-1], \\ \beta \in [p-2]}} c_{\alpha_1 \dots \alpha_t \beta} \left(\bigoplus_{i=1}^{t} \alpha_i h_i \oplus h_{t+1} \oplus \beta \right) + \kappa \tag{4.12}$$

Recall that each h_i is a p-expressions over variables $x_1, \dots, x_n$. By substituting back for each h_i and rearranging terms, each p-expression $\bigoplus_{i=1}^{t} \alpha_i h_i \oplus h_{t+1} \oplus \beta$ is equivalent to $\alpha'_1 x_1 \oplus \alpha'_2 x_2 \oplus \cdots \oplus \alpha'_n x_n \oplus \beta'$ for some $\alpha'_1, \dots, \alpha'_n, \beta' \in [p-1]$. By Lemma 4.5.5, such expression can be written as a $\mathbb{R}$-linear combination of basis in $\mathcal{F}_n$.

Note that number of p-expressions in (4.12) is at most p^{d+1}. By Lemma 4.5.5, each p-expression can be written as a $\mathbb{R}$-linear combination of basis in $\mathcal{F}_n$ in time $O(np^{O(1)})$. Therefore, in time $O(np^{O(d)})$ we can write $\mathscr{M} = h_1 \cdot h_2 \cdots h_d$ as a $\mathbb{R}$-linear combination of basis in $\mathcal{F}_n$ which is polynomial in n for fixed p and d. □

4.5.3 The correctness of the conversion algorithm

Now we have enough ingredients to prove Algorithm 1 runs in polynomial time and correctly decides if $q|_{G_1} = \sum_j k_j b_j|_{G_1}$ for every q. In the following theorem, suppose q is the current monomial considered by the algorithm. Furthermore, suppose sets G_2 and $B(G_2)$ have been constructed correctly so far.

Theorem 4.5.7. *Let $q = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ be a monomial of degree at most d. Suppose q is not divisible by any leading monomial of polynomials in the current set G_2. Then, there exists a polynomial time algorithm that can decide whether $q|_{G_1} = \sum_j k_j b_j|_{G_1}$ where b_j are in the current set $B(G_2)$ in Algorithm 1.*

Proof. First, we discuss the case where for some $r < i$ we have $p - 1 < \alpha_i$. Set $q = q' \cdot x_i^{\alpha_i}$ where $q' = x_1^{\alpha_1} \cdots x_{i-1}^{\alpha_{i-1}} \cdot x_{i+1}^{\alpha_{i+1}} \cdots x_n^{\alpha_n}$. Then $q|_{G_1} = q'|_{G_1} \cdot x_i^{\alpha_i}|_{G_1}$. Note that $x_i^{\alpha_i}|_{G_1}$ is a linear combination of $x_i^{p-1}, x_i^{p-2}, \dots, x_i$. This is because the domain polynomial $\prod_{a \in \mathbb{Z}_p} (x_i - a)$ is

in G_1. Therefore,

$$\begin{aligned} q|_{G_1} &= q'|_{G_1} \cdot x_i^{\alpha_i}|_{G_1} \\ &= q'|_{G_1} \cdot (c_{p-1}x_i^{p-1} + \cdots + c_1 x_i) \\ &= (q' \cdot c_{p-1}x_i^{p-1})|_{G_1} + \cdots + (q' \cdot c_1 x_i)|_{G_1} \end{aligned} \tag{4.13}$$

All of $q' \cdot x_i^j$ in (4.13) have degree less than q, and hence they have been considered by Algorithm 1 before reaching q. Now, none of $q' \cdot x_i^j$ can be a multiple of the leading monomial of a polynomial in G_2 as otherwise q divides the leading monomial of a polynomial in G_2. This implies that all $q' \cdot x_i^j$ with $c_j \neq 0$ in (4.13) are in $B(G_2)$ and we have $q|_{G_1} = \sum_j k_j b_j|_{G_1}$.

We continue by assuming for all $r < i$ we have $\alpha_i \leq p-1$. Note that if $q|_{G_1} = \sum_j k_j b_j|_{G_1}$, then $q|_{G_1} - \sum_j k_j b_j|_{G_1} = 0$ and hence $q - \sum_j k_j b_j \in \mathcal{I}(\mathcal{P})$. We proceed by checking, in a systematic way, if there exist coefficients k_j such that $q|_{G_1} - \sum_j k_j b_j|_{G_1} = 0$ holds. We will construct a system of linear equations over $\mathbb{R}$ for coefficients k_j so that this system has a solution if and only if $q|_{G_1} = \sum_j k_j b_j|_{G_1}$.

By Observation 4.5.3, we have $q|_{G_1} = f_1^{\alpha_1} \cdots f_n^{\alpha_n}$ where each f_i is the p-expression associated to x_i. Similarly, for each b_j we have $b_j|_{G_1} = \mathscr{M}_j$ where $\mathscr{M}_j = h_{j1} \cdot h_{j2} \cdots h_{jd}$ is a multiplication of at most d (not necessary distinct) p-expressions.

$$f_1^{\alpha_1} \cdots f_n^{\alpha_n} = \sum_j k_j b_j|_{G_1} = \sum_j k_j \mathscr{M}_j \tag{4.14}$$

Recall that degree of q is at most d and, by Lemma 4.5.6, q can be written as a $\mathbb{R}$-linear combination of the basis in $\mathcal{F}_{n-1}$, say $\mathscr{L}_q$. Similarly, by Lemma 4.5.6, each product $\mathscr{M}_j = h_{j1} \cdot h_{j2} \cdots h_{jd}$ can be written as a $\mathbb{R}$-linear combination of the basis in $\mathcal{F}_{n-1}$ in polynomial time. Therefore, (4.14) is equivalent to

$$f_1^{\alpha_1} \cdots f_n^{\alpha_n} = \mathscr{L}_q = \sum_j k_j b_j|_{G_1} = \sum_j k_j \mathscr{M}_j = \sum_j k_j \mathscr{L}_j \tag{4.15}$$

where each $\mathscr{L}_j$ is a $\mathbb{R}$-linear combination equivalent to $\mathscr{M}_j$ via p-expression basis in $\mathcal{F}_n$. Rearranging terms in $\sum_j k_j \mathscr{L}_j$ and $\mathscr{L}_q$ yields

$$0 = \sum_j k_j \mathscr{L}_j - \mathscr{L}_q = \sum_j k'_j \mathscr{L}'_j \tag{4.16}$$

where each k'_j is linear combination of k_js. Since $\mathcal{F}_{n-1}$ is linearly independent and the left hand side of (4.16) is a constant we deduce that all k'_j should be zero. Hence, we are left with (possibly more than one) linear equations with respect to k_j over $\mathbb{R}$. Note that at this

point there is not any term with a p-expression. If such a system has a (unique) solution for k_j then we conclude $q|_{G_1} = \sum_j k_j b_j|_{G_1}$, else the equality does not hold.

Note that, by Lemma 4.5.6, time complexity of finding a $\mathbb{R}$-linear combination of basis in $\mathcal{F}_{n-1}$ for a multiplication of d p-expressions is $O(np^{O(d)})$. Moreover, we use Lemma 4.5.6 at most $O(n^{O(d)})$ many times. Hence, the whole process requires $O(n^{O(d)})$ time complexity. □

Theorem 4.5.8. *Let Γ be a constraint language where each relation in Γ is expressed as a system of linear equations modulo a prime number p. Then, the $IMP_d(\Gamma)$ can be solved in polynomial time for fixed d and p.*

4.5.4 Proof of Theorem 4.5.4

In this section we give a proof of Theorem 4.5.4. Also, it turns out in the case $p = 3$ there is another basis of $\mathcal{V}_{n,p}$ that has a particularly clear structure. We give a proof of this result in Section 4.5.4.

Linear independence

Recall that we consider the set $\mathcal{V}_{k,p}$ of functions from $\mathbb{Z}_p^k$ to $\mathbb{C}$ as a p^k-dimensional vector space, whose components are values of the function at the corresponding point of $\mathbb{Z}_p^k$. Let

$$F_k = \left\{ \bigoplus_{i=1}^{k} \alpha_i x_i \oplus x_{k+1} \oplus \beta \mid \alpha_i \in \{0, \dots, p-1\}, i \in [n], \beta \in \{0, \dots, p-2\} \right\}$$

be a collection of linear functions over $\mathbb{Z}_p$, and let

$$\mathcal{F}_k = F_k \cup \dots \cup F_0 \cup \{1\}.$$

The result we are proving in this section is

Theorem 4.5.9. *For any k, the collection $\mathcal{F}_k$ of p-expressions is linearly independent as a set of vectors from $\mathcal{V}_{k+1,p}$ and forms a basis of $\mathcal{V}_{k+1,p}$.*

Monomials and projections

As a vector space the set $\mathcal{V}_{k,p}$ has several natural bases. Let $\mathsf{Mon}(k,p)$ denote the set of all monomials $u = x_1^{d_1} \dots x_k^{d_k}$ where $d_i \in \{0, \dots, p-1\}$; and let $\mathsf{cont}(u)$ denote the set $\{i \mid d_i \neq 0\}$. For $\overline{a} \in \mathbb{Z}_p^k$ we denote by $r_{\overline{a}}$ the function given by $r_{\overline{a}}(\overline{a}) = 1$ and $r_{\overline{a}}(\overline{x}) = 0$ when $\overline{x} \neq \overline{a}$. We start with a simple observation.

Lemma 4.5.10. *The sets $\mathsf{Mon}(k,p)$ and $R(k,p) = \{r_{\overline{a}} \mid \overline{a} \in \mathbb{Z}_p^k\}$ are bases of $\mathcal{V}_{k,p}$.*

As is easily seen, both sets contain p^k elements, $R(k,p)$ obviously spans $\mathcal{V}_{k,p}$. That $\mathsf{Mon}(k,p)$ also spans $\mathcal{V}_{k,p}$ follows from polynomial interpolation properties.

Our goal here is to find yet another basis of $\mathcal{V}_{k,p}$ suitable for our needs, which is the set $\mathcal{F}_k$ defined above.

In this section we view functions from $\mathcal{F}_k$ as elements of $\mathcal{V}_{k+1,p}$. In particular, we will use coordinates of such functions in the bases $\mathsf{Mon}(k+1,p)$ and $R(k+1,p)$. The latter is of course just the collection of values of a function in points from $\mathbb{Z}_p^{k+1}$, while the former is the polynomial interpolation of a function, which is unique when we restrict ourselves to polynomials of degree at most $p-1$ in each variable.

The number of functions in $\mathcal{F}_k$ is

$$1+\sum_{\ell=0}^{k}|F_\ell| = 1+\sum_{\ell=0}^{k}p^\ell(p-1) = 1+(p-1)\sum_{\ell=0}^{k}p^\ell = 1+(p-1)\frac{p^{k+1}-1}{p-1} = p^{k+1}.$$

Thus, we only need to prove that $\mathcal{F}_k$ spans $\mathcal{V}_{k+1,p}$.

We will need a finer partition of sets F_k: for $S \subseteq [k]$ let F_k^S denote the set of functions $\bigoplus_{i=1}^k \alpha_i x_i \oplus x_{k+1} \oplus b$ such that $\alpha_i \neq 0$ if and only if $i \in S$.

We prove by induction on $|S|$, $S \subseteq [k+1]$, that any monomial u with $\mathsf{cont}(u) = S$ is in the span of

$$\mathcal{F}_k^S = \bigcup_{\ell \leq k, T \subseteq S \cap [\ell]} F_\ell^T.$$

Since up to renaming the variables $\mathcal{F}_k^S$ can be viewed as $\mathcal{F}_{|S|}$, it suffices to prove the result for $S = [k]$, and our inductive process is actually on k.

For $f \in F_k$ let f' denote the sum of all monomials u of f (with the same coefficients) for which $\mathsf{cont}[u] = [k+1]$. In other words, f' can be viewed as a projection f onto the subspace $\mathcal{V}_1$ spanned by $\mathsf{Mon}^*(k+1,p) = \{u \in \mathsf{Mon}(k+1,p) \mid \mathsf{cont}(u) = [k+1]\}$ parallel to the subspace $\mathcal{V}_2$ spanned by $\mathsf{Mon}^\dagger(k+1,p) = \mathsf{Mon}(k+1,p) - \mathsf{Mon}^*(k+1,p)$. Let $F_k' = \{f' \mid f \in F_k\}$. Note that $\mathcal{V}_1$ is also the subspace of $\mathcal{V}_{n,p}$ spanned by the set $\{r_{\overline{a}} \mid \overline{a} \in (\mathbb{Z}_p^*)^{k+1}\}$, and the dimensionality of $\mathcal{V}_1$ is $(p-1)^{k+1} = |F_k| = |F_k'|$. Since by the induction hypothesis $f - f'$ is in the span of $\mathcal{F}_k$, it suffices to prove that vectors in F_k' are linearly independent, and therefore generate $\mathcal{V}_1$. This will be proved in the rest of this section.

Proposition 4.5.11. *The set F_k' is linearly independent.*

We prove Proposition 4.5.11 by constructing a matrix containing the values of functions from F_k' and find its rank by finding all its eigenvalues. We do it in three steps. First, let $F_k^\dagger$ denote the superset of F_k that apart from functions from F_k also contains functions of the form $\bigoplus_{i=1}^k \alpha_i x_i \oplus x_{k+1} \oplus (p-1)$ for $\alpha_i \in \mathbb{Z}_p^*$, $i \in [k]$. Then, let N_k be the $p(p-1)^k \times p(p-1)^k$-dimensional matrix whose rows are labeled with $(x_1, \ldots, x_{k+1}) \in (\mathbb{Z}_p^*)^k \times \mathbb{Z}_p$ representing values of the arguments of functions from $F_k^\dagger$, and the columns are labeled with $f \in F_k^\dagger$. The entry of N_k in row $(x_1, \ldots, x_{k+1})$ and column f is $f(x_1, \ldots, x_{k+1})$. In the next section we find the eigenvectors and eigenvalues of N_k. In the second step we use the properties of N_k to study the matrix N_k'' obtained from N_k by replacing every entry of the form $f(x_1, \ldots, x_{k+1})$

with the value $f''(x_1, \ldots, x_{k+1})$, where f'' is the sum of all the monomials u from f with $x_1, \ldots, x_k \in \mathsf{cont}(u)$. We again find the eigenvectors and eigenvalues of N''_k. Finally, we transform N''_k to obtain a new matrix N'_k in such a way that the entry of N'_k in the row $(x_1, \ldots, x_{k+1})$ with $x_{k+1} \neq 0$ and column f equals $f'(x_1, \ldots, x_{k+1})$. We then finally prove that all the rows of N'_k labeled $(x_1, \ldots, x_{k+1})$, $x_{k+1} \neq 0$, are linearly independent.

Kronecker sum and the eigenvalues of N_k

We first introduce some useful notation.

Let A, B be $q \times r$ and $s \times t$ matrices with entries in $\mathbb{Z}_p$. The *Kronecker sum* of A and B denoted $A \boxplus B$ is the $qs \times rt$ matrix whose entry in row $is + i'$ and column $jt + j'$ equals $A(i, j) \oplus B(i', j')$. In other words, $A \boxplus B$ is defined the same way as Kronecker product, except using addition modulo p rather than multiplication. We also use $A^{\boxplus k}$ to denote $A \boxplus \cdots \boxplus A$.

We consider two matrices, matrix B_p essentially consists of values of unary functions αx on $[p-1]$, except that we rearrange its rows and columns as follows. Let a be a primitive residue modulo p, that is, a generator of $\mathbb{Z}_p^*$. Then

$$B_p = \begin{pmatrix} 1 & a & a^2 & a^3 & \ldots & a^{p-2} \\ a^{p-2} & 1 & a & a^2 & \ldots & a^{p-3} \\ a^{p-3} & a^{p-2} & 1 & a & \ldots & a^{p-4} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ a & a^2 & a^3 & a^4 & \ldots & 1 \end{pmatrix},$$

where a^i denotes exponentiation modulo p. Matrix C_p is again the operation table of addition modulo p with rearranged rows

$$C_p = \begin{pmatrix} 0 & 1 & 2 & \ldots & p-1 \\ p-1 & 0 & 1 & \ldots & p-2 \\ p-2 & p-1 & 0 & \ldots & p-3 \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & 2 & 3 & \ldots & 0 \end{pmatrix}.$$

As is easily seen, the values of functions $\bigoplus_{i=1}^k \alpha_i x_i$, $\alpha_1, \ldots, \alpha_k \in \mathbb{Z}_p^*$ on $(\mathbb{Z}_p^*)^k$ can be viewed as $B^{\boxplus k}$; and those of $\bigoplus_{i=1}^k \alpha_i x_i \oplus x_{k+1} \oplus b$, $\alpha_1, \ldots, \alpha_k \in \mathbb{Z}_p^*$, $b \in \mathbb{Z}_p$, on $x_1, \ldots, x_k \in \mathbb{Z}_p^*$, $x_{k+1} \in \mathbb{Z}_p$ can be represented as $N_k = C_p \boxplus B_p^{\boxplus k}$. Next we find the eigenvectors and eigenvalues of N_k.

Recall that a square matrix of the form

$$A = \begin{pmatrix} a_1 & a_2 & a_3 & \dots & a_n \\ a_n & a_1 & a_2 & \dots & a_{n-1} \\ a_{n-1} & a_n & a_1 & \dots & a_{n-2} \\ \vdots & \vdots & \vdots & & \vdots \\ a_2 & a_3 & a_4 & \dots & a_1 \end{pmatrix} \tag{4.17}$$

is called *circulant*. The eigenvectors and eigenvalues of circulant matrices are well known. We give a brief proof of the following fact for the sake of completeness.

Lemma 4.5.12. *The eigenvectors of the matrix A in (4.17) have the form $\vec{v}_\xi = (1, \xi, \xi^2, \dots, \xi^{n-1})$ for nth roots of unity ξ. The eigenvalue corresponding to $\vec{v}_\xi$ is $\mu_\xi = a_1 + a_2\xi + a_3\xi^2 + \cdots + a_n\xi^{n-1}$.*

Proof. We compute $A \cdot \vec{v}_\xi$

$$\begin{aligned} A \cdot \vec{v}_\xi &= \begin{pmatrix} a_1 & a_2 & a_3 & \dots & a_n \\ a_n & a_1 & a_2 & \dots & a_{n-1} \\ a_{n-1} & a_n & a_1 & \dots & a_{n-2} \\ \vdots & \vdots & \vdots & & \vdots \\ a_2 & a_3 & a_4 & \dots & a_1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ \xi \\ \xi^2 \\ \vdots \\ \xi^{n-1} \end{pmatrix} \\ &= \begin{pmatrix} a_1 + a_2\xi + a_3\xi^2 + \cdots + a_n\xi^{n-1} \\ a_n + a_1\xi + a_2\xi^2 + \cdots + a_{n-1}\xi^{n-1} \\ a_{n-1} + a_n\xi + a_1\xi^2 + \cdots + a_{n-2}\xi^{n-1} \\ \vdots \\ a_2 + a_3\xi + a_4\xi^2 + \cdots + a_1\xi^{n-1} \end{pmatrix} \\ &= (a_1 + a_2\xi + a_3\xi^2 + \cdots + a_n\xi^{n-1}) \begin{pmatrix} 1 \\ \xi \\ \xi^2 \\ \vdots \\ \xi^{n-1} \end{pmatrix}, \end{aligned}$$

as required. □

A generalization of circulant matrices is obtained by replacing entries of a circulant matrix by matrices. More precisely, a matrix of the form

$$A = \begin{pmatrix} A_1 & A_2 & A_3 & \dots & A_n \\ A_n & A_1 & A_2 & \dots & A_{n-1} \\ A_{n-1} & A_n & A_1 & \dots & A_{n-2} \\ \vdots & \vdots & \vdots & & \vdots \\ A_2 & A_3 & A_4 & \dots & A_1 \end{pmatrix}, \tag{4.18}$$

where $A_1, \dots, A_n$ are square matrices of the same size is said to be *block-circulant*. Note that N_k can be viewed as a block-circulant matrix:

$$N_k = C_p \boxplus B_p^{\boxplus k} = \begin{pmatrix} B_p^{\boxplus k} & B_p^{\boxplus k} \oplus 1 & \dots & B_p^{\boxplus k} \oplus (p-1) \\ B_p^{\boxplus k} \oplus (p-1) & B_p^{\boxplus k} & \dots & B_p^{\boxplus k} \oplus (p-2) \\ \vdots & \vdots & & \vdots \\ B_p^{\boxplus k} \oplus 2 & B_p^{\boxplus k} \oplus 3 & \dots & B_p^{\boxplus k} \end{pmatrix}.$$

Similarly, $B_p^{\boxplus \ell}$ can also be viewed as a block-circulant matrix:

$$B_p^{\boxplus(\ell-1)} \boxplus B_p = \begin{pmatrix} B_p^{\boxplus(\ell-1)} \oplus 1 & B_p^{\boxplus(\ell-1)} \oplus a & \dots & B_p^{\boxplus(\ell-1)} \oplus a^{p-1} \\ B_p^{\boxplus(\ell-1)} \oplus a^{p-1} & B_p^{\boxplus(\ell-1)} \oplus 1 & \dots & B_p^{\boxplus(\ell-1)} \oplus a^{p-2} \\ \vdots & \vdots & & \vdots \\ B_p^{\boxplus(\ell-1)} \oplus a & B_p^{\boxplus(\ell-1)} \oplus a^2 & \dots & B_p^{\boxplus(\ell-1)} \oplus 1 \end{pmatrix}.$$

In some cases the eigenvectors and eigenvalues of block-circulant matrices can also be found.

Lemma 4.5.13. *Let A be a block-circulant matrix with blocks $A_1, \dots, A_n$ as in (4.18) such that $A_1, \dots, A_n$ have the same eigenvectors. Then every eigenvector $\vec{w}$ of A has the form $\vec{w}_{v,\xi} = (1, \xi, \dots, \xi^{n-1}) \otimes \vec{v}$, where ξ is an nth root of unity and $\vec{v}$ is an eigenvector of $A_1, \dots, A_n$. Conversely, for every eigenvector $\vec{v}$ of $A_1, \dots, A_n$ and an nth root of unity ξ, $\vec{w}_{v,\xi}$ is an eigenvector of A. The eigenvalue of A associated with $\vec{w}_{v,\xi}$ is $\mu_{1,\vec{v}} + \mu_{2,\vec{v}}\xi + \mu_{3,\vec{v}}\xi^2 + \cdots + \mu_{n,\vec{v}}\xi^{n-1}$, where $\mu_{i,\vec{v}}$ is the eigenvalue of A_i associated with $\vec{v}$.*

Proof. As in the proof of Lemma 4.5.12 we compute $A \cdot \vec{w}_{v,\xi}$:

$$\begin{aligned} A \cdot \vec{w}_{v,\xi} &= \begin{pmatrix} A_1 & A_2 & A_3 & \dots & A_n \\ A_n & A_1 & A_2 & \dots & A_{n-1} \\ A_{n-1} & A_n & A_1 & \dots & A_{n-2} \\ \vdots & \vdots & \vdots & & \vdots \\ A_2 & A_3 & A_4 & \dots & A_1 \end{pmatrix} \cdot \begin{pmatrix} \vec{v} \\ \xi\vec{v} \\ \xi^2\vec{v} \\ \vdots \\ \xi^{n-1}\vec{v} \end{pmatrix} \\ &= \begin{pmatrix} A_1\vec{v} + \xi A_2\vec{v} + \xi^2 A_3\vec{v} + \cdots + \xi^{n-1}A_n\vec{v} \\ A_n\vec{v} + \xi A_1\vec{v} + \xi^2 A_2\vec{v} + \cdots + \xi^{n-1}A_{n-1}\vec{v} \\ A_{n-1}\vec{v} + \xi A_n\vec{v} + \xi^2 A_1\vec{v} + \cdots + \xi^{n-1}A_{n-2}\vec{v} \\ \vdots \\ A_2\vec{v} + \xi A_3\vec{v} + \xi^2 A_4\vec{v} + \cdots + \xi^{n-1}A_1\vec{v} \end{pmatrix} \\ &= (\mu_{1,\vec{v}} + \xi\mu_{2,\vec{v}} + \xi^2\mu_{3,\vec{v}} + \cdots + \xi^{n-1}\mu_{n,\vec{v}}) \begin{pmatrix} \vec{v} \\ \xi\vec{v} \\ \xi^2\vec{v} \\ \vdots \\ \xi^{n-1}\vec{v} \end{pmatrix}, \end{aligned}$$

as required. □

We now apply these techniques to N_k. Let $\vec{v}(\eta) = (1, \eta, \eta^2, \dots, \eta^{p-1})$ for a $(p-1)$th root of unity, and let $\otimes$ denote Kronecker product.

Lemma 4.5.14. *(1) For any k and any $i, j \in \mathbb{Z}_p$ the matrices $B_p^{\boxplus k} \oplus i$ and $B_p^{\boxplus k} \oplus j$ have the same eigenvectors, and every eigenvector has the form*

$$\vec{v}(\eta_1, \dots, \eta_k) = \vec{v}(\eta_1) \otimes \cdots \otimes \vec{v}(\eta_k)$$

for some $(p-1)$th roots of unity.

(2) For any k and any $j \in \mathbb{Z}_p$ the eigenvalues of the matrix $B_p^{\boxplus k} \oplus j$ are

$$\lambda(\eta_1, \dots, \eta_k; j) = \sum_{i_1, \dots, i_k = 0}^{p-2} (a^{i_1} \oplus \cdots \oplus a^{i_k} \oplus j)\eta_1^{i_1} \dots \eta_k^{i_k}$$

for $(p-1)$th roots of unity $\eta_1, \dots, \eta_k$.

(3) Let $i_1, \dots, i_s \in [k]$ be such that if $\eta_i \neq 1$ then $i = i_r$, $r \in [s]$, and $s \neq 0$. Then

$$\lambda(\eta_1, \dots, \eta_k; j) = (-1)^{k-s}\lambda(\eta_{i_1}, \dots, \eta_{i_k}; j).$$

Proof. We proceed by induction on k to prove all three claims simultaneously. If $k = 1$ then as $B_p \oplus j$ is a circulant matrix whose first row is $(1 \oplus j, a \oplus j, a^2 \oplus j, \ldots, a^{p-2} \oplus j)$, by Lemma 4.5.12 its eigenvalues have the form

$$\lambda(\eta_1) = (1 \oplus j) + (a \oplus j)\eta_1 + (a^2 \oplus j)\eta_1^2 + \cdots + (a^{p-2} \oplus j)\eta^{p-2}$$

for a $(p-1)$th root of unity η_1, and the corresponding eigenvector is $\vec{v}(\eta_1) = (1, \eta_1, \eta_1^2, \ldots, \eta_1^{p-1})$ regardless of j.

Now, suppose that the lemma is true for $k-1$. Also, suppose that every eigenvector of $B_p^{\boxplus(k-1)} \oplus j$ has the form $\vec{v}(\eta_2, \ldots, \eta_k)$ for $(p-1)$th roots of unity $\eta_2, \ldots, \eta_k$. Then $B_p^{\boxplus k} \oplus j$ is a block-circulant matrix with the first row $(B_p^{\boxplus(k-1)} \oplus 1 \oplus j, B_p^{\boxplus(k-1)} \oplus a \oplus j, B_p^{\boxplus(k-1)} \oplus a^2 \oplus j, \ldots, B_p^{\boxplus(k-1)} \oplus a^{p-1} \oplus j)$. As by the induction hypothesis the blocks in this row have the same eigenvectors, by Lemma 4.5.13 the eigenvalues of $B_p^{\boxplus k} \oplus j$ have the form

$$\mu_{0,\vec{v}} + \mu_{1,\vec{v}}\eta_1 + \mu_{1,\vec{v}}\eta_1^2 + \cdots + \mu_{p-2,\vec{v}}\eta_1^{p-2},$$

where $\vec{v}$ is an eigenvector of $B^{\boxplus(k-1)}$ and $\mu_{i,\vec{v}}$ is the eigenvalue of $B^{\boxplus(k-1)} \oplus a^i \oplus j$ associated with $\vec{v}$. Thus, plugging in the inductive hypothesis we obtain the result.

To prove item (3) we need to inspect the case when $\eta_k = 1$. In this case

$$\begin{aligned}
\lambda(\eta_1, \ldots, \eta_k) &= \sum_{i_1,\ldots,i_k=0}^{p-2} (a^{i_1} \oplus \cdots \oplus a^{i_k} \oplus j)\eta_1^{i_1} \ldots \eta_k^{i_k} \\
&= \sum_{i_1,\ldots,i_k=0}^{p-2} (a^{i_1} \oplus \cdots \oplus a^{i_k} \oplus j)\eta_1^{i_1} \ldots \eta_{k-1}^{i_{k-1}} \cdot 1 \\
&= \sum_{i_1,\ldots,i_{k-1}=0}^{p-2} \eta_1^{i_1} \ldots \eta_{k-1}^{i_{k-1}} \left(\sum_{i_k=0}^{p-2} (a^{i_1} \oplus \cdots \oplus a^{i_k} \oplus j) \right) \\
&= \sum_{i_1,\ldots,i_{k-1}=0}^{p-2} \eta_1^{i_1} \ldots \eta_{k-1}^{i_{k-1}} \left(\frac{p(p-1)}{2} - (a^{i_1} \oplus \cdots \oplus a^{i_{k-1}} \oplus j) \right) \\
&= \frac{p(p-1)}{2} \sum_{i_1,\ldots,i_{k-1}=0}^{p-2} \eta_1^{i_1} \ldots \eta_{k-1}^{i_{k-1}} - \lambda(\eta_1, \ldots, \eta_{k-1}) \\
&= -\lambda(\eta_1, \ldots, \eta_{k-1}).
\end{aligned}$$

The last equality is due to fact that

$$\sum_{i_1,\ldots,i_{k-1}=0}^{p-2} \eta_1^{i_1} \ldots \eta_{k-1}^{i_{k-1}} = \sum_{i_1=0}^{p-2} \eta_1^{i_1} \cdot \cdots \cdot \sum_{i_{k-1}=0}^{p-2} \eta_{k-1}^{i_{k-1}} = 0.$$

By the induction hypothesis the result follows. □

Since the matrix N_k can be represented as $C_p \boxplus B_p^{\boxplus k}$, its eigenvalues can be found by Lemma 4.5.13.

Lemma 4.5.15. *The eigenvalues of N_k can be represented in one of the following forms.*

- *for a pth root of unity ξ and $(p-1)$th roots of unity $\eta_1, \ldots, \eta_k$*

$$\mu(\eta_1, \ldots, \eta_k; \xi) = \sum_{j=0}^{p-1} \xi^j \sum_{i_1,\ldots,i_k=0}^{p-2} (a^{i_1} \oplus \cdots \oplus a^{i_k} \oplus j) \eta_1^{i_1} \ldots \eta_k^{i_k}.$$

- *for a pth root of unity ξ and $(p-1)$th roots of unity $\eta_1, \ldots, \eta_k$*

$$\mu(\eta_1, \ldots, \eta_k; \xi) = P(\xi) \cdot Q(\eta_1, \xi) \cdot \cdots \cdot Q(\eta_k, \xi),$$

where $P(\xi) = \frac{p}{\xi - 1}$ unless $\xi = 1$, in which case $P(1) = \frac{p(p-1)}{2}$, and

$$Q(\eta, \xi) = \sum_{i=0}^{p-2} \eta^i \xi^{a^i}.$$

Proof. (1) By Lemma 4.5.13 the eigenvalues of N_k have the form $\lambda_{0,\vec{v}} + \lambda_{1,\vec{v}}\xi + \lambda_{2,\vec{v}}\xi^2 + \cdots + \lambda_{p-1,\vec{v}}\xi^{p-1}$, where $\vec{v}$ is an eigenvector of $B_p^{\boxplus k} \oplus i$ for all $i \in \mathbb{Z}_p$, and $\lambda_{i,\vec{v}}$ is the eigenvalue of $B_p^{\boxplus k} \oplus i$ associated with $\vec{v}$, and ξ is a pth root of unity. By Lemma 4.5.14 we obtain item (1) of the lemma.

(2) Consider the values $a^{i_1} \oplus \cdots \oplus a^{i_k} \oplus j$ in the formula from part (1). For $j = 0, \ldots, p-1$ they constitute the set $\mathbb{Z}_p$ regardless of $i_1, \ldots, i_k$, and the sequence, when j grows from 0 to $p-1$, is a sequence of consequent residues modulo p. Therefore

$$\sum_{j=0}^{p-1} (a^{i_1} \oplus \cdots \oplus a^{i_k} \oplus j) \xi^j = \xi^{a^{i_1} \oplus \cdots \oplus a^{i_k}} \sum_{j=0}^{p-1} j \xi^j,$$

and let $P(\xi) = \sum_{j=0}^{p-1} j\xi^j$. Therefore by part (1)

$$\begin{aligned}
\mu(\eta_1, \ldots, \eta_k; \xi) &= \sum_{j=0}^{p-1} \xi^j \sum_{i_1,\ldots,i_k=0}^{p-2} (a^{i_1} \oplus \cdots \oplus a^{i_k} \oplus j) \eta_1^{i_1} \ldots \eta_k^{i_k} \\
&= \sum_{i_1,\ldots,i_k=0}^{p-2} \xi^{a^{i_1} \oplus \cdots \oplus a^{i_k}} P(\xi) \eta_1^{i_1} \ldots \eta_k^{i_k} \\
&= P(\xi) \sum_{i_1,\ldots,i_k=0}^{p-2} (\eta^{i_1} \xi^{a^{i_1}}) \cdot \cdots \cdot (\eta_k^{i_k} \xi^{a^{i_k}}) \\
&= P(\xi) \left(\sum_{i_1=0}^{p-2} \eta^{i_1} \xi^{a^{i_1}} \right) \cdot \cdots \cdot \left(\sum_{i_k=0}^{p-2} \eta_k^{i_k} \xi^{a^{i_k}} \right) \\
&= P(\xi) \cdot Q(\eta_1, \xi) \cdot \cdots \cdot Q(\eta_k, \xi).
\end{aligned}$$

Finally, we show that $P(\xi)$ has the required form. Since $\sum_{j=0}^{p-1}\xi^j = 0$, we have $P(\xi) = P'(\xi)$, where $P'(x) = \sum_{j=0}^{p-1}(j+1)x^j$. Then

$$\begin{aligned}\sum_{j=0}^{p-1}(j+1)x^j &= \frac{d}{dx}\left(\sum_{j=0}^{p-1}x^{j+1}\right)\\ &= \frac{d}{dx}\left(\frac{x^{p+1}-x}{x-1}\right)\\ &= \frac{((p+1)x^p-1)(x-1)-(x^{p+1}-x)}{(x-1)^2}\\ &= \frac{px^{p+1}-(p+1)x^p+1}{(x-1)^2}.\end{aligned}$$

Since ξ is a pth root of unity, if $\xi \neq 1$ we have

$$P(\xi) = \frac{p\xi-(p+1)+1}{(\xi-1)^2} = \frac{p}{\xi-1}.$$

Finally, $P(1) = \frac{p(p-1)}{2}$, as is easily seen. □

Clearly, the co-rank of N_k equals the multiplicity of the eigenvalue 0. Thus, we need to find the number of combinations of $\xi, \eta_1, \ldots, \eta_k$ such that $\mu(\eta_1, \ldots, \eta_k; \xi) = 0$. For some of them it is easy.

Lemma 4.5.16. *If $\eta_i \neq 1$ for some $i \in [k]$ then $\mu(\eta_1, \ldots, \eta_k; 1) = 0$.*

Proof. We use Lemma 4.5.15. Let $\xi = 1$, and, say, $\eta_1 \neq 1$. Then

$$Q(\eta_1, 1) = \sum_{i=0}^{p-2}\eta_1^i = \frac{\eta_1^{p-1}-1}{\eta_1-1} = 0,$$

as η_1 is a $(p-1)$th root of unity and $\eta_1 \neq 1$. □

Lemma 4.5.17. *Let $\xi \neq 1$ be a pth root of unity and η a $(p-1)$th root of unity. Then $Q(\eta, \xi) \neq 0$.*

Proof. Let χ be a primitive $p(p-1)$th root of unity. Then η, ξ can be represented as $\eta = \chi^{up}$, $\xi = \chi^{v(p-1)}$ and $Q(\eta, \xi)$ can be rewritten as

$$Q^*(\chi) = \sum_{j=1}^{p-1}\chi^{jup+a^jv(p-1)}.$$

Note that all the arithmetic operations in the exponent including a^j can be treated as regular ones rather than modular, as $\chi^b = \chi^c$ whenever $b \equiv c \pmod{p(p-1)}$. Therefore if

there are η, ξ, $\xi \neq 1$ such that $Q(\eta, \xi) = 0$, then there exists a primitive $p(p-1)$th root of unity χ that is also a root of the polynomial

$$Q^*(x) = \sum_{j=1}^{p-1} x^{jup + a^j v(p-1)}.$$

This means that $Q^*(x)$ is divisible by $p(p-1)$ cyclotomic polynomial $C_{p(p-1)}$. The degree of $C_{p(p-1)}$ equals $\varphi(p(p-1))$, where φ is Euler's totient function. In particular, the degree of $C_{p(p-1)}$ is divisible by $p-1$, and so is the degree of Q^*. Since a and $p-1$ are relatively prime with p, it is only possible iv u is divisible by $p-1$, that is, $\eta = 1$, in which case, as is easily seen, $Q(1, \xi) = -1$ if $\xi \neq 1$ and $Q(1, 1) = p - 1$. □

The next proposition follows from Lemma 4.5.17 and the observation that $P(\xi) \neq 0$ whenever ξ is a pth root of unity.

Proposition 4.5.18. *The rank of N_k is $(p-1)^k + 1$.*

Changed matrices

In this subsection we make the second step in our proof.

Lemma 4.5.19. *Let $f = \bigoplus_{i=1}^{k} \alpha_i x_i \oplus x_{k+1} \oplus b$ then*

$$f'' = \sum_{S \subseteq [k]} (-1)^{k-|S|} \left(\bigoplus_{i \in S} \alpha_i x_i \oplus x_{k+1} \oplus b \right). \tag{4.19}$$

Proof. We need to show that $f''(x_1, \ldots, x_{k+1}) = 0$ whenever $x_i = 0$ for some $i \in [k]$. In order to do that observe that the terms in (4.19) can be paired up so that every S containing i is paired with $S - \{i\}$. Then $\bigoplus_{i \in S} \alpha_i x_i \oplus x_{k+1} \oplus b$ and $\bigoplus_{i \in S - \{i\}} \alpha_i x_i \oplus x_{k+1} \oplus b$ appear in (4.19) with opposite signs, and, as $x_i = 0$ are equal. □

Let N_k'' denote the matrix constructed the same way as N_k only with f'', $f \in F^\dagger$, in place of f. More precisely, N_k'' is the $p(p-1)^k \times p(p-1)^k$-dimensional matrix whose rows are labeled with $(x_1, \ldots, x_{k+1}) \in (\mathbb{Z}_p^*)^k \times \mathbb{Z}_p$ representing values of the arguments of functions from $F_k^\dagger$, and the columns are labeled with $f \in F_k^\dagger$. The entry of N_k'' in row $(x_1, \ldots, x_{k+1})$ and column f is $f''(x_1, \ldots, x_{k+1})$.

Using Lemma 4.5.19 we represent N_k'' as a sum of matrices. Let $f = \bigoplus_{i=1}^{k} \alpha_i x_i \oplus x_{k+1} \oplus b \in F_k^\dagger$ and $S \subseteq [k]$. Then let f_S denote the function $\bigoplus_{i \in S} \alpha_i x_i \oplus x_{k+1} \oplus b$. In other words, by Lemma 4.5.19

$$f = \sum_{S \subseteq [k]} (-1)^{k-|S|} f_S.$$

By $N_k(S)$, $S \subseteq [k]$, we denote the matrix constructed in a similar way to N_k and N_k''. Again, its rows are labeled with $(x_1, \ldots, x_{k+1}) \in (\mathbb{Z}_p^*)^k \times \mathbb{Z}_p$, and the columns are labeled

with $f \in F_k^\dagger$. The entry of $N_k(S)$ in row $(x_1, \ldots, x_{k+1})$ and column f is $f_S(x_1, \ldots, x_{k+1})$. It is now easy to see that

$$N_k'' = \sum_{S \subseteq [k]} (-1)^{k-|S|} N_k(S).$$

In order to determine the structure of $N_k(S)$ we need one further observation. Let $\mathbf{0}_\ell$ denote the square ℓ-dimensional matrix whose entries are all 0. Note that for a matrix B and $\mathbf{0}_\ell$

$$B \boxplus \mathbf{0}_\ell = \begin{pmatrix} B & \ldots & B \\ \vdots & & \vdots \\ B & \ldots & B \end{pmatrix}.$$

Lemma 4.5.20. *Let B an n-dimensional diagonalizable matrix. Then the eigenvectors of $B \boxplus \mathbf{0}_\ell$ are of the form $(\beta_1 \vec{v}, \ldots, \beta_\ell \vec{v})$ where v is an eigenvector of B and either $\beta_1 = \cdots = \beta_\ell$ or $\beta_1 + \cdots + \beta_\ell = 0$. The corresponding eigenvalue in the former case is $\ell\lambda$, where λ is the eigenvalue of B associated with $\vec{v}$, and 0 in the latter case.*

The following lemma establishes the structure of $N_k(S)$, its eigenvalues and eigenvectors.

Lemma 4.5.21. *(a) $N_k(S) = C_p \boxplus D_1 \boxplus \cdots \boxplus D_k$, where*

$$D_i = \begin{cases} B_p, & \text{if } i \in S, \\ \mathbf{0}_{p-1}, & \text{otherwise.} \end{cases}$$

(b) Every eigenvector of N_k is also an eigenvector of $N_k(S)$.

(c) The eigenvalue $\mu(\eta_1, \ldots, \eta_; \xi; S)$ of $N_k(S)$ associated with eigenvector $\vec{v}(\eta_1, \ldots, \eta_k; \xi)$ equals

$$\mu(\eta_1, \ldots, \eta_; \xi; S) = \begin{cases} 0 & \text{if } \eta_i \neq 1 \text{ for some } i \in [k] - S, \\ \mu(1, \ldots, 1; 1), & \text{if } \xi = \eta_1 = \cdots = \eta_k = 1, \\ (1-p)^{|[k]-S|} \mu(\eta_1, \ldots, \eta_k; \xi), & \text{otherwise.} \end{cases}$$

Proof. We will construct the matrix $N_k(S)$ inductively and prove the three claims of the lemma as we go. Let $N_k(S, \ell)$, $\ell \leq k$, denote the $(p-1)^\ell \times (p-1)^\ell$-matrix whose rows are labeled with $(x_1, \ldots, x_\ell) \in (\mathbb{Z}_p^*)^\ell$, columns are labeled with functions $f = \bigoplus_{i=1}^\ell \alpha_i x_i$. The entry of $N_k(S, \ell)$ in row $(x_1, \ldots, x_\ell)$ and column f is $f_S(x_1, \ldots, x_\ell)$. We show that

(a') $N_k(S, \ell) = D_1 \boxplus \cdots \boxplus D_\ell$, where the D_i's are defined as in the lemma.

(b') Every vector of the form $\vec{v}(\eta_1, \ldots, \eta_\ell) = \vec{v}(\eta_1) \otimes \cdots \otimes \vec{v}(\eta_\ell)$, where η_i is a $(p-1)$th root of unity is an eigenvector of $N_k(S, \ell) \oplus j$ for $j \in \mathbb{Z}_p$.

(c') The eigenvalue $\mu(\eta_1, \ldots, \eta_\ell, S, j)$ of $N_k(S, \ell) \oplus j$ associated with eigenvector $\vec{v}(\eta_1, \ldots, \eta_\ell)$ equals

$$\mu(\eta_1, \ldots, \eta_\ell, S, j) = \begin{cases} (p-1)^\ell \cdot j & \text{if } [\ell] \cap S = \emptyset \text{ and } \eta_1 = \cdots = \eta_\ell, \\ 0, & \text{if } \eta_i \neq 1 \text{ for some } i \in [\ell] - S, \\ (p-1)^{|[\ell]-S|} \lambda(\eta_{i_1}, \ldots, \eta_{i_t}; j), & \text{otherwise (see Lemma 4.5.14),} \\ & \text{where } \{j_1, \ldots, j_t\} = [\ell] \cap S. \end{cases}$$

If $\ell = 1$ then either $N_k(S, 1) = B_p$ if $1 \in S$, or $N_k(S, 1) = \mathbf{0}_{p-1}$ if $1 \notin S$. In the former case we have the result by Lemma 4.5.14, and in the latter case by Lemma 4.5.20 every vector of the form $\vec{v}(\eta)$, η is a $(p-1)$th root of unity, is an eigenvector with eigenvalue $(p-1)j$ if $\eta = 1$ and 0 otherwise.

Suppose the statement is true for some ℓ. If $[\ell+1] \cap S = \emptyset$, the claim is straightforward, as $f_S(x_1, \ldots, x_{\ell+1}) = 0$ for all $x_1, \ldots, x_{\ell+1} \in \mathbb{Z}_p^*$, and the result follows by Lemma 4.5.20.

Next, suppose that $[\ell] \cap S \neq \emptyset$, but $\ell+1 \notin S$. In this case the entry of $N_k(S, \ell+1)$ indexed with row $(x_1, \ldots, x_{\ell+1})$ and column $f = \bigoplus_{i=1}^{\ell+1} \alpha_i x_i$ is $f_S(x_1, \ldots, x_{\ell+1}) = f_S^*(x_1, \ldots, x_\ell) = \bigoplus_{i \in S \cap [\ell]} \alpha_i x_i$, where $f^* = \bigoplus_{i=1}^{\ell} \alpha_i x_i$.This implies that $N_k(S, \ell+1) \oplus j = (N_k(S, \ell) \oplus j) \boxplus \mathbf{0}_{p-1}$. By Lemma 4.5.20 the eigenvectors of $N_k(S, \ell+1) \oplus j$ are of the two types: $\vec{v}' = (\vec{v}, \ldots, \vec{v})$ or $(\beta_1 \vec{v}, \ldots, \beta_{p-1} \vec{v})$ with $\beta_1 + \cdots + \beta_{p-1} = 0$, where $\vec{v}$ is an eigenvector of $N_k(S, \ell) \oplus j$. In the first case $\vec{v}' = \vec{v} \otimes \vec{v}(\eta)$ for $\eta = 1$ and by the induction hypothesis has the required form. The corresponding eigenvalue of $N_k(S, \ell+1) \oplus j$ equals $(p-1)\lambda$, where λ is the eigenvalue of $N_k(S, \ell) \oplus j$ associated with $\vec{v}$, and so also has the required form. In the latter case $v \otimes (1, \eta, \ldots, \eta^{p-2})$, $\eta \neq 1$ and $\vec{v}$ is an eigenvector of $N_k(S, \ell) \oplus j$, satisfies the condition $1 + \eta + \cdots + \eta^{p-2} = 0$ and has eigenvalue 0. By the induction hypothesis and Lemma 4.5.14(3) we get the result.

Finally, let $\ell+1 \in S$. In this case for any $x_1, \ldots, x_{\ell+1} \in \mathbb{Z}_p^*$ and $f = \bigoplus_{i=1}^{\ell+1} \alpha_i x_i$ the entry of $N_k(S, \ell+1)$ equals

$$f_S(x_1, \ldots, x_{\ell+1}) = \bigoplus_{i=1}^{\ell+1} \alpha_i x_i = f_S^*(x_1, \ldots, x_\ell) \oplus \alpha_{\ell+1} x_{\ell+1},$$

which implies $N_k(S, \ell+1) = N_k(S, \ell) \boxplus B_p$ proving (a'). Therefore $N_k(S, \ell+1) \oplus j$ is a block-circulant matrix and we can apply Lemma 4.5.13 to show that eigenvectors of $N_k(S, \ell+1)$ are of the form

$$\vec{v} \otimes (1, \eta, \ldots, \eta^{p-2}) = \vec{v} \otimes \vec{v}(\eta),$$

where η is a $(p-1)$th root of unity and $\vec{v}$ is any eigenvector of $N_k(S, \ell) \oplus j$. The eigenvalue of such a vector can be found using the inductive hypothesis and the last part of the proof

of Lemma 4.5.14 as follows. We have

$$\mu(\eta_1,\ldots,\eta_{\ell+1},S,j)=\mu_{0,\vec{v}}+\mu_{1,\vec{v}}\eta_{\ell+1}+\mu_{1,\vec{v}}\eta_{\ell+1}^2+\cdots+\mu_{p-2,\vec{v}}\eta_{\ell+1}^{p-2},$$

where $\vec{v}$ is an eigenvector of $N_k(S,\ell)$ and $\mu_{i,\vec{v}}$ is the eigenvalue of $N_k(S,\ell)\oplus a^i\oplus j$ associated with $\vec{v}$. If there is $i\in S\cap[\ell]$ such that $\eta_i\neq 1$, then $\mu(\eta_1,\ldots,\eta_{\ell+1},S,j)=0$. If $S\cap[\ell]=\emptyset$ and $\eta_1=\cdots=\eta_\ell=1$ then

$$\mu(\eta_1,\ldots,\eta_{\ell+1},S,j)=\sum_{i=0}^{p-2}(p-1)^\ell(a^i\oplus j)\eta_{\ell+1}^i=(p-1)^\ell\lambda(\eta_{\ell+1};j).$$

If $S\cap[\ell]=\{i_1,\ldots,i_t\}\neq\emptyset$, then by the induction hypothesis

$$\begin{aligned}\mu(\eta_1,\ldots,\eta_{\ell+1},S,j) &= \sum_{i=0}^{p-2}(p-1)^{|[\ell]-S|}\lambda(\eta_{i_1},\ldots,\eta_{i_t};a^i\oplus j)\eta_{\ell+1}^i\\ &= (p-1)^{|[\ell]-S|}\sum_{i=0}^{p-2}\sum_{j_1,\ldots,j_t=0}^{p-2}(a^{j_1}\oplus\cdots\oplus a^{j_t}\oplus a^i\oplus j)\eta_{i_1}\ldots\eta_{i_t}\eta_{\ell+1}^i\\ &= (p-1)^{|[\ell]-S|}\lambda(\eta_{i_1},\ldots,\eta_{i_t},\eta_{\ell+1};j).\end{aligned}$$

We now consider the last step in constructing $N_k(S)$, from $N_k(S,k)$ to $N_k(S)$. As is easily seen, $N_k(S)=C_p\boxplus N_k(S,k)$, implying item (a) of the lemma, and by Lemma 4.5.13 every vector of the form $(1,\xi,\ldots,\xi^{p-1})\otimes\vec{v}$, where $\vec{v}$ is an eigenvector of $N_k(S,k)$ and ξ is a pth root of unity is an eigenvector of $N_k(S)$. By the induction hypothesis this implies item (b) of the lemma. Finally, again by Lemma 4.5.13 and the induction hypothesis the eigenvalue associated with the vector $(1,\xi,\ldots,\xi^{p-1})\otimes\vec{v}(\eta_1,\ldots,\eta_k)$ equals

$$\begin{aligned}&\mu(\eta_1,\ldots,\eta_k;\xi;S)\\ &\quad=\ \mu(\eta_1,\ldots,\eta_k,S,0)+\mu(\eta_1,\ldots,\eta_k,S,1)\xi+\cdots+\mu(\eta_1,\ldots,\eta_k,S,p-1)\xi^{p-1}.\end{aligned}$$

If $S=\emptyset$, $\xi\neq 1$, and $\eta_1=\cdots=\eta_k=1$ then

$$\begin{aligned}\mu(1,\ldots,1;\xi;\emptyset) &= \sum_{j=0}^{p-1}(p-1)^k j\xi^j=(p-1)^kP(\xi)\\ &= (p-1)^k(-1)^k\mu(1,\ldots,1;\xi)\\ &= (1-p)^k\mu(1,\ldots,1;\xi),\end{aligned}$$

as $Q(1,\xi)=-1$, as is easily seen. Also,

$$\mu(1,\ldots,1;1;\emptyset)=(p-1)^kP(1)=\mu(1,\ldots,1;1).$$

If $\eta_i \neq 1$ for some $i \in [k] - S$ then $\mu(\eta_1, \ldots, \eta_k, S, j) = 0$, and so $\mu(\eta_1, \ldots, \eta_k; \xi; S) = 0$. Otherwise if $S = \{i_1, \ldots, i_s\}$,

$$\begin{aligned}\mu(\eta_1, \ldots, \eta_k; \xi; S) &= (p-1)^{|[k]-S|} \sum_{j=0}^{p-1} \lambda(\eta_{i_1}, \ldots, \eta_{i_s}; j)\xi^j \\ &= (p-1)^{|[k]-S|} \mu(\eta_{i_1}, \ldots, \eta_{i_s}; \xi)\end{aligned}$$

Finally, by Lemma 4.5.14 $\mu(\eta_{i_1}, \ldots, \eta_{i_s}; \xi) = (-1)^{|[k]-S|}\mu(\eta_1, \ldots, \eta_k; \xi)$, if $\xi \neq 1$, $\mu(\eta_{i_1}, \ldots, \eta_{i_s}; 1) = 0$ if $\eta_{i_j} \neq 1$ for some j, and $\mu(1, \ldots, 1; 1) = (p-1)^{|S|}P(1)$, and the result follows. □

Now, we are ready to find the eigenvalues of N_k''.

Lemma 4.5.22. *Let $\vec{w} = \vec{v}(\eta_1, \ldots, \eta_k, \xi)$ and $T \subseteq [k]$ be such that $i \in T$ iff $\eta_i \neq 1$. Then*

$$\mu''(\eta_1, \ldots, \eta_k; \xi) = \begin{cases} p^{k-|T|}\mu(\eta_1, \ldots, \eta_k; \xi), & \text{if } \xi \neq 1, \\ 0, & \text{if } \xi = 1 \text{ and } \eta_i \neq 1 \text{ for some } i \in [k]. \end{cases}$$

Proof. Assume first that $\xi \neq 1$. By Lemmas 4.5.19 and 4.5.21 we have

$$\begin{aligned}\mu''(\eta_1, \ldots, \eta_k; \xi) &= \sum_{S \subseteq [k]} (-1)^{|[k]-S|}\mu(\eta_1, \ldots, \eta_k; \xi; S) \\ &= \sum_{[k] \supseteq S \supseteq T} (-1)^{|[k]-S|}\mu(\eta_1, \ldots, \eta_k; \xi; S) \\ &= \sum_{\ell=0}^{k-|T|} (-1)^\ell \binom{k-|T|}{\ell} (1-p)^\ell \mu(\eta_1, \ldots, \eta_k; \xi) \\ &= \mu(\eta_1, \ldots, \eta_k; \xi) \sum_{\ell=0}^{k-|T|} \binom{k-|T|}{\ell} (p-1)^\ell \\ &= \mu(\eta_1, \ldots, \eta_k; \xi) p^{k-|T|},\end{aligned}$$

as required.

Now let $\xi = 1$. If $T \neq \emptyset$, then $\mu(\eta_1, \ldots, \eta, \xi; S) = 0$ for any $S \subseteq [k]$. Otherwise, we have

$$\begin{aligned}\mu'(\eta_1, \ldots, \eta_k; \xi) &= \sum_{T \subseteq [k]} (-1)^{k-|T|}\mu(1, \ldots, 1; 1; T) \\ &= \sum_{T \subseteq [k]} (-1)^{k-|T|}\mu(1, \ldots, 1; 1) \\ &= \mu(1, \ldots, 1; 1) \sum_{\ell=0}^{k-1} (-1)^\ell \binom{k}{\ell} \\ &= 0.\end{aligned}$$

□

Since f' consists of all the monomials u of f with $x_1, \ldots, x_{k+1} \in \mathsf{cont}(U)$ and f'' consists of those with $x_1, \ldots, x_k \in \mathsf{cont}(u)$, it is easy to see that

$$f'(x_1, \ldots, x_k, x_{k=1}) = f''(x_1, \ldots, x_k, x_{k+1}) - f''(x_1, \ldots, x_k, 0). \tag{4.20}$$

Let N_k''' denote the matrix obtained from N_k'' by subtracting the row labeled $(x_1, \ldots, x_k, 0)$ from every row labeled by $(x_1, \ldots, x_k, x_{k+1})$, $x_{k+1} \in \mathbb{Z}_p^*$. By (4.20) the rows of N_k''' labeled $(x_1, \ldots, x_k, x_{k+1})$, $x_{k+1} \in \mathbb{Z}_p^*$ contain the values of $f'(x_1, \ldots, x_k, x_{k+1})$. Let N_k' be the submatrix of N_k''' containing only such rows. We need to prove that the columns of N_k' labeled with $f \in F_k$ are linearly independent. We do it by first proving that the rank of N_k' equals $(p-1)^{k+1}$ and then demonstrating that the column labeled $f^{(p-1)} = \bigoplus_{i=1}^k \alpha_i x_i \oplus x_{k+1} \oplus (p-1)$ is a linear combination of columns labeled $f^{(b)} = \bigoplus_{i=1}^k \alpha_i x_i \oplus x_{k+1} \oplus b$ for $b \in \{0, \ldots, p-2\}$.

Lemma 4.5.23. *Let* $f = \bigoplus_{i=1}^k \alpha_i x_i \oplus x_{k+1} \oplus b$.

(a)

$$\Sigma' f = \sum_{x_{k+1}=0}^{p-1} f''(x_1, \ldots, x_k, x_{k+1}) = 0.$$

(b) Let $f^{(a)}$ *denote the function* $f^{(a)} = \bigoplus_{i=1}^k \alpha_i x_i \oplus x_{k+1} \oplus a$ *(and so* $f = f^{(b)}$*). Then*

$$\Sigma f = \sum_{b \in \mathbb{Z}_p} f''^{(b)}(x_1, \ldots, x_{k+1}) = 0.$$

Proof. (a) We have

$$\begin{aligned} \Sigma' f &= \sum_{x_{k+1}=0}^{p-1} f''(x_1, \ldots, x_k, x_{k+1}) \\ &= \sum_{x_{k+1}=0}^{p-1} \sum_{S \subseteq [k]} (-1)^{k-|S|} f_S(x_1, \ldots, x_{k+1}) \\ &= \sum_{S \subseteq [k]} (-1)^{k-|S|} \sum_{x_{k+1}=0}^{p-1} f_S(x_1, \ldots, x_{k+1}). \end{aligned}$$

Let $S \subseteq [k]$, $x_1, \ldots, x_k \in \mathbb{Z}_p^*$, and let us denote $A = f_S(x_1, \ldots, x_k, 0) = \bigoplus_{i \in S}^k \alpha_i x_i \oplus b$. Then

$$\begin{aligned} \sum_{x_{k+1}=0}^{p-1} f_S(x_1, \ldots, x_{k+1}) &= \sum_{a=0}^{p-1} (A \oplus a) \\ &= \frac{p(p+1)}{2}. \end{aligned}$$

Now,

$$\Sigma' f = \frac{p(p+1)}{2} \sum_{S \subseteq [k]} (-1)^{k-|S|} = 0.$$

(b) We have

$$\begin{aligned} \Sigma f &= \sum_{b \in \mathbb{Z}_p} f''^{(b)}(x_1, \ldots, x_{k+1}) \\ &= \sum_{b \in \mathbb{Z}_p} \sum_{S \subseteq [k]} (-1)^{k-|S|} f_S^{(b)}(x_1, \ldots, x_{k+1}) \\ &= \sum_{S \subseteq [k]} (-1)^{k-|S|} \sum_{b \in \mathbb{Z}_p} f_S^{(b)}(x_1, \ldots, x_{k+1}). \end{aligned}$$

Let $S \subseteq [k]$, $x_1, \ldots, x_{k+1} \in \mathbb{Z}_p^*$, and let us denote $A = \bigoplus_{i \in S} \alpha_i x_i \oplus x_{k+1}$. Then

$$\sum_{b \in \mathbb{Z}_p} f_S^{(b)}(x_1, \ldots, x_{k+1}) = \sum_{b \in \mathbb{Z}_p} (A \oplus b) = \sum_{b \in \mathbb{Z}_p} b = \frac{p(p+1)}{2}.$$

Now,

$$\Sigma f = \frac{p(p+1)}{2} \sum_{S \subseteq [k]} (-1)^{k-|S|} = 0.$$

□

We are now in a position to complete the proof of Proposition 4.5.11. Let $\vec{a}(x_1, \ldots, x_k, x_{k+1})$ denote the row of N_k'' labeled with $(x_1, \ldots, x_k, x_{k+1})$. Then by Lemma 4.5.23(a)

$$\vec{a}(x_1, \ldots, x_k, 0) = -\sum_{b \in \mathbb{Z}_p^*} \vec{a}(x_1, \ldots, x_k, b),$$

and the row of N_k''' labeled with $(x_1, \ldots, x_k, x_{k+1})$ is

$$\vec{b}(x_1, \ldots, x_k, x_{k+1}) = \vec{a}(x_1, \ldots, x_k, x_{k+1}) + \sum_{b \in \mathbb{Z}_p^*} \vec{a}(x_1, \ldots, x_k, b).$$

As is easily seen the row $\vec{a}(x_1, \ldots, x_k, 0)$ is still a linear combination of $\vec{b}(x_1, \ldots, x_k, x_{k+1})$, $x_{k+1} \in \mathbb{Z}_p^*$, implying that the rows of N_k''' labeled $(x_1, \ldots, x_{k+1}) \in (\mathbb{Z}_p^*)^{p+1}$ are linearly independent and N_k' has rank $(p-1)^{k+1}$. Finally, by Lemma 4.5.23(a) the columns of N_k' labeled with $f \in F_k$ are also linear independent.

Linear equations mod 3

In this section we consider the case where $p = 3$ and provide linearly independent p-expressions that span the space of functions from $\mathbb{Z}_3^n$ to $\mathbb{C}$. The p-expressions we consider

here are different from the ones considered in Theorem 4.5.4 and we prove they are linearly independent using somewhat a simpler approach.

In this subsection set $p = 3$ and $\oplus$, $\odot$ denote addition and multiplication modulo 3, respectively. Let $x_1, \ldots, x_n$ be variables that take values from the ternary domain $\{0, 1, 2\}$. Here we prove that all the linear expressions of the form

$$(a_1 \odot x_1) \oplus (a_2 \odot x_2) \oplus \cdots \oplus (a_n \odot x_n)$$

with $a_i \in \{0, 1, 2\}$ are linearly independent, except the zero expression. For instance, in the case where $n = 1$, the following matrix has rank 2 meaning that x_1 and $2 \odot x_1$ are linearly independent.

$$A = \begin{array}{c} \begin{matrix} 0 & x_1 & 2 \odot x_1 \end{matrix} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 2 \\ 0 & 2 & 1 \end{pmatrix} \end{array} \begin{matrix} \\ x_1{=}0 \\ x_1{=}1 \\ x_1{=}2 \end{matrix}$$

Now define sequence of matrices as follows. Set $C_1 = A$ and recursively define C_n to be the following $3^n \times 3^n$ matrix

$$C_n = \begin{pmatrix} C_{n-1} & C_{n-1} & C_{n-1} \\ C_{n-1} & C_{n-1} \oplus \mathbf{1} & C_{n-1} \oplus \mathbf{2} \\ C_{n-1} & C_{n-1} \oplus \mathbf{2} & C_{n-1} \oplus \mathbf{1} \end{pmatrix}$$

Observation 4.5.24. *For any real numbers $a, b \neq 0$ and any integer n we have $\mathsf{rank}(aC_n + \mathbf{b}) = \mathsf{rank}(C_n) + 1$.*

Proof. Note that the first row and the first column of C_n contain only zeros. That is

$$C_n = \left(\begin{array}{c|ccc} 0 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & B & \\ 0 & & & \end{array}\right)$$

where B is a $3^n - 1 \times 3^n - 1$ matrix and has the same rank as C_n. Now, $\mathsf{rank}(aC_n + B) = \mathsf{rank}(C_n + \frac{\mathbf{b}}{\mathbf{a}})$.

$$C_n + \frac{\mathbf{b}}{\mathbf{a}} = \left(\begin{array}{c|ccc} \frac{b}{a} & \frac{b}{a} & \cdots & \frac{b}{a} \\ \hline \frac{b}{a} & & & \\ \vdots & & B + \frac{\mathbf{b}}{\mathbf{a}} & \\ \frac{b}{a} & & & \end{array}\right) \rightarrow \left(\begin{array}{c|ccc} \frac{b}{a} & \frac{b}{a} & \cdots & \frac{b}{a} \\ \hline 0 & & & \\ \vdots & & B + \mathbf{0} & \\ 0 & & & \end{array}\right) \rightarrow \left(\begin{array}{c|ccc} \frac{b}{a} & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & B & \\ 0 & & & \end{array}\right)$$

Hence, $\mathsf{rank}(aC_n + \mathbf{b}) = \mathsf{rank}(C_n + \frac{\mathbf{b}}{\mathbf{a}}) = \mathsf{rank}(C_n) + 1$. □

Lemma 4.5.25. *For any integer n, C_n has rank $3^n - 1$ i.e., all the linear expressions of the form $(a_1 \odot x_1) \oplus (a_2 \odot x_2) \oplus \cdots \oplus (a_n \odot x_n)$ with $a_i \in \{0, 1, 2\}$ are linearly independent, except the zero expression.*

Proof. The proof is by induction. Clearly, for $n = 1$, the matrix $C_1 = A$ has rank 2. Suppose C_i has rank $3^i - 1$ for all $1 \leq i \leq n$. For a matrix M with $0, 1, 2$ entries, let

$$p_1(M) = \frac{3}{2}M \circ M + \frac{5}{2}M + \mathbf{1}$$
$$p_2(M) = -\frac{3}{2}M \circ M - \frac{7}{2} + \mathbf{2}$$

where $\circ$ denotes the *Hadamard* product or the *element-wise* product of two matrices. Observe that $M \oplus \mathbf{1} = p_1(M)$ and $M \oplus \mathbf{2} = p_2(M)$. Hence, we can write C_{n+1} as follow

$$C_{n+1} = \begin{pmatrix} C_n & C_n & C_n \\ C_n & p_1(C_n) & p_2(C_n) \\ C_n & p_2(C_n) & p_1(C_n) \end{pmatrix}$$

Next, we perform a series of row and column operations to transform C_n into a block-diagonal matrix.

$$
\begin{aligned}
C_{n+1} &= \begin{pmatrix} C_n & C_n & C_n \\ C_n & p_1(C_n) & p_2(C_n) \\ C_n & p_2(C_n) & p_1(C_n) \end{pmatrix} \to \begin{pmatrix} C_n & C_n & C_n \\ \mathbf{0} & p_1(C_n) - C_n & p_2(C_n) - C_n \\ \mathbf{0} & p_2(C_n) - C_n & p_1(C_n) - C_n \end{pmatrix} \\
&\to \begin{pmatrix} C_n & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & p_1(C_n) - C_n & p_2(C_n) - C_n \\ \mathbf{0} & p_2(C_n) - C_n & p_1(C_n) - C_n \end{pmatrix} \\
&\to \begin{pmatrix} C_n & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & p_1(C_n) - C_n & p_2(C_n) - C_n \\ \mathbf{0} & p_1(C_n) + p_2(C_n) - 2C_n & p_1(C_n) + p_2(C_n) - 2C_n \end{pmatrix} \\
&\to \begin{pmatrix} C_n & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & p_1(C_n) - C_n & p_2(C_n) - C_n \\ \mathbf{0} & -3C_n + \mathbf{3} & -3C_n + \mathbf{3} \end{pmatrix} \\
&\to \begin{pmatrix} C_n & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & p_1(C_n) - C_n & p_1(C_n) + p_2(C_n) - 2C_n \\ \mathbf{0} & -3C_n + \mathbf{3} & -6C_n + \mathbf{6} \end{pmatrix} \\
&\to \begin{pmatrix} C_n & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & p_1(C_n) - C_n & -3C_n + \mathbf{3} \\ \mathbf{0} & -3C_n + \mathbf{3} & -6C_n + \mathbf{6} \end{pmatrix} \to \begin{pmatrix} C_n & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & p_1(C_n) - C_n - \frac{1}{2}(-3C_n + \mathbf{3}) & -3C_n + \mathbf{3} \\ \mathbf{0} & \mathbf{0} & -6C_n + \mathbf{6} \end{pmatrix} \\
&\to \begin{pmatrix} C_n & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & p_1(C_n) - C_n - \frac{1}{2}(-3C_n + \mathbf{3}) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -6C_n + \mathbf{6} \end{pmatrix} \\
&\to \begin{pmatrix} C_n & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\frac{3}{2}C_n \circ C_n + 3C_n - \mathbf{\frac{1}{2}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 6C_n - \mathbf{6} \end{pmatrix} \\
&\to \begin{pmatrix} C_n & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -3C_n \circ C_n + 6C_n - \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & C_n - \mathbf{1} \end{pmatrix}
\end{aligned}
$$

Hence, rank of C_{n+1} is $\mathsf{rank}(C_n) + \mathsf{rank}(C_n - \mathbf{1}) + \mathsf{rank}(-3C_n \circ C_n + 6C_n - \mathbf{1})$. Moreover, Observation 4.5.24 yields

$$
\begin{aligned}
\mathsf{rank}(C_{n+1}) &= \mathsf{rank}(C_n) + \mathsf{rank}(C_n - \mathbf{1}) + \mathsf{rank}(-3C_n \circ C_n + 6C_n - \mathbf{1}) \\
&= 3^n - 1 + 3^n + \mathsf{rank}(-3C_n \circ C_n + 6C_n - \mathbf{1})
\end{aligned}
$$

In what follows we prove that $\mathsf{rank}(-3C_n \circ C_n + 6C_n - \mathbf{1}) = 3^n$. Let us define the following two matrices associated to a matrix M with $\{0,1,2\}$ entries.

$$M^{\dagger}[i,j] = \begin{cases} 0 & \text{if } M[i,j] = 0 \\ 1 & \text{if } M[i,j] = 1 \\ 0 & \text{if } M[i,j] = 2 \end{cases} \quad \text{and} \quad M^{\dagger\dagger}[i,j] = \begin{cases} 0 & \text{if } M[i,j] = 0 \\ 0 & \text{if } M[i,j] = 1 \\ 2 & \text{if } M[i,j] = 2 \end{cases}$$

Note that $C_n = C_n^{\dagger} + C_n^{\dagger\dagger}$. For instance in the case $C_1 = A$ the two matrices $A^{\dagger}$ and $A^{\dagger\dagger}$ are $A^{\dagger} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ and $A^{\dagger\dagger} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 2 \\ 0 & 2 & 0 \end{pmatrix}$. Here, we simplify the expression $-3C_n \circ C_n + 6C_n - \mathbf{1}$ and write it in terms of $C_n^{\dagger}$ and $C_n^{\dagger\dagger}$.

$$\begin{aligned} -3C_n \circ C_n + 6C_n - \mathbf{1} &= -3(C_n^{\dagger} + C_n^{\dagger\dagger}) \circ (C_n^{\dagger} + C_n^{\dagger\dagger}) + 6(C_n^{\dagger} + C_n^{\dagger\dagger}) - \mathbf{1} \\ &= -3(C_n^{\dagger} \circ C_n^{\dagger} + C_n^{\dagger\dagger} \circ C_n^{\dagger\dagger}) + 6C_n^{\dagger} + 6C_n^{\dagger\dagger} - \mathbf{1} \\ &= -3(C_n^{\dagger} + 2C_n^{\dagger\dagger}) + 6C_n^{\dagger} + 6C_n^{\dagger\dagger} - \mathbf{1} \\ &= -3C_n^{\dagger} - 6C_n^{\dagger\dagger} + 6C_n^{\dagger} + 6C_n^{\dagger\dagger} - \mathbf{1} \\ &= 3C_n^{\dagger} - \mathbf{1} \end{aligned}$$

Claim 4.5.26. *For every positive integer n, the matrix $3C_n^{\dagger} - \mathbf{1}$ has full rank. This implies that $-3C_n \circ C_n + 6C_n - \mathbf{1}$ has full rank.*

Proof. For the base case $n = 1$, the matrix $3A^{\dagger} - \mathbf{1} = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix}$ has full rank i.e., $\mathsf{rank}(3A^{\dagger} - \mathbf{1}) = 3$. For our induction hypothesis suppose the claim is correct for every n. Next we show $3C_{n+1}^{\dagger} - \mathbf{1}$ has rank 3^{n+1}.

$$
\begin{aligned}
3C_{n+1}^{\dagger} &= 3\begin{pmatrix} C_n^{\dagger} & C_n^{\dagger} & C_n^{\dagger} \\ C_n^{\dagger} & (C_n \oplus \mathbf{1})^{\dagger} & (C_n \oplus \mathbf{2})^{\dagger} \\ C_n^{\dagger} & (C_n \oplus \mathbf{2})^{\dagger} & (C_n \oplus \mathbf{1})^{\dagger} \end{pmatrix} - \mathbf{1} \\
&= \begin{pmatrix} 3C_n^{\dagger} - \mathbf{1} & 3C_n^{\dagger} - \mathbf{1} & 3C_n^{\dagger} - \mathbf{1} \\ 3C_n^{\dagger} - \mathbf{1} & 3(C_n \oplus \mathbf{1})^{\dagger} - \mathbf{1} & 3(C_n \oplus \mathbf{2})^{\dagger} - \mathbf{1} \\ 3C_n^{\dagger} - \mathbf{1} & 3(C_n \oplus \mathbf{2})^{\dagger} - \mathbf{1} & 3(C_n \oplus \mathbf{1})^{\dagger} - \mathbf{1} \end{pmatrix} \\
&\to \begin{pmatrix} 3C_n^{\dagger} - \mathbf{1} & 3C_n^{\dagger} - \mathbf{1} & 3C_n^{\dagger} - \mathbf{1} \\ \mathbf{0} & 3(C_n \oplus \mathbf{1})^{\dagger} - 3C_n^{\dagger} & 3(C_n \oplus \mathbf{2})^{\dagger} - 3C_n^{\dagger} \\ \mathbf{0} & 3(C_n \oplus \mathbf{2})^{\dagger} - 3C_n^{\dagger} & 3(C_n \oplus \mathbf{1})^{\dagger} - 3C_n^{\dagger} \end{pmatrix} \\
&\to \begin{pmatrix} 3C_n^{\dagger} - \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 3(C_n \oplus \mathbf{1})^{\dagger} - 3C_n^{\dagger} & 3(C_n \oplus \mathbf{2})^{\dagger} - 3C_n^{\dagger} \\ \mathbf{0} & 3(C_n \oplus \mathbf{2})^{\dagger} - 3C_n^{\dagger} & 3(C_n \oplus \mathbf{1})^{\dagger} - 3C_n^{\dagger} \end{pmatrix} \\
&\to \begin{pmatrix} 3C_n^{\dagger} - \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (C_n \oplus \mathbf{1})^{\dagger} - C_n^{\dagger} & (C_n \oplus \mathbf{2})^{\dagger} - C_n^{\dagger} \\ \mathbf{0} & (C_n \oplus \mathbf{2})^{\dagger} - C_n^{\dagger} & (C_n \oplus \mathbf{1})^{\dagger} - C_n^{\dagger} \end{pmatrix}
\end{aligned}
$$

For a matrix M with $0, 1, 2$ entries, define $p_1'(M) = \frac{1}{2}M \circ M - \frac{3}{2}M + \mathbf{1}$ and $p_2'(M) = \frac{1}{2}M \circ M - \frac{1}{2}M$ where $\circ$ denotes the Hadamard product or the element-wise product of two matrices. Observe that $(M \oplus \mathbf{1})^{\dagger} = p_1'(M)$ and $(M \oplus \mathbf{2})^{\dagger} = p_2'(M)$.

$$
\begin{aligned}
p_1'(M) &= \frac{1}{2}(M^{\dagger} + M^{\dagger\dagger}) \circ (M^{\dagger} + M^{\dagger\dagger}) + \frac{3}{2}(M^{\dagger} + M^{\dagger\dagger}) + \mathbf{1} = -M^{\dagger} - \frac{1}{2}M^{\dagger\dagger} + \mathbf{1} \\
p_2'(M) &= \frac{1}{2}(M^{\dagger} + M^{\dagger\dagger}) \circ (M^{\dagger} + M^{\dagger\dagger}) - \frac{1}{2}(M^{\dagger} + M^{\dagger\dagger}) = \frac{1}{2}M^{\dagger\dagger}
\end{aligned}
$$

We continue by performing row and column operation to to transform $3C_{n+1} - \mathbf{1}$ into a block-diagonal matrix.

$$
\begin{pmatrix} 3C_n^\dagger - \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (C_n \oplus \mathbf{1})^\dagger - C_n^\dagger & (C_n \oplus \mathbf{2})^\dagger - C_n^\dagger \\ \mathbf{0} & (C_n \oplus \mathbf{2})^\dagger - C_n^\dagger & (C_n \oplus \mathbf{1})^\dagger - C_n^\dagger \end{pmatrix}
$$

$$
= \begin{pmatrix} 3C_n^\dagger - \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -2C_n^\dagger - \frac{1}{2}C_n^{\dagger\dagger} + \mathbf{1} & -C_n^\dagger + \frac{1}{2}C_n^{\dagger\dagger} \\ \mathbf{0} & -C_n^\dagger + \frac{1}{2}C_n^{\dagger\dagger} & -2C_n^\dagger - \frac{1}{2}C_n^{\dagger\dagger} + \mathbf{1} \end{pmatrix}
$$

$$
\to \begin{pmatrix} 3C_n^\dagger - \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -2C_n^\dagger - \frac{1}{2}C_n^{\dagger\dagger} + \mathbf{1} & -C_n^\dagger + \frac{1}{2}C_n^{\dagger\dagger} \\ \mathbf{0} & -3C_n^\dagger + \mathbf{1} & -3C_n^\dagger + \mathbf{1} \end{pmatrix}
$$

$$
\to \begin{pmatrix} 3C_n^\dagger - \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -C_n^\dagger - C_n^{\dagger\dagger} + \mathbf{1} & -C_n^\dagger + \frac{1}{2}C_n^{\dagger\dagger} \\ \mathbf{0} & \mathbf{0} & -3C_n^\dagger + \mathbf{1} \end{pmatrix} \to \begin{pmatrix} 3C_n^\dagger - \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -C_n^\dagger - C_n^{\dagger\dagger} + \mathbf{1} & -\frac{3}{2}C_n^\dagger + \mathbf{\frac{1}{2}} \\ \mathbf{0} & \mathbf{0} & -3C_n^\dagger + \mathbf{1} \end{pmatrix}
$$

$$
\to \begin{pmatrix} 3C_n^\dagger - \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -C_n^\dagger - C_n^{\dagger\dagger} + \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -3C_n^\dagger + \mathbf{1} \end{pmatrix} \to \begin{pmatrix} 3C_n^\dagger - \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & C_n^\dagger + C_n^{\dagger\dagger} - \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 3C_n^\dagger - \mathbf{1} \end{pmatrix}
$$

$$
= \begin{pmatrix} 3C_n^\dagger - \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & C_n - \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 3C_n^\dagger - \mathbf{1} \end{pmatrix}
$$

By the induction hypothesis, $\mathsf{rank}(3C_n^\dagger - \mathbf{1}) = 3^n$. Moreover, by the induction hypothesis and Observation 4.5.24 $\mathsf{rank}(C_n - \mathbf{1}) = 3^n$. As a result, $\mathsf{rank}(3C_{n+1}^\dagger - \mathbf{1}) = 3^n + 3^n + 3^n = 3^{n+1}$ and $-3C_n \circ C_n + 6C_n - \mathbf{1}$ has full rank. □

Recall that $\mathsf{rank}(C_{n+1}) = \mathsf{rank}(C_n) + \mathsf{rank}(C_n - \mathbf{1}) + \mathsf{rank}(-3C_n \circ C_n + 6C_n - \mathbf{1})$. By the induction hypothesis and Observation 4.5.24, we have $\mathsf{rank}(C_n - \mathbf{1}) = \mathsf{rank}(C_n) + 1 = 3^n$. Moreover, Claim 4.5.26 yields $\mathsf{rank}(-3C_n \circ C_n + 6C_n - \mathbf{1}) = 3^n$. Hence,

$$\mathsf{rank}(C_{n+1}) = 3^n - 1 + 3^n + 1 + 3^n + 1 = 3^{n+1} - 1.$$

□

Chapter 5

Finding membership proofs and applications

In this section, we study a variation of the IMP and show similar reductions for pp-definability notion as well as pp-interpretability notion. These reductions provide a very strong tool for solving the search version of the IMP. More generally, using the algebraic approach we present a general framework for computing d-truncated Gröbner Bases for combinatorial ideals. Moreover, in the later sections we discuss applications of this variation of the IMP in studying bit complexity of SOS proofs, and how it can be used to construct theta bodies of combinatorial problems.

5.1 The IMP with indeterminate coefficients

In the new variant of the IMP we are given a polynomial with unknown coefficients and the goal is to decide if there is an assignment for coefficients such that the resulting polynomial belongs to an ideal. Formally speaking,

Definition 5.1.1 (χIMP)**.** *Given an ideal* $\mathtt{I} \subseteq \mathbb{F}[x_1, \ldots, x_n]$ *and a vector of* ℓ *polynomials* $M = (g_1, \ldots, g_\ell)$*, the* χ*IMP asks if there exist coefficients* $\mathbf{c} = (c_1, \ldots, c_\ell) \in \mathbb{F}^\ell$ *such that* $\mathbf{c}M = \sum_{i=1}^{\ell} c_i g_i$ *belongs to the ideal* $\mathtt{I}$*.*

In a similar fashion, χIMP associated with a constraint language Γ over a set D is the problem $\chi\mathrm{IMP}(\Gamma)$ in which the input is a pair $(M, \mathcal{P})$ where $\mathcal{P} = (X, D, C)$ is a $\mathrm{CSP}(\Gamma)$ instance and M is a vector of ℓ polynomials. The goal is to decide whether there are coefficients $\mathbf{c} = (c_1, \ldots, c_\ell) \in \mathbb{F}^\ell$ such that $\mathbf{c}M$ lies in the combinatorial ideal $\mathtt{I}(\mathcal{P})$. We use $\chi\mathrm{IMP}_d(\Gamma)$ to denote $\chi\mathrm{IMP}(\Gamma)$ when the vector M contains polynomials of degree at most d. By the search χIMP we understand the following problem.

> **Search version of χIMP.** Let $(M, \mathcal{P})$ be an instance of $\chi\mathrm{IMP}(\Gamma)$ where there are coefficients $\mathbf{c} = (c_1, \ldots, c_\ell) \in \mathbb{F}^\ell$ so that $\mathbf{c}M \in \mathtt{I}(\mathcal{P})$, the problem is to find a $\mathbf{c} = (c_1, \ldots, c_\ell) \in \mathbb{F}^\ell$ such that $\mathbf{c}M \in \mathtt{I}(\mathcal{P})$.

Next, we show that the main reductions from Section 3.1 work for the χIMP as well.

Theorem 5.1.2. *If Γ pp-defines Δ, then $\chi IMP(\Delta)$ is polynomial time reducible to $\chi IMP(\Gamma)$.*

Proof. The proof of Theorem 5.1.2 closely follows that of Theorem 3.1.11. Let $(M, \mathcal{P}_\Delta)$, $\mathcal{P}_\Delta = (X, D, C_\Delta)$, be an instance of χIMP(Δ) where $X = \{x_{m+1}, \ldots, x_{m+k}\}$, M is a vector of ℓ polynomials in $x_{m+1}, \ldots, x_{m+k}$, $k = |X|$, and m will be defined later, and $\mathcal{I}(\mathcal{P}_\Delta) \subseteq \mathbb{F}[x_{m+1}, \ldots, x_{m+k}]$. From this we construct an instance $(M', \mathcal{P}_\Gamma)$ of χIMP(Γ) where M' is a vector of ℓ' polynomials in $x_1, \ldots, x_{m+k}$ and $\mathcal{I}(\mathcal{P}_\Gamma) \subseteq \mathbb{F}[x_1, \ldots, x_{m+k}]$ such that

$$\exists \mathbf{c} \in \mathbb{F}^\ell \text{ with } \mathbf{c}M \in \mathcal{I}(\mathcal{P}_\Delta) \iff \exists \mathbf{c}' \in \mathbb{F}^{\ell'} \text{ with } \mathbf{c}'M' \in \mathcal{I}(\mathcal{P}_\Gamma).$$

Using pp-definitions of relations from Δ we convert the instance $\mathcal{P}_\Delta$ into an instance $\mathcal{P}_\Gamma = (\{x_1, \ldots, x_{m+k}\}, D, C_\Gamma)$ of CSP(Γ) such that every solution of $\mathcal{P}_\Delta, \mathcal{P}_\Gamma$ satisfy the Extension Condition 3.1.2. Such an instance $\mathcal{P}_\Gamma$ can be constructed in polynomial time as follows.

By the assumption each $S \in \Delta$, say, t_S-ary, is pp-definable in Γ by a pp-formula involving relations from Γ and the equality relation, $=_D$. Thus,

$$S(y_{q_S+1}, \ldots, y_{q_s+t_S}) = \exists y_1, \ldots, y_{q_S}(R_1(w_1^1, \ldots, w_{l_1}^1) \wedge \cdots \wedge R_r(w_1^r, \ldots, w_{l_r}^r)),$$

where $w_1^1, \ldots, w_{l_1}^1, \ldots, w_1^k, \ldots, w_{l_k}^k \in \{y_1, \ldots, y_{m_S+t_S}\}$ and $R_1, \ldots, R_r \subseteq \Gamma \cup \{=_D\}$.

Now, for every constraint $B = \langle \mathbf{s}, S \rangle \in C_\Delta$, where $\mathbf{s} = (x_{i_1}, \ldots, x_{i_t})$ create a fresh copy of $\{y_1, \ldots, y_{q_S}\}$ denoted by Y_B, and add the following constraints to C_Γ

$$\langle (w_1^1, \ldots, w_{l_1}^1), R_1 \rangle, \ldots, \langle (w_1^r, \ldots, w_{l_r}^r), R_r \rangle.$$

We then set $m = \sum_{B \in C} |Y_B|$ and assume that $\cup_{B \in C} Y_B = \{x_1, \ldots, x_m\}$. Note that the problem instance obtained by this procedure belongs to CSP($\Gamma \cup \{=_D\}$). All constraints of the form $\langle (x_i, x_j), =_D \rangle$ can be eliminated by replacing all occurrences of the variable x_i with x_j. Moreover, it can be checked (see also Theorem 2.16 in [39]) that $\mathcal{P}_\Delta, \mathcal{P}_\Gamma$ satisfy the Extension Condition 3.1.2.

Let $\mathcal{I}(\mathcal{P}_\Gamma) \subseteq \mathbb{F}[x_1, \ldots, x_{m+k}]$ be the ideal corresponding to $\mathcal{P}_\Gamma$ and set $M' = M$. Now, $(M', \mathcal{P}_\Gamma)$ is an instance of χIMP(Γ). We prove that, for every $\mathbf{c} \in \mathbb{F}^\ell$, $\mathbf{c}M \in \mathcal{I}(\mathcal{P}_\Delta)$ if and only if $\mathbf{c}M \in \mathcal{I}(\mathcal{P}_\Gamma)$.

Consider an arbitrary $\mathbf{c} \in \mathbb{F}^\ell$ and set $f_0 = \mathbf{c}M$. Suppose $f_0 \notin \mathcal{I}(\mathcal{P}_\Delta)$, this means there exists $\varphi \in \mathbf{V}(\mathcal{I}(\mathcal{P}_\Delta))$ such that $f_0(\varphi) \neq 0$. By Theorem 3.1.9, φ can be extended to a point $\varphi' \in \mathbf{V}(\mathcal{I}(\mathcal{P}_\Gamma))$. This in turn implies that $f_0 \notin \mathcal{I}(\mathcal{P}_\Gamma)$. Conversely, suppose $f_0 \notin \mathcal{I}(\mathcal{P}_\Gamma)$. Hence, there exists $\varphi' \in \mathbf{V}(\mathcal{I}(\mathcal{P}_\Gamma))$ such that $f_0(\varphi') \neq 0$. Projection of φ' to its last k coordinates gives a point $\varphi \in \mathbf{V}(\mathcal{I}_X)$. By Lemma 3.1.10, $\varphi \in \mathbf{V}(\mathcal{I}(\mathcal{P}_\Delta))$ which implies $f_0 \notin \mathcal{I}(\mathcal{P}_\Delta)$. □

The reduction for pp-interpretable languages remain valid in the case of χIMP as well.

Theorem 5.1.3. *Let Γ, Δ be constraint languages on sets D, E, respectively, and let Γ pp-interprets Δ. Then $\chi IMP_d(\Delta)$ is polynomial time reducible to $\chi IMP_{d\ell|E|}(\Gamma)$.*

Proof. Let $(M, \mathcal{P}_\Delta)$ be an instance of $\chi\mathrm{IMP}(\Delta)$ where M is a vector of r polynomials in $x_1, \ldots, x_n$, $\mathcal{P}_\Delta = (\{x_1, \ldots, x_n\}, E, C_\Delta)$, an instance of $\mathrm{CSP}(\Delta)$, and $\mathcal{I}(\mathcal{P}_\Delta) \subseteq \mathbb{F}[x_1, \ldots, x_n]$.

The properties of the mapping π from Definition 3.1.13 allow us to rewrite an instance of $\mathrm{CSP}(\Delta)$ to an instance of $\mathrm{CSP}(\Gamma')$ over the constraint language Γ'. Recall that, by Definition 3.1.13, Γ' contains all the ℓk-ary relations $\pi^{-1}(S)$ on D where $S \in \Delta$ is k-ary relation, as well as the 2ℓ-ary relation $\pi^{-1}(=_E)$.

Note that Γ' is pp-definable from Γ. By Theorem 5.1.2, $\chi\mathrm{IMP}(\Gamma')$ is reducible to $\chi\mathrm{IMP}(\Gamma)$. It remains to show $\chi\mathrm{IMP}(\Delta)$ is reducible to $\chi\mathrm{IMP}(\Gamma')$. To do so, from instance $(M, \mathcal{P}_\Delta)$ of $\chi\mathrm{IMP}(\Delta)$ we construct an instance $(M', \mathcal{P}_{\Gamma'})$ of $\chi\mathrm{IMP}(\Gamma')$ such that

$$\exists \mathbf{c} \in \mathbb{F}^r \text{ with } \mathbf{c}M \in \mathcal{I}(\mathcal{P}_\Delta) \iff \exists \mathbf{c}' \in \mathbb{F}^{r'} \text{ with } \mathbf{c}'M' \in \mathcal{I}(\mathcal{P}'_\Gamma).$$

Let p be a polynomial of total degree at most $\ell|E|$ that interpolates mapping π. For each monomial $\mathbf{x}^\alpha = \prod x_i^{\alpha_i}$ in M replace each indeterminate x_i with $p(x_{1i}, \ldots, x_{\ell i})$. This yields the following polynomial

$$\prod_{i=1}^{n} [p(x_{1i}, \ldots, x_{\ell i})]^{\alpha_i} \tag{5.1}$$

Note that for a monomial of total degree at most d, the maximal degree of monomials appearing in the polynomial (5.1) is at most $d\ell|E|$. Let M' be the vector of monomials consisting monomials in (5.1) for all monomials in M. Observe that M' contains at most $O(n^{d\ell|E|})$ monomials and each monomial in M' consists of indeterminates $x_{11}, \ldots, x_{\ell 1}, \ldots, x_{1n}, \ldots, x_{\ell n}$. Now, $(M', \mathcal{P}_{\Gamma'})$ is an instance of $\chi\mathrm{IMP}(\Gamma')$.

Consider an arbitrary $\mathbf{c} \in \mathbb{F}^r$ and set $f_0 = \mathbf{c}M \in \mathbb{F}[x_1, \ldots, x_n]$. Let

$$f'_0 \in \mathbb{F}[x_{11}, \ldots, x_{\ell 1}, \ldots, x_{1n}, \ldots, x_{\ell n}]$$

be the polynomial that is obtained from f_0 by replacing each indeterminate x_i with $p(x_{1i}, \ldots, x_{\ell i})$. Note that there exists $\mathbf{c}'$ such that $f'_0 = \mathbf{c}'M'$. Clearly, for any assignment $\varphi : \{x_1, \ldots, x_n\} \to E$, $f_0(\varphi) = 0$ if and only if $f'_0(\psi) = 0$ for every $\psi : \{x_{11}, \ldots, x_{\ell n}\} \to D$ such that

$$\varphi(x_i) = \pi(\psi(x_{1i}), \ldots, \psi(x_{\ell i}))$$

for every $i \leq n$. Moreover, for any such φ, ψ it holds $\varphi \in \mathbf{V}(\mathcal{I}(\mathcal{P}_\Delta))$ if and only if $\psi \in \mathbf{V}(\mathcal{I}(\mathcal{P}_{\Gamma'}))$. This yields that

$$(\exists \varphi \in \mathbf{V}(\mathcal{I}(\mathcal{P}_\Delta)) \wedge f_0(\varphi) \neq 0) \iff (\exists \psi \in \mathbf{V}((\mathcal{P}_{\Gamma'})) \wedge f'_0(\psi) \neq 0)$$

This completes the proof of the theorem. □

Recall that one drawback of the reductions for the IMP, Theorems 3.1.11 and 3.1.15, is the issue of recovering a proof which is a subtle point in the search version of the IMP. A nice property of the reductions in Theorems 5.1.2 and 5.1.3 is that they provide reductions for the search version of the χIMP as well. To elaborate, consider the reduction for pp-interpretablility in the proof of Theorem 5.1.3. The entries of vector $\mathbf{c}'$ are linear combination of $c_1, \ldots, c_\ell$. Hence, if there exists a polynomial time algorithm that finds $\mathbf{c}'$ such that $\mathbf{c}'M' \in \mathcal{I}(\mathcal{P}_{\Gamma'})$ then a vector $\mathbf{c}$ with $\mathbf{c}M \in \mathcal{I}(\mathcal{P}_{\Delta})$ can be computed by simply solving a system of linear equations with $c_1, \ldots, c_\ell$ as unknowns. This is formalized as follows.

Theorem 5.1.4. *Let Γ and Δ be constraint languages on (possibly similar) sets D, E, respectively. Suppose there exists a polynomial time algorithm that solves the search version of χIMP(Γ). Then, there exists a polynomial time algorithm that solves the search version of χIMP(Δ) if*

1. *$D = E$ and Γ pp-defines Δ, or*
2. *Γ pp-interprets Δ.*

Proof. It follows from a similar argument in the proofs of Theorems 5.1.2, 5.1.3, and noting that $\mathbf{c}'$ is a linear combination of $c_1, \ldots, c_\ell$. □

5.1.1 Sufficient conditions for tractability of χIMP

We first show that having a Gröbner Basis yields a polynomial time algorithm for solving the search version of χIMP. Next, we use the reductions from Theorem 5.1.4 to establish the tractability of $\chi\text{IMP}_d(\Gamma)$ for languages closed under various polymorphisms.

Theorem 5.1.5. *Let $\mathtt{I}$ be an ideal, and let $\{g_1, \ldots, g_s\}$ be a given (d-truncated) Gröbner Basis for $\mathtt{I}$ with respect to a grlex. Then the (search version of) χIMP_d is polynomial time solvable.*

Proof. Recall that a polynomial p belongs to $\mathcal{I}$ if and only if the remainder on division of p by $g_1, \ldots, g_s$ is zero. Let $M = (m_1, \ldots, m_\ell)$ be a vector of ℓ polynomials and $\mathbf{c} = (c_1, \ldots, c_\ell) \in \mathbb{F}^\ell$ be a vector of unknown coefficients. Set $f = \mathbf{c}M = \sum c_i m_i$. We do the division algorithm to obtain the reminder of dividing f by $g_1, \ldots, g_s$. Repeatedly, choose a $g_i \in \{g_1, \ldots, g_s\}$ such that $\text{LT}(g_i)$ divides some term t of f and replace f with $f - \frac{t}{\text{LT}(g_i)} g_i$, until it cannot be further applied. Hence,

$$f = q_1 g_1 + \cdots + q_r g_s + r$$

where r is a linear combination, with unknown coefficients in $\mathbb{F}$, of monomials, none of which is divisible by any of $\text{LT}(g_1), \ldots, \text{LT}(g_s)$. The key observation is that the coefficients

of monomials in r are linear combination of $c_1, \ldots, c_\ell$. Now, we want r to be the zero polynomial. Hence, we set every unknown coefficient of monomials in r to be zero. This in turn yields a system of linear equations in $c_1, \ldots, c_\ell$. Such a system of linear equations has a solution if and only if there exists $\mathbf{c} = (c_1, \ldots, c_\ell)$ such that $f = \mathbf{c}M \in \mathcal{I}$. □

The above theorem and the results by Mastrolilli [161, 25] give the following corollary.

Corollary 5.1.6. *Let Γ be a finite constraint language over domain $\{0,1\}$. Then the (search version of) $\chi IMP_d(\Gamma)$ can be solved in polynomial time if*

1. *Γ has a semilattice polymorphism, or*
2. *Γ has a majority polymorphism, or*
3. *Γ has a minority polymorphism.*

Now we use our reductions to prove the same tractability results for languages over arbitrary finite domain. Note that the only majority polymorphism over $\{0,1\}$ is the dual-discriminator.

Theorem 5.1.7. *Let Γ be a finite constraint language over domain D. Then the (search version of) $\chi IMP_d(\Gamma)$ can be solved in polynomial time if*

1. *Γ has a semilattice polymorphism, or*
2. *Γ has the dual-discriminator polymorphism, or*
3. *Γ is expressed as a system of linear equations over $\mathsf{GF}(p)$, p prime.*

Proof. In the first case where Γ has a semilattice polymorphism we reduce the problem to the Boolean case similar to Theorem 4.2.3. That is, there exists a finite constraint language Δ over $\{0,1\}$ with a semilattice polymorphism so that Δ pp-interprets Γ. Now by Theorem 5.1.3 $\chi\mathrm{IMP}_d(\Gamma)$ is polynomial time reducible to $\chi\mathrm{IMP}_{ud}(\Delta)$ for a constant u. Then using Theorem 5.1.4 and Corollary 5.1.6 we can solve $\chi\mathrm{IMP}_d(\Gamma)$.

In the second case, let $\mathcal{P} = (X, D, C)$ be an instance of $\mathrm{CSP}(\Gamma)$ where $\nabla \in \mathsf{Pol}(\Gamma)$, and $(M, \mathcal{P})$ be an instance of $\chi\mathrm{IMP}_d(\Gamma)$ where M has length ℓ. Recall the preprocessing step in Lemma 4.1.4 that yields an instance $\mathcal{P}' = (X', D, C')$ where C' contains no permutation constraints. In a similar fashion as Lemma 4.1.4 we can transform the vector of polynomials M and obtain a vector of polynomials M' which consists of polynomials of degree at most $O(d)$, and in addition we have

$$\exists \mathbf{c} \in \mathbb{F}^\ell \text{ with } \mathbf{c}M \in \mathcal{I}(\mathcal{P}) \iff \exists \mathbf{c}' \in \mathbb{F}^\ell \text{ with } \mathbf{c}'M' \in \mathcal{I}(\mathcal{P}').$$

Note that in the above each entry of $\mathbf{c}'$ is a linear combination of $c_1, \ldots, c_\ell$, entries of $\mathbf{c}$. Now by Theorem 4.1.6, we can compute an $O(d)$-truncated Gröbner Basis with respect to

grlex for $\mathcal{I}(\mathcal{P}')$. Hence, by Theorem 5.1.9, the instance $(M', \mathcal{P}')$ is polynomial time solvable. This in turn, gives a system of linear equations over $c_1, \ldots, c_\ell$. Solving this system of linear equations yields a solution for $(M, \mathcal{P})$, if one exists.

For the third case we use a reduction similar to Theorems 4.3.3 to transform the instances into instances over roots of unities, where we can compute truncated Gröbner Bases. The key observation here is that under these reductions, similar to the reduction for pp-interpretability in Theorem 5.1.4 or the transformation in the second case, we end up with system of linear equations over unknown variables $c_1, \ldots, c_\ell$. □

We prove a similar result for constraint languages invariant under the affine operation of an Abelian group. Indeed, in Section 5.2, we provide a unifying framework based on substitution techniques that covers all the cases here.

5.1.2 A framework for constructing d truncated Gröbner Bases

We observe that constructing a d-truncated Gröbner Basis for an ideal $\mathcal{I}$ is reducible to solving χIMP_d for the ideal $\mathcal{I}$. With this reduction at hand, we design algorithms to construct d-truncated Gröbner Basis for many combinatorial ideals, namely, combinatorial ideals arising from languages invariant under a semilattice, or the dual-discriminator, or languages expressible as linear equations over $\mathsf{GF}(p)$. Some basic notation are in order.

Let $\mathcal{I} \in \mathbb{F}[X]$ be an ideal. We say two polynomials f, g are congruent modulo $\mathcal{I}$ and write $f \equiv g \ mod \ \mathcal{I}$ if $f - g \in \mathcal{I}$. It is easy to see that congruence modulo $\mathcal{I}$ is an equivalence relation on $\mathbb{F}[X]$. The quotient of $\mathbb{F}[X]$ modulo $\mathcal{I}$, written $\mathbb{F}[X]/\mathcal{I}$ is a ring with the base set consisting of the cosets $[f] = f + \mathcal{I} = \{f + q \mid q \in \mathcal{I}\}$. $\mathbb{F}[X]/\mathcal{I}$ is a commutative ring under addition $[f] + [g] = [f + g]$ and multiplication $[f] \cdot [g] = [fg]$ (product in $\mathbb{F}[X]$). We consider $\mathbb{F}[X]/\mathcal{I}$ as a $\mathbb{F}$-vector space with addition defined as above and scalar multiplication given by $c \cdot [f] = [c \cdot f]$, $c \in \mathbb{F}$. We also consider the subset $\mathbb{F}[X]_d/\mathcal{I}$ of all polynomials of total degree at most d. As is easily seen it is also an $\mathbb{F}$-vector space. Note that $\mathbb{F}[X]/\mathcal{I}$ is infinitely dimensional in general. However, if $\mathcal{I}$ is zero-dimensional and radical then the quotient ring $\mathbb{F}[X]/\mathcal{I}$ is a finite dimensional vector space. Moreover, for any bound d on the total degree of polynomials $\mathbb{F}[X]_d/\mathcal{I}$ is finitely dimensional. We also have a natural basis for those spaces.

Proposition 5.1.8 (Proposition 1 on page 248 of [59])**.** *Fix a monomial ordering on* $\mathbb{F}[X]$ *and let* $\mathtt{I} \subseteq \mathbb{F}[X]$ *be an ideal. Let* $\langle \text{LT}(I) \rangle$ *denote the ideal generated by the leading terms of elements of* $\mathtt{I}$*.*

1. *Every* $f \in \mathbb{F}[X]$ *is congruent modulo* $\mathtt{I}$ *to a* unique *polynomial* r *which is a* $\mathbb{F}$*-linear combination of the monomials in the complement of* $\langle \text{LT}(\mathtt{I}) \rangle$*,*

2. *The elements of* $\{\mathbf{x}^\alpha \mid \mathbf{x}^\alpha \notin \langle \text{LT}(\mathtt{I}) \rangle\}$ *are linearly independent modulo* $\mathtt{I}$*, i.e., if we have*

$$\sum_\alpha c_\alpha \mathbf{x}^\alpha \equiv 0 \ mod \ \mathtt{I},$$

Algorithm 2 d-Truncated Gröbner Bases

Require: $\mathcal{I}$, degree d.

Let Q be the list of all monomials of degree at most d arranged in increasing grlex order.
$G = \emptyset, B(G) = \{1\}$ (we assume $1 \notin \mathcal{I}$).
Let b_i (arranged in increasing grlex order) be the elements of $B(G)$.
for $q \in Q$ **do**
    if q is divisible by some LM in G **then**
        Discard it and go to Step 4,
    Let M be the vector of length ℓ whose entries are monomials in $B(G)$,
    if there exists $\mathbf{c} \in \mathbb{F}^\ell$ such that $q - \mathbf{c}M \in \mathcal{I}$ **then**
        $G = G \cup \{q - \mathbf{c}M\}$
    else
        $B(G) = B(G) \cup \{q\}$
return G

where the $\mathbf{x}^\alpha$ *are all in the complement of* $\langle \mathrm{LT}(\mathtt{I}) \rangle$*, then* $c_\alpha = 0$ *for all* α*.*

Proposition 5.1.8 suggests a simple algorithm to construct a d-truncated Gröbner Basis. Let $\mathcal{I} \subseteq \mathbb{F}[X]$ be an ideal. At the beginning of the algorithm, there will be two sets: G, which is initially empty but will become the d-truncated Gröbner Basis with respect to the grlex order, and $B(G)$, which initially contains 1 and will grow to be the grlex monomial basis of the quotient ring $\mathbb{F}[x_1, \ldots, x_n]/\mathcal{I}$ as a $\mathbb{F}$-vector space i.e., $B(G) = \{\mathbf{x}^\alpha \mid |\alpha| \leq d, \mathbf{x}^\alpha \notin \langle \mathrm{LT}(\mathcal{I}) \rangle\}$. In fact, $B(G)$ contains the reduced monomials (of degree at most d) with respect to G. Every $f \in \mathbb{F}[x_1, \ldots, x_n]$ is congruent modulo $\mathcal{I}$ to a unique polynomial r which is a $\mathbb{F}$-linear combination of the monomials in the *complement* of $\langle \mathrm{LT}(\mathcal{I}) \rangle$. Furthermore, $\mathbb{F}[X]_d/\mathcal{I}$ is isomorphic as a $\mathbb{F}$-vector space to $\mathtt{Span}(\mathbf{x}^\alpha \mid \mathbf{x}^\alpha \notin \langle \mathrm{LT}(\mathcal{I}) \rangle)$ via mapping $\Phi([f]) = f|_G$. Here, $\mathtt{Span}(\mathbf{x}^\alpha \mid \mathbf{x}^\alpha \notin \langle \mathrm{LT}(\mathcal{I}) \rangle)$ means the set of all $\mathbb{F}$-linear combinations of $\{\mathbf{x}^\alpha \mid \mathbf{x}^\alpha \notin \langle \mathrm{LT}(\mathcal{I}) \rangle)\}$. Hence, for every $f \in \mathbb{F}[x_1, \ldots, x_n]$, we have

$$f|_G \in \mathtt{Span}(\mathbf{x}^\alpha \mid \mathbf{x}^\alpha \notin \langle \mathrm{LT}(\mathcal{I}) \rangle).$$

This suggests the following algorithm, inspired by the famous FGLM algorithm [75] and the conversion algorithm in [25]. In Algorithm 2, Q is the list of all monomials of degree at most d arranged in increasing order with respect to grlex ordering. The algorithm iterates over monomials in Q in increasing grlex order and at each iteration decides exactly one of the following actions given the current sets G and $B(G)$.

1. q should be discarded (if q is divisible by some LM in G), or
2. a polynomial with q as its leading monomial should be added to G, or
3. q should be added to $B(G)$.

Theorem 5.1.9. *Let $\mathcal{H}$ be a class of ideals for which the search version of χIMP_d is polynomial time solvable. Then there exists a polynomial time algorithm (see Algorithm 2) that constructs a degree d Gröbner Basis of an ideal $\mathtt{I} \in \mathcal{H}$ with respect to a grlex order, $\mathtt{I} \subseteq \mathbb{F}[x_1, \ldots, x_n]$, in time $O(n^d)$.*

Proof. Algorithm 2 is clearly a polynomial time algorithm assuming χIMP_d for ideal $\mathtt{I}$ is polynomial time solvable. We prove G returned by the algorithm is d-truncated Gröbner Basis, and set of monomials $B(G)$ is so that

$$B(G) = \{\mathbf{x}^\alpha \mid |\alpha| \leq d, \mathbf{x}^\alpha \notin \langle \text{LT}(\mathtt{I})\rangle\}.$$

We prove this by induction. The induction base is correct as $B(G) = \{1\}$ and $G = \emptyset$. Suppose sets G and $B(G)$ are computed correctly up to the i-th iteration and let q be the current monomial.

First, if q is a multiple of some LM in G then $q \in \langle \text{LT}(G)\rangle$. Furthemore, no polynomial with q as its leading monomial is in a reduced Gröbner Basis of $\mathtt{I}$ (recall the definition of a reduced Gröbner Basis). Therefore, in this case, Algorithm 2 correctly discards monomial q.

Second, suppose q is not divisible by any LM in G then by the division algorithm the normal form of q by G, $q|_G$, is q itself. Now the algorithm decides if a polynomial with q as its leading monomial can be in G. Let $g = q + f$ be a polynomial such that $\text{LM}(g) = q$. Therefore, by Proposition 5.1.8 and the inductive hypothesis, if $g \in \mathtt{I}$ then with the current G and $B(G)$ we must have

$$0 = g|_G = q|_G + f|_G = q + \sum k_i b_i$$

where all

$$b_i \in M = \{\mathbf{x}^\alpha \mid deg(\mathbf{x}^\alpha) < deg(q), \mathbf{x}^\alpha \notin \langle \text{LT}(\mathtt{I})\rangle\}$$

and $k_i \in \mathbb{R}$. This yields $g \in \mathtt{I}$ if there exists $\mathbf{c} \in \mathbb{R}^\ell$ such that $q - \mathbf{c}M \in \mathtt{I}(\mathcal{P})$ then $\{q - \mathbf{c}M\} \in \mathtt{I}$. Furthermore, if such $\mathbf{c} \in \mathbb{R}^\ell$ does not exists then it implies there is no polynomial in $\mathtt{I}$ with q as its leading monomial. Hence, q must be added to $B(G)$. □

We point out that in Theorem 5.1.9, if only the decision version of χIMP is polynomial time solvable then a slight modification of Algorithm 2 returns basis monomials $\{\mathbf{x}^\alpha \mid |\alpha| \leq d, \mathbf{x}^\alpha \notin \langle \text{LT}(\mathtt{I})\rangle\}$.

Theorem 5.1.10. *Let Γ be a finite constraint language over domain D. For an instance $\mathcal{P}$ of CSP(Γ) a d-truncated Gröbner Basisof $\mathtt{I}(\mathcal{P})$ with respect to a grlex order can be computed in time $O(n^d)$ if*

1. *Γ has a semilattice polymorphism, or*

2. Γ *has the dual-discriminator polymorphism, or*

3. Γ *is expressed as a system of linear equations over* $\mathsf{GF}(p)$*,* p *prime.*

Proof. Follows from Theorems 5.1.7 and 5.1.9. □

We prove a similar result for constraint languages invariant under the affine operation of an Abelian group. Indeed, in the following section we formalize the main idea of the above theorem in terms of a unifying framework based on substitution techniques that covers all the cases here. We believe our substitution techniques will find further applications in the study of IMP.

5.2 Finding a proof and the substitution technique

In the previous sections we introduced a framework to bridge the gap between the decision of IMP and finding a proof of membership. Indeed, the framework gives a polynomial time algorithm to construct a truncated Gröbner Basis provided that the search version of the χIMP_d is polynomial time solvable. Then it was observed in Theorem 5.1.5 that having a (truncated) Gröbner Basis yields a polynomial time algorithm for solving the search version of χIMP_d. Given this, to solve the χIMP_d one might reduce the problem at hand to a problem for which a (truncated) Gröbner Basis can be constructed in a relatively simple way. This approach was successfully applied in various cases in Theorem 5.1.7 in an ad hoc manner. However, the core idea in all of them is a substitution technique. Here we provide a unifying construction based on *substitution reductions* that covers all the useful cases so far.

5.2.1 Reduction by substitution

We call a class of IMPs or χIMPs *CSP-based* if its instances are of the form $(f, \mathcal{P})$ or $(M, \mathcal{P})$, where $\mathcal{P}$ is a CSP instance over a fixed set D. Let $\mathcal{X}, \mathcal{Y}$ be restricted CSP-based classes of the χIMP. The classes $\mathcal{X}, \mathcal{Y}$ can be defined by various kinds of restrictions, for example, as $\chi\text{IMP}(\Gamma), \chi\text{IMP}(\Delta)$, but not necessarily. Let the domain of $\mathcal{X}$ be D and the domain of $\mathcal{Y}$ be E. Let also $\mu_1, \ldots, \mu_k$ be a collection of surjective functions $\mu_i : E^{\ell_i} \to D$, $i \in [k]$. Each mapping μ_i can be interpolated by a polynomial h_i. We call the collection $\{h_1, \ldots, h_k\}$ a *substitution collection*.

We define substitution reductions for the χIMP, for the IMP it is quite similar. The problem $\mathcal{X}$ is said to be *substitution reducible* to $\mathcal{Y}$ if there exists a substitution collection $\{h_1, \ldots, h_k\}$ and a polynomial time algorithm A such that for every instance $(M, \mathcal{P})$ of $\mathcal{X}$ the instance constructed as follows belongs to $\mathcal{Y}$.

(1) Let X be the set of variables of $(M, \mathcal{P})$. For every $x \in X$ the algorithm A selects a polynomial h_{i_x} and a set of variables Y_x such that

(a) $|Y_x| = \ell_{i_x}$;

(b) for any $x, y \in X$ either $Y_x = Y_y$ or $Y_x \cap Y_y = \emptyset$;

(c) if $x_1, \ldots, x_r \in X$ are such that $Y_{x_1} = \cdots = Y_{x_r} = \{y_1, \ldots, y_{\ell_j}\}$ then for any solution φ of $\mathcal{P}$ there are values $a_1, \ldots, a_{\ell_j} \in E$ such that $\varphi(x_i) = h_{i_{x_i}}(a_1, \ldots, a_{\ell_j})$.

(2) If $M = (g_1, \ldots, g_\ell)$ then $M' = (g'_1, \ldots, g'_\ell)$, where for $g_i(x_1, \ldots, x_t)$

$$g'_i = g_i(h_{i_{x_1}}(Y_{x_1}), \ldots, h_{i_{x_y}}(Y_{x_t})).$$

(3) Let $Y = \bigcup_{x \in X} Y_x$. The instance $\mathcal{P}'$ is given by $(Y, E, \mathcal{C}')$, where for every constraint $\langle \mathbf{s}, R \rangle$, $\mathbf{s} = (x_1, \ldots, x_t)$, $\mathcal{P}'$ contains the constraint $\langle \mathbf{s}', R' \rangle$ such that

- $\mathbf{s}' = (x_{1,1}, \ldots, x_{1,\ell_{x_1}}, x_{2,1}, \ldots, x_{t,\ell_{x_t}})$, where $Y_{x_j} = \{x_{j,1}, \ldots, x_{j,\ell_j}\}$;
- R' is an ℓ-ary relation, $\ell = \ell_{x_1} + , \ldots, + \ell_{x_t}$, such that $(a_{1,1}, \ldots, a_{1,\ell_{x_1}}, a_{2,1}, \ldots, a_{t,\ell_{x_t}}) \in R'$ if and only if $(h_{i_{x_1}}(a_{1,1}, \ldots, a_{1,\ell_{x_1}}), \ldots, h_{i_{x_t}}(a_{t,1}, \ldots, a_{t,\ell_{x_t}})) \in R$.

Lemma 5.2.1. *Let $\mathcal{X}, \mathcal{Y}$ be restricted CSP-based classes of the χIMP_d and χIMP_{rd}, respectively, $\ell \geq 1$. If $\mathcal{X}$ is substitution reducible to $\mathcal{Y}$ with a substitution collection $\{h_1, \ldots, h_k\}$, and $r \geq \ell_i$ for each $i \in [k]$, then there is a polynomial time reduction from $\mathcal{X}$ to $\mathcal{Y}$.*

Proof. Let $(M, \mathcal{P})$ with $M = (g_1, \ldots, g_\ell)$ be an instance of $\mathcal{X}$. Moreover, suppose polynomials in M have total degree at most d. By the above definition, in polynomial time, we construct an instance $(M', \mathcal{P}')$ with $M' = (g'_1, \ldots, g'_\ell)$ of $\mathcal{Y}$ that satisfies the conditions in the definition. Note that each polynomial h_i in the substitution collection has degree at most $\ell_i |D|$, therefore, each g'_i in M' has a bounded degree. We now prove $(M, \mathcal{P})$ is a **yes** instance if and only if $(M', \mathcal{P}')$ is a **yes** instance.

Recall that $Y = \bigcup_{x \in X} Y_x$ and set $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_m\}$. Let $\mathcal{I}(\mathcal{P}) \subseteq \mathbb{F}[X]$ and $\mathcal{I}(\mathcal{P}') \subseteq \mathbb{F}[Y]$ be the corresponding ideals to $\mathcal{P}$ and $\mathcal{P}'$, respectively. Consider an arbitrary $\mathbf{c} \in \mathbb{F}^\ell$ and set $f \in \mathbb{F}[X]$ to be

$$\begin{aligned} f(x_1, \ldots, x_n) &= \sum_{i=1}^{\ell} c_i g_i(x_1, \ldots, x_n) \\ &= \mathbf{c}M. \end{aligned}$$

Now define the polynomial $f' \in \mathbb{F}[Y]$ to be

$$\begin{aligned} f'(y_1, \ldots, y_m) &= \sum_{i=1}^{\ell} c_i g_i(h_{i_{x_1}}(Y_{x_1}), \ldots, h_{i_{x_n}}(Y_{x_n})) \\ &= \sum_{i=1}^{\ell} c_i g'_i(Y) \\ &= \mathbf{c}M'. \end{aligned}$$

In what follows, we prove that we can construct a satisfying assignment for $\mathcal{P}'$ from a satisfying assignment for $\mathcal{P}$, and vice versa. Consider a satisfying assignment ψ for the instance $\mathcal{P}'$. We find a mapping $\varphi : X \to D$ and show it is a satisfying assignment for $\mathcal{P}$. Define φ as follows. For any $x \in X$ with $Y_x = \{y_{x,1}, \ldots, y_{x,\ell_{i_x}}\}$ let

$$\varphi(x) = h_{i_x}(\psi(y_{x,1}), \ldots, \psi(y_{x,\ell_{i_x}})).$$

Consider a constraint from $\mathcal{P}$, say $\langle \mathbf{s}, R\rangle$ with $\mathbf{s} = (x_1, \ldots, x_t)$. By definition, there exists a constraint $\langle \mathbf{s}', R'\rangle$ such that

- $\mathbf{s}' = (x_{1,1}, \ldots, x_{1,\ell_{x_1}}, x_{2,1}, \ldots, x_{t,\ell_{x_t}})$, where $Y_{x_j} = \{x_{j,1}, \ldots, x_{j,\ell_j}\} \subseteq Y$;
- R' is an ℓ-ary relation, $\ell = \ell_{x_1}+, \ldots, +\ell_{x_t}$, such that $(a_{1,1}, \ldots, a_{1,\ell_{x_1}}, a_{2,1}, \ldots, a_{t,\ell_{x_t}}) \in R'$ if and only if $(h_{i_{x_1}}(a_{1,1}, \ldots, a_{1,\ell_{x_1}}), \ldots, h_{i_{x_t}}(a_{t,1}, \ldots, a_{t,\ell_{x_t}})) \in R$.

Now since ψ is a satisfying assignment of $\mathcal{P}'$ then

$$(\psi(x_{1,1}), \ldots, \psi(x_{1,\ell_{x_1}}), \psi(x_{2,1}), \ldots, \psi(x_{t,\ell_{x_t}})) \in R'$$

if and only if

$$\Big(h_{i_{x_1}}(\psi(x_{1,1}), \ldots, \psi(x_{1,\ell_{x_1}})), \ldots, h_{i_{x_t}}(\psi(x_{t,1}), \ldots, \psi(x_{t,\ell_{x_t}}))\Big) \in R.$$

Hence, since ψ is a satisfying assignment for $\mathcal{P}'$ then φ is a satisfying assignment for $\mathcal{P}$.

Conversely, consider a satisfying assignment φ for the instance $\mathcal{P}$. We find a mapping $\psi : Y \to E$ and show it is a satisfying assignment for $\mathcal{P}'$. Define ψ as follows. Let $x_1, \ldots, x_r$ be all the variables such that $y \in Y_{x_i}$, $i \in [r]$. According to the definition, item 1(b), we must have $Y_{x_1} = \cdots = Y_{x_r} = \{y_1, y_2, \ldots, y_{\ell_j}\}$. Without loss of generality, suppose $y = y_1$. Now, according to item 1(b) of the definition, since φ is a solution of $\mathcal{P}$ there are values $a_1, \ldots, a_{\ell_j} \in E$ such that $\varphi(x_i) = h_{i_{x_i}}(a_1, \ldots, a_{\ell_j})$ for all $i \in [r]$. Hence, in this case we set $\psi(y) = a$.

Now we show ψ is a satisfying assignment for $\mathcal{P}'$. Consider a constraint from $\mathcal{P}'$, let say $\langle \mathbf{s}', R'\rangle$. By item (3) in the definition, there exists a constrain $\langle \mathbf{s}, R\rangle$ with $\mathbf{s} = (x_1, \ldots, x_t)$ in $\mathcal{P}$ that gives rise to $\langle \mathbf{s}', R'\rangle$ such that

- $\mathbf{s}' = (x_{1,1}, \ldots, x_{1,\ell_{x_1}}, x_{2,1}, \ldots, x_{t,\ell_{x_t}})$, where $Y_{x_j} = \{x_{j,1}, \ldots, x_{j,\ell_j}\} \subseteq Y$;
- R' is an ℓ-ary relation, $\ell = \ell_{x_1}+, \ldots, +\ell_{x_t}$, such that $(a_{1,1}, \ldots, a_{1,\ell_{x_1}}, a_{2,1}, \ldots, a_{t,\ell_{x_t}}) \in R'$ if and only if $(h_{i_{x_1}}(a_{1,1}, \ldots, a_{1,\ell_{x_1}}), \ldots, h_{i_{x_t}}(a_{t,1}, \ldots, a_{t,\ell_{x_t}})) \in R$.

Now since φ is a satisfying assignment of $\mathcal{P}$ then

$$\begin{aligned} &(\varphi(x_1), \ldots, \varphi(x_n)) \in R \\ \implies &\Big(h_{i_{x_1}}(\psi(x_{1,1}), \ldots, \psi(x_{1,\ell_{x_1}})), \ldots, h_{i_{x_t}}(\psi(x_{t,1}), \ldots, \psi(x_{t,\ell_{x_t}}))\Big) \in R. \end{aligned}$$

if and only if

$$(\psi(x_{1,1}), \ldots, \psi(x_{1,\ell_{x_1}}), \psi(x_{2,1}), \ldots, \psi(x_{t,\ell_{x_t}})) \in R'$$

Hence, since φ is a satisfying assignment for $\mathcal{P}$ then ψ is a satisfying assignment for $\mathcal{P}'$.

It remains to show $f'(\psi) = 0$ if and only if $f(\varphi) = 0$. This is the case because

$$\begin{aligned} f'(\psi) &= \sum_{i=1}^{\ell} c_i g_i'(\psi) \\ &= \sum_{i=1}^{\ell} c_i g_i \left(h_{i_{x_1}}(\psi(Y_{x_1})), \ldots, h_{i_{x_n}}(\psi(Y_{x_n})) \right) \\ &= \sum_{i=1}^{\ell} c_i g_i(\varphi(x_1), \ldots, \varphi(x_n)) \\ &= f(\varphi) \end{aligned}$$

Therefore we have proved

$$\exists \mathbf{c} \in \mathbb{F}^{\ell} \text{ with } \mathbf{c}M \in \mathcal{I}(\mathcal{P}) \iff \exists \mathbf{c}' \in \mathbb{F}^{\ell} \text{ with } \mathbf{c}'M' \in \mathcal{I}(\mathcal{P}').$$

This finishes the proof. □

Theorem 5.1.5 and the above lemma provide a powerful tool for solving the χIMP. That is, if $\mathcal{X}$ is substitution reducible to $\mathcal{Y}$ and furthermore $\mathcal{Y}$ is such that it admits a polynomial time algorithm to construct a Gröbner Basis, then instances of $\mathcal{X}$ are solvable in polynomial time too. More formally,

Theorem 5.2.2. *Let $\mathcal{X}, \mathcal{Y}$ be restricted CSP-based classes of the χIMP_d and χIMP_{rd}, $r \geq 1$ respectively, such that $\mathcal{X}$ is substitution reducible to $\mathcal{Y}$ with a substitution collection $\{h_1, \ldots, h_k\}$ and $r \geq \ell_i$ for $i \in [k]$. Suppose there exists a polynomial time algorithm that for any instance $(M', \mathcal{P}')$ of $\mathcal{Y}$ constructs a (truncated) Gröbner Basis, then*

1. *there is a polynomial time algorithm that solves every instance $(M, \mathcal{P})$ of $\mathcal{X}$; and*
2. *there exists a polynomial time algorithm that for any instance $(M, \mathcal{P})$ of $\mathcal{X}$ constructs a d-truncated Gröbner Basis for $\mathtt{I}(\mathcal{P})$.*

Proof. Suppose M contains ℓ polynomials $g_1, \ldots, g_\ell$. From instance $(M, \mathcal{P})$ of $\mathcal{X}$ we construct an instance $(M', \mathcal{P}')$ of $\mathcal{Y}$ as explained above. Now by Lemma 5.2.1 these two instances are equivalent.

The objective is to find $\mathbf{c} \in \mathbb{F}^{\ell}$ such that $f = \mathbf{c}M \in \mathcal{I}(\mathcal{P})$, if one exists. After carrying out the construction the coefficients in all the polynomials $g_1', \ldots, g_\ell' \in M'$ are linear combination of elements of $\mathbf{c}$. Hence, we have polynomial $f' = \mathbf{c}'M'$ where each entry of $\mathbf{c}'$ is a linear combination of elements of $\mathbf{c}$. By our assumption, we can construct a Gröbner Basis for

$\mathcal{I}(\mathcal{P}')$ then we can check in polynomial time if such $\mathbf{c}'$ exists. If no such $\mathbf{c}'$ exists then $(M, \mathcal{P})$ is a **no** instance, else we can solve a system of linear equations over the elements of $\mathbf{c}$ and find a solution $\mathbf{c}$.

Finally, for the second part of the theorem since $(M, \mathcal{P})$ is polynomial time solvable, by Theorem 5.1.9, we can construct a d-truncated Gröbner Basis for $\mathcal{I}(\mathcal{P})$ in polynomial time. □

5.2.2 Applications of reduction by substitution

In this section we demonstrate that the notion of reduction by substitution introduced in the previous section is applicable to various cases, in particular the case of CSPs over constraint languages closed under the affine operation of an finite Abelian group. We start off with the case of pp-interpretation.

Lemma 5.2.3. *Let Δ and Γ be multi-sorted constraint languages over finite collection of sets $\mathcal{D} = \{D_t \mid t \in T\}$, $\mathcal{E} = \{E_s \mid s \in S\}$, respectively. Let $\mathcal{X}, \mathcal{Y}$ be classes of CSP-based χIMP with $\mathcal{X}$ defined as $\chi IMP_d(\Delta)$ and $\mathcal{Y}$ defined as $\chi IMP(\Gamma)$. Suppose Γ pp-interprets Δ, then $\mathcal{X}$ is substitution reducible to $\mathcal{Y}$.*

Proof. Let $(M, \mathcal{P})$ be an instance of $\mathcal{X}$. We provide an algorithm that construct an instance $(M', \mathcal{P}')$ of $\mathcal{Y}$ in such a way that it satisfies the conditions for reduction by substitution.

Recall Definition 3.3.4. Define the substitution collection $\{h_s \mid s \in S\}$ to be the set of polynomials where each h_s interpolates the onto mapping $\pi_s : F_s \to E_s$. For every constraint $\langle \mathbf{v}, R \rangle$ in $\mathcal{P}$ with $\mathbf{v} = (x_1, \ldots, x_t)$, $\mathcal{P}'$ contains the constraint $\langle \mathbf{v}', R' \rangle$ with

- $\mathbf{v}' = (x_{1,1}, \ldots, x_{1,\ell_{s_1}}, \ldots, x_{t,1}, \ldots, x_{t,\ell_{s_t}})$, and
- R' is such that

$$\pi^{-1}(R)(x_{1,1}, \ldots, x_{1,\ell_{s_1}}, x_{2,1}, \ldots, x_{2,\ell_{s_2}}, \ldots, x_{t,1}, \ldots, x_{t,\ell_{s_t}}) \qquad \text{is true}$$

if and only if

$$R(h_{s_1}(x_{1,1}, \ldots, x_{1,\ell_{s_1}}), \ldots, h_{s_k}(x_{k,1}, \ldots, x_{k,\ell_{s_t}})) \qquad \text{is true.}$$

Now for each x_i with $\delta_\Delta(x_i) = s$ we have $Y_{x_i} = \{x_{i,1}, \ldots, x_{i,\ell_s}\}$. Note that for every distinct x_i and x_j we have $Y_{x_i} \cap Y_{x_j} = \emptyset$. This satisfies conditions 1(b),(c). Moreover, according to pp-interpretability and the way $(M', \mathcal{P}')$ is constructed condition (3) is also satisfied. □

From the above lemma and Theorem 5.2.2 we obtain the following corollary.

Corollary 5.2.4. *Let Δ and Γ be multi-sorted constraint languages over finite collection of sets $\mathcal{D} = \{D_t \mid t \in T\}$, $\mathcal{E} = \{E_s \mid s \in S\}$, respectively. Suppose Γ pp-interprets Δ and*

there exists a polynomial time algorithm that for any instance $(M', \mathcal{P}')$ *of* $\chi IMP_{O(d)}(\Gamma)$ *constructs a (truncated) Gröbner Basis, then*

1. *there is a polynomial time algorithm that solves every instance* $(M, \mathcal{P})$ *of* $\chi IMP_d(\Delta)$*; and*
2. *there exists a polynomial time algorithm that for any instance* $(M, \mathcal{P})$ *of* $\chi IMP_d(\Delta)$ *constructs a d-truncated Gröbner Basis for* $\mathtt{I}(\mathcal{P})$*.*

Reduction by substitution for languages over Abelian groups

In this section we prove that our reductions for constraint languages over finite Abelian groups is an example of reduction by substitution. Lemma 5.2.3 states that reductions under pp-interpretability can be seen as reduction by substitution. This means the part of our reduction where we transform an instance of CSP over an Abelian group to an instance of CSP over $\mathbb{Z}_{p_1^{m_1}}, \dots, \mathbb{Z}_{p_s^{m_s}}$ can be seen as a reduction by substitution. We will show that the reduction to roots of unities is also a reduction by substitution.

Let $(M, \mathcal{P})$ be such that $M = (g_1, \dots, g_\ell)$ is a vector of polynomials of length ℓ where each $g_i \in M$ is from $\mathbb{C}[x_{1,1}, \dots, x_{1,k_1}, \dots, x_{s,1}, \dots, x_{s,k_s}]$ and $\mathcal{P}$ is an instance of $\mathrm{CSP}(\Gamma)$. Here Γ is a constraint language invariant under the affine operation of $\mathbb{Z}_{p_1^{m_1}}, \dots, \mathbb{Z}_{p_s^{m_s}}$. Moreover, $\mathcal{P}$ can be represented as a collection of systems of linear equations $\mathcal{L}_1, \dots, \mathcal{L}_s$ where

1. each $\mathcal{L}_i$ is a system of linear equations over $\mathbb{Z}_{p_i^{m_i}}$ with variables $X(\mathcal{L}_i) \cup Y(\mathcal{Y}_i)$, $X(\mathcal{L}_i) = \{x_{i,1}, \dots, x_{i,k_i}\}$, $Y(\mathcal{L}_i) = \{y_{i,1}, \dots, y_{i,r_i}\}$;
2. each $\mathcal{L}_i$ is of the following form

$$(\mathbf{1}_{k_i \times k_i} \quad M_i)(x_{i,1}, \dots, x_{i,k_i}, y_{i,1}, \dots, y_{i,r_i}, 1)^T = \mathbf{0};$$

3. $X(\mathcal{L}_i) \cap X(\mathcal{L}_j) = \emptyset$, $Y(\mathcal{L}_i) \cap Y(\mathcal{L}_j) = \emptyset$, for all $1 \leq i, j \leq s$ and $i \neq j$;
4. an assignment φ to variables from X is a solution of $\mathcal{P}$ if and only if for every $i \in [s]$ there are values of variables from $Y(\mathcal{L}_i)$ that together with $\varphi_{|X(\mathcal{L}_i)}$ satisfy $\mathcal{L}_i$.

Now for each $i \in [s]$ let h_i be a polynomial that interpolates the mapping

$$(0, \omega_i^0), (1, \omega_i), \dots, (p_i^{m_i} - 1, \omega_i^{(p_i^{m_i}-1)})$$

where ω_i is a primitive $p_i^{m_i}$-th root of unity. The substitution collection consists of all h_i, $i \in [s]$. For every variable $x_{i,j}, y_{i,j}$ we set $Y_{i,j} = \{x_{i,j}\}$ and $Y'_{i,j} = \{y_{i,j}\}$ satisfying condition 1(b). Then, for each variable $x_{i,j}, y_{i,j}$ we choose $h_{i_{x_{i,j}}} = h_i$, $h_{i_{y_{i,j}}} = h_i$. Thus, condition 1(c) is satisfied.

Now for every constraint in $\mathcal{P}$ which is of the form

$$x_{i,t} + \alpha_{t,1}\ y_{i,1} + \cdots + \alpha_{t,r_i}\ y_{i,r_i} + \alpha_t = 0 \pmod{p_i^{m_i}}$$

we add the following constraint in $\mathcal{P}'$

$$x_{i,t} - \omega_i^{\alpha_t} \cdot \left(y_{i,1}^{\alpha_{t,1}} \cdot \ldots \cdot y_{i,r_i}^{\alpha_{t,r_i}} \right) = 0.$$

Such construction of $\mathcal{P}'$ guarantees that conditions in (3) hold. Now it is immediate that our transformation of the problem to an equivalent problem over roots of unities is indeed a reduction by substitution. This together with the fact that there exists an algorithm to construct a Gröbner Basis for the equivalent problem over roots of unities, see Lemma 4.4.12, give us the following theorem.

Theorem 5.2.5. *Let $\mathbb{A}$ be an Abelian group. Then $IMP_d(\Delta)$ is polynomial time solvable for any finite constraint language Δ which is invariant under the affine operation of $\mathbb{A}$.*

Moreover, given an instance $(f_0, \mathcal{P})$ of $IMP_d(\Delta)$ a (d-truncated) Gröbner Basis of $\mathtt{I}(\mathcal{P})$ (with respect to a grlex order) can be constructed in polynomial time.

Proof. The discussion above tells us that $\chi\mathrm{IMP}_d(\Delta)$ is substitution reducible to a class of χIMP, say $\mathcal{Y}$, where for every instance $(M', \mathcal{P}')$ of $\mathcal{Y}$ we can construct a Gröbner Basis. Hence, by Theorem 5.2.2, every instance $(M, \mathcal{P})$ of $\chi\mathrm{IMP}_d(\Delta)$ is polynomial time solvable. Moreover, by item 2 of Theorem 5.2.2, we can construct a d-truncated Gröbner Basis for $\mathcal{I}(\mathcal{P})$ thus $\mathrm{IMP}_d(\Delta)$ is polynomial time solvable. □

5.3 SOS proofs: bit complexity and automatability

The focus of this section is on designing efficient algorithms to find proofs of nonnegativity of polynomials over (semi)algebraic sets. Sum-of-squares certificates of nonnegativity is a popular and powerful framework to provide a proof that a polynomial is nonnegative. In this section we present a very light introduction to SOS proofs of nonnegativity and lay down some notation and background. The main appeal of this proof system is that it can be transformed into an SDP feasibility problem and hopefully be solved efficiently using methods such as the Ellipsoid method. However, we discuss a recently discovered issue with the bit complexity of the coefficients appearing in polynomials in an SOS proof which could cause algorithms such as the Ellipsoid method to run in exponential time. This issue affects the automatability of SOS proofs. Our objective is to characterize algebraic sets i.e., constraint languages, for which SOS proofs are automatable. We first observe that the existence of a degree d SOS proof implies existence of a degree d SOS proof on quotient rings. Hence, we only need to look at SOS on quotient rings. Using this we then prove that for radical ideals and ideals arising from CSP instances we can guarantee SOS proofs with

low bit complexity, provided a low degree one exists. This leads us to the third part of this section where we show that degree bounds on SOS proofs can be relaxed preserving automatability provided the IMP part is polynomial time solvable. We believe this could potentially lead to more expressive SOS proofs. Throughout this section we work with $\mathbb{F} = \mathbb{R}$.

5.3.1 SOS proofs on quotient ring

In this subsection we provide a light introduction to SOS proofs and discuss the issue with the bit complexity of the coefficients of polynomials appearing in a proof.

It is known that, in general, nonnegativity of a polynomial is not equivalent to having a representation as a sum-of-squares polynomials [112]. The most famous example of these type of polynomials is the *Motzkin polynomial* [174]. The Motzkin polynomial $r(x, y) = 1 + x^4y^2 + x^2y^4 - 3x^2y^2$ is nonnegative while it cannot be represented as a sum-of-squares [193]. However, the situation is different when one is concerned with proofs of nonnegativity over (semi)algebraic sets. Let S be the following semialgebraic set.

$$S = \{\mathbf{x} \in \mathbb{R}^n \mid p_1(\mathbf{x}) = 0, \ldots, p_m(\mathbf{x}) = 0, q_1(\mathbf{x}) \geq 0, \ldots, q_\ell(\mathbf{x}) \geq 0\} \tag{5.2}$$

where the ideal $\mathtt{I} = \langle p_1, \ldots, p_m \rangle$ is zero-dimensional and radical. In this case we have the following lemma. Note that radicality is crucial in the next lemma.

Lemma 5.3.1 ([183])**.** *Every nonnegative polynomial on S is of the form $s_0^2 + \sum_{j=1}^{\ell} s_j^2 q_j + g$ with $g \in \mathtt{I}$.*

This leads to a uniform and powerful tool for low degree polynomial optimization. Formally,

`Minimize`	$r(\mathbf{x})$
`Subject to:`	$\mathbf{x} \in S = \{\mathbf{x} \in \mathbb{R}^n \mid p_1(\mathbf{x}) = 0, \ldots, p_m(\mathbf{x}) = 0, q_1(\mathbf{x}) \geq 0, \ldots, q_\ell(\mathbf{x}) \geq 0\}$

An SOS proof of a lower bound $r(\mathbf{x}) \geq \theta$ is given by a polynomial identity of the form

$$r(\mathbf{x}) - \theta = \sum_{i=1}^{t_0} h_i^2(\mathbf{x}) + \sum_{k=1}^{\ell} (\sum_{j=1}^{t_k} s_j^2(\mathbf{x})) q_k(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i(\mathbf{x}) p_i(\mathbf{x}). \tag{5.3}$$

The degree of an SOS certificate is often defined to be the maximum degree of the polynomials involved in the proofs i.e., $\max\{deg(h_i^2), deg(s_j^2 q_k), deg(\lambda_i p_i)\}$.

A common misconception was that if a degree d SOS proof of nonnegativity exists then the corresponding SDP feasibility is solvable by the Ellipsoid method in time $O(n^d)$. This is sometimes referred to as the SOS proof system being *automatabe*. Unfortunately, this is not true in general [178]. Technically, the Ellipsoid method is guaranteed to work in time

$poly(n^d)$ if the SDP's feasible region (should it exist) intersects a ball of radius $2^{poly(n^d)}$ [93]. Thus, it is not sufficient for an SOS proof to exist, we also need one to exist in which all the SOS polynomials can be written down with $poly(n^d)$ bits i.e., coefficients involved in an SOS certificate have *low bit complexity*.

Let $\mathcal{I} \subseteq \mathbb{R}[X]$ be a zero-dimensional ideal. Fix a monomial order, here we consider grlex order. Suppose we know $\mathcal{B} = \{\mathbf{x}^\alpha \mid \mathbf{x}^\alpha \notin \langle LT(\mathcal{I})\rangle\}$ so that $\mathbb{R}[X] = \mathtt{Span}_{\mathbb{R}}(\mathcal{B}) \oplus \mathcal{I}$. If $r = \sum s_i^2 + q$ with $s_i \in \mathbb{R}[X]$, $q \in \mathcal{I}$, write $s_i = u_i + v_i$ with $u_i \in \mathtt{Span}_{\mathbb{R}}(\mathcal{B})$ and $v_i \in \mathcal{I}$. Hence, $r = \sum u_i^2 + g$ where $g = q + \sum(v_i^2 + 2u_iv_i)$ is in $\mathcal{I}$. This yields the following well known lemma.

Lemma 5.3.2 ([145])**.** *Let* $\mathtt{I} = \langle p_1 \ldots, p_m\rangle$ *be an ideal in* $\mathbb{R}[X]$ *and* $\mathcal{Q} = \{q_1, \ldots, q_\ell\}$*. Define* $S = \{\mathbf{x} \in \mathbb{R}^n \mid p_1(\mathbf{x}) = 0, \ldots, p_m(\mathbf{x}) = 0, q_1(\mathbf{x}) \geq 0, \ldots, q_\ell(\mathbf{x}) \geq 0\}$.

Suppose $r(\mathbf{x})$ *has a degree* d SOS *proof of nonnegativity on* S*. Then, it has a degree* d SOS *proof of nonnegativity*

$$r(\mathbf{x}) = \sum_{i=1}^{t_0} h_i^2(\mathbf{x}) + \sum_{k=1}^{\ell}(\sum_{j=1}^{t_k} s_j^2(\mathbf{x}))q_i(\mathbf{x}) + g$$

such that all h_i, s_j *are in* $\mathtt{Span}(\mathbf{x}^\alpha \mid |\alpha| \leq \frac{d}{2}, \mathbf{x}^\alpha \notin \langle \mathrm{LT}(\mathtt{I})\rangle)$ *and* $g \in \mathtt{I}$.

This tells us the existence of a degree d SOS proof modulo the ideal implies existence of an SOS proof modulo the ideal where all the SOS polynomials are linear combination of the monomials from $\mathcal{B}$.

5.3.2 Automatability on quotient ring

Here we wish to provide conditions that guarantee low bit complexity for polynomials appearing in an SOS proof. This in turn guarantees automatability i.e., if there exists a degree d SOS proof of nonnegativity then the Ellipsoid method is guaranteed to find such a proof. The first systematic approach to this problem is due to Raghavendra and Weitz [192]. Let $\mathcal{P} = \{p_1 \ldots, p_m\}$ and $\mathcal{Q} = \{q_1, \ldots, q_\ell\}$ and define $S \subseteq \{\mathbf{a} \in \mathbb{R}^n \mid \forall p \in \mathcal{P} : p(\mathbf{a}) = 0\}$. We write $\mathbf{u}_d$ for the vector of polynomials whose entries are the elements of the usual monomial basis of $\mathbb{R}[X]_d$. Similarly, we use $\mathbf{u}_d(\mathbf{a})$ for the vector of reals whose entries are the entries of $\mathbf{u}_d$ evaluated at $\mathbf{a}$. Let $\mathcal{U}$ denote the uniform distribution over S and define the moment matrix as

$$M_d = \mathbb{E}_{\mathbf{a}\sim\mathcal{U}}[\mathbf{u}_d(\mathbf{a})\mathbf{u}_d(\mathbf{a})^T]. \tag{5.4}$$

In some sense, the main condition that Raghavendra and Weitz provided is that *every* polynomial of degree d has a proof of membership in the ideal from $p_1, \ldots, p_m$ of degree at most kd. In this case we say $p_1, \ldots, p_m$ are k-effective or $\mathcal{P}$ is k-effective. As an example, if $p_1, \ldots, p_m$ form a Gröbner Basis with respect to grlex then they are 1-effective. If furthermore

the ideal $\mathcal{I}(\mathcal{P}) = \langle p_1, \ldots, p_m \rangle$ is radical, then in this case we say that $(\mathcal{P}, \mathcal{Q})$ is kd-complete on S up to degree d. Ideally, we want k to be small e.g., $k = O(d)$. This is a very strong condition on the structure of the ideal $\mathcal{I}$ while the other conditions in [192] are considered to be mild. The conditions are as follows,

1. $(\mathcal{P}, \mathcal{Q})$ is kd-complete on S up to degree d,
2. S is ε-robust for Q. This means $\forall q \in \mathcal{Q}, \forall \mathbf{a} \in S : q(\mathbf{a}) > \varepsilon$,
3. S is δ-spectrally rich for $(\mathcal{P}, \mathcal{Q})$ up to degree d. This means every nonzero eigenvalue of M_d is at least δ[1].

Provided the above conditions are met, they show that any polynomial r that is nonnegative on S and admits a degree d SOS of form (5.3), is guaranteed to have a degree kd SOS certificate of form (5.3) where all coefficients have polynomial bit complexity. While [218] developed a proof strategy for proving that a set of polynomials are k-effective, unfortunately, as marked in [218]: " this strategy is by no means universally applicable, and it had to be applied on a case-by-case basis". It is not obvious how to verify effectiveness even if we are given a Gröbner Basis of the ideal $\langle p_1, \ldots, p_m \rangle$ [2].

The situation is different if we are interested in SOS proofs of nonnegativity on S modulo the ideal, rather than an SOS proof modulo $\{p_1, \ldots, p_m\}$. Recall that as long as the ideal is radical, by Lemma 5.3.1, every nonnegative polynomial on S is of the form $s_0^2 + \sum_{j=1}^{\ell} s_j^2 q_j + g$ with $g \in \mathcal{I}(\mathcal{P})$. We say a polynomial $r(\mathbf{x})$ is SOS modulo S (or modulo $\mathcal{I}(\mathcal{P})$, when the ideal is radical) if there are h_i, s_j, and $g \in \mathcal{I}(\mathcal{P})$ such that

$$r(\mathbf{x}) = \sum_{i=1}^{t_0} h_i^2(\mathbf{x}) + \sum_{k=1}^{\ell} (\sum_{j=1}^{t_k} s_j^2(\mathbf{x})) q_k(\mathbf{x}) + g. \tag{5.5}$$

Turning our attention to SOS proofs of form (5.5) modulo CSP-based ideals, we note these ideals are radical and moreover we realize that the k-effectiveness condition is avoidable and somewhat irrelevant. In addition, the discrete nature of the varieties of CSP-based ideals implies the δ-spectrally richness. In fact, one can prove similar to Lemma 7 of [192] that $\frac{1}{\delta} = 2^{poly(n^{|D|})}$ where D is the domain of the constraint language at hand. We follow a similar approach to [192] proving the following theorem.

Theorem 5.3.3. *Let $\mathcal{P}$ be an instance of CSP(Γ) with domain D, and $\mathtt{I}(\mathcal{P}) = \langle p_1 \ldots, p_m \rangle$ be the corresponding ideal to $\mathcal{P}$. Define $S = \boldsymbol{V}(\mathtt{I}(\mathcal{P})) = \{\mathbf{a} \in \mathbb{R}^n \mid \forall p \in \mathcal{P} : p(\mathbf{a}) = 0\}$ and suppose S is ε-robust for $\mathcal{Q} = \{q_1, \ldots, q_\ell\}$.*

[1] A zero eigenvector of M corresponds to a polynomial which is zero on S.

[2] Obviously, if $p_1, \ldots, p_m$ form a Gröbner Basis then they are 1-effective i.e. every degree d polynomial in the ideal has a degree d derivation from $p_1, \ldots, p_m$.

Let $r(\mathbf{x})$ be a polynomial nonnegative on S, and assume r has a degree d SOS proof of nonnegativity

$$r(\mathbf{x}) = \sum_{i=1}^{t_0} h_i^2(\mathbf{x}) + \sum_{k=1}^{\ell}(\sum_{j=1}^{t_k} s_j^2(\mathbf{x}))q_k(\mathbf{x}) + g \qquad (g \in \mathcal{I}(\mathcal{P}))$$

Then r has a degree d SOS proof of nonnegativity modulo $\mathtt{I}(\mathcal{P})$ such that the coefficients of every polynomial appearing in the proof are bounded by $2^{poly(n^d, n^{|D|}, \log\frac{1}{\varepsilon})}$. In particular, if $\mathcal{Q} = \emptyset$ then every coefficient can be written down with only $poly(n^d, n^{|D|})$ bits.

Proof. Let $\mathcal{I} = \mathcal{I}(\mathcal{P})$ and $\mathcal{B}_{\frac{d}{2}} = \{\mathbf{x}^\alpha \mid |\alpha| \leq \frac{d}{2}, \mathbf{x}^\alpha \notin \langle \mathrm{LT}(I)\rangle\}$. Let $\mathbf{v}$ be the vector whose entries are the elements of $\mathcal{B}_d$, so any polynomial in $\mathtt{Span}_{\mathbb{R}}(\mathcal{B}_{\frac{d}{2}})$ can be expressed as $c^T\mathbf{v}$, where c is a vector of reals. By Lemma 5.3.2, there is a proof of nonnegativity as follows.

$$\begin{aligned} r(\mathbf{x}) &= \sum_{i=1}^{t_0}(c_i^T\mathbf{v})^2(\mathbf{x}) + \sum_{k=1}^{\ell}\left(\sum_{j=1}^{t_k}(d_{kj}^T\mathbf{v})^2(\mathbf{x})\right)q_k(\mathbf{x}) + g \qquad (g \in \mathcal{I}) \\ &= \langle C, \mathbf{v}\mathbf{v}^T\rangle + \sum_{k=1}^{\ell}\langle D_k, \mathbf{v}\mathbf{v}^T\rangle q_k + g \end{aligned}$$

for PSD matrices $C, D_1, \ldots, D_\ell$. Next, we average this polynomial identity over all the points $\mathbf{a} \in S$:

$$\mathbb{E}_{\mathbf{a}\in S}[r(\mathbf{a})] = \langle C, \mathbb{E}_{\mathbf{a}\in S}[\mathbf{v}(\mathbf{a})\mathbf{v}(\mathbf{a})^T]\rangle + \sum_{k=1}^{\ell}\langle D_k, \mathbb{E}_{\mathbf{a}\in S}[\mathbf{v}(\mathbf{a})\mathbf{v}(\mathbf{a})^T q_k(\mathbf{a})]\rangle + 0$$

Let $||r||$ denote the maximum absolute value of coefficients of r and $||S|| = \max_{\mathbf{a}\in S}||\mathbf{a}||_\infty$. Then the LHS is at most $poly(||r||, ||S||) \leq 2^{poly(n^d)}$. The RHS is a sum of positive numbers, since the inner products are over pairs of PSD matrices. Thus the LHS is an upper bound on each term of the RHS. We would like to say that the LHS also provides an upper bound on the entries of matrices C, D_k or an upper bound on the trace of these matrices. Recall that the trace of a matrix is sum of its eigenvalues. Hence, we prove that the averaged matrix $M' = \mathbb{E}_{\mathbf{a}\in S}[\mathbf{v}(\mathbf{a})\mathbf{v}(\mathbf{a})^T]$ has no zero eigenvector so none of the eigenvalues of matrices C and D_k's are being dismissed by a zero eigenvector of M'.

We now claim that the averaged matrix $M' = \mathbb{E}_{\mathbf{a}\in S}[\mathbf{v}(\mathbf{a})\mathbf{v}(\mathbf{a})^T]$ has no zero eigenvector. Any zero eigenvector c of M' can be associated with a polynomial $c^T\mathbf{v}$. Since $c^TM'c = \mathbb{E}_{\mathbf{a}\in S}[(c^T\mathbf{v}(\mathbf{a}))^2]$ and $c^TM'c = 0$, we must have $c^T\mathbf{v}(\mathbf{a}) = 0$ for each $\mathbf{a} \in S$. Therefore, as $\mathcal{I}$ is radical, by the Strong Nullstellensatz 2.1.9, $c^T\mathbf{v}$ must be in the ideal $\mathcal{I}$. This is a contradiction to $\mathbf{v}$ being a vector of monomials from $\mathcal{B}_{\frac{d}{2}}$.

We proceed by assuming the averaged matrix $M' = \mathbb{E}_{\mathbf{a}\in S}[\mathbf{v}(\mathbf{a})\mathbf{v}(\mathbf{a})^T]$ has no zero eigenvector. Therefore, none of the eigenvalues of matrix C are being dismissed by a zero eigenvector of M'. The same is true for all D_k. Furthermore, similar to M (5.4), every nonzero

eigenvalue of M' is at least δ, so

$$\langle C, \mathbb{E}_{\mathbf{a}\in S}[\mathbf{v}(\mathbf{a})\mathbf{v}(\mathbf{a})^T]\rangle \geq \delta \cdot Tr(C).$$

Also, $q_k(\mathbf{a}) > \varepsilon$ for each k and $\mathbf{a}$. Thus,

$$\begin{aligned}\langle D_k, \mathbb{E}_{\mathbf{a}\in S}[\mathbf{v}(\mathbf{a})\mathbf{v}(\mathbf{a})^T q_k(\mathbf{a})]\rangle &\geq \varepsilon \cdot \langle D_k, \mathbb{E}_{\mathbf{a}\in S}[\mathbf{v}(\mathbf{a})\mathbf{v}(\mathbf{a})^T]\rangle \\ &\geq \varepsilon \cdot \delta \cdot Tr(D_k)\end{aligned}$$

Thus, after averaging we have

$$\delta \cdot Tr(C) + \varepsilon \cdot \delta \cdot \sum_{k=1}^{\ell} Tr(D_k) \leq poly(||r||, ||S||) \leq 2^{poly(n^d)}$$

Every entry of a PSD matrix is bounded by the trace, so C and each D_k have entries bounded by $poly(||r||, ||S||, \frac{1}{\varepsilon}, \frac{1}{\delta})$. Noting that $poly(||r||, ||S||) \leq 2^{poly(n^d)}$ and $\frac{1}{\delta} = 2^{poly(n^{|D|})}$ yields the desire bound on the entries of C and each D_k. It remains to give an upper bound on the coefficients appearing in g. Consider the system of linear equations induced by

$$r(\mathbf{x}) - \langle C, \mathbf{v}\mathbf{v}^T\rangle + \sum_{k=1}^{\ell} \langle D_k, \mathbf{v}\mathbf{v}^T\rangle q_k = g$$

where we take the coefficients appearing in g as variables. Note that this system is feasible. By Lemma 5.3.2, g has degree at most d and therefore there are at most $O(n^d)$ variables and the coefficients on the LHS are bounded by $poly(||r||, ||S||, \frac{1}{\delta}, \frac{1}{\varepsilon})$. Let $||\mathcal{P}, \mathcal{Q}||$ denote the maximum absolute value of coefficients appearing in polynomials $p_1, \ldots, p_m$ and $q_1, \ldots, q_\ell$. By Cramer's rule, the coefficients appearing in g can be taken to be bounded by $poly(||\{\mathcal{P}, \mathcal{Q}\}||, ||r||, ||S||, \frac{1}{\delta}, \frac{1}{\varepsilon}, n!)$. Noting that $poly(||\{\mathcal{P}, \mathcal{Q}\}||, ||r||, ||S||) \leq 2^{poly(n^d)}$, as they are part of the input, and $\frac{1}{\delta} = 2^{poly(n^{|D|})}$ gives that this bound is at most $2^{poly(n^d, n^{|D|}, \log \frac{1}{\varepsilon})}$ as desired. □

The required conditions in Theorem 5.3.3 are quite mild and covers almost all algebraic sets arising from CSP-based ideals, particularly when $\mathcal{Q} = \emptyset$. We remark that, Theorem 5.3.3 is applicable to radical ideals as long as the δ-spectrally richness is met.

5.3.3 Automatability and CSP-based ideals

Note that in [192] if the conditions for low bit complexity are met then one can formulate (5.3) as an SDP which is guaranteed to be solved by the Ellipsoid method in time $poly(n^d)$. However, Theorem 5.3.3 does not tell us how to decide in polynomial time if a polynomial r is SOS modulo S. One approach suggested by Raghavendra and Weitz and Mastrolilli, is to express the polynomial g in (5.5) in terms of "nice" generating sets of polynomials such

as Gröbner Bases and then use an SDP formulation, see Problem 1.2.1. Here, we suggest a different perspective through the lens of χIMP to decide the existence of an SOS proof modulo S. The following theorem puts forward the idea that the χIMP is the main player in SOS proofs automatability and allow us to use a much larger tool box than the usual Gröbner Basis.

Theorem 5.3.4. *Let Γ be a constraint language such that $\chi IMP_d(\Gamma)$ is polynomial time solvable. Let $\mathcal{P}$ be an instance of $CSP(\Gamma)$ and $\mathtt{I}(\mathcal{P}) = \langle p_1 \dots, p_m \rangle$ be the corresponding ideal to $\mathcal{P}$. Assume that $\forall q_i, \forall \mathbf{a} \in S$ we have $q_i(\mathbf{a}) > \varepsilon$.*

Then for a polynomial r, the existence of an SOS proof of form (5.5) with $\max\{deg(h_i^2), deg(s_j^2 q_k)\} \leq d$ is polynomial time decidable. Furthermore, such a proof can be found in polynomial time, if one exists.

Proof. By Theorem 5.3.3 if an SOS proof exists then there is an SOS proof such that its coefficients can be written down with at most $poly(n^d, n^{|D|}, \log \frac{1}{\epsilon})$ bits. Moreover, by Theorem 5.1.9, we can compute a d-truncated Gröbner Basis for $\mathcal{I}(\mathcal{P})$ in time $O(n^d)$. Then using the d-truncated Gröbner Basis, a proof of form (5.5) can be formulated as an SDP. Finally, the Ellipsoid method is guaranteed to find a proof in time $poly(n^d)$, if one exists. □

The above results cover a wide range of problems for which a low bit complexity guarantee and automatability of SOS proofs was not known. Here we give an example of such problems, the H-Coloring problem.

Example 5.3.5. Fix a (di)graph H with $V(H) = \{0, 1, \dots, d\}$. For any (di)graph G with vertex set $V(G) = \{x_1, \dots, x_n\}$ we are interested in homomorphisms from G to H. The set of all homomorphisms from G to H can be captured by the following polynomial system. The first set of polynomials are the domain polynomials and ensures each vertex of G is assigned a label from $V(H)$. The second set of polynomials are the edge constraints where they ensure that each edge $x_i x_j$ of G is mapped to an edge $\alpha\beta \in E(H)$.

$$\begin{aligned}\mathcal{P} =& \Big\{ \prod_{\alpha \in V(H)} (x_i - \alpha) \mid x_i \in V(G) \Big\} \\ & \cup \Big\{ \prod_{\alpha\beta \in E(H)} (1 - \prod_{\substack{\lambda \in V(H), \\ \lambda \neq \alpha}} \frac{\lambda - x_i}{\lambda - \alpha} \prod_{\substack{\lambda \in V(H), \\ \lambda \neq \beta}} \frac{\lambda - x_j}{\lambda - \beta}) \mid x_i x_j \in E(G) \Big\}\end{aligned}$$

Here S is the set of all homomorphisms from G to H. This setting is very general and many important optimization problems are captured by this specification: k Coloring where H is a clique of size k, Vertex Cover where $H = (V = \{0, 1\}, E = \{(0, 1), (1, 0), (1, 1)\})$. For both of these examples it is known that $\mathcal{P}$ is in fact a Gröbner Basis. Our results imply that if there exists a degree d SOS proof of nonnegativity on S then there exists one of degree d with low bit complexity. Furthermore, our result implies that when H is closed

under a semilattice polymorphism then a truncated Gröbner Basis can be constructed in polynomial time. This includes many graph classes such as bi-arc digraphs, interval graphs, signed interval digraphs, threshold tolerance graphs, etc. [107].

5.4 Theta bodies for combinatorial ideals

One of the core problems in optimization is to understand the $\texttt{conv}(S)$ or a relaxation of $\texttt{conv}(S)$, where S the set of feasible solutions to a given problem and $\texttt{conv}(S)$ denotes the convex hull of S. For the CSPs, an instance $\mathcal{P}$ of CSP(Γ) can be associated with an ideal $\mathcal{I}(\mathcal{P}) = \langle f_1, \ldots, f_s \rangle$ and the $\mathsf{Sol}(\mathcal{P}) = \mathbf{V}(\mathcal{I}(\mathcal{P}))$ is a *finite* subset of $\mathbb{R}^n$. Hence, in this setting we are interested in computing $\texttt{conv}(\mathbf{V}(\mathcal{I}(\mathcal{P})))$. This is cut out by the inequalities $f(\mathbf{x}) \geq 0$ as f runs over all linear polynomials that are nonnegative over $\mathbf{V}(\mathcal{I}(\mathcal{P}))$. Thus, a natural relaxation for $\texttt{conv}(\mathbf{V}(\mathcal{I}(\mathcal{P})))$ is [3]

$$\{\mathbf{x} \in \mathbb{R}^n \mid f(\mathbf{x}) \geq 0 \text{ for all } f \text{ linear and } \mathsf{SOS} \bmod \mathcal{I}\} \tag{5.6}$$

Here we wish to provide a systematic approach using the algebraic property of the solution sets and our results on the IMP and χIMP to compute/approximate $\texttt{conv}(\mathsf{Sol}(\mathcal{P}))$. We follow the notion of *theta bodies* introduced and advanced by Gouveia, Parrilo and Thomas [90]. Throughout this section we work with $\mathbb{F} = \mathbb{R}$. Some definitions are in order.

Definition 5.4.1. *Let* $\texttt{I}$ *be an ideal in* $\mathbb{R}[X]$.

1. *A polynomial* $f \in \mathbb{R}[X]$ *is called* k-SOS mod $\mathcal{I}$ *if there exist* $s_1, \ldots, s_t \in \mathbb{R}[X]_k$ *such that* $f - \sum_{i=1}^{t} s_i^2 \in \texttt{I}$.
2. *The ideal* $\texttt{I}$ *is called* k-SOS *if* every *nonnegative polynomial on* $\boldsymbol{V}(\texttt{I})$ *is* k-SOS *mod* $\texttt{I}$. *If* every degree d *nonnegative polynomial on* $\boldsymbol{V}(\texttt{I})$ *is* k-SOS *mod* $\texttt{I}$ *we say* $\texttt{I}$ *is* (d,k)-SOS.

Lovász [155] asked the following question: Which ideals in $\mathbb{R}[X]$ are $(1,1)$-SOS? How about $(1,k)$-SOS?. We propose a restricted version of this question for vanishing ideals of constraint languages. Formally, we say a language Γ is (d,k)-SOS if for every instance $\mathcal{P}$ of CSP(Γ) the corresponding ideal $\mathcal{I}(\mathcal{P})$ is (d,k)-SOS. An analogue of Lovász's question for constraint languages is,

Problem 5.4.2. *Which languages are* $(1,1)$-SOS*? How about* $(1,k)$-SOS*?*

Gouveia, Parrilo and Thomas [90] quite elegantly present an equivalent geometric notion to an ideal being $(1,k)$-SOS.

[3] Note that, in general, for an ideal $\mathcal{I} \in \mathbb{R}[X]$ the convex hull of $\mathbf{V}(\mathcal{I})$ may not be closed. Hence, (5.6) is a relaxation for the closure of $\texttt{conv}(\mathbf{V}(\mathcal{I}))$, denoted by $cl(\texttt{conv}(\mathbf{V}(\mathcal{I})))$. Here, closure of the convex hull of a set C means the intersection of the closed halfspaces containing C.

Definition 5.4.3. *For a positive integer k, the k-th* theta body *of an ideal $\mathtt{I} \in \mathbb{R}[X]$ is*

$$TH_k(\mathtt{I}) := \{\mathbf{x} \in \mathbb{R}^n \mid f(\mathbf{x}) \geq 0 \text{ for every linear } f \text{ that is } k\text{-SOS mod } \mathtt{I}\}. \tag{5.7}$$

Observe that, by definition, $TH_1(\mathcal{I}) \supseteq TH_2(\mathcal{I}) \supseteq \cdots \supseteq \mathtt{conv}(\mathbf{V}(\mathcal{I}))$. We say a *combinatorial* ideal is TH_k*-exact* if $TH_k(\mathcal{I}) = \mathtt{conv}(\mathbf{V}(\mathcal{I}))$[4]. Zero-dimensional ideals are TH_k-exact for some finite k [145]. We say a language Γ is TH_k-exact if for any instance $\mathcal{P}$ of CSP(Γ) the ideal $\mathcal{I}(\mathcal{P})$ is TH_k-exact. An intriguing question is characterizing TH_k-exact languages, for constant k.

Problem 5.4.4. *Which languages are TH_k-exact, for some constant k?*

Gouveia, Parrilo and Thomas [90] proved that a radical ideal is TH_k-exact if and only if it is $(1, k)$-SOS. They have also provided an answer to Problem 5.4.2 characterizing finite sets $S \subset \mathbb{R}^n$ that are $(1, 1)$-SOS (see Theorem 4.2 in [90]).

Although the results of [90] provide a characterization of TH_k-exact ideal, they do little to the computational side of theta bodies. Theta bodies provide a set of relaxations of solution sets and helps to formulate many combinatorial problems as optimizing a (linear) polynomial over theta bodies which could lead to a better understanding of approximability of many combinatorial problems in a unified way. In order to use theta bodies in this way, we need a method of efficiently 'construct' a theta body. One way to describe the k-th theta body of $\mathcal{I}$ is in terms of the so called *combinatorial moment matrices* which are matrices indexed by a basis of $\mathcal{B}_k$ [145]. Let us elaborate on this via an example, see [145, 90] for further details. First we explain the construction of combinatorial moment matrices and then discuss how we can use them to describe theta bodies for the Clique problem. We may assume $\mathcal{B} = \{\mathbf{x}^\alpha \mid \mathbf{x}^\alpha \notin \langle \mathrm{LT}(\mathcal{I})\rangle\}$ and $\mathcal{B}_k = \{\mathbf{x}^\alpha \mid |\alpha| \leq k, \mathbf{x}^\alpha \notin \langle \mathrm{LT}(\mathcal{I})\rangle\}$ are vectors with their elements listed in increasing grlex order. Hence, any polynomial modulo $\mathcal{I}$ is $\mathbf{c} \cdot \mathcal{B}$ with $\mathbf{c} \in \mathbb{R}^{|\mathcal{B}|}$. Now, let $\mathbf{y} = (y_1, \ldots, y_m) \in \mathbb{R}^{|\mathcal{B}_{2k}|}$ and define $M_{\mathcal{B}_k}(\mathbf{y})$ to be the matrix indexed by $\mathcal{B}_k$ whose $(\mathbf{x}^{\alpha_i}, \mathbf{x}^{\alpha_j})$ entry is $\mathbf{c} \cdot \mathbf{y}$ where $\mathbf{c} \in \mathbb{R}^{|\mathcal{B}_{2k}|}$ is so that

$$\mathbf{x}^{\alpha_i + \alpha_j} = \mathbf{x}^{\alpha_i}\mathbf{x}^{\alpha_j} \equiv \mathbf{c} \cdot \mathcal{B}_{2k} \mod \mathcal{I}.$$

The matrix $M_{\mathcal{B}_k}(\mathbf{y})$ is known as the *k-th truncated combinatorial moment matrix*. Having $M_{\mathcal{B}_k}$, we define the following

$$TH_k(\mathcal{I}) = \{\mathbf{y} = (1, y_1, \ldots, y_m) \mid \mathbf{y} \in \mathbb{R}^{\mathcal{B}_{2k}} \text{ and } M_{\mathcal{B}_k}(\mathbf{y}) \succeq 0\}. \tag{5.8}$$

[4]In [90], an ideal in $\mathbb{R}[X]$ is called TH_k*-exact* if $TH_k(\mathcal{I}) = cl(\mathtt{conv}(\mathbf{V}(\mathcal{I})))$. In general, $\mathtt{conv}(\mathbf{V}(\mathcal{I}))$ may not be closed while the theta bodies are. Therefore, the theta body sequence of $\mathcal{I}$ can converge, if at all, only to $cl(\mathtt{conv}(\mathbf{V}(\mathcal{I})))$.

We now explain this construction using the Clique problem. Let $G = (V, E)$ be a graph with vertex set $V = \{x_1, \ldots, x_n\}$ and edge set E. A clique in G is a set of vertices U such that for all $x_i, x_j \in U$, $x_i x_j \in E$. The corresponding ideal is

$$\mathcal{I}_{\mathsf{Clique}} = \left\langle x_i^2 - x_i : x_i \in V, x_i \cdot x_j : x_i x_j \notin E \right\rangle.$$

For a subset $U \subseteq V$ let $\mathbf{x}^U = \prod_{x_i \in U} x_i$. From definition of $\mathcal{I}$, it is clear that $\mathcal{B} = \{\mathbf{x}^U \mid U \text{ is a clique in } G\}$. In particular, $\mathcal{B}_1 = \{1, x_1, \ldots, x_n\}$. Denote by $y_0, y_1, \ldots, y_n$ the first $n+1$ coordinates of $\mathbf{y} \in \mathbb{R}^{|\mathcal{B}_{2k}|}$. Then the k-th theta body of $\mathcal{I}$ is described as

$$TH_k(\mathcal{I}_{\mathsf{Clique}}) = \{(1, y_1, \ldots, y_n) \mid \mathbf{y} \in \mathbb{R}^{|\mathcal{B}_{2k}|} \text{ and } M_{\mathcal{B}_k}(\mathbf{y}) \succeq 0\}.$$

Equivalently, in a more intuitive way, we can describe the k-th theta body as follows too.

$$TH_k(\mathcal{I}_{\mathsf{Clique}}) = \left\{ \mathbf{y} \in \mathbb{R}^n : \begin{array}{l} \exists\, M \succeq 0,\, M \in \mathbb{R}^{|\mathcal{B}_k| \times |\mathcal{B}_k|} \text{ such that} \\ M_{\emptyset\emptyset} = 1, \\ M_{\emptyset\{i\}} = M_{\{i\}\emptyset} = M_{\{i\}\{i\}} = y_i \\ M_{UU'} = 0 \text{ if } U \cup U' \text{ is not clique in } G \\ M_{UU'} = M_{WW'} \text{ if } U \cup U' = W \cup W' \end{array} \right\}.$$

Another way to compute a representation of theta bodies is to work with the *truncated quadratic module*. The *quadratic module of* $\mathtt{I}$ is

$$\mathcal{M}(\mathcal{I}) = \{s + \mathcal{I} \mid s \text{ is } \mathsf{SOS} \text{ in } \mathbb{R}[X]\},$$

and the k-th truncation of $\mathcal{M}(\mathcal{I})$ is

$$\mathcal{M}_k(\mathcal{I}) = \{s + \mathcal{I} \mid s \text{ is } k\text{-}\mathsf{SOS} \text{ in } \mathbb{R}[X]\}.$$

Both $\mathcal{M}(\mathcal{I})$ and $\mathcal{M}_k(\mathcal{I})$ are cones in the $\mathbb{R}$-vector space $\mathbb{R}[X]/I$. Recall that $\mathbb{R}[X]/I$ as a $\mathbb{R}$-vector space is isomorphic to $\mathtt{Span}(\mathbf{x}^\alpha \mid \mathbf{x}^\alpha \notin \langle \mathrm{LT}(\mathcal{I})\rangle)$. Hence, having a monomial basis $\mathcal{B} = \{\mathbf{x}^\alpha \mid \mathbf{x}^\alpha \notin \langle \mathrm{LT}(\mathcal{I})\rangle\}$ or $\mathcal{B}_k = \{\mathbf{x}^\alpha \mid |\alpha| \leq k, \mathbf{x}^\alpha \notin \langle \mathrm{LT}(\mathcal{I})\rangle\}$ would lead to an efficient computation of sum-of-squares in $\mathbb{R}[X]/I$ [184].

If the ideals arising from the combinatorial problem are produced through instance of CSP(Γ) for some language Γ, our methods make it possible to efficiently compute a representation of the corresponding theta body of one of the types described above. We say the class of k-th theta bodies arising from instances of CSP(Γ), denoted by $TH_k(\Gamma)$, is efficiently computable if for any instance $\mathcal{P}$ of CSP(Γ) a representation of the k-th theta body $TH_k(\mathcal{I}(\mathcal{P}))$ can be constructed in polynomial time. Therefore, we propose the following research problem.

Problem 5.4.5. *For which languages Γ the k-th theta body $TH_k(\mathtt{I}(\mathcal{P}))$ is computable in polynomial time where $\mathcal{P}$ is an instance of CSP(Γ)?*

We point out that the connection between theta bodies and the IMP was first reported in [161] and efficient constructions of theta bodies of Boolean problems have been addressed by Mastrolilli. Here we improve/extend upon this by employing our results on χIMP. Our results on χIMP imply that a monomial basis $\mathcal{B}_k$ as well as a k-truncated Gröbner Basis can be computed in polynomial time. This together with our result on the bit complexity of SOS proofs obtain the following.

Theorem 5.4.6. *Let $\mathcal{H}$ be a class of ideals for which χIMP_d is polynomial time decidable. Then there exists a polynomial time algorithm that constructs the d-th theta body of an ideal $\mathtt{I} \in \mathcal{H}$.*

Proof. Let $\mathtt{I}$ be an ideal in $\mathcal{H}$. By Theorem 5.1.9 we can construct the d-truncated Gröbner Basis as well as $\mathcal{B}_d$. This together with the result of Laurent [145] leads to a polynomial time algorithm to compute the d-th theta body of $\mathtt{I}$. □

Using the reductions from Theorems 5.1.4 and 5.1.10 we obtain the following results.

Corollary 5.4.7. *Let Γ and Δ be constraint languages on (possibly different) sets D, E, respectively. Suppose there exists a polynomial time algorithm that solves the search version of $\chi IMP_k(\Gamma)$. Then, $O(k)$-th theta body for language Δ can be constructed in polynomial time if*

1. *$D = E$ and Γ pp-defines Δ, or*
2. *Γ pp-interprets Δ.*

Corollary 5.4.8. *Let Γ be a finite constraint language over domain D. For constant k and an instance $\mathcal{P}$ of CSP(Γ) the k-th theta body of $\mathtt{I}(\mathcal{P})$, $TH_k(\mathtt{I}(\mathcal{P}))$, can be computed in polynomial time if*

1. *Γ has a semilattice polymorphism, or*
2. *Γ has the dual-discriminator polymorphism, or*
3. *Γ is expressed as a system of linear equations over $\mathsf{GF}(p)$, p prime.*
4. *Γ is invariant under the affine operation of an Abelian group.*

Our results yield efficient construction of theta bodies for many well-studied combinatorial problems. Note that, in this case, Theorem 5.3.3 guarantees low bit complexity of the coefficients in SOS proofs which leads to a polynomial time execution of the Ellipsoid method. As a result optimizing a linear polynomial over such theta bodies is guaranteed

to be polynomial time using the Ellipsoid method. In what follows we provide examples of well-studied problems for which the objective is optimizing a linear polynomial over integer points and (re)discover multiple positive results on computation of theta bodies. We point out there are only a handful of specific problems for which theta bodies are known to be efficiently computed.

The Maximum Stable Set problem. Let $G = (V, E)$ be an undirected graph with vertex set $V = \{x_1, \ldots, x_n\}$ and edge set E. A stable set (a.k.a independent set) in G is a set of vertices U such that for all $x_i x_j \in U$, $x_i x_j \notin E$. The Maximum Stable Set problem seeks a stable set of largest cardinality in G. This can be seen as an optimization problem over language $\Gamma = \{R_\Gamma := \{(0,0),(0,1),(1,0)\}\}$. More precisely, the objective is maximizing $\sum_i x_i$ subject to $(x_i x_j) \in R_\Gamma$, for all $x_i x_j \in E$. One can observe that Γ admits the semilattice polymorphism Min and hence by Corollary 5.4.8 the k-th theta body of the corresponding ideal to this problem, $TH_k(G)$, can be constructed in polynomial time. We point out that, for this problem, it is known that the 1-st theta body provides a convex relaxation for the (characteristic vectors) of all stable sets in G [154]. Therefore, the problem $\max_{\mathbf{x} \in TH_1(G)} \sum x_i$ is a SDP which can be solved to arbitrary precision in polynomial time in the size of G. The optimal value of this SDP provides an upper bound on the size of a maximum stable set.

Binary matroid and its theta bodies. Let $\mathcal{M} = (E, \mathcal{C})$ be a binary matroid where E is called the ground set and $\mathcal{C}$ is a collection of subsets of E that is closed under taking symmetric differences. Each member of $\mathcal{C}$ is called a *cycle*. (Often members of $\mathcal{C}$ are called independent sets, however, here we call them cycles to avoid confusion.) One can view the binary matroid $\mathcal{M} = (E, \mathcal{C})$ as the $\mathsf{GF}(2)$-vector subspaces of $\mathsf{GF}(2)^E$. Let $\mathbf{1}_F \in \{0,1\}^E$ denote the characteristic vector of $F \subseteq E$, thus the cycles of the binary matroid $\mathcal{M}$ arise as the solutions in $\mathsf{GF}(2)^E$ of a linear system $M\mathbf{x} = 0$, where M is a matrix with columns indexed by E. The convex hull of the vectors $\mathbf{1}_F$, $F \in \mathcal{C}$, is called the *matroid polytope* or *cycle polytope*. Let $\mathcal{I}(\mathcal{M}) \subseteq \mathbb{R}[x_i \mid i \in E]$ be the vanishing ideal of the cycle vectors of $\mathcal{M}$. Our result in Theorem 5.1.10 implies the following theorem.

Theorem 5.4.9. *The set* $\mathcal{B}_k = \{\mathbf{x}^\alpha \mid |\alpha| \leq k, \mathbf{x}^\alpha \notin \langle \mathrm{LT}(\mathtt{I}(\mathcal{M}))\rangle\}$*, and the* k*-truncated Gröbner Basis of the ideal* $\mathtt{I}(\mathcal{M})$ *can be computed in polynomial time.*

This result has numerous consequences including a polynomial time algorithm to construct $TH_k(\mathcal{I}(\mathcal{M}))$. This in turn can be viewed as computing the moment matrices for the cycle ideal $\mathcal{I}(\mathcal{M})$. One classical and well studied application is *cut function* of graphs. When $\mathcal{M}$ is the cut polytope then $TH_k(\mathcal{I}(\mathcal{M}))$ provides a relaxation for the *cut polytope*. In particular, $TH_1(\mathcal{I}(\mathcal{M}))$ coincides with the edge-relaxation considered by Rendl and Wiegele [219] and numerical experiments there indicates that it is often tighter than the Goemans-Williamson SDP relaxation [87]. See [89] for more detailed discussions and applications.

The Min/Max Ones problem and a generalization. Min Ones and Max Ones are Boolean CSP problems where the objective is to find a feasible solution (a 0-1 assignment satisfying all constraints) minimizing/maximizing the number of variables assigned the label 1 [130]. Classical examples are the Minimum Vertex Cover, the Maximum Stable Set, and many packing and covering problems [202]. In almost all the cases where Boolean CSP is polynomial time decidable our results imply that we can construct theta bodies as relaxations of the solution space. This motivates the question of studying the power of the theta body relaxations in designing approximation algorithms for this class of problems.

The MinHOM problem. Recall the MinHOM problem in which $H = (V, E)$ with $V(H) = \{0, 1, \ldots, d\}$ is a fixed digraph and $G = (V, E)$ with $V(G) = \{x_1, \ldots, x_n\}$ is an input digraph. Given a cost function $c : V(G) \times V(H) \to \mathbb{R}_+ \cup \{+\infty\}$, the objective is to find a homomorphism from G to H with minimum cost which is defined as $\sum_{i \in [n], j \in V(H)} c(x_i, j)$. Our result implies that we can construct theta bodies for many cases of H, in particular all the approximable cases considered in this thesis, and have a convex relaxation of all possible homomorphisms. This motivates the question of checking if formulating the MinHOM problem as an optimization problem over theta bodies help to achieve (better) approximation algorithms.

The Strict CSPs problem. A natural generalization of the above problems, both in terms of domain and arity of the relations, is the class of Strict CSPs [142]. As usual we have a constraint language Γ on domain $D = \{0, \ldots, d\}$. Given an instance $\mathcal{P} = (X, D, C)$ of CSP(Γ) and a cost function $c : X \times D \to \mathbb{R}_+ \cup \{+\infty\}$, the objective is to minimize/maximize $\sum_{x_i \in X, j \in D} c(x_i, j)$ subject to $\mathbf{V}(\langle \mathcal{P} \rangle)$ i.e., finding a solution to $\mathcal{P}$ that minimizes/maximizes $\sum_{x_i \in X} x_i$. The approximability of this problem and it special cases have been studied intensively and (Unique Game) hardness results are known [142]. It is tempting to study the power of theta body relaxations for approximation of Strict CSPs as our results imply in many cases a theta body can be constructed in polynomial time.

Chapter 6

Approximation of minimum cost H-coloring

6.1 Introduction

The complexity of *exact minimization* of MinHOM(H) was studied in a series of papers, and complete complexity classifications were given in [95] for undirected graphs, in [109] for digraphs, and in [206] for more general structures. Certain minimum cost homomorphism problems have polynomial time algorithms [95, 96, 97, 109], but most are **NP**-hard. We remark that, the complexity of *exact minimization* of VCSPs is well understood [136, 207]. In terms of approximation, Hell *et al.,* [103] proved a dichotomy for approximating MinHOM(H) when H is a bipartite graph by transforming the MinHOM(H) to a linear program, and rounding the fractional values to get a homomorphism to H. Their characterization is best described in terms of polymorphisms or equivalently an ordering on the vertices of H. They say a (di)graph admits a min-ordering when the (di)graph is invariant under a conservative semilattice polymorphism, more on this latter.

Theorem 6.1.1 (Dichotomy for bipartite graphs [103])**.** *For a fixed bipartite graph H, MinHOM(H) admits a constant factor approximation algorithm if H admits a* min-ordering *(complement of H is a* circular arc *graph), otherwise MinHOM(H) is not approximable unless $\boldsymbol{P = NP}$.*

Beyond this, there is no result concerning the approximation of MinHOM(H).

6.1.1 Overview of our contributions

One can show that if LHOM(H) is not polynomial time solvable then there is no approximation algorithm for MinHOM(H) [103, 162].

Observation 6.1.2. *If LHOM(H) is not polynomial time solvable, then there is no approximation algorithm for MinHOM(H).*

The complexity of the LHOM problems for graphs, digraphs, and relational structures (with arity two and higher) have been classified in [76, 108, 33] respectively. LHOM(H) is polynomial time solvable if the digraph H does not contain a *digraph asteroidal triple (DAT)*[1] as an induced sub-digraph, and **NP**-complete when H contains a DAT [108].

MinHOM(H) is polynomial time solvable when digraph H admits a k-min-max-ordering, a subclass of DAT-free digraphs, and otherwise, **NP**-complete [109, 110].

Here, in this thesis, we take an important step towards closing the gap between DAT-free digraphs and the one that admit a k-min-max-ordering. First, we consider digraphs that admit a min-ordering i.e., conservative semilattice polymorphism. Digraphs that admit a min-ordering have been studied under the name of *bi-arc* digraphs [107] and *signed interval* digraphs [101, 102]. Deciding if digraph H has a min-ordering and finding a min-ordering of H is in **P** [107]. We provide a constant factor approximation algorithm for MinHOM(H) where H admits a min-ordering.

Theorem 6.1.3 (Digraphs with a min-ordering)**.** *If digraph H admits a min-ordering i.e., conservative semilattice polymorphism, then MinHOM(H) has a constant factor approximation algorithm.*

Sections 6.4, 6.5 are dedicated to the proof of Theorem 6.1.3. In section 6.6, we turn our attention to digraphs with k-min-orderings, for integer $k > 1$. They are also called digraphs with *extended X-underbar* [9, 98, 156]. It was shown in [98] that if H has the X-underbar property, then the HOM(H) problem is polynomial time solvable. In Lemmas 6.6.2 and 6.6.1, we show that if H admits a k-min-ordering, then H is a DAT-free digraph, and provide a simple proof that LHOM(H) is polynomial time solvable. Finally, we have the following theorem.

Theorem 6.1.4 (Digraphs with a k-min-ordering)**.** *If digraph H admits a k-min-ordering for some integer $k > 1$, then MinHOM(H) has a constant factor approximation algorithm.*

Considering graphs, Feder *et al.,* [76] proved that LHOM(H) is polynomial time solvable if H is a *bi-arc* graph, and is **NP**-complete otherwise. In the same paper, they showed graph H is a bi-arc graph if and only if it admits a conservative majority polymorphism. In Section 6.7, we show that the same dichotomy classification holds in terms of approximation.

Theorem 6.1.5 (Dichotomy for graphs)**.** *Let H be a graph. There exists a constant factor approximation algorithm for MinHOM(H) if H is a bi-arc graph i.e. admits a conservative majority polymorphism, otherwise, MinHOM(H) is inapproximable unless $\boldsymbol{P = NP}$.*

By combining the approach for obtaining the dichotomy in the graph case, together with the idea of getting an approximation algorithm for digraphs admitting a min-ordering, we

[1]The definition of DAT (Definition 6.2.4) is rather technical and it is not necessary to fully understand in this thesis.

might be able to achieve a constant factor approximation algorithm for MinHOM(H) when H is DAT-free. Note that DAT-free digraphs can be characterized in terms of conservative semilattice and majority polymorphisms [108]. Thus, we conjecture the following dichotomy.

Conjecture 6.1.6. *Let H be a digraph. MinHOM(H) admits a constant factor approximation algorithm when H is a DAT-free digraph, otherwise, MinHOM(H) is not approximable unless $\boldsymbol{P = NP}$.*

Our constant factors depend on the size of H. However, the implementation of the LP and the ILP would yield a small integrality gap (See [187]). This indicates perhaps a better analysis of the performance of our algorithm is possible.

Problem 6.1.7. *For which digraphs MinHOM(H) is approximable within a constant factor independent of size of H?*

6.2 Preliminaries

Recall that a polymorphism of H of arity k is a mapping f from the set of k-tuples over $V(H)$ to $V(H)$ such that if $x_iy_i \in A(H)$ for $i = 1, 2, \ldots, k$, then $f(x_1, x_2, \ldots, x_k)f(y_1, y_2, \ldots, y_k) \in A(H)$, and a polymorphism f is *conservative* if $f(x_1, x_2, \ldots, x_k) \in \{x_1, x_2, \ldots, x_k\}$. If f is a polymorphism of H we also say that H admits f. A conservative semilattice polymorphism of H naturally defines a binary relation $x \leq y$ on the vertices of H by $x \leq y$ if and only if $f(x, y) = x$; by associative, the relation $\leq$ is a linear order on $V(H)$, which we call a *min-ordering* of H.

Definition 6.2.1. *The ordering $v_1 < v_2 < \cdots < v_n$ of $V(H)$ is a*

- min-ordering *if and only if $uv \in A(H), u'v' \in A(H)$ and $u < u', v' < v$ implies that $uv' \in A(H)$;*
- max-ordering *if and only if $uv \in A(H), u'v' \in A(H)$ and $u < u', v' < v$ implies that $u'v \in A(H)$;*
- min-max-ordering *if and only if $uv \in A(H), u'v' \in A(H)$ and $u < u', v' < v$ implies that $uv', u'v \in A(H)$.*

For a bipartite graph $H = (B, W)$ let $\overrightarrow{H}$ be the digraph obtained by orienting all the edges of H from B to W. If $\overrightarrow{H}$ admits a min-ordering then we say H admits a min-ordering. It is worth mentioning that, a bipartite graph H admits a conservative majority, if and only if it admits a min-ordering [103]. Moreover, the complement of H is a circular arc graph with clique cover two [76].

Definition 6.2.2. *Let $H = (V, E)$ be a digraph that admits a homomorphism $f : V(H) \to \overrightarrow{C_k}$ (here $\overrightarrow{C_k}$ is the induced directed cycle on $\{0, 1, 2, \ldots, k-1\}$ (i.e., arc set $\{(01, 12, 23, ..., (k-2)(k-1), (k-1)0\}$). Let $V_i = f^{-1}(i)$, $0 \leq i \leq k-1$.*

– *A* k-min-ordering *of* H *is a linear ordering* $<$ *of the vertices of* H*, so that* $<$ *is a min-ordering on the subgraph induced by any two circularly consecutive* V_i, V_{i+1} *(subscript addition modulo* k*).*

– *A* k-min-max-ordering *of* H *is a linear ordering* $<$ *of the vertices of* H*, so that* $<$ *is a min-max-ordering on the subgraph induced by any two circularly consecutive* V_i, V_{i+1} *(subscript addition modulo* k*).*

We close this section by giving a formal definition of a digraph asteroidal triple (DAT). The definition is rather technical and it is not necessary to fully understand it in this thesis. We give a brief discussion on DAT for the sake of completeness.

Definition 6.2.3 (Invertible pair)**.** *Let* H *be a digraph. Define* $\widehat{H^k}$ *to be the digraph with the vertex set* $\{(a_1, a_2, \ldots, a_k) \mid a_i \in V(H), 1 \leq i \leq k\}$ *and the arc set*

$$\begin{aligned} A(\widehat{H^k}) =& \{(a_1, a_2, \ldots, a_k)(b_1, b_2, \ldots, b_k) \mid a_i b_i (b_i a_i) \in A(H), 1 \leq i \leq k, \\ & a_1 b_j (b_j a_1) \notin A(H) \forall j, 2 \leq j \leq k\}. \end{aligned}$$

When $k = 2$*, we say* (x, y) *is an invertible pair if* $(x, y), (y, x)$ *belong to the same strong component of* $\widehat{H^2}$*.*

Definition 6.2.4 (DAT)**.** *A digraph asteroidal triple of* H *is an induced sub-digraph of* $\widehat{H^3}$*, on three directed paths* P_1, P_2, P_3 *where* P_1 *goes from* (a, b, c) *to* (α, β, β)*,* P_2 *goes from* (b, a, c) *to* (α, β, β)*, and* P_3 *goes from* (c, a, b) *to* (α, β, β) *and* (α, β) *is an invertible pair.*

If H contains a DAT then all three pairs $(a, b), (b, c), (c, a)$ are invertible. Note that an invertible pair is an obstruction to existence of min-orderings [76, 103]. Moreover, H does not admit a conservative majority polymorphism g because of the directed path P_1, $g(a, b, c) \neq a$, and because of P_2, $g(a, b, c) \neq b$, and finally because of P_3, $g(a, b, c) \neq c$. Therefore, the value of $g(a, b, c)$ can not be any of the a, b, c [108].

DAT-free digraph can also be characterized in terms of polymorphisms.

Theorem 6.2.5 ([108])**.** *Let* H *be a digraph.* H *is DAT-free if and only if there exist a conservative binary polymorphism* f *and a conservative ternary polymorphism* g *of* H *such that for every* $a, b \in V(H)$*,*

1. *either* $f|_{\{a,b\}}$ *is a semilattice polymorphism or*
2. $g|_{\{a,b\}}$ *is a majority polymorphism.*

6.3 LP for digraphs with a min-max-ordering

Before presenting the LP, we give a procedure to modify lists associated to the vertices of D. To each vertex $x \in D$, we associate a list $L(x)$ that initially contains $V(H)$. Think of

$L(x)$ as the set of possible images for x in a homomorphism from D to H. Apply the *arc consistency* procedure as follows. Take an arbitrary arc $xy \in A(D)$ ($yx \in A(D)$) and let $a \in L(x)$. If there is no out-neighbor (in-neighbor) of a in $L(y)$ then remove a from $L(x)$. Repeat this until a list becomes empty or no more changes can be made. Note that if we end up with an empty list after arc consistency then there is no homomorphism of D to H.

Let $a_1, a_2, a_3, \ldots, a_p$ be a min-max-ordering $<$ of the target digraph H. Define $\ell^+(i)$ to be the smallest subscript j such that a_j is an out-neighbor of a_i (and $\ell^-(i)$ to be the smallest subscript j such that a_j is an in-neighbor of a_i).

Consider the following linear program. For every vertex v of D and every vertex a_i of H define variable v_i. We also define variable v_{p+1} for every $v \in D$ whose value is set to zero.

`min`	$\sum_{v,i} c(v,a_i)(v_i - v_{i+1})$		
`subject to:`	$v_i \geq 0$		(C1)
	$v_1 = 1$		(C2)
	$v_{p+1} = 0$		(C3)
	$v_{i+1} \leq v_i$		(C4)
	$v_{i+1} = v_i$	if $a_i \notin L(v)$	(C5)
	$u_i \leq v_{l^+(i)}$	$\forall uv \in A(D)$	(C6)
	$v_i \leq u_{l^-(i)}$	$\forall uv \in A(D)$	(C7)

Table 6.1: LP with constraint set $\mathcal{S}$

Let us denote the set of constraints of the above LP by $\mathcal{S}$. In what follows, we prove that there is a one-to-one correspondence between integer solutions of $\mathcal{S}$ and homomorphisms from D to H when H admits a min-max-ordering.

Theorem 6.3.1. *If digraph H admits a min-max-ordering, then there is a one-to-one correspondence between homomorphisms of D to H and integer solutions of $\mathcal{S}$.*

Proof. For homomorphism $f : D \to H$, if $f(v) = a_t$ we set $v_i = 1$ for all $i \leq t$, otherwise we set $v_i = 0$. We set $v_1 = 1$ and $v_{p+1} = 0$ for all $v \in V(D)$. Now all the variables are nonnegative and we have $v_{i+1} \leq v_i$. Note that if $a_i \notin L(v)$ then $f(v) \neq a_i$ and we have $v_i - v_{i+1} = 0$. It remains to show that $u_i \leq v_{l^+(i)}$ for every uv arc in D. Suppose for contradiction that $u_i = 1$ and $v_{l^+(i)} = 0$ and let $f(u) = a_r$ and $f(v) = a_s$. This implies that $u_r = 1$, whence $i \leq r$; and $v_s = 1$, whence $s < l^+(i)$. Since $a_i a_{l^+(i)}$ and $a_r a_s$ both are arcs of H with $i \leq r$ and $s < l^+(i)$, the fact that H has a min-ordering implies that $a_i a_s$ must also be an arc of H, contradicting the definition of $l^+(i)$. The proof for $v_i \leq u_{l^-(i)}$ is analogous.

Conversely, if there is an integer solution for $\mathcal{S}$, we define a homomorphism f as follows: we let $f(v) = a_i$ when i is the largest subscript with $v_i = 1$. We prove that this is indeed a homomorphism by showing that every arc of D is mapped to an arc of H. Let uv be an arc of D and assume $f(u) = a_r$, $f(v) = a_s$. We show that $a_r a_s$ is an arc in H. Observe that $1 = u_r \leq v_{l^+(r)} \leq 1$ and $1 = v_s \leq u_{l^-(s)} \leq 1$, therefore we must have $v_{l^+(r)} = u_{l^-(s)} = 1$.

Since r and s are the largest subscripts such that $u_r = v_s = 1$ then $l^+(r) \leq s$ and $l^-(s) \leq r$. Since $a_r a_{l^+(r)}$ and $a_{l^-(s)} a_s$ are arcs of H, we must have the arc $a_r a_s$, as H admits a max-ordering.

Furthermore, $f(v) = a_i$ if and only if $v_i = 1$ and $v_{i+1} = 0$, so, $c(v, a_i)$ contributes to the sum if and only if $f(v) = a_i$. □

We have translated the minimum cost homomorphism problem to a linear program. In fact, this linear program corresponds to a minimum cut problem in an auxiliary network, and can be solved by network flow algorithms [95, 162]. In [103], a similar result to Theorem 6.3.1 was proved for the MinHOM(H) problem on undirected graphs when target the graph H is bipartite and admits a min-max-ordering. We shall enhance the above system $\mathcal{S}$ to obtain an approximation algorithm for the case where H is only assumed to have a min-ordering.

6.4 LP for digraphs with a min-ordering

In the rest of the section assume lists are not empty. Moreover, non-empty lists guarantee a homomorphism when H admits a min-ordering. For the sake of completeness we present the proof of the following lemma. The argument is simple and perhaps could have appeared in earlier literature.

Lemma 6.4.1 ([105])**.** *Let H be a digraph that admits a min-ordering. If all the lists are non-empty after arc consistency, then there exists a homomorphism from D to H.*

Proof. Let $a_1, a_2, \ldots, a_p$ be a min-ordering of H. For every vertex x of D, define $f(x) = a_i$ where a_i is the smallest element (according to the ordering) in $L(x)$. We show that f is a homomorphism from D to H. Let xy be an arc of D. Suppose $f(x) = a_i$ and $f(y) = a_j$. Because of the arc-consistency, there exist $a_{j'}$ in $L(y)$ such that $a_i a_{j'} \in A(H)$ and there exists $a_{i'} \in L(x)$ such that $a_{i'} a_j \in A(H)$. Note that $j \leq j'$ and $i \leq i'$. Since $a_1, a_2, \ldots, a_p$ is a min-ordering, then $a_i a_j \in A(H)$ and $f(x)f(y) \in A(H)$. □

Suppose $a_1, a_2, \cdots, a_p$ is a min-ordering of H. Let E' denote the set of all the pairs (a_i, a_j) such that $a_i a_j$ is not an arc of H, but there is an arc $a_i a_{j'}$ of H with $j' < j$ and an arc $a_{i'} a_j$ of H with $i' < i$. Let $E = A(H)$ and define H' to be the digraph with vertex set $V(H)$ and arc set $E \cup E'$. Note that E and E' are disjoint sets.

Observation 6.4.2. *The ordering $a_1, a_2, \cdots, a_p$ is a min-max-ordering of H'.*

Proof. We show that for every pair of arcs $e = a_i a_{j'}$ and $e' = a_{i'} a_j$ in $E \cup E'$, with $i' < i$ and $j' < j$, both $g = a_i a_j$ and $g' = a_{i'} a_{j'}$ are in $E \cup E'$. If both e and e' are in E, $g \in E \cup E'$ and $g' \in E$.

If only one of the arcs e, e', say e, is in E', there is a vertex $a_{j''}$ with $a_i a_{j''} \in E$ and $j'' < j'$, and a vertex $a_{i''}$ with $a_{i''} a_{j'} \in E$ and $i'' < i$. Now, $a_{i'} a_j$ and $a_i a_{j''}$ are both in E,

so $g \in E \cup E'$. We may assume that $i'' \neq i'$, otherwise $g' = a_{i''}a_{j'} \in E$. If $i'' < i'$, then $g' \in E \cup E'$ because $a_{i'}a_{j''} \in E$; and if $i'' > i'$, then $g' \in E$ because $a_{i'}a_j \in E$.

If both edges e, e' are in E', then the earliest out-neighbor of a_i and earliest in-neighbor of a_j in E imply that $g \in E \cup E'$, and the earliest out-neighbors of $a_{i'}$ and earliest in-neighbor of $a_{j'}$ in E imply that $g' \in E \cup E'$. □

Observation 6.4.3. *Let $e = a_ia_j \in E'$. Then a_i does not have any out-neighbor in H after a_j, or a_j does not have any in-neighbor in H after a_i.*

Observation 6.4.3 easily follows from the fact that H has a min-ordering. Since H' has a min-max-ordering, we can form system of linear inequalities $\mathcal{S}$, for H' as described in Section 6.3. Homomorphisms of D to H' are in a one-to-one correspondence with integer solutions of $\mathcal{S}$, by Theorem 6.3.1. However, we are interested in homomorphisms of D to H, not H'. Therefore, we shall add further inequalities to $\mathcal{S}$ to ensure that we only admit homomorphisms from D to H, i.e., avoid mapping arcs of D to the arcs in E'. For every arc $e = a_ia_j \in E'$ and every arc $uv \in A(D)$, by Observation 6.4.3, two of the following set of inequalities will be added to $\mathcal{S}$ (i.e. either (C8), (C11) or (C9), (C10)).

$v_j \leq u_s + \sum_{\substack{t<i \\ a_ta_j \in E \\ a_t \in L(u)}} (u_t - u_{t+1})$	if $a_s \in L(u)$ is the first in-neighbor of a_j after a_i	(C8)
$v_j \leq v_{j+1} + \sum_{\substack{t<i \\ a_ta_j \in E \\ a_t \in L(u)}} (u_t - u_{t+1})$	if a_j has no in-neighbor after a_i	(C9)
$u_i \leq v_s + \sum_{\substack{t<j \\ a_ia_t \in E \\ a_t \in L(v)}} (v_t - v_{t+1})$	if $a_s \in L(v)$ is the first out-neighbor of a_i after a_j	(C10)
$u_i \leq u_{i+1} + \sum_{\substack{t<j \\ a_ia_t \in E \\ a_t \in L(v)}} (v_t - v_{t+1})$	if a_i has no out-neighbor after a_j	(C11)

Table 6.2: Extension of $\mathcal{S}$

Additionally, for every pair $(x, y) \in V(D) \times V(D)$ consider a list $L(x, y)$ of possible pairs (a, b), $a \in L(x)$ and $b \in L(y)$. Perform *pair consistency* procedure as follows. Consider three vertices $x, y, z \in V(D)$. For $(a, b) \in L(x, y)$ if there is no $c \in L(z)$ such that $(a, c) \in L(x, z)$ and $(c, b) \in L(z, y)$ then remove (a, b) from $L(x, y)$. Repeat this until a pair list becomes empty or no more changes can be made. Here, we assume that after pair consistency procedure no pair list is empty, as otherwise there is no homomorphism of D to H. Therefore, by pair consistency, add the following constraints for every $u \neq v$ in $V(D)$ and $a_i \in L(u)$:

$$u_i - u_{i+1} \leq \sum_{\substack{j: \\ (a_i,a_j) \in L(u,v)}} (v_j - v_{j+1}) \quad \text{(C12)}$$

Lemma 6.4.4. *If H admits a min-ordering, then there is a one-to-one correspondence between homomorphisms of D to H and the integer solutions of the extended system $\mathcal{S}$.*

Proof. In the proof of Theorem 6.3.1 we shown that from an integer solution for $\mathcal{S}$, one can obtained a homomorphism from D to H'. Let f be such a homomorphism. We show that f is a homomorphism from D to H. Let uv be an arc of D and let $f(u) = a_i, f(v) = a_j$. We have $u_i = 1$, $u_{i+1} = 0$, $v_j = 1$, $v_{j+1} = 0$, and for all $a_t a_j \in E$ with $t < i$ we have $u_t - u_{t+1} = 0$. We show that $a_i a_j \in E$. If it is not the case, then either constraints (C8),(C9) or constraints (C10),(C11) should hold in the LP. Consider the former case. If a_s is the first in-neighbor of a_j after a_i, then we will also have $u_s = 0$, and so inequality (C8) fails. Else, if a_j has no in-neighbor after a_i, then inequality (C9) fails. The other case is similar.

Conversely, suppose f is a homomorphism of D to H (i.e., f maps the edges of D to the edges in E). We show that the inequalities hold. For a contradiction, assume that the first inequality fails (the other inequalities are similar). This means that for some arc $uv \in A(D)$ and some edge $a_i a_j \in E'$, we have $v_j = 1$, $u_s = 0$, and the sum of $(u_t - u_{t+1}) = 0$, summed over all $t < i$ such that a_t is an in-neighbor of a_j. The latter two facts easily imply that $f(u) = a_i$. Since a_j has an in-neighbor after a_i, Observation 6.4.3 tells us that a_i has no out-neighbors after a_j, whence $f(v) = a_j$ and thus $a_i a_j \in E$, contradicting the fact that $a_i a_j \in E'$. Note that if there is a homomorphism from D to H then inequality (C12) is a necessary condition for having such a homomorphism. □

6.5 Approximation for digraphs with a min-ordering

In what follows, we describe our approximation algorithm for MinHOM(H) where the fixed digraph H has a min-ordering. We start off with an overview of our algorithm. The proofs of the correctness and approximation bound are postponed for the later subsections.

Let D be the input digraph together with a costs function c, and let H be a fixed target digraph H, let $a_1, \cdots, a_p$ be a min-ordering of the vertices of H. Algorithm 3, first constructs digraph H' from H as explained in Section 6.4. By Observation 6.4.2, $a_1, \ldots, a_p$ is a min-max-ordering for H'. By Lemma 6.4.4, the integral solutions of the extended LP are in one-to-one correspondence to homomorphisms from D to H. At this point, our algorithm will minimize the cost function over extended $\mathcal{S}$ in polynomial time using a linear programming algorithm. This will generally result in a fractional solution (Even though the original system $\mathcal{S}$ is known to be totally unimodular [95, 162] and hence has integral optima, we have added inequalities, thus losing this advantage). We will obtain an integer solution by a randomized procedure called *rounding*. Choose, uniformly at random, a random variable $X \in [0, 1]$, and define the rounded values $u'_i = 1$ when $u_i \geq X$ (u_i is the returned value by the LP), and $u'_i = 0$ otherwise. It is easy to check that the rounded values satisfy the original inequalities, i.e., correspond to a homomorphism f of D to H'.

Now the algorithm will modify the solution f to become a homomorphism from D to H, i.e., to avoid mapping the arcs of D to the arcs in E'. This will be accomplished by another randomized procedure, which we call SHIFT. We choose, uniformly at random, another random variable $Y \in [0,1]$, which will guide the shifting. Let F denote the set of all arcs in E' to which some arcs of D are mapped by f. If F is empty, we need no shifting. Otherwise, let a_ia_j be an arc of F. Since $F \subseteq E'$, Observation 6.4.3 implies that either a_j has no in-neighbor after a_i or a_i has no out-neighbor after a_j. Suppose the first case happens (the shifting process is similar in the other case).

Consider a vertex v in D such that $f(v) = a_j$ (i.e. $v'_j = 1$ and $v'_{j+1} = 0$) and v has an in-neighbor u in D with $f(u) = a_i$ (i.e. $u'_i = 1$ and $u'_{i+1} = 0$). For such a vertex v, let $S_v = \{a_{t_1}, a_{t_2}, \ldots, a_{t_k}\}$ be the set of all vertices a_t with $t < j$ such that $a_ia_t \in E$ and $a_t \in L(v)$. We will show in Lemma 6.5.1 that S_v is not empty. Suppose S_v consists of a_t with subscripts t ordered as $t_1 < t_2 < \cdots < t_k$. The algorithm now selects one vertex from this set as follows. Let $P_{v,t} = \frac{v_t - v_{t+1}}{P_v}$, where

$$P_v = \sum_{\substack{t<j \\ a_ia_t \in E \\ a_t \in L(v)}} (v_t - v_{t+1}).$$

Note that $P_v > 0$ because of constraints (C9) and (C10). Then a_{t_q} is selected if $\sum_{p=1}^{q} P_{v,t_p} < Y \leq \sum_{p=1}^{q+1} P_{v,t_p}$. Thus a concrete a_t is selected with probability $P_{v,t}$, which is proportional to the difference of the fractional values $v_t - v_{t+1}$. When the selected vertex is a_t, we shift the image of the vertex v from a_j to a_t, and set $v'_r = 1$ if $r \leq t$, else set $v'_r = 0$. Note that a_t is before a_j in the min-ordering[2]. Now we might need to shift images of the neighbors of v. In this case, repeat the shifting procedure for neighbors of v. This processes continues in a Breadth-first search (BFS) like manner, until no more shift is required (see Figure 6.1 for an illustration). Note that a vertex might be visited multiple times in procedure shift while a pair $(v, a_i) \in V(D) \times V(H)$ is considered at most one time.

Lemma 6.5.1. *During procedure SHIFT, the set of indices $t_1 < \cdots < t_k$ considered in Line 19 of the Algorithm 3 is non-empty.*

Proof. In procedure SHIFT, consider vz such that $f(v)f(z) \notin E(H')$ and $f(v) = a_t$ and $f(z) = a_l$. This means $0 < v_t - v_{t+1}$, and together with constraint (C12), it implies

$$0 < v_t - v_{t+1} \leq \sum_{\substack{j: \\ (a_t, a_j) \in L(v,z)}} (z_j - z_{j+1}).$$

[2]The images are always shifted towards smaller elements.

Therefore, there must be an index l' such that $(a_t, a_{l'}) \in L(v, z)$. It remains to show that $a_{l'}$ appears before a_l in the min-ordering. There are two cases to consider. First is $f(v)$ is set to a_t in rounding step (Line 5). Second is image of v was shifted from a_j to a_t in procedure SHIFT.

For the first case, note that, since f is a homomorphism from D to H', $a_t a_l \in E(H') \setminus E(H)$. Arc vz is mapped to $a_t a_l$ in rounding step (Line 5) according to random variable X. Note that, during procedure SHIFT, we do not map any arc of D to edges in $E(H') \setminus E(H)$. Therefore, we have $X \leq v_t, z_l$. Consider the situation where a_l has no in-neighbor after a_t. Let a_s be the first out-neighbor of a_t after a_l, then we have $z_s < X \leq v_t$. This together with inequality (C10) implies that

$$0 < \sum_{\substack{l'<l \\ a_t a_l \in E \\ a_{l'} \in L(z)}} (z_{l'} - z_{l'+1}).$$

Hence, there exists an index $l' < l$ as we wanted. The argument for the case where a_t has no out-neighbor after a_l is similar.

For the second case, before mapping v to a_t, there was an index a_j such that $a_t < a_j$. There are two cases regarding $a_j a_l$. Either it is in $E(H)$ or it is in $E(H') \setminus E(H)$. In both cases, $a_{l'}$ must appear before a_l as otherwise, min-max-ordering implies $a_t a_l \in E(H')$, contradicting our assumption. □

Lemma 6.5.2. *Procedure* SHIFT *runs in polynomial time and returns a homomorphism from D to H'.*

Proof. It it easy to see that, if there exists a homomorphism from D to H, then there is a homomorphism from D to H that maps every vertex of D to the smallest vertex in its list (Lemma 6.4.1). We show that a sequence of shifting, either stops at some point, or it keeps shifting to a smaller vertex in each list. In the latter case, after finite (polynomially many) steps, we end up mapping every vertex of D to the smallest vertex in its list.

Consider an arc $vz \in A(D)$. Suppose $f(v) = a_t$ and $f(z) = a_l$. Assume that we have shifted the image of v from a_t to $a_{t'} \in L(v)$ where $a_{t'}$ is before a_t in the min-ordering. If $a_{t'} a_l$ is in $E(H)$ then we do not have to shift the image of z. Note that, since $a_{t'}$ is in $L(v)$ then it has to have an out-neighbor in $L(z)$. Let say $a_{l'} \in L(z)$ is an out-neighbor of $a_{t'}$. If $a_{l'}$ is after a_l in the min-ordering then it implies $a_{t'} a_{l'} \in A(H)$. Else, $a_{l'}$ is before a_l in the min-ordering and we shift the image of z to a smaller vertex in its list. □

Lemma 6.5.2 shows that this shifting modifies the homomorphism f, and hence, the corresponding values of the variables. Namely, $v'_{t+1}, \ldots, v'_j$ are reset to 0, keeping all other values the same. Note that these modified values still satisfy the original set of constraints $\mathcal{S}$, i.e., the modified mapping is still a homomorphism from D to H'.

We repeat the same process for the next v with these properties, until no edge of D is mapped to an edge in E'. Each iteration involves at most $|V(H)| \cdot |V(D)|$ shifts. After at most $|E'|$ iterations, no edge of D is mapped to an edge in F and we no longer need to shift. See Figure 6.1 for an example. Next theorem follows from Lemma 6.5.1 and 6.5.2.

Theorem 6.5.3. *Algorithm 3, in polynomial time, returns a homomorphism of D to H.*

Algorithm 3 Approximation MinHOM(H)

procedure Approx–MinHOM(H)
  Construct H' from H (as in Section 6.3)
  Let u_is be the (fractional) values returned by the extended LP
  Choose a random variable $X \in [0, 1]$
  For all u_is: if $X \leq u_i$ let $u'_i = 1$, else let $u'_i = 0$
  Let $f(u) = a_i$ where i is the largest subscript with $u'_i = 1$ ▷ f is a homomorphism from D to H'
  Choose a random variable $Y \in [0, 1]$
  while $\exists uv \in A(D)$ such that $f(u)f(v) \in A(H') \setminus A(H)$ **do**
    if $f(v)$ does not have an in-neighbor after $f(u)$ **then**
      Shift(f, v)
    else if $f(u)$ does not have an out-neighbor after $f(v)$ **then**
      Shift(f, u)
  return f ▷ f is a homomorphism from D to H

procedure Shift(f, x)
  Let Q be a Queue, $Q.enqueue(x)$
  while Q is not empty **do**
    $v \leftarrow Q.dequeue()$
    for $uv \in A(D)$ with $f(u)f(v) \notin A(H)$ **or** $vu \in A(D)$ with $f(v)f(u) \notin A(H)$ **do**
        ▷ Here we assume the first condition hold, the other case is similar
          ▷ Plus, we assume $f(v)$ does not have an in-neighbor after $f(u)$
      Let $t_1 < \cdots < t_k$ be indices so that $a_{t_j} < f(v), a_{t_j} \in L(v), f(u)a_{t_j} \in A(H)$
      Let $P_v \leftarrow \sum_{j=1}^{j=k} (v_{t_j} - v_{t_j+1})$ and $P_{v,t} \leftarrow (v_t - v_{t+1}) \,/\, P_v$
      if $\sum_{p=1}^{q} P_{v,t_p} < Y \leq \sum_{p=1}^{q+1} P_{v,t_p}$ **then**
        $f(v) \leftarrow a_{t_q}$, set $v'_i = 1$ for $1 \leq i \leq t_q$, and set $v'_i = 0$ for $t_p < i$
      for $vz \in A(D)$ $(zv \in A(D))$ with $f(v)f(z) \notin A(H)$ $(f(z)f(v) \notin A(H))$ **do**
        $Q.enqueue(z)$
  return f ▷ f is a homomorphism from D to H'

Example 6.5.4 (Figure 6.1: two examples for Algorithm 3)**.** In the right example, the target digraph is H_1 and the input is D_1. The right digraphs (D_1, H_1) both can be view as bipartite graphs and $1, 2, 3, 4, 5, 6, 7$ is a min-ordering of H. When x is mapped to 3 and w is mapped to 6 then the algorithm should shift the image of w from 6 to 5 and since 35

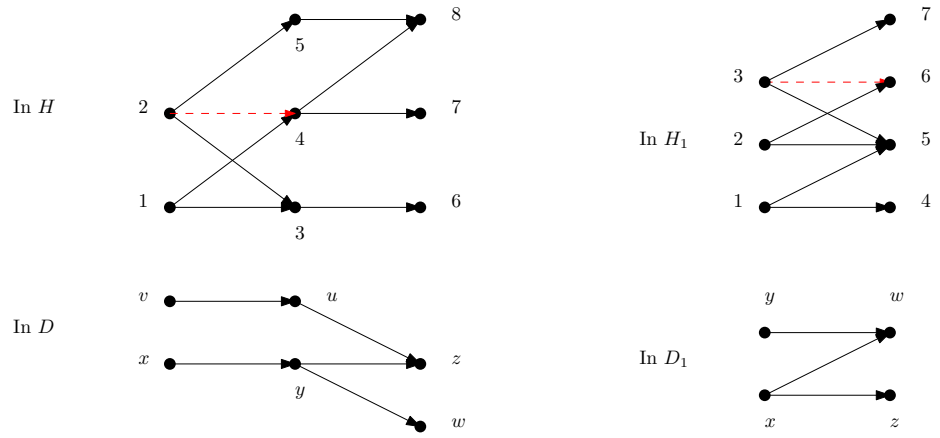


Figure 6.1: Two examples for Algorithm 3.

is an arc there is no need to shift the image of y. In the left example, the target digraph is H and the input is D. In H, $1, 2, 3, 4, 5, 6, 7, 8$ is a min-ordering and 24 is a missing arc. Suppose x is mapped to 2, y to 4, w to 7, z to 8, u to 5 and v to 2. Then we should shift the image of y to 3 and then w to 6 and z to 6 and then u to 3 and v to one of the $1, 2$.

6.5.1 Analyzing the approximation Ratio

We now claim that, the cost of this homomorphism is at most $|V(H)|^2$ times the minimum cost of a homomorphism. Let w denote the value of the objective function with the fractional optimum u_i, v_j, and w' denote the value of the objective function with the final values u'_i, v'_j, after the rounding and all the shifting. Also, let w^* be the minimum cost of a homomorphism of D to H. Obviously, $w \leq w^* \leq w'$.

We now show that the expected value of w' is at most a constant times w. Let us focus on the contribution of one summand, say $v'_t - v'_{t+1}$, to the calculation of the cost.

In any integer solution, $v'_t - v'_{t+1}$ is either 0 or 1. The probability that $v'_t - v'_{t+1}$ contributes to w' is the probability of the event that $v'_t = 1$ and $v'_{t+1} = 0$. This can happen in the following situations:

1. v is mapped to a_t by rounding, and is not shifted away. In other words, we have $v'_t = 1$ and $v'_{t+1} = 0$ after rounding, and these values do not change by procedure Shift.

2. v is first mapped to some $a_j, j > t$, by rounding, and then re-mapped to a_t by procedure Shift.

Lemma 6.5.5. *The expected contribution of one summand, say $v'_t - v'_{t+1}$, to the expected cost of w' is at most $|V(H)|^2 c(v, a_t)(v_t - v_{t+1})$.*

Proof. Vertex v is mapped to a_t in two cases. The first case is where v is mapped to a_t by rounding Line 5, and is not shifted away. In other words, we have $v'_t = 1$ and $v'_{t+1} = 0$ after

rounding, and these values do not change by procedure SHIFT. Hence, for this case we have:

$$\begin{aligned}\mathbb{P}[f(v) = a_t] &= \mathbb{P}[v_{t+1} < X \leq v_t] \cdot \mathbb{P}[v \text{ is not shifted in procedure SHIFT}] \\ &\leq v_t - v_{t+1}\end{aligned}$$

Whence this situation occurs with probability at most $v_t - v_{t+1}$, and the expected contribution is at most $c(v, a_t)(v_t - v_{t+1})$.

Second case is where $f(v)$ is set to a_t during procedure SHIFT. The algorithm calls SHIFT if there exists $u^0u^1 \in A(D)$ such that $f(u^0)f(u^1) \in E(H') \setminus E(H)$ (Line 8). Let us assume it calls SHIFT(f, u^1). Procedure SHIFT modifies images of vertices $u^1, u^2, \cdots$. Consider the last time that SHIFT changes image of v. Note that $u^1, \cdots, u^k = v$ is an oriented walk, meaning that there is an arc between every two consecutive vertices of the sequence and the u^is are not necessarily distinct.

We first compute the contribution for a fixed j, that is the contribution of shifting v from a fixed a_j to a_t. We use induction on k. Consider the simplest case where $k = 1$. In this case v is first mapped to $a_j, j > t$, by rounding, and then re-mapped to a_t during procedure SHIFT. This happens if there exist i and u such that uv is an arc of D mapped to $a_ia_j \in E'$, and then the image of v is shifted to a_t ($a_t < a_j$ in the min-ordering), where $a_ia_t \in E = A(H)$. In other words, we have $u'_i = v'_j = 1$ and $u'_{i+1} = v'_{j+1} = 0$ after rounding (Line 5); and then v is shifted from a_j to a_t. Therefore,

$$\begin{aligned}\mathbb{P}[u'_i = v'_j = 1, u'_{i+1} = v'_{j+1} = 0] &= \mathbb{P}[\max\{u_{i+1}, v_{j+1}\} < X \leq \min\{u_i, v_j\}] \\ &= \min\{u_i, v_j\} - \max\{u_{i+1}, v_{j+1}\} \\ &\leq v_j - v_{j+1} \\ &\leq \sum_{\substack{t<j \\ a_ia_t \in E \\ a_t \in L(v)}} (v_t - v_{t+1}) = P_v\end{aligned}$$

The last inequality is because a_j has no in-neighbor after a_i and it follows from inequality (C9). Having uv mapped to a_ia_j in the rounding step, we shift v to a_t with probability $P_{v,t} = \frac{(v_t - v_{t+1})}{P_v}$. Note that the upper bound P_v is independent from the choice of u and a_i. Therefore, for a fixed a_j, the probability that v is shifted from a_j to a_t is at most $\frac{v_t - v_{t+1}}{P_v} \cdot P_v = v_t - v_{t+1}$.

For $k > 1$, consider oriented walk $u^0, \cdots, u^k = v$. Before calling SHIFT(f, u^1), this walk is mapped to some vertices in H. Without loss of generality, let us assume these vertices are $a_0, a_1, \cdots, a_k$. Note that a_is may not be distinct. Once again we compute the contribution for a fixed $k = j$, that is the contribution of shifting v from a fixed $a_k = a_j$ to a_t. First, we

give an upper bound on the probability of existence of such a situation after rounding step (Line 5),

$$\begin{aligned}
&\mathbb{P}[u_0^{0\prime} = \cdots = u_k^{k\prime} = 1, u_1^{0\prime} = \cdots = u_{k+1}^{k}{}' = 0] \\
&= \mathbb{P}[\max\{u_1^0, \cdots, u_{k+1}^k\} < X \leq \min\{u_0^0, \cdots, u_k^k\}] \\
&= \min\{u_0^0, \cdots, u_k^k\} - \max\{u_1^0, \cdots, u_{k+1}^k\} \\
&\leq (u_k^k) - u_{k+1}^k \\
&= v_j - v_{j+1} \\
&\leq \sum_{\substack{t<k-1 \\ a_{k-1}a_t \in A(H) \\ a_t \in L(u^{k-1})}} (u_t^{k-1} - u_{t+1}^{k-1}) \\
&= P_v
\end{aligned}$$

Now the algorithm calls SHIFT(f, u^1) and, in procedure SHIFT, images of $u^1, u^2, \cdots, u^k = v$ are changed in this order. We are interested in probability of mapping v from fixed $a_k = a_j$ to a_t. Analyzing the situation for u^1 is the same as the case for $k = 2$. As induction hypothesis, assume for $u^1, \cdots, u^{k-1}$, the probability that the algorithm shifts image of u^i to some a_i is at most $u_i^i - u_{i+1}^i$, particularly for $u^{k-1} = u$. At this point $f(u) = a_i$ and $f(v) = a_k$. Note that $a_i a_k$ is not an edge in H, as otherwise no change is required for image of v. Here, the algorithm chooses a_t where $a_t \in L(v), a_t < a_k$ and $a_i a_t \in E(H)$ with probability

$$\frac{v_t - v_{t+1}}{\sum\limits_{\substack{j<k \\ a_i a_j \in A(H) \\ a_j \in L(v)}} (v_j - v_{j+1})}$$

It remains to argue that

$$u_i - u_{i+1} \leq \sum_{\substack{j<k \\ a_i a_j \in A(H) \\ a_j \in L(v)}} (v_j - v_{j+1}).$$

Having that gives us the probability of shifting v from a_j to a_t is at most $v_t - v_{t+1}$.

Observe that a_i does not have any neighbor a_s after a_k. This is because $a_{k-1}a_k$, $a_i a_s \in A(H')$ and the min-ordering implies $a_i a_k \in A(H)$ which contradicts our assumption. Thus,

by inequality (C11), we get

$$u_i - u_{i+1} \leq \sum_{\substack{j<k \\ a_i a_j \in A(H) \\ a_j \in L(v)}} (v_j - v_{j+1})$$

This completes this part of the proof.

Let $L(v) = \{a_1^v \cdots, a_k^v\}$. Clearly, during procedure SHIFT, image of v can be shifted to a_i^v from any of vertices $a_{i+1}^v, \cdots, a_k^v$. For any fixed $a_j \in \{a_{i+1}^v, \cdots, a_k^v\}$, this shift is initiated from vertices in $V(H)$ that are incident with some edges in E', and reaches to a_j to shift image of v. Shifting of image of v happens because of missing edges from a_j that is at most $|V(H)| - d^+(a_j) - d^-(a_j) \leq |V(H)|$ ($d^+(a_j)$ and $d^-(a_j)$ are out-degree and in-degree of a_j respectively). Therefore, the contribution of v and a_i^v to the expected value of w' is at most $(1 + |V(H)|(k-i))c(v, a_i^v)(v_{a_i^v} - v_{a_{i+1}^v})$ where $(v_{a_i^v} - v_{a_{i+1}^v})$ is the upper bound on the probability provided before. □

Theorem 6.5.6. *Algorithm3 returns a homomorphism with expected cost $|V(H)|^2$ times the optimal cost. The algorithm can be de-randomized to obtain a deterministic $|V(H)|^2$-approximation algorithm.*

Proof. By Lemma 6.5.5 the expected value of w' is

$$\begin{aligned}
\mathbb{E}[w'] &= \mathbb{E}\left[\sum_{v,i} c(v, a_i)(v_i' - v_{i+1}')\right] \\
&= \sum_{v,i} c(v, a_i)\mathbb{E}[v_i' - v_{i+1}'] \\
&\leq |V(H)|^2 \sum_{v,i} c(v, a_i)(v_i - v_{i+1}) \\
&\leq |V(H)|^2 w \\
&\leq |V(H)|^2 w^*.
\end{aligned}$$

At this point we have proved that Algorithm 3 produces a homomorphism whose expected cost is at most $|V(H)|^2$ times the minimum cost. It can be transformed to a deterministic algorithm as follows. There are only polynomially many values v_t (at most $|V(D)| \cdot |V(H)|$). When X lies anywhere between two such consecutive values, all computations will remain the same. Thus we can de-randomize the first phase by trying all these values of X and choosing the best solution. Similarly, there are only polynomially many values of the partial sums $\sum_{i=1}^{p} P_{u,t_i}$ (again at most $|V(D)| \cdot |V(H)|$), and when Y lies between two such consecutive values, all computations remain the same. Thus we can also de-randomize the second phase by trying all possible values and choosing the best. Since the expected value is at most $|V(H)|^2$ times the minimum cost, this bound also applies to this best solution. □

6.6 Approximation for digraphs with a k-min-ordering

Digraphs admitting k-min-ordering ($k > 1$) do not admit a min-ordering or a conservative majority polymorphism. However, this does not rule out the possibility of a constant factor approximation algorithm. We show that they are in fact DAT-free digraphs and the List Homomorphism problem is polynomial time solvable for this class of digraphs.

It turns out that digraphs admitting a k-min-ordering do not contain a DAT. Furthermore, List Homomorphism problem is polynomial time solvable for this class of digraphs (Lemmas 6.6.2 and 6.6.1).

In the rest of this section $\overrightarrow{C_k}$ denotes an induced directed cycle with vertices $\{0, 1, \ldots, k-1\}$ and the arc set $\{01, 12, \ldots, (k-2)(k-1), (k-1)0\}$.

Lemma 6.6.1. *Let H be a digraph that admits a k-min-ordering. Then LHOM(H) is polynomial time solvable.*

Proof. Let D, H, L be an instance of LHOM(H) where D is the input digraph and L is the set of lists, i.e. for every $x \in V(D)$, $L(x) \subseteq V(H)$. We run the arc consistency procedure and suppose the lists are not empty after the arc consistency procedure. Let $V_0, V_1, \ldots, V_{k-1}$ be the sets of vertices of H according to the k-min-ordering $<$. We also note that if there exists a homomorphism $\phi : V(D) \to V(H)$, then D must be homomorphic to $\overrightarrow{C_k}$ because H is homomorphic to $\overrightarrow{C_k}$. This means the vertices of D are partitioned into $D_0, D_1, \ldots, D_{k-1}$ where the arcs of D go from some D_i to D_{i+1}, $0 \leq i \leq k-1$ (sum modulo k). For simplicity we may assume that D is weakly connected; i.e. the underlying graph of D is connected. Moreover, without loss of generality let x be an arbitrary vertex in D_0 (D_0 is not empty). Now the vertices of D_0 are mapped to some V_ℓ, for some $0 \leq \ell \leq k-1$. In other words, $L(x) \cap V_\ell \neq \emptyset$.

Now for every $y \in D_{j+\ell}$ and every $0 \leq j \leq k$, set $f(y)$ to be the smallest element in $L(y) \cap V_{j+\ell}$ according to $<$. Observe that the restriction of $<$ on $V_i \cup V_{i+1}$, $0 \leq i \leq k-1$, is a min-ordering. Suppose yz is an arc of D with $y \in D_{j+\ell}$ and $z \in D_{j+\ell+1}$. We show that $f(y)f(z)$ is an arc of H. Suppose $f(y) = a$ and $f(z) = b$. Since we run the arc-consistency procedure, there exists some element $b' \in L(z) \cap V_{\ell+j+1}$ such that $ab' \in A(H)$, and there exists some $a' \in L(y) \cap V_{\ell+j+1}$ so that $a'b \in A(H)$. The ordering $<$ on $V_{\ell+j} \cup V_{\ell+j+1}$ is a min-ordering, and hence, ab is an arc of H. □

Lemma 6.6.2. *Let H be a digraph that admits a k-min-ordering. Then H does not contain a DAT.*

Proof. It was shown in [108] that digraph H_1 is DAT-free if and only if $V(H_1) \times V(H_1)$ can be partitioned into two sets V_f, V_g where there exist two polymorphisms f, g over H_1 such that f is a semilattice on V_f and g is a majority over V_g. Let $V_0, V_1, \ldots, V_{k-1}$ be the vertices of H according to the k-min-ordering $<$. Define the binary polymorphism f over H as follows.

1. $f(x,y) = f(y,x) = x$ when $x, y \in V_i$ and $x < y$ (in the ordering $<$),
2. $f(x,y) = x$, $f(y,x) = y$ when $x \in V_i$ and $y \in V_j$, $0 \leq i \neq j \leq k-1$,
3. $f(x,x) = x$ for every $x \in V(H)$.

First we show that f is a polymorphism on H and it is semilattice on $V_f = \{(x,y) \mid x, y \in V_i, 0 \leq i \leq k-1\}$. Let $xy, x'y' \in A(H)$ where $x, y \in V_i$ and $x', y' \in V_{i+1}$. Now $f(x,y)f(x',y') \in A(H)$ because between V_i, V_{i+1} we have a min-ordering, implying that f is a polymorphism. It is also easy (since $<$ is min-ordering) to see that f is associative. Now, define the ternary polymorphism g over H as follows :

1. $g(x,y,z) = x$ when $x, y, z \in V_i$,
2. $g(x,y,z) = x$ when $x \in V_i$, $y \in V_j$, $z \in V_\ell$ and i, j, ℓ are all distinct,
3. $g(x,y,z) = g(z,x,y) = g(x,z,y) = g(z,y,x) = g(y,z,x) = g(y,x,z) = x$ when $x, y \in V_i$, $x < y$ (in the ordering $<$), and $z \in V_j$, $i \neq j$,
4. $g(x,y,z) = g(z,x,y) = g(x,z,y) = g(z,y,x) = g(y,z,x) = g(y,x,z) = y$ when $x, y \in V_i$, $y < x$, and $z \in V_j$, $i \neq j$,
5. $g(x,x,y) = g(x,y,x) = g(y,x,x) = x$ when $x \in V_i$ and $y \in V_j$, $i \neq j$,
6. $g(x,x,x) = x$ for all $x \in V(H)$.

We show that g is a polymorphism over H, and show that it is a majority polymorphism over the pairs in $V_g = \{(x,y) \mid x \in V_i, y \in V_j, i \neq j\}$. By definition, we need to show that

$$\forall xx', yy', zz' \in A(H) \implies g(x,y,z)g(x',y',z') \in A(H)$$

Case 1. If x, y, z all belong to the same V_i, then $x', y', z' \in V_{i+1}$, and hence, by definition

$$g(x,y,z)g(x',y',z') = xx' \in A(H).$$

Case 2. If x, y, z belong to three different partite sets, then x', y', z' belong to three distinct partite sets, and hence,

$$g(x,y,z)g(x',y',z') = xx' \in A(H).$$

Case 3. If x, y belong to V_i (possibly $x = y$) and $z \in V_j$, then $x', y' \in V_{i+1}$ and $z' \in V_{j+1}$. When $x < y$ and $x' < y'$, then by definition $g(x,y,z)g(x',y',z') \in A(H)$. Now suppose that $x < y$ and $y' < x'$. Since $<$ is a min-ordering on V_i, V_{i+1}, we have $xy' \in A(H)$, and hence,

$$g(x,y,z)g(x',y',z') = xy' \in A(H).$$

By symmetry, the other remaining cases can be handled similarly. □

Theorem 6.6.3. *There is a $|V(H)|^2$-approximation algorithm for MinHOM(H) when the target digraph H admits a k-min-ordering, $k > 1$.*

Proof. Let $V_0, V_1, \ldots, V_{k-1}$ be a partition of the vertices of H according to the k-min-ordering; i.e. every arc of H is from a vertex in V_l to a vertex in V_{l+1}, $0 \leq k-1$ (sum module k). Clearly a mapping $g : V(H) \to \overrightarrow{C_k}$ with $g(a) = l$ when $a \in V_l$, $l \in \{0, 1, \ldots, k-1\}$, is a homomorphism from H to $\overrightarrow{C_k}$

Let D be the input digraph together with the costs. Observe that if D is homomorphic to H, then D must be homomorphic to $\overrightarrow{C_k}$. We may assume that D is weakly connected. Otherwise, each weakly connected component of D is treated separately.

Let x be a fixed vertex in D and let ψ_ℓ be a homomorphism from D to $\overrightarrow{C_k}$ where $\psi_\ell(x) = \ell$, $\ell \in \{0, 1, \ldots, k-1\}$. We design an approximation algorithm for MinHOM(H) in which x is mapped to V_i of H. In order to find the approximation algorithm for MinHOM(H) for the given digraph D, we consider each homomorphism $\psi_\ell(x) = \ell$, $\ell \in \{0, 1, \ldots, k-1\}$ and find an approximation algorithm from D to H corresponding to ψ_ℓ and output the one with best performance. For simplicity of notations we work with $\phi = \psi_0$. Let $U_0, U_1, U_2, \ldots, U_{k-1}$ be the partition of the vertices in D under ϕ, i.e. $\phi^{-1}(\ell) = U_\ell$.

Consider the following LP with set of constraint called $\mathcal{S}$. For every $u \in U_\ell$ and every $a_i \in V_\ell$, $\ell \in \{0, 1, \ldots, k\}$ define variable $0 \leq u_i \leq 1$. For every vertex $a_i \in V_j$, $j \in \{0, 1, \ldots, k-1\}$, let $\ell^+(i)$ be the first $b_{i'} \in U_{j+1}$ in the ordering $<$ such that $a_i b_{i'} \in A(H)$ and let $\ell^{-1}(i)$ be the first $c_r \in V_{j-1}$ in the ordering $<$ such that $c_r a_i \in A(H)$.

`min`	$\sum_{\substack{\ell \in \{0,1,\ldots,k-1\} \\ v \in U_\ell, i \in V_\ell}} c(v, a_i)(v_i - v_{i+1})$		
`subject to:`	$v_i \geq 0$		(A1)
	$v_1 = 1$	$\forall \ell$ and every $v \in U_\ell, a_i \in V_\ell$	(A2)
	$v_{p+1} = 0$	$\lvert V(H)\rvert = p$	(A3)
	$v_{i+1} \leq v_i$	$\forall \ell$ and every $v \in U_\ell, a_i \in V_\ell$	(A4)
	$v_{i+1} = v_i$	if $a_i \notin L(v)$	(A5)
	$u_i \leq v_{l^+(i)}$	$\forall\ uv \in A(D)$	(A6)
	$v_i \leq u_{l^-(i)}$	$\forall\ uv \in A(D)$	(A7)

Let $a_1, a_2, \ldots, a_p$ be the vertices in V_ℓ according to the k-min-ordering $<$, and let $b_1, b_2, \ldots, b_q$ be the vertices in $V_{\ell+1}$ according to $<$.

Let $E = A(H)$ and define H' to be the digraph with vertex set $V(H)$ and arc set $E \cup E'$. Here E' is the set of arcs added into $A(H)$ so that the resulting digraph admit a k-min-max-ordering. Note that E and E' are disjoint sets. Let E'_ℓ denote the set of all the pairs $(a_i, b_j) \in V_\ell \times V_{\ell+1}$ such that $a_i b_j$ is not an arc of H, but there is an arc $a_i b_{j'}$ of H with $j' < j$ and an arc $a_{i'} b_j$ of H with $i' < i$. Observe that $E' = \bigcup_{\ell=0}^{\ell=k-1} E'_\ell$.

For every arc $e = a_i a_j \in E'_\ell$ and every arc $uv \in A(D)$, $u \in U_\ell, v \in U_{\ell+1}$ two of the following set of inequalities is added to $\mathcal{S}$ (i.e. either (A8), (A9) or (A10), (A11)).

$v_j \leq u_s$	$+$	$\sum_{\substack{a_t \in L(u) \\ a_t b_j \in E_\ell \\ t<i}} (u_t - u_{t+1})$	if a_s is the first in-neighbor of b_j after a_i	(A8)
$v_j \leq v_{j+1}$	$+$	$\sum_{\substack{a_t \in L(u) \\ a_t b_j \in E_\ell \\ t<i}} (u_t - u_{t+1})$	if b_j has no in-neighbor after a_i	(A9)
$u_i \leq v_s$	$+$	$\sum_{\substack{b_t \in L(v) \\ a_i b_t \in E_\ell \\ t<j}} (v_t - v_{t+1})$	if b_s is the first out-neighbor of a_i after b_j	(A10)
$u_i \leq u_{i+1}$	$+$	$\sum_{\substack{b_t \in L(v) \\ a_i b_t \in E_\ell \\ t<j}} (v_t - v_{t+1})$	if a_i has no out-neighbor after b_j	(A11)

Moreover, by pair consistency, we can add the following constraints for every $u \in U_\ell$ and every $v \in U_{\ell'}$ in $V(D)$ and $a_i \in L(u)$:

$$u_i - u_{i+1} \leq \sum_{\substack{j: \\ (a_i, b_j) \in L(u,v)}} (v_j - v_{j+1}) \quad \text{(A12)}$$

By similar argument as in the previous section, one can show the following. There is a one-to-one correspondence between the homomorphisms from D to H, and integer solutions of the extended system $\mathcal{S}$.

In what follows we outline the process of rounding the fractional values of the LP to obtain an integral solution, and hence, a homomorphism from D to H (see fig. 6.2). In the first stage of the algorithm, we use a random variable $X \in [0, 1]$ and round the fractional values according to X. This means, if $u_i < X$ then u'_i is set to zero, otherwise we set $u'_i = 1$.

The intention is to map v to vertex a_i of H when $u'_i = 1$ and $u'_{i+1} = 0$. However, we may set $u'_i = v'_j = 1$, $u'_{i+1} = v'_{j+1} = 0$ where $u \in U_\ell, v \in U_{\ell+1}$, $a_i \in V_\ell$, $b_j \in V_\ell + 1$ and $a_i b_j \in E'_\ell$, i.e. $a_i b_j$ is not an arc of H but it is one of the added arcs into H. In other words, what we have obtained would not be a homomorphism, and hence, we have to fix this partial integral assignment. To keep track of fixings, we may assume sum $i + j$ is maximum.

We may assume that b_j does not have any in-neighbor in V_ℓ after a_i. Now we use a random variable $Y \in [0, 1]$ to select an out-neighbor $b_t \in V_{\ell+1}$ of a_i before (in the ordering $<$) b_j and shift the image of v from b_j to b_t. The vertex b_t is selected according to random variable Y with the same rule as the one described in Section 6.5 (see the description after Lemma 6.5.1). However, this could force us to shift the image of some out-neighbor of v, say $w \in V_{\ell+2}$ (subscript in modulo k). Therefore, we deploy a BFS search (applying a version of shift procedure in Algorithm 3) to fix the images of the vertices of D that may need to be changed because of the initial change in shifting the image of v to b_t (see the Figure

6.2). We use the same strategy as used in the case of the min-ordering to round the values of $\mathcal{S}$ and obtain an integral solution. □

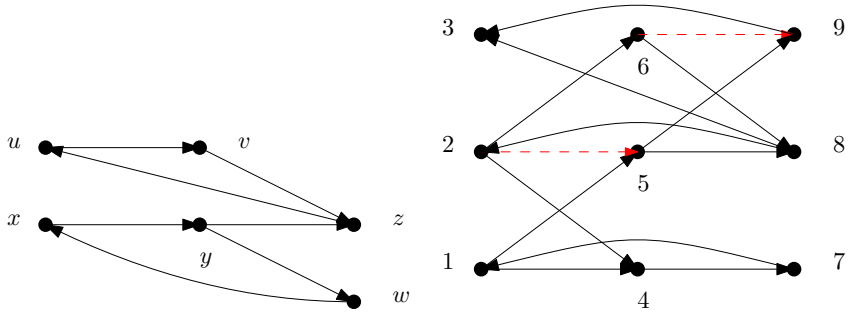


Figure 6.2: An illustration of the algorithm for k-min-ordering.

Example 6.6.4 (Figure 6.2: an illustration of the algorithm for k-min-ordering)**.** In digraph H, $1, 2, 3, 4, 5, 6, 7, 8, 9$ is a 3-min-ordering. The dash arcs are the missing arcs. Suppose after the first step of rounding $u'_2 = v'_5 = z'_7 = 1$, $v'_6 = u'_3 = z'_9 = 0$. Then the algorithm shift the image of v from 5 to 4 and consequently the image of z from 8 to 7.

6.7 A dichotomy for graphs

Feder and Vardi [78] proved that if a graph H admits a conservative majority polymorphism, then LHOM(H) is polynomial time solvable. Later, Feder *et al.,* [76] showed that LHOM(H) is polynomial time solvable if and only if H is a *bi-arc* graph. Bi-arc graphs are defined as follows.

Let C be a circle with two specified points p and q on C. A bi-arc is an ordered pair of arcs (N, S) on C such that N contains p but not q, and S contains q but not p. A graph H is a bi-arc graph if there is a family of bi-arcs $\{(N_x, S_x) : x \in V(H)\}$ such that, for any $x, y \in V(H)$, not necessarily distinct, the following hold:

- if x and y are adjacent, then neither N_x intersects S_y nor N_y intersects S_x;
- if x and y are not adjacent, then both N_x intersects S_y and N_y intersects S_x.

We shall refer to $\{(N_x, S_x) : x \in V(H)\}$ as a bi-arc representation of H. Note that a bi-arc representation cannot contain bi-arcs $(N, S), (N', S')$ such that N intersects S' but S does not intersect N' and vice versa. Furthermore, by the above definition a vertex may have a self loop.

Theorem 6.7.1 ([29, 76])**.** *A graph admits a conservative majority polymorphism if and only if it is a bi-arc graph.*

Definition 6.7.2 (G^*)**.** *Let $G = (V, E)$ be a graph. Let G^* be a bipartite graph with partite sets V, V' where V' is a copy of V. Two vertices $u \in V$, and $v' \in V'$ of G^* are adjacent in G^* if and only if uv is an edge of G.*

A *circular arc* graph is a graph that is the intersection graph of a family of arcs on a circle. We interpret the concept of an intersection graph literally, thus any intersection graph is automatically reflexive (i.e. there is a loop at every vertex), since a set always intersects itself. A bipartite graph whose complement is a circular arc graph, is called a *co-circular arc* graph. Note that co-circular arc graphs are irreflexive, meaning no vertex has a loop.

Lemma 6.7.3. *Let H^* be the bipartite graph constructed from a bi-arc graph H, according to Definition 6.7.2. Then the following hold.*

- *H^* is a co-circular arc graph.*
- *H^* admits a min-ordering.*

Proof. It is easy to see that H^* is a co-circular arc graph. From a bi-arc representation $\{(N_i, S_i) : i \in V(H)\}$ of H, we obtain a co-circular arc representation of H^* by choosing the arc N_i, $i \in H$ for vertex $i \in H^*$ and arc S_i for vertex $i' \in H^*$. A bipartite graph admits a min-ordering if and only if it is co-circular arc graph [103]. H^* is a co-circular arc graph, and hence, it admits a min-ordering. □

Let H be a bi-arc graph, with vertex set I, and let $H^* = (I, I')$ be the bipartite graph constructed from H according to Definition 6.7.2. Let $a_1, a_2, \ldots, a_p$ be an ordering of the vertices in I and $b_1, b_2, \ldots, b_p$ be an ordering of the vertices of I'. Note that each a_i has a copy $b_{\pi(i)}$ in $\{b_1, b_2, \ldots, b_n\}$ where π is a permutation on $\{1, 2, 3, \ldots, p\}$. By Lemma 6.7.3, let us assume $a_1, a_2, \ldots, a_p, b_1, b_2, \ldots, b_p$ is a min-ordering for H^*.

Let G be the input graph with vertex set V and a cost function c. Construct G^* from G with vertex set $V \cup V'$ as in Definition 6.7.2. Now construct an instance of the MinHOM(H^*) for the input graph G^* and set $c(v', b_{\pi(i)}) = c(v, a_i)$ for $v \in V$, $v' \in V'$. Further, make H^* a digraph by orienting all its edges from I to I', and similarly make G^* a digraph by orienting all its edges from V to V'. The following lemma immediately follows from the construction of H^* and G^*.

Lemma 6.7.4. *There exists a homomorphism $f : G \to H$ with cost $\mathfrak{C}$ if and only if there exists homomorphism $f^* : G^* \to H^*$ with cost $2\mathfrak{C}$ such that, if $f^*(v) = a_i$ then $f^*(v') = b_j$ with $j = \pi(i)$.*

We first perform the arc-consistency and pair-consistency procedures for the vertices in G^*. Note that if $L(u)$ contains element a_i then $L(u')$ contains $b_{\pi(i)}$ and when $L(u')$ contains some b_j then $L(u)$ contains $a_{\pi^{-1}(j)}$. Next, we define the system of linear equations $\widehat{S^*}$ with

the same construction as in Sections 6.3, 6.4. Equivalently, one can use the LP formulation in [103]. However, for the sake of completeness we present the entire LP in this section.

Consider the following linear program. For every vertex $v \in V$ from $G^* = (V, V')$ and every vertex $a_i \in I$ from $H^* = (I, I')$ define variable v_i. For every vertex $v' \in V'$ from G^* and every vertex $b_i \in I'$ from H^* define variable v'_i. We also define variable v_{p+1} for every $v \in V$ whose value is set to zero. Now the goal is to minimize the following objective function:

`min`	$\sum_{v,i} c(v, a_i)(v_i - v_{i+1}) + \sum_{v',j} c(v', b_j)(v'_j - v'_{j+1})$			
`subject to:`	$v_i, v'_{\pi(i)} \geq 0$			(CM1)
	$v_1 = v'_1 = 1$			(CM2)
	$v_{p+1} = 0$			(CM3)
	$v_{i+1} \leq v_i$	and $v'_{\pi(i)+1} \leq v'_{\pi(i)}$		(CM4)
	$v_{i+1} = v_i$	and $v'_{\pi(i)+1} = v'_{\pi(i)}$	if $a_i \notin L(v)$	(CM5)
	$u_i \leq v'_{l^+(i)}$		$\forall\ uv' \in A(G^*)$	(CM6)
	$v'_i \leq u_{l^-(i)}$		$\forall\ uv' \in A(G^*)$	(CM7)
	$u_i - u_{i+1} = u'_{\pi(i)} - u'_{\pi(i)+1}$		$\forall u, u' \in G^*, \forall a_i, b_{\pi(i)} \in H^*$	(CM8)

Let E' denote the set of all the pairs (a_i, b_j) such that $a_i b_j$ is not an arc of H^*, but there is an arc $a_i b_{j'}$ of H^* with $j' < j$ and an arc $a_{i'} b_j$ of H^* with $i' < i$. Let $E = A(H^*)$ and define H'^* to be the digraph with vertex set $V(H^*)$ and arc set $E \cup E'$. Note that E and E' are disjoint sets. For every arc $e = a_i b_j \in E'$ and every arc $uv \in A(G^*)$, by Observation 6.4.3, two of the following set of inequalities will be added to $\widehat{S^*}$ (i.e. either (CM9), (CM12) or (CM10), (CM11)).

$v'_j \leq u_s$	$+ \sum_{\substack{a_t \in L(u) \\ a_t b_j \in E \\ t<i}} (u_t - u_{t+1})$		if a_s is the first in-neighbor of b_j after a_i	(CM9)
$v'_j \leq v'_{j+1}$	$+ \sum_{\substack{a_t \in L(u) \\ a_t b_j \in E \\ t<i}} (u_t - u_{t+1})$		if b_j has no in-neighbor after a_i	(CM10)
$u_i \leq v'_s$	$+ \sum_{\substack{b_t \in L(v') \\ a_i b_t \in E \\ t<j}} (v'_t - v'_{t+1})$		if b_s is the first out-neighbor of a_i after b_j	(CM11)
$u_i \leq u_{i+1}$	$+ \sum_{\substack{b_t \in L(v') \\ a_i b_t \in E \\ t<j}} (v'_t - v'_{t+1})$		if a_i has no out-neighbor after b_j	(CM12)

Lemma 6.7.5. *If H is a bi-arc graph, then there is a one-to-one correspondence between homomorphisms from G to H and integer solutions of $\widehat{S^*}$.*

Proof. For homomorphism $f : G \to H$, if $f(v) = a_t$ we set $v_i = 1$ for all $i \leq t$, otherwise we set $v_i = 0$, we also set $v'_j = 1$ for all $j \leq \pi(i)$ and $v'_{j+1} = 0$ where $\pi(i) = j$. We set $v_1 = 1$,

$v'_1 = 1$ and $v_{p+1} = v'_{p+1} = 0$ for all $v, v' \in V(G^*)$. Now all the variables are non-negative and we have $v_{i+1} \leq v_i$ and $v'_{j+1} \leq v'_j$. Note that by this assignment constraint (CM12) is satisfied. It remains to show that $u_i \leq v'_{l^+(i)}$ for every arc $uv' \in A(G^)*$ Suppose for contradiction that $u_i = 1$ and $v'_{l^+(i)} = 0$ and let $f(u) = a_r$ and $f(v) = a_s$. This implies that $u_r = 1$, whence $i \leq r$; and $v'_s = 1$, whence $s < l^+(i)$. Since $a_i b_{l^+(i)}$ and $a_r b_s$ both are arcs of H^* with $i \leq r$ and $s < l^+(i)$, the fact that H^* has a min-ordering implies that $a_i b_s$ must also be an arc of H^*, contradicting the definition of $l^+(i)$. The proof for $v'_j \leq u_{l^-(i)}$ is analogous.

Conversely, suppose there is an integer solution for $\widehat{S^*}$. First we define a homomorphism $g : G^* \to H^*$ as follows : let $g(u) = a_i$ where i is the largest subscript with $v_i = 1$, and $g(v') = b_j$ when j is the largest subscript with $v_j = 1$. We prove that this is indeed a homomorphism by showing that every arc of G^* is mapped to an arc of H^*. Let uv' be an arc of G^* and assume $g(u) = a_r$, $g(v') = b_s$ We show that $a_r b_s$ is an arc in H^*. Observe that, by (CM6) and (CM7), $1 = u_r \leq v'_{l^+(r)} \leq 1$ and $1 = v'_s \leq u_{l^-(s)} \leq 1$, therefore we must have $v'_{l^+(r)} = u_{l^-(s)} = 1$. Since r and s are the largest subscripts such that $u_r = v'_s = 1$ then $l^+(r) \leq s$ and $l^-(s) \leq r$. Since $a_r b_{l^+(r)}$ and $a_{l^-(s)} b_s$ are arcs of H^*, we must have the arc $a_r b_s$ in H^* because H^* admits a min-ordering. Furthermore, $g(u) = a_i$ if and only if $u_i = 1$ and $u_{i+1} = 0$, so, $c(u, a_i)$ contributes to the sum if and only if $g(u) = a_i$ and $c(v', b_j)$ contributes to the sum if and only if $g(v') = b_j$.

Now let $f(u) = a_i$ when $g(u) = a_i$. We show that if uv is an esge of G then $f(u)f(v)$ is an edge of H. Since g is a homomorphism from G^* to H^*, $g(u)g(v') \in A(H^*)$. Suppose $g(v') = b_j$. This means $u_i = v'_j = 1$, $u_{i+1} = v'_{j+1} = 0$. Now by constraint (CM12), we have $v_{\pi^{-1}(j)} = 1$, and $v_{\pi^{-1}(j)+1} = 0$, and hence, we have $f(v) = a_{\pi^{-1}(j)}$. Now by definition of H^*, $a_i a_{\pi^{-1}(j)}$ is an arc of H because $a_i b_j$ is an arc of H^*. Furthermore, $f(u) = a_i$ if and only if $u_i = 1$ and $u_{i+1} = 0$, so, $c(u, a_i)$ contributes to the sum if and only if $f(u) = a_i$. □

Once again we round an optimal fractional solution of $\widehat{S^*}$, using random variable $X \in [0, 1]$. Let $\mathcal{F}$ be a mapping form $V(G^*)$ to $V(H^*)$ obtained after rounding. We propose an algorithm that modifies $\mathcal{F}$ and achieves a homomorphism $f : G \to H$ (i.e. an integral solution that satisfies $\widehat{S^*}$).

Theorem 6.7.6. *There exists a randomized algorithm that modifies $\mathcal{F}$ and obtain a homomorphism $f : G \to H$. Moreover, the expected cost of the homomorphism returned by this algorithm is at most $2|V(H)| \cdot OPT$.*

Proof. For every variable u_i, $u \in V(G^*)$, set $\hat{u}_i = 1$ if $X \leq u_i$ else $\hat{u}_i = 0$. Similarly for every v'_j, $v' \in V(G^*)$, set $\hat{v}'_j = 1$ if $X \leq v'_j$ else $\hat{v}'_i = 0$. The algorithm has two stages after rounding the variables using random variable X.

Stage 1. Fixing the arcs uv' of G^* that have been mapped to non-arcs $a_i b_j$ of H^*: Suppose for some arc uv' of G^*, $\hat{u}_i = 1$, $\hat{u}_{i+1} = 0$, $\hat{v}'_j = 1$, $\hat{v}'_{j+1} = 0$. By Observation 6.4.3,

either b_j has no in-neighbor after a_i or a_i has no out-neighbor after b_j. Suppose the former is the case. We also note that because of the constrains (CM5), (CM6), a_ib_j is one of the arcs that should be added into H^* in order to obtain a min-max-ordering for H'^*. Suppose, for edge $uv' \in A(G^*)$, $\mathcal{F}(u) = a_i, \mathcal{F}(v) = a_j$ where $a_ia_j \in E'$; i.e. $a_ib_{\pi(j)} \notin A(H^*)$. We may assume that a_ia_j is the first such non-edge in H when we look at the min-ordering of H^*.

Choose a random variable $Y \in [0, 1]$, which will guide us to shift the image of v' from b_j to some b_t where $a_ib_t \in E$, and b_t appears before b_j in the min-ordering of H^*. Consider the set of such b_ts (by definition of the min-ordering of H^*, this set is non-empty), and suppose it consists of b_t with subscripts t ordered as $t_1 < t_2 < \ldots t_k$. Let $P_{v',t} = \frac{v'_t - v'_{t+1}}{P_{v'}}$ with $P_{v'} = \sum_{a_ib_t \in E(H^*),\ t<j} (v'_t - v'_{t+1})$. Select b_{t_q} if $\sum_{p=1}^{q} P_{v',t_p} < Y \leq \sum_{p=1}^{q+1} P_{v',t_p}$. Thus, a concrete b_t is selected with probability $P_{v',t}$, which is proportional to the difference of the fractional values $v'_t - v'_{t+1}$. Observe that there is no need to shift the image of some vertex w which is an in-neighbor of v' from its current value to some other vertex (because of shifting the image of v).

Now we note that the probability of shifting the image of some v' from b_j to b_t is at most $v'_t - v'_{t+1}$. Note that as long as such arcs uv' exists, we repeat the shifting procedure. At the end of this stage we have obtained a homomorphism f^* from G^* to H^*.

Stage 2. Making the assignment consistent with respect to both orderings: We say a vertex $u \in V$ of $G^* = (V, V')$ is *unstable* if $\hat{u}_i = 1$, $\hat{u}_{i+1} = 0$, and $\hat{u}'_q = 1$, $\hat{u}'_{q+1} = 0$ with $q \neq \pi(i)$. Now we start a BFS in $V(G^*)$ and continue as long as there exists an unstable vertex u in G^*. We start from the biggest subscripts i for which there exists an unstable u with $\hat{u}_i = 1$, $\hat{u}_{i+1} = 0$. We put all such vertices u with respect to index i in a queue. During the BFS, one of the following is performed:

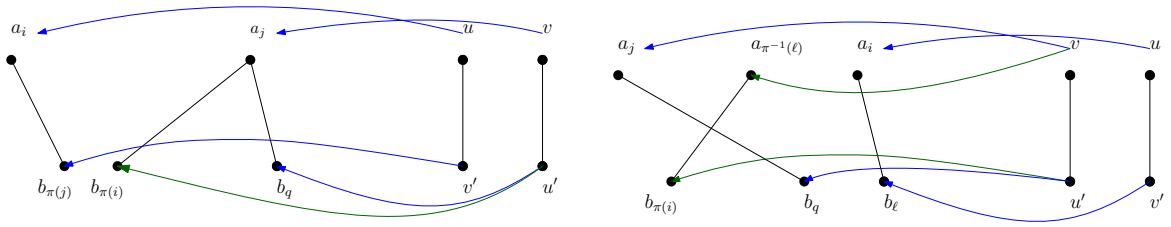
1. shift the image of u' from b_q to $b_{\pi(i)}$.
2. shift the image of u from a_i to $a_{\pi^{-1}(q)}$.

As a consequence of the above actions we would have the following cases:

Case 1. We change the image of u' from b_q to $b_{\pi(i)}$ (with $\hat{u}_i = 1$, $\hat{u}_{i+1} = 0$), and there exists some uv' such that $\hat{v}_j = \hat{v}'_\ell = 1$ and $\hat{v}_{j+1} = \hat{v}'_{\ell+1} = 0$ with $\ell = \pi(j)$.

We note that $a_ib_{\pi(j)}$ is an arc because uv' is an arc, and hence, $a_jb_{\pi(i)}$ is an arc of H^*. This would mean there is no need to shift the image of v from a_j to something else (see the Figure 6.3 (a)).

Case 2. We change the image of u' from b_q to $b_{\pi(i)}$ (with $\hat{u}_i = 1$, $\hat{u}_{i+1} = 0$). Let j be a biggest subscript such that there exists some vu' where $\hat{v}_j = \hat{v}'_\ell = 1$ and $\hat{v}_{j+1} = \hat{v}'_{\ell+1} = 0$ and $\ell \neq \pi(j)$. Note that here $j < i$. Such vertex v is added into the queue, and once



(a) Case 1 (b) Case 2

Figure 6.3: Illustrating the shifting process in Stage 2 of the algorithm.

we retrieve v from the queue we do the following: Moving the image of v from a_j to $a_{\pi^{-1}(\ell)}$ (see the Figure 6.3 (b)).

Note that $a_i b_\ell \in A(H^*)$ because vu' is an arc of G^*, and hence $a_{\pi^{-1}(\ell)} b_{\pi(i)}$ is an arc of H^*, i.e. $a_i a_\ell$ is an edge of H.

Case 3. We change the image of v from a_j to some $a_{\pi^{-1}(\ell)}$ (with $\hat{v}'_\ell = 1, \hat{v}'_{\ell+1} = 0$), and there exists some vw' such that $\hat{w}_t = \hat{w}'_r = 1$ and $\hat{w}_{t+1} = \hat{w}'_{r+1} = 0$ with $r = \pi(t)$. We note that $a_t b_\ell \in A(H^*)$ because $v'w$ is an arc, and hence, $a_{\pi^{-1}(\ell)} b_r$ is an arc of H^*. This would mean there is no need to shift the image of w' to something else.

Case 4. We change the image of v from a_j to some $a_{\pi^{-1}(\ell)}$ (with $\hat{v}'_\ell = 1, \hat{v}'_{\ell+1} = 0$) Let r be a biggest subscripts such that there exists some vw' where $\hat{w}_t = \hat{w}'_r = 1$ and $\hat{w}_{t+1} = \hat{w}'_{r+1} = 0$ with $r \neq \pi(t)$, $t < i$. Such vertex w' is added into the queue, and once we retrieve w' from the queue we do the following: Moving the image of w' from b_r to $b_{\pi^{-1}(t)}$.

Note that $a_t b_\ell \in A(H^*)$ because wv' is an arc of G^*. Therefore, $a_{\pi^{-1}(\ell)} b_{\pi^{-1}(t)}$ is an arc of H^*, i.e. an edge of H.

When Case (2) occurs, we continue the shifting. This would mean we may need to shift the image of some out-neighbor w' of v accordingly. We continue the BFS from v, and modify the images of out-neighbors of v, say w', to be consistent with new image of v. This means we encounter either case 3 or 4. Suppose $\hat{w'}_t = 1$, $\hat{w'}_{t+1} = 0$ or $\hat{w}'_{\pi(t)} = 1$, $\hat{w}_{\pi(t)+1} = 0$. Then there is no need to change the image of w'. Otherwise, we change the image of w' from b_t to b_j where $a_{\pi^{-1}(\ell)} b_j$ is an arc of H^* and we need to consider Cases 3,4 for the current vertex w. When we are in Case 4, then we would consider Cases 1,2 and proceed accordingly.

Note that during the BFS, if we encounter a vertex x (x') that has been visited before, then we would be at Case 1 or 3 and hence, no further action is needed for in-neighbors (out-neighbors) of x. We also note that at each step an unstable vertex y is associated to some a_ℓ where ℓ is decreasing. Therefore, this procedure would eventually stop, and we no longer have an unstable vertex y in V.

Estimating the ratio: Vertex v (v') is mapped to a_t (b_t) in three situations. The first scenario is where v is mapped to a_t by rounding (according to random variable X in Stage 1) and is not shifted away. In other words, we have $\hat{v}_t = 1$ and $\hat{v}_{t+1} = 0$ (i.e. $v_{t+1} \leq X < v_t$) and these values do not change by the shifting procedure. Hence, for this case we have:

$$\begin{aligned}\mathbb{P}[f(v) = a_t] &= \mathbb{P}[v_{t+1} < X \leq v_t] \\ &\leq v_t - v_{t+1}\end{aligned}$$

Whence this situation occurs with probability at most $v_t - v_{t+1}$, and the expected contribution is at most $c(v, a_t)(v_t - v_{t+1})$.
The second scenario is where $f(v)$ is set to a_t according to random variable Y in Stage 1. In this case v is first mapped to $a_j, j > t$, by rounding according to variable X and then re-mapped to a_t during the shifting according to variable Y. We first compute the expected contribution for a fixed j, that is the contribution of shifting v from a fixed a_j to a_t.

This happens if there exist i and $u' \in V(H^*)$ such that vu' is an arc of D^* mapped to $a_j b_i \in E'$, and then the image of v is shifted to a_t ($a_t < a_j$ in the min-ordering), where $a_t b_i \in E = A(H^*)$. In other words, we have $\hat{u}'_i = \hat{v}_j = 1$ and $\hat{u}'_{i+1} = \hat{v}_{j+1} = 0$ after rounding; and then v is shifted from a_j to a_t. Therefore,

$$\begin{aligned}\mathbb{P}[\hat{u}'_i = \hat{v}_j = 1, \hat{u}'_{i+1} = \hat{v}_{j+1} = 0] &= \mathbb{P}[\max\{u'_{i+1}, v_{j+1}\} < X \leq \min\{u'_i, v_j\}] \\ &= \min\{u'_i, v_j\} - \max\{u'_{i+1}, v_{j+1}\} \\ &\leq v_j - v_{j+1} \\ &\leq \sum_{\substack{t<j \\ a_t b_i \in E \\ a_t \in L(v)}} (v_t - v_{t+1}) \\ &= P_v\end{aligned}$$

The last inequality is because a_j has no out-neighbor after b_i and it follows from inequality (CM9). Having vu' mapped to $a_j b_i$ in the rounding step, we shift v to a_t with probability $P_{v,t} = \frac{(v_t - v_{t+1})}{P_v}$. Note that the upper bound P_v is independent from the choice of u and b_i. Therefore, for a fixed a_j, the probability that v is shifted from a_j to a_t is at most $\frac{v_t - v_{t+1}}{P_v} \cdot P_v = v_t - v_{t+1}$. There are at most $|V(H)|$ of such b_i's, (causing the shift to a_j) and hence, the expected contribution of $v_t - v_{t+1}$ to the objective function is at most $|V(H)|c(v,t)(v_t - v_{t+1})$.
The third scenario is when the image of v is shifted from some a_j to a_t in the second Stage of the shifting . More precisely, when one of the actions 1,2 occurs.

This happens because the image of v' has been shifted from b_q to $b_{\pi(t)}$ in Stage 2 according to variables X or Y (i.e. BFS). As we argued, in the previous scenarios, the overall expected value of shifting v' from b_q to $b_{\pi(t)}$ is $|V(H)|c(v,t)(v'_{\pi(t)} - v'_{\pi(t)+1})$. Since

$v_t - v_{t+1} = v'_{\pi(t)} - v'_{\pi(t)+1}$, the overall expected value of shifting v to a_t is $|V(H)|(v_t - v_{t+1})$. In conclusion, the expected contribution of $v_t - v_{t+1}$ to the objective function is $2|V(H)|c(v,t)(v_t - v_{t+1})$. □

We remark that, as in the proof of Theorem 6.5.6, the above algorithm can be derandomized. By Lemma 6.7.3 and Theorem 6.7.6 we obtain the following classification theorem.

Theorem 6.7.7. *If H admits a conservative majority polymorphism, then MinHOM(H) has a (deterministic) $2|V(H)|$-approximation algorithm, otherwise, MinHOM(H) is inapproximable unless $\boldsymbol{P} = \boldsymbol{NP}$.*

Chapter 7

Sparsification of submodular functions

7.1 Introduction

Submodularity allows one to efficiently find provably (near-)optimal solutions. In particular, a wide range of efficient approximation algorithms have been developed for maximizing or minimizing submodular functions subject to different constraints. Unfortunately, these algorithms require number of function evaluations which in many data intensive applications are infeasible or extremely inefficient. Fortunately, several submodular optimization problems arising in machine learning have structure that allows solving them more efficiently. A novel class of submodular functions are decomposable submodular functions. These are functions that can be written as sums of several "simple" submodular functions, i.e.,

$$F(S) = \sum_{i=1}^{N} f_i(S) \qquad \forall S \subseteq E,$$

where each $f_i \colon 2^E \to \mathbb{R}$ is a submodular function on the ground set E with $|E| = n$.

Decomposable submodular functions encompass many of the examples of submodular functions studied in the context of machine learning as well as economics. For example, they are extensively used in economics in the problem of welfare maximization in combinatorial auctions [64, 80, 81, 180, 216].

Example 7.1.1 (Welfare maximization)**.** Let E be a set of n resources and $a_1, \ldots, a_N$ be N agents. Each agent has an interest over subsets of resources which is expressed as a submodular function $f_i : 2^E \to \mathbb{R}$. The objective is to select a small subset of resources that maximizes the happiness across all the agents, the "social welfare". More formally, the goal is to find a subset $S \subseteq E$ of size at most k that maximizes $F(S) = \sum_{i=1}^{N} f_i(S)$, where k is a positive integer.

Decomposable submodular functions appear in various machine learning tasks such as data summarization, where we seek a representative subset of elements of small size. This has numerous applications, including exemplar-based clustering [65, 88], image summarization [209], recommender systems [182], and document and corpus summarization [152]. The problem of maximizing decomposable submodular functions has been studied under different constraints such as cardinality and matroid constraints in various data summarization settings [167, 169, 170], and differential privacy settings [49, 171, 188].

In many of these applications, the number of underlying submodular functions are too large (i.e., N is too large) to even fit in the main memory, and building a compressed representation that preserves relevant properties of the submodular function is appealing. This motivates us to find a *sparse* representation for a decomposable submodular function F. In this chapter, we propose a simple and very effective algorithm that yields a sparse and accurate representation of a decomposable submodular function. To the best of our knowledge this work is the first to study sparsification of decomposable submodular functions.

7.1.1 Overview of our contributions

General setting. Given a decomposable submodular function $F = \sum_{i=1}^{N} f_i$, we present a randomized algorithm that yields a sparse representation that approximates F. Our algorithm chooses each submodular function f_i with probability proportional to its "importance" in the sum $\sum_{i=1}^{N} f_i$ to be in the sparsifier. Moreover, each selected submodular function will be assigned a weight which also is proportional to its "importance". We prove this simple algorithm yields a sparsifier of small size (***independent of*** N) with a very good approximation of F. Let $|\mathcal{B}(f_i)|$ denote the number of extreme points in the base polytope of f_i, and $B = \max_{i\in[N]} |\mathcal{B}(f_i)|$.

Theorem 7.1.2. *Let $F = \sum_{i=1}^{N} f_i$ be a decomposable submodular function. For any $\epsilon > 0$, there exists a vector $\boldsymbol{w} \in \mathbb{R}^N$ with at most $O(\frac{B\cdot n^2}{\epsilon^2})$ non-zero entries such that for the submodular function $F' = \sum_{i=1}^{N} \boldsymbol{w}_i f_i$ we have*

$$(1-\epsilon)F'(S) \leq F(S) \leq (1+\epsilon)F'(S) \quad \forall S \subseteq E.$$

Moreover, if all f_i's are monotone, then there exists a polynomial-time randomized algorithm that outputs a vector $\boldsymbol{w} \in \mathbb{R}^N$ with at most $O(\frac{B\cdot n^{2.5}\log n}{\epsilon^2})$ non-zero entries in expectation such that for the submodular function $F' = \sum_{i=1}^{N} \boldsymbol{w}_i f_i$, with high probability, we have

$$(1-\epsilon)F'(S) \leq F(S) \leq (1+\epsilon)F'(S) \quad \forall S \subseteq E.$$

Remark 7.1.3 (Tightness)**.** *The existential result is almost tight because in the special case of directed graphs, we have $\max_i |\mathcal{B}(f_i)| = 2$ and it is known that we need $\Omega(n^2)$ edges to construct a sparsifier [56].*

Sparsifying under constraints. We consider the setting where we only are interested in evaluation of F on particular sets. For instance, the objective is to optimize F on subsets of size at most k, or it is to optimize F over subsets that form a matroid. Optimizing submodular functions under these constraints has been extensively studied and has an extremely rich theoretical landscape. Our algorithm can be tailored to these types of constraints and constructs a sparsifier of even smaller size.

Theorem 7.1.4. *Let $F = \sum_{i=1}^{N} f_i$ be a decomposable submodular function. For any $\epsilon > 0$ and a matroid $\mathcal{M}$ of rank r, there exists a vector $\boldsymbol{w} \in \mathbb{R}^N$ with at most $O(\frac{B \cdot r \cdot n}{\epsilon^2})$ non-zero entries such that for the submodular function $F' = \sum_{i=1}^{N} \boldsymbol{w}_i f_i$ we have*

$$(1-\epsilon)F'(S) \leq F(S) \leq (1+\epsilon)F'(S) \quad \forall S \subseteq \mathcal{M}.$$

Moreover, if all f_i's are monotone, then there exists a polynomial-time randomized algorithm that outputs a vector $\boldsymbol{w} \in \mathbb{R}^N$ with at most $O(\frac{B \cdot r \cdot n^{1.5} \log n}{\epsilon^2})$ non-zero entries in expectation such that for the submodular function $F' = \sum_{i=1}^{N} \boldsymbol{w}_i f_i$, with high probability, we have

$$(1-\epsilon)F'(S) \leq F(S) \leq (1+\epsilon)F'(S) \quad \forall S \subseteq \mathcal{M}.$$

Applications, speeding up maximization/minimization. Our sparsifying algorithm can be used as a preprocessing step in many settings in order to speed up algorithms. To elaborate on this, we consider the classical greedy algorithm of [176] for maximizing monotone submodular functions under cardinality constraints. We observe that sparsifying the instance reduces the number of function evaluations from $O(knN)$ to $O(\frac{Bk^2n^2}{\epsilon^2})$, which is a significant speed up when $N \gg n$. Regarding minimization, we prove our algorithm gives an approximation on the *Lovász extension*, thus it can be used as a preprocessing step for algorithms working on Lovász extensions such as the ones in [8, 73]. One particular regime that has been considered in many results is where each submodular function f_i acts on $O(1)$ elements of the ground set which implies $B = \max_i |\mathcal{B}(f_i)|$ is $O(1)$. Using our sparsifier algorithm as a preprocessing step is quite beneficial here. For instance, it improves the running time of [8] from $\widetilde{O}(T_{\text{maxflow}}(n, n+N) \log \frac{1}{\epsilon})$ to $\widetilde{O}(T_{\text{maxflow}}(n, n+\frac{n^2}{\epsilon^2}) \log \frac{1}{\epsilon})$. Here, $T_{\text{maxflow}}(n, m)$ denotes the time required to compute the maximum flow in a directed graph of n vertices and m arcs with polynomially bounded integral capacities.

Well-known examples. In practice, the bounds on the size of sparsifiers are often better than the ones presented in Theorems 7.1.2 and 7.1.4 e.g. B is a constant. We consider several examples of decomposable submodular functions that appear in many applications, namely, Maximum Coverage, Facility Location, and Submodular Hypergraph Min Cut problems. For the first two examples, sparsifiers of size $O(\frac{n^2}{\epsilon^2})$ can be constructed in time linear in N. For Submodular Hypergraph Min Cut when each hyperedge is of constant size sparsifiers of size

$O(\frac{n^2}{\epsilon^2})$ exist, and in several specific cases with various applications efficient algorithms are employed to construct them.

Empirical results. Finally, we empirically examine our algorithm and demonstrate that it constructs a concise sparsifier on which we can efficiently perform algorithms.

7.1.2 Related work

To the best of our knowledge there is no prior work on sparsification algorithms for decomposable submodular functions. However, special cases of this problem have attracted much attention, most notably *cut sparsifiers* for graphs. The cut function of a graph $G = (V, E)$ can be seen as a decomposable submodular function $F(S) = \sum_{e \in E} f_e$, where $f_e(S) = 1$ if and only if $e \cap S \neq \emptyset$ and $e \cap (V \setminus S) \neq \emptyset$. The problem of sparsifying a graph while approximately preserving its cut structure has been extensively studied, (See [1, 2, 12, 23] and references therein.) The pioneering work of Benczúr and Karger [22] showed for any graph G with n vertices one can construct a weighted subgraph G' in nearly linear time with $O(n \log n/\epsilon^2)$ edges such that the weight of every cut in G is preserved within a multiplicative $(1 \pm \epsilon)$-factor in G'. Note that a graph on n vertices can have $N = \Omega(n^2)$ edges. The bound on the number of edges was later improved to $O(n/\epsilon^2)$ [19] which is tight [3].

A more general concept for graphs called *spectral sparsifier* was introduced by Spielman and Teng [201]. This notion captures the spectral similarity between a graph and its sparsifiers. A spectral sparsifier approximates the *quadratic form of the Laplacian* of a graph. Note that a spectral sparsifier is also a cut sparsifier. This notion has numerous applications in linear algebra [158, 149, 57, 146], and it has been used to design efficient approximation algorithms related to cuts and flows [23, 127, 157]. Spielman and Teng's sparsifier has $O(n \log^c n)$ edges for a large constant $c > 0$ which was improved to $O(n/\epsilon^2)$ [147].

In pursuing a more general setting, the notions of cut sparsifier and spectral sparsifier have been studied for *hypergraphs*. Observe that a hypergraph on n vertices can have exponentially many hyperedges i.e., $N = \Omega(2^n)$. For hypergraphs, Kogan and Krauthgamer [133] provided a polynomial-time algorithm that constructs an ϵ-cut sparsifier with $O(n(r + \log n)/\epsilon^2)$ hyperedges where r denotes the maximum size of a hyperedge. The current best result is due to [52] where their ϵ-cut sparsifier uses $O(n \log n/\epsilon^2)$ hyperedges and can be constructed in time $O(Nn^2 + n^{10}/\epsilon^2)$ where N is the number of hyperedges. Recently, Soma dn Yoshida [200] initiated the study of spectral sparsifiers for hypergraphs and showed that every hypergraph admits an ϵ-spectral sparsifier with $O(n^3 \log n/\epsilon^2)$ hyperedges. For the case where the maximum size of a hyperedge is r, Bansal, Svensson, and Trevisan [12] showed that every hypergraph has an ϵ-spectral sparsifier of size $O(nr^3 \log n/\epsilon^2)$. Recently, this bound has been improved to $O(nr(\log n/\epsilon)^{O(1)})$ and then to $O(n(\log n/\epsilon)^{O(1)})$ [125, 126]. This leads to the study of sparsification of submodular functions which is our focus and provides a unifying framework for these previous works.

7.2 Preliminaries

For a positive integer n, let $[n] = \{1, 2, \ldots, n\}$. Let E be a set of elements of size n which we call the *ground set*. For a set $S \subseteq E$, $\mathbf{1}_S \in \mathbb{R}^E$ denotes the characteristic vector of S. For a vector $\boldsymbol{x} \in \mathbb{R}^E$ and a set $S \subseteq E$, $\boldsymbol{x}(S) = \sum_{e \in S} \boldsymbol{x}(e)$.

Submodular functions. Let $f\colon 2^E \to \mathbb{R}_+$ be a set function. We say that f is *monotone* if $f(S) \leq f(T)$ holds for every $S \subseteq T \subseteq E$. We say that f is *submodular* if $f(S \cup \{e\}) - f(S) \geq f(T \cup \{e\}) - f(T)$ holds for any $S \subseteq T \subseteq E$ and $e \in E \setminus T$. The *base polytope* of a submodular function f is defined as

$$\mathcal{B}(f) = \{\boldsymbol{y} \in \mathbb{R}^E \mid \boldsymbol{y}(S) \leq f(S) \ \ \forall S \subseteq E, \boldsymbol{y}(E) = f(E)\},$$

and $|\mathcal{B}(f)|$ denotes the number of *extreme points* in the base polytope $\mathcal{B}(f)$.

Definition 7.2.1 (ϵ-sparsifier)**.** *Let f_i $(i \in D)$ be a set of N submodular functions, and $F(S) = \sum_{i \in D} f_i(S)$ be a decomposable submodular function. A vector $\boldsymbol{w} \in \mathbb{R}^N$ is called an ϵ-sparsifier of F if, for the submodular function $F' := \sum_{i \in D} \boldsymbol{w}_i f_i$, the following holds for every $S \subseteq E$*

$$(1 - \epsilon)F'(S) \leq F(S) \leq (1 + \epsilon)F'(S). \tag{7.1}$$

The size *of an ϵ-sparsifier $\boldsymbol{w}$,* size$(\boldsymbol{w})$*, is the number of indices i's with $\boldsymbol{w}_i \neq 0$.*

Matroids and matroid polytopes. A pair $\mathcal{M} = (E, \mathcal{I})$ of a set E and $\mathcal{I} \subseteq 2^E$ is called a *matroid* if

(1) $\emptyset \in \mathcal{I}$,

(2) $A \in \mathcal{I}$ for any $A \subseteq B \in \mathcal{I}$, and

(3) for any $A, B \in \mathcal{I}$ with $|A| < |B|$, there exists $e \in B \setminus A$ such that $A \cup \{e\} \in \mathcal{I}$.

We call a set in $\mathcal{I}$ an *independent set*. The *rank function* $r_{\mathcal{M}}\colon 2^E \to \mathbb{Z}_+$ of $\mathcal{M}$ is $r_{\mathcal{M}}(S) = \max\{|I| : I \subseteq S, I \in \mathcal{I}\}$. An independent set $S \in \mathcal{I}$ is called a *base* if $r_{\mathcal{M}}(S) = r_{\mathcal{M}}(E)$. We denote the rank of $\mathcal{M}$ by $r(\mathcal{M})$. The *matroid polytope* $\mathcal{P}(\mathcal{M}) \subseteq \mathbb{R}^E$ of $\mathcal{M}$ is

$$\mathcal{P}(\mathcal{M}) = \mathrm{conv}\{\mathbf{1}_I : I \in \mathcal{I}\},$$

where conv denotes the convex hull. Or equivalently [71],

$$\mathcal{P}(\mathcal{M}) = \{\boldsymbol{x} \geq \mathbf{0} : \boldsymbol{x}(S) \leq r_{\mathcal{M}}(S) \ \forall S \subseteq E\}.$$

Algorithm 4

Require: Submodular functions f_i in dataset D where each $f_i : \{0,1\}^E \to \mathbb{R}$, $\epsilon, \delta \in (0,1)$

$\boldsymbol{w} \leftarrow \mathbf{0}$
$\kappa \leftarrow 3\log(2^{n+1}/\delta)/\epsilon^2$
for f_i in D **do**
    $p_i \leftarrow \max_{A \subseteq E} f_i(A)/F(A)$
    $\kappa_i \leftarrow \min\{1, \kappa \cdot p_i\}$
    $\boldsymbol{w}_i \leftarrow 1/\kappa_i$ with probability κ_i ▷ do nothing with probability $1 - \kappa_i$
return $\boldsymbol{w} \in \mathbb{R}^D$.

Concentration bound. We use the following concentration bound:

Theorem 7.2.2 (Chernoff bound, see e.g. [173])**.** *Let $X_1, \ldots, X_n$ be independent random variable in range $[0, a]$. Let $T = \sum_{i=1}^n X_i$. Then for any $\epsilon \in [0,1]$ and $\mu \geq \mathbb{E}[T]$,*

$$\mathbb{P}\left[|T - \mathbb{E}[T]| \geq \epsilon\mu\right] \leq 2\exp\left(-\frac{\epsilon^2\mu}{3a}\right).$$

7.3 Constructing a sparsifier

In this section, we propose a probabilistic argument that proves the existence of an accurate sparsifier and turn this argument into an (polynomial-time) algorithm that finds a sparsifier with high probability.

For each submodular function f_i, let

$$p_i = \max_{A \subseteq E} \frac{f_i(A)}{F(A)}. \tag{7.2}$$

The values p_i's are our guide on how much weight should be allocated to a submodular function f_i and with what probability it might happen. To construct an ϵ-sparsifier of F, for each submodular function f_i, we assign weight $1/(\kappa \cdot p_i)$ to $\boldsymbol{w}_i$ with probability $\kappa \cdot p_i$ and do nothing for the complement probability $1 - \kappa \cdot p_i$ (see Algorithm 4). Here κ depends on n, ϵ and δ where δ is the failure probability of our algorithm. Observe that, for each f_i, the expected weight $\boldsymbol{w}_i$ is exactly one. We show that the expected number of entries of $\boldsymbol{w}$ with $\boldsymbol{w}_i > 0$ is $n^2 \cdot \max_{i \in D} |\mathcal{B}(f_i)|$. Let $B = \max_{i \in D} |\mathcal{B}(f_i)|$ in the rest of this chapter.

Lemma 7.3.1. *Algorithm 4 returns $\boldsymbol{w}$ which is an ϵ-sparsifier of F with probability at least $1 - \delta$.*

Proof. We prove that for every $S \subseteq E$ with high probability it holds that $(1-\epsilon)F'(S) \leq F(S) \leq (1+\epsilon)F'(S)$.

Observe that by our choice of p_i and $\boldsymbol{w}_i$ we have $\mathbb{E}[F'(S)] = F(S)$, for all subsets $S \subseteq E$. Consider a subset S_k. Using Theorem 7.2.2, we have

$$\begin{aligned}
&\mathbb{P}\Big[|F'(S_k) - \mathbb{E}[F'(S_k)]| \geq \epsilon \mathbb{E}[F'(S_k)]\Big] \\
&= \mathbb{P}\Big[|F'(S_k) - F(S_k)| \geq \epsilon F(S_k)\Big] \qquad (7.3) \\
&\leq 2\exp\left(\frac{-\epsilon^2 F(S_k)}{3a}\right) \qquad (7.4)
\end{aligned}$$

where $a = \max_i \boldsymbol{w}_i f_i(S_k)$. We bound the right hand side of (7.4) by providing an upper bound for a.

$$\begin{aligned}
a &= \max_i \boldsymbol{w}_i f_i(S_k) = \max_i \frac{f_i(S_k)}{\kappa \cdot p_i} \\
&= \max_i \frac{f_i(S_k)}{\kappa \cdot \max\limits_{A \subseteq E} \frac{f_i(A)}{F(A)}} \qquad (7.5) \\
&\leq \max_i \frac{f_i(S_k)}{\kappa \cdot \frac{f_i(S_k)}{F(S_k)}} = \frac{F(S_k)}{\kappa} \qquad (7.6)
\end{aligned}$$

Given the above upper bound for a and the inequality in (7.4) yields

$$\begin{aligned}
&\mathbb{P}\Big[|F'(S_k) - F(S_k)| \geq \epsilon F(S_k)\Big] \leq 2\exp\left(-\frac{\epsilon^2 F(S_k)}{3a}\right) \\
&\leq 2\exp\left(-\frac{\epsilon^2 F(S_k)}{3F(S_k)/\kappa}\right) = 2\exp\left(\frac{-\kappa\epsilon^2}{3}\right).
\end{aligned}$$

Recall that $\kappa = 3\log(2^{n+1}/\delta)/\epsilon^2$. Hence, taking a union bound over all 2^n possible subsets yields that Algorithm 4 with probability at least $1-\delta$ returns a spectral sparsifier for F. □

Lemma 7.3.2. *Algorithm 4 outputs an ϵ-sparsifier with the expected size $O(\frac{B \cdot n^2}{\epsilon^2})$.*

Proof. In Algorithm 4, each $\boldsymbol{w}_i$ is greater than zero with probability κ_i and it is zero with probability $1 - \kappa_i$. Hence,

$$\mathbb{E}[\text{size}(\boldsymbol{w})] = \sum_{i \in D} \kappa_i \leq \kappa \sum_{i \in D} p_i \leq O\left(\frac{n}{\epsilon^2}\right) \sum_{i \in D} p_i \qquad (7.7)$$

It suffices to show an upper bound for $\sum_{i \in D} p_i$.

Claim 7.3.3. $\sum_{i \in D} p_i \leq n \cdot \max_{i \in D} |\mathcal{B}(f_i)| = n \cdot B$.

Claim 7.3.3 and inequality (7.7) yield the desired bound. □

Lemmas 7.3.1 and 7.3.2 proves the existence part of Theorem 7.1.2. That is, for every $\epsilon, \delta \in (0,1)$, there exists an ϵ-sparsifier of size at most $O(\frac{B \cdot n^2}{\epsilon^2})$ with probability at least $1-\delta$.

Polynomial time algorithm. Observe that computing p_i's (7.2) may not be a polynomial-time task in general. However, to guarantee that Algorithm 4 outputs an ϵ-sparsifier with high probability it is sufficient to instantiate it with an upper bound for each p_i (see proof of Lemma 7.3.1). Fortunately, the result of [10] provides an algorithm to approximate the ratio of two monotone submodular functions.

Theorem 7.3.4 ([10])**.** *Let f and g be two monotone submodular functions. Then there exists a polynomial-time algorithm that approximates $\max_{S \subseteq E} \frac{f(S)}{g(S)}$ within $O(\sqrt{n} \log n)$ factor.*

Hence, when all f_i's are monotone we can compute $\hat{p}_i$'s with $p_i \leq \hat{p}_i \leq O(\sqrt{n} \log n) p_i$ in polynomial time which leads to a polynomial-time randomized algorithm that constructs an ϵ-sparsifier of the expected size at most $O(\frac{B \cdot n^{2.5} \log n}{\epsilon^2})$. This proves the second part of Theorem 7.1.2.

As we will see, in various applications, the expected size of the sparsifier is often much better than the ones presented in this section. Also, we emphasize that once a sparsifier is constructed it can be reused many times (possibly for maximization/minimization under several different constraints). Hence computing or approximating p_i's should be regarded as a preprocessing step, see Example 7.3.5 for a motivating example. Finally, it is straightforward to adapt our algorithm to sparsify decomposable submodular functions of the form $\sum_{i \in D} \alpha_i f_i$, known as *mixtures of submodular functions* [11, 209].

Example 7.3.5 (Knapsack constraint)**.** Consider the following optimization problem

$$\max_{S \subseteq I} \{F(S) : \sum_{i \in S} c_i \leq B\} \tag{7.8}$$

where $I = \{1, \ldots, n\}$, B and c_i, $i \in I$, are nonnegative integers. In scenarios where the items costs c_i or B are dynamically changing and $F = \sum_{i=1}^{N} f_i$ is decomposable, it is quite advantageous to use our sparsification algorithm and reuse a sparsifier. That is, instead of maximizing F whenever item costs or B are changed, we can maximize F', a sparsification of F.

7.4 Constructing a sparsifier under constraints

Here we are interested in constructing a sparsifier for a decomposable submodular function F while the goal is to optimize F subject to constraints. One of the most commonly used

Algorithm 5

Require: Submodular functions $f_i : \{0,1\}^E \to \mathbb{R}$ in dataset D, matroid $\mathcal{M} = (E, \mathcal{I})$, and $\epsilon, \delta \in (0,1)$

$\boldsymbol{w} \leftarrow \mathbf{0}$

$\kappa \leftarrow 3\log(2n^{r+1}/\delta)/\epsilon^2$, where r is the rank of $\mathcal{M}$.

for f_i in D **do**

  $p_i \leftarrow \max_{A \in \mathcal{I}} f_i(A)/F(A)$

  $\kappa_i \leftarrow \min\{1, \kappa \cdot p_i\}$

  $\boldsymbol{w}_i \leftarrow 1/\kappa_i$ with probability κ_i ▷ do nothing with probability $1 - \kappa_i$

return $\boldsymbol{w} \in \mathbb{R}^D$.

and general constraints are matroid constraints. That is, for a matroid $\mathcal{M} = (E, \mathcal{I})$, the objective is finding $S^* = \operatorname{argmax}_{S \subseteq E, S \in \mathcal{I}} F(S)$.

In this setting it is sufficient to construct a sparsifier that approximates F only on independent sets. It turns out that we can construct a *smaller* sparsifier than the one constructed to approximate F everywhere. For each submodular function f_i, let

$$p_i = \max_{A \in \mathcal{I}} \frac{f_i(A)}{F(A)}. \tag{7.9}$$

Other than different definition for p_i's and different κ, Algorithm 5 is the same as Algorithm 4.

Theorem 7.4.1. *Algorithm 5 returns a vector $\boldsymbol{w}$ with expected size at most $O(\frac{B \cdot r \cdot n}{\epsilon^2})$ such that, with probability at least $1 - \delta$, for $F' = \sum_{i \in D} \boldsymbol{w}_i f_i$ we have*

$$(1-\epsilon)F'(S) \leq F(S) \leq (1+\epsilon)F'(S) \quad \forall S \subseteq \mathcal{M}.$$

Theorem 7.4.1 proves the existence part of Theorem 7.1.4. Algorithm 5 can be turned into a polynomial-time algorithm if one can approximate p_i's (7.9). By modifying the proof of Theorem 7.3.4 we prove the following.

Theorem 7.4.2. *Let f and g be two monotone submodular functions and $\mathcal{M} = (E, \mathcal{I})$ be a matroid. Then there exists a polynomial-time algorithm that approximates $\max_{S \subseteq E, S \in \mathcal{I}} \frac{f(S)}{g(S)}$ within $O(\sqrt{n}\log n)$ factor.*

By this theorem, when all f_is are monotone we can compute $\hat{p}_i$'s with $p_i \leq \hat{p}_i \leq O(\sqrt{n}\log n)p_i$ in polynomial time which leads to a polynomial-time randomized algorithm that constructs an ϵ-sparsifier of the expected size at most $O(\frac{B \cdot r \cdot n^{1.5} \log n}{\epsilon^2})$. This proves the second part of Theorem 7.1.4.

Algorithm 6

Require: Submodular function $F = \sum_{i \in D} f_i$ with each $f_i : \{0,1\}^E \to \mathbb{R}$, constant k, and $\epsilon, \delta \in (0,1)$

Compute $F' = \sum_{i \in D} \boldsymbol{w}_i f_i$, an ϵ-sparsifier for F.
$A \leftarrow \emptyset$.
while $|A| \leq k$ **do**
    $a_i \leftarrow \text{argmax}_{a \in E \setminus A}(F'(A \cup \{a\}) - F'(A))$.
    $A \leftarrow A \cup \{a_i\}$.
return A.

7.5 Applications

7.5.1 Submodular function maximization with cardinality constraint

Our sparsification algorithm can be used as a preprocessing step and once a sparsifier is constructed it can be reused many times (possibly for maximization/minimization under several different constraints). To elaborate on this, we consider the problem of maximizing a submodular function subject to a cardinality constraint. That is finding $S^* = \text{argmax}_{S \subseteq E, |S| \leq k} F(S)$. Cardinality constraint is a special case of matroid constraint where the independent sets are all subsets of size at most k and the rank of the matroid is k. A celebrated result of [176] states that for non-negative monotone submodular functions a simple greedy algorithm provides a solution with $(1 - 1/\text{e})$ approximation guarantee to the optimal (intractable) solution. For a ground set E of size n and a monotone submodular function $F = \sum_{i \in D} f_i$, this greedy algorithm needs $O(knN)$ function evaluations to find S of size k such that $F(S) \geq (1 - 1/\text{e})F(S^*)$. We refer to this algorithm as `GreedyAlg`. In many applications where $N \gg n$, having a sparsifier is beneficial. Applying `GreedyAlg` on an ϵ-sparsifier of size $O(Bkn/\epsilon^2)$ improves the number of function evaluations to $O(Bk^2n^2/\epsilon^2)$ and yields S of size k such that $F(S) \geq (1 - 1/\text{e} - \epsilon)F(S^*)$ with high probability (see Algorithm 6).

We point out that sampling techniques such as [172, 168] sample elements from the ground set E rather than sampling from functions $f_1, \ldots, f_N$. Hence their running time depend on N, which could be slow when N is large — the regime we care about. Besides, our algorithm can be used as a preprocessing step for these algorithms. For instance, the lazier than lazy greedy algorithm [168] requires $O(nN \log \frac{1}{\epsilon})$ function evaluations. However, when N is much larger than n it is absolutely beneficial to use our sparsification algorithm and reduce the number of submodular functions that one should consider.

7.5.2 Two well-known examples

Maximum Coverage problem. Let $[N]$ be a universe and $E = \{S_1, \ldots, S_n\}$ with each $S_i \subseteq N$ be a family of sets. Given a positive integer k, in the Max Coverage problem the

objective is to select at most k of sets from E such that the maximum number of elements are covered, i.e., the union of the selected sets has maximal size. One can formulate this problem as follows. For every $i \in [N]$ and $A \subseteq [n]$ define $f_i(A)$ as

$$f_i(A) = \begin{cases} 1 & \text{if there exists } a \in A \text{ such that } i \in S_a, \\ 0 & \text{otherwise.} \end{cases}$$

Note that f_i's are monotone and submodular. Furthermore, define $F : 2^n \to \mathbb{R}_+$ to be $F(A) = \sum_{i \in [N]} f_i(A)$ which is monotone and submodular as well. Now the Max Coverage problem is equivalent to $\max_{A \subseteq [n], |A| \leq k} F(A)$. For each submodular function f_i, the corresponding p_i is

$$\begin{aligned} p_i &= \max_{A \subseteq [n], |A| \leq k} \frac{f_i(A)}{F(A)} = \max_{S_a \in E, i \in S_a} \frac{f_i(\{a\})}{F(\{a\})} \\ &= \max_{S_a \in E, i \in S_a} \frac{1}{F(\{a\})} = \max_{S_a \in E, i \in S_a} \frac{1}{|S_a|}. \end{aligned}$$

We can compute all the p_i's in $O(\sum |S_i|)$ time, which is the input size. Then we can construct a sparsifier in $O(N)$ time. In total, the time required for sparsification is $O(\sum |S_i| + N)$. On the other hand, for this case we have

$$\sum_{i=1}^{N} p_i = \sum_{i=1}^{N} \max_{S_a \in \mathcal{S}, i \in S_a} \frac{1}{|S_a|} \leq \sum_{i=1}^{n} \frac{|S_i|}{|S_i|} = n.$$

By Lemma 7.3.2, this upper bound provides that our algorithm constructs an ϵ-sparsifier of size at most $O(kn/\epsilon^2)$. Algorithm 6 improves the running time of the `GreedyAlg` from $O(knN)$ to $O(k^2n^2/\epsilon^2)$. Furthermore, Algorithm 6 returns a set A of size at most k such that $(1 - 1/\mathrm{e} - \epsilon)\texttt{OPT} \leq F(A)$. (`OPT` denotes $F(S^*)$ where $S^* = \operatorname{argmax}_{S \subseteq E, |S| \leq k} F(S)$.)

Facility Location problem. Let I be a set of N clients and E be a set of facilities with $|E| = n$. Let $c : I \times E \to \mathbb{R}$ be the cost of assigning a given client to a given facility. For each client i and each subset of facilities $A \subseteq E$, define $f_i(A) = \max_{j \in A} c(i, j)$. For any non-empty subset $A \subseteq E$, the value of A is given by

$$F(A) = \sum_{i \in I} f_i(A) = \sum_{i \in I} \max_{j \in A} c(i, j).$$

For completeness, we define $F(\emptyset) = 0$. An instance of the Max Facility Location problem is specified by a tuple (I, E, c). The objective is to choose a subset $A \subseteq E$ of size at most k

maximizing $F(A)$. For each submodular function f_i, the corresponding p_i is

$$p_i = \max_{A \subseteq E, |A| \leq k} \frac{f_i(A)}{F(A)} = \max_{A \subseteq E, |A| \leq k} \frac{\max_{j \in A} c(i,j)}{F(A)} = \max_{j \in E} \frac{c(i,j)}{F(\{j\})}$$

It is clear that p_i's can be computed in $O(|I| \cdot |E|)$ time, which is the input size. In this case, we have

$$\begin{aligned} \sum_{i \in I} p_i &= \sum_{i \in I} \max_{j \in E} \frac{c(i,j)}{F(\{j\})} \leq \sum_{j=1}^{|E|} \frac{\sum_{i \in I} c(i,j)}{F(\{j\})} = \sum_{j=1}^{|E|} \frac{F(\{j\})}{F(\{j\})} \\ &= |E| = n. \end{aligned}$$

Hence, by Lemma 7.3.2, our algorithm construct a sparsifier of size $O(kn/\epsilon^2)$. Algorithm 6 improves the running time of the `GreedyAlg` from $O(knN)$ to in $O(k^2n^2/\epsilon^2)$. Furthermore, Algorithm 6 returns a set A of size at most k such that $(1 - 1/\mathrm{e} - \epsilon)\texttt{OPT} \leq F(A)$.

Remark 7.5.1. *[153] sparsify an instance of the Facility Location problem by zeroing out entries in the cost matrix — this is not applicable to the general setting. The runtime of the `GreedyAlg` applied on their sparsified instance is $O(nN/\epsilon)$. This runtime is huge when N is large — the regime we care about. Moreover, we can first construct our sparsifier and apply the algorithm of [153] on it.*

7.5.3 Submodular function minimization

Besides the applications regarding submodular maximization, our sparsification algorithm can be used as a preprocessing step for submodular minimization as well. In many applications of the submodular minimization problem such as image segmentation [198], Markov random field inference [83, 134, 215], hypergraph cuts [214], covering functions [203], the submodular function at hand is a decomposable submodular function. Many of recent advances on decomposable submodular minimization such as [73, 8] have leveraged a mix of ideas coming from both discrete and continuous optimization. Here we discuss that our sparsifying algorithm approximates the so called *Lovász extension*, a natural extension of a submodular function to the continuous domain $[0,1]^n$.

Lovász extension. Let $\boldsymbol{x} \in [0,1]^n$ be the vector $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$. Let $\pi : [n] \to [n]$ be a sorting permutation of $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$, which means if $\pi(i) = j$, then $\boldsymbol{x}_j$ is the i-th largest element in the vector $\boldsymbol{x}$. Hence, $1 \geq \boldsymbol{x}_{\pi(1)} \geq \cdots \geq \boldsymbol{x}_{\pi(n)} \geq 0$. Let $\boldsymbol{x}_{\pi(0)} = 1$ and $\boldsymbol{x}_{\pi(n+1)} = 0$. Define sets $S_0^\pi = \emptyset$ and $S_i^\pi = \{\pi(1), \ldots, \pi(i)\}$. The *Lovász extension* of f is defined as follows $f^L(\boldsymbol{x}) = \sum_{i=0}^n (\boldsymbol{x}_{\pi(i)} - \boldsymbol{x}_{\pi(i+1)}) f(S_i^\pi)$. It is well-known that $f^L(\boldsymbol{x}) = \max_{\boldsymbol{y} \in \mathcal{B}(f)} \langle \boldsymbol{y}, \boldsymbol{x} \rangle$.

For a decomposable submodular function $F = \sum_{i \in D} f_i$, its Lovász extension is

$$F^L(\boldsymbol{x}) = \sum_{j=0}^{n} \sum_{i \in D} (\boldsymbol{x}_{\pi(j)} - \boldsymbol{x}_{\pi(j+1)}) f_i(S_j^\pi).$$

Recall the definition of p_i's (7.2), they can be expressed in an equivalent way in terms of permutations as follow

$$p_i = \max_{A \subseteq E} \frac{f_i(A)}{F(A)} = \max_{\pi} \max_{j \in [n]} \frac{f_i(S_j^\pi)}{F(S_j^\pi)}. \tag{7.10}$$

Furthermore, note that $F^L(\boldsymbol{x})$ is a linear combination of $F(S)$, $S \subseteq E$. Given these, we prove Algorithm 4 outputs a sparsifier that not only approximates the function itself but also approximates its Lovász extension.

Theorem 7.5.2. *Algorithm 4 returns a vector $\boldsymbol{w}$ with expected size at most $O(\frac{B \cdot n^2}{\epsilon^2})$ such that, with probability at least $1 - \delta$, for $F' = \sum_{i \in D} \boldsymbol{w}_i f_i$ it holds that*

$$(1 - \epsilon) F'^L(\boldsymbol{x}) \leq F^L(\boldsymbol{x}) \leq (1 + \epsilon) F'^L(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in [0, 1]^n.$$

Remark 7.5.3 (Relation to spectral sparsification of graphs)**.** *The cut function of a graph $G = (V, E)$ can be seen as a decomposable submodular function $F(S) = \sum_{e \in E} f_e$, where $f_e(S) = 1$ if and only if $e \cap S \neq \emptyset$ and $e \cap (V \setminus S) \neq \emptyset$. The goal of spectral sparsification of graphs [201] is to preserve the quadratic form of the Laplacian of G, which can be rephrased as $\sum_{e \in E} f_e^L(\boldsymbol{x})^2$. In contrast, our sparsification preserves $F^L(\boldsymbol{x}) = \sum_{e \in E} f_e^L(\boldsymbol{x})$. Although we can construct a sparsifier that preserves $\sum_{e \in E} f_e^L(\boldsymbol{x})^2$ in the general submodular setting, we adopted the one used here because, in many applications where submodular functions are involved, we are more interested in the value of $\sum_{e \in E} f_e^L(\boldsymbol{x})$ than $\sum_{e \in E} f_e^L(\boldsymbol{x})^2$, and the algorithm for preserving the former is simpler than that for preserving the latter.*

Because our algorithm gives an approximation on the Lovász extension, it can be used as a preprocessing step for algorithms working on Lovász extensions such as the ones in [8, 73]. For instance, it improves the running time of [8] from $\widetilde{O}(T_{\text{maxflow}}(n, n+N) \log \frac{1}{\epsilon})$ to $\widetilde{O}(T_{\text{maxflow}}(n, n + \frac{n^2}{\epsilon^2}) \log \frac{1}{\epsilon})$ in cases where each submodular function $f_i \in D$ acts on $O(1)$ elements of the ground set which implies $B = \max_i |\mathcal{B}(f_i)|$ is $O(1)$. An example of such cases is hypergraph cut functions with $O(1)$ sized hyperedges.

Next we discuss several examples for which computing p_i's is a computationally efficient task, thus achieving a polynomial-time algorithm to construct sparsifiers. Recall that the cut function of a graph $G = (V, E)$ can be seen as a decomposable submodular function. In this case, computing each p_e for an edge $e = st \in E$ is equivalent to finding the minimum s-t cut in the graph, which is a polynomial time task. A more general framework is the *submodular hypergraph minimum s-t cut* problem discussed in what follows.

Submodular hypergraphs [151, 221]. Let $\mathcal{H}$ be a hypergraph with vertex set V and set of hyperedges E where each hyperedge is a subset of vertices V. A submodular function f_e is associated to each hyperedge $e \in E$. In the submodular hypergraph minimum s-t cut problem the objective is

$$\text{minimize}_{S \subset V} \sum_{e \in E} f_e(e \cap S) \tag{7.11}$$

subject to $s \in S, t \in V \setminus S$. This problem has been studied by [213] and its special cases where submodular functions f_e take particular forms have been studied with applications in semi-supervised learning, clustering, and rank learning (see [151, 213] for more details). Examples of such special cases include:

- Linear penalty: $f_e(S) = \min\{|S|, |e \setminus S|\}$
- Quadratic Penalty: $f_e(S) = |S| \cdot |e \setminus S|$

We refer to Table 1 in [213] for more examples. These examples are cardinality-based, that is, the value of the submodular function depends on the cardinality of the input set (see Definition 3.2 of [213]). It is known that if all the submodular functions are cardinality-based, then computing the s-t minimum cut in the submodular hypergraph can be reduced to that in an auxiliary (ordinary) graph (Theorem 4.6 of [213]), which allows us to compute p_e's in polynomial time.

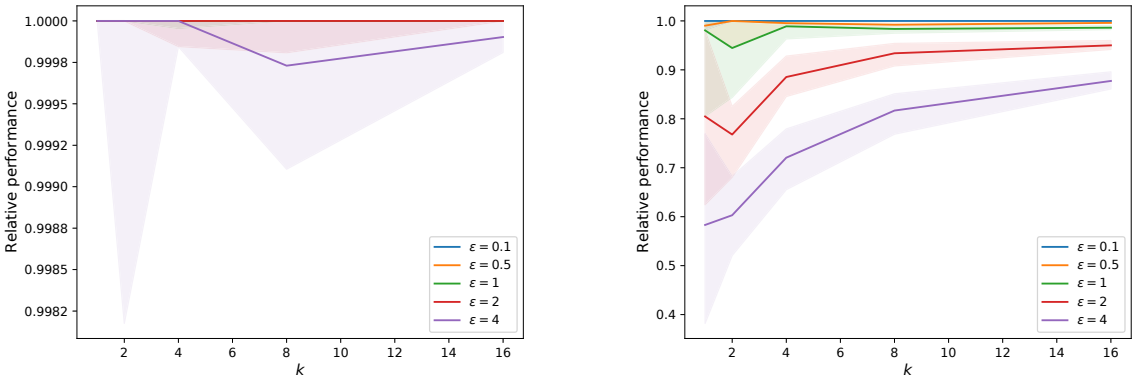
Remark 7.5.4. *Our sparsification algorithm can also be used to construct* submodular Laplacian *based on the Lovász extension of submodular functions. Submodular Laplacian was introduced by [221] and has numerous applications in machine learning, including in learning ranking data, clustering based on network motifs [150], network analysis [220], and etc.*

7.6 Experimental results

In this section, we empirically demonstrate that our algorithm (Algorithm 5) generates a sparse representation of a decomposable submodular function $F : 2^E \to \mathbb{R}_+$ with which we can efficiently obtain a high-quality solution for maximizing F. We consider the following two settings.

Uber pickup. We used a database of Uber pickups in New York city in May 2014 consisting of a set R of 564,517 records[1]. Each record has a pickup position, longitude and latitude. Consider selecting k locations as waiting spots for idle Uber drivers. To formalize this problem, we selected a set L of 36 popular pickup locations in the database, and

[1]Available at `https://www.kaggle.com/fivethirtyeight/uber-pickups-in-new-york-city`



(a) Uber pickup (b) Discogs

Figure 7.1: Relative performance of the greedy method on sparsifiers.

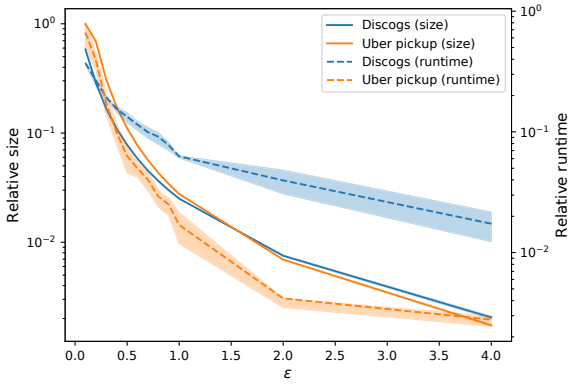


Figure 7.2: Relative size of sparsifiers and relative runtime of the greedy method on sparsifiers.

constructed a facility location function $F : 2^L \to \mathbb{R}_+$ as $F(S) = \sum_{v \in R} f_v(S)$, where $f_v(S) = \max_{u \in L} d(u, v) - \min_{u \in S} d(u, v)$ and $d(u, v)$ is the Manhattan distance between u and v. Then, the goal of the problem is to maximize $F(S)$ subject to $|S| \leq k$.

Discogs [143]. This dataset provides information about audio records as a bipartite graph $G = (L, R; E)$, where each edge $(u, v) \in L \times R$ indicates that a label v was involved in the production of a release of a style u. We have $|L| = 383$ and $|R| = 243,764$, and $|E| = 5,255,950$. Consider selecting k styles that cover the activity of as many labels as possible. To formalize this problem, we constructed a maximum coverage function $F : 2^L \to \mathbb{R}$ as $F(S) = \sum_{v \in R} f_v(S)$, where $f_v(S)$ is 1 if v has a neighbor in S and 0 otherwise. Then, the goal is to maximize $F(S)$ subject to $|S| \leq k$.

Figure 7.1 shows the objective value of the solution obtained by the greedy method on the sparsifier relative to that on the original input function with its 25th and 75th

percentiles. Although our theoretical results do not give any guarantee when $\epsilon > 1$, we tried constructing our sparsifier with $\epsilon > 1$ to see its performance. The solution quality of our sparsifier for Uber pickup is more than 99.9% even when $\epsilon = 4$, and that for Discogs is more than 90% performance when $\epsilon \leq 1.0$. The performance for Uber pickup is higher than that for Discogs because the objective function of the former saturates easily. These results suggest that we get a reasonably good solution quality by setting $\epsilon = 1$.

Number of functions and speedups. Figure 7.2 shows the size, that is, the number of functions with positive weights, of our sparsifier relative to that of the original function and the runtime of the greedy method on the sparsifier relative to that on the original function with their 25th and 75th percentiles when $k = 8$. The size and runtime are decreased by a factor of 30–50 when $\epsilon = 1$. To summarize, our experimental results suggest that our sparsifier highly compresses the original function without sacrificing the solution quality.

7.7 Missing proofs

7.7.1 Proof of Claim 7.3.3

Proof.

$$\begin{aligned}
\sum_{i\in D} p_i &= \sum_{i\in D} \max_{A\subseteq E} \frac{f_i(A)}{F(A)} = \sum_{i\in D} \max_{A\subseteq E} \frac{f_i(A)}{\sum\limits_{j\in D} f_j(A)} \\
&= \sum_{i\in D} \max_{A\subseteq E} \frac{\max\limits_{\boldsymbol{y}\in\mathcal{B}(f_i)} \langle \boldsymbol{y}, \mathbf{1}_A\rangle}{\sum\limits_{j\in D} \max\limits_{\boldsymbol{y}\in\mathcal{B}(f_j)} \langle \boldsymbol{y}, \mathbf{1}_A\rangle} \\
&\leq \sum_{i\in D} \max_{A\subseteq E} \frac{\sum\limits_{\boldsymbol{y}\in\mathcal{B}(f_i)} \langle \boldsymbol{y}, \mathbf{1}_A\rangle}{\sum\limits_{j\in D} \frac{1}{|\mathcal{B}(f_j)|} \sum\limits_{\boldsymbol{y}\in\mathcal{B}(f_j)} \langle \boldsymbol{y}, \mathbf{1}_A\rangle} \\
&\leq \sum_{i\in D} \max_{A\subseteq E} \max_{e\in A} \frac{\sum\limits_{\boldsymbol{y}\in\mathcal{B}(f_i)} \boldsymbol{y}(e)}{\sum\limits_{j\in D} \frac{1}{|\mathcal{B}(f_j)|} \sum\limits_{\boldsymbol{y}\in\mathcal{B}(f_j)} \boldsymbol{y}(e)} \\
&= \sum_{i\in D} \max_{e\in E} \frac{\sum\limits_{\boldsymbol{y}\in\mathcal{B}(f_i)} \boldsymbol{y}(e)}{\sum\limits_{j\in D} \frac{1}{|\mathcal{B}(f_j)|} \sum\limits_{\boldsymbol{y}\in\mathcal{B}(f_j)} \boldsymbol{y}(e)} \\
&\leq \sum_{i\in D} \sum_{e\in E} \frac{\sum\limits_{\boldsymbol{y}\in\mathcal{B}(f_i)} \boldsymbol{y}(e)}{\sum\limits_{j\in D} \frac{1}{|\mathcal{B}(f_j)|} \sum\limits_{\boldsymbol{y}\in\mathcal{B}(f_j)} \boldsymbol{y}(e)} \\
&= \sum_{e\in E} \sum_{i\in D} \frac{\sum\limits_{\boldsymbol{y}\in\mathcal{B}(f_i)} \boldsymbol{y}(e)}{\sum\limits_{j\in D} \frac{1}{|\mathcal{B}(f_j)|} \sum\limits_{\boldsymbol{y}\in\mathcal{B}(f_j)} \boldsymbol{y}(e)} \\
&\leq \sum_{e\in E} \max_{j\in D} |\mathcal{B}(f_j)| \sum_{i\in D} \frac{\sum\limits_{\boldsymbol{y}\in\mathcal{B}(f_i)} \boldsymbol{y}(e)}{\sum\limits_{j\in D} \sum\limits_{\boldsymbol{y}\in\mathcal{B}(f_j)} \boldsymbol{y}(e)} \\
&\leq \sum_{e\in E} \max_{j\in D} |\mathcal{B}(f_j)| \frac{\sum\limits_{i\in D} \sum\limits_{\boldsymbol{y}\in\mathcal{B}(f_i)} \boldsymbol{y}(e)}{\sum\limits_{j\in D} \sum\limits_{\boldsymbol{y}\in\mathcal{B}(f_j)} \boldsymbol{y}(e)} \\
&= \sum_{e\in E} \max_{j\in D} |\mathcal{B}(f_j)| = n \cdot (\max_{j\in D} |\mathcal{B}(f_j)|).
\end{aligned}$$

□

7.7.2 Proof of Theorem 7.4.1

Proof. The proof is almost identical to the proof of Lemma 7.3.1. We prove that for every $S \in \mathcal{I}$ with high probability it holds that $(1-\epsilon)F'(S) \leq F^L(S) \leq (1+\epsilon)F'(S)$. Observe

that by our choice of p_i and $\boldsymbol{w}_i$ we have $\mathbb{E}[F'(S)] = F(S)$, for all subsets $S \in \mathcal{M}$. Consider a subset S_k. Using Theorem 7.2.2, we have

$$\mathbb{P}\left[|F'(S_k) - \mathbb{E}[F'(S_k)]| \geq \epsilon \mathbb{E}[F'(S_k)]\right] \tag{7.12}$$

$$= \mathbb{P}\left[|F'(S_k) - F(S_k)| \geq \epsilon F(S_k)\right] \tag{7.13}$$

$$\leq 2\exp\left(\frac{-\epsilon^2 F(S_k)}{3a}\right) \tag{7.14}$$

where $a = \max_i \boldsymbol{w}_i f_i(S_k)$. We bound the right hand side of (7.14) by providing an upper bound for a.

$$\begin{aligned} a &= \max_i \boldsymbol{w}_i f_i(S_k) = \max_i \frac{f_i(S_k)}{\kappa \cdot p_i} \\ &= \max_i \frac{f_i(S_k)}{\kappa \cdot \max\limits_{A \in \mathcal{I}} \frac{f_i(A)}{F(A)}} \\ &\leq \max_i \frac{f_i(S_k)}{\kappa \cdot \frac{f_i(S_k)}{F(S_k)}} = \frac{F(S_k)}{\kappa}. \end{aligned}$$

Given the above upper bound for a and the inequality in (7.14) yields

$$\begin{aligned} &\mathbb{P}\left[|F'(S_k) - F(S_k)| \geq \epsilon F(S_k)\right] \leq 2\exp\left(-\frac{\epsilon^2 F(S_k)}{3a}\right) \\ &\leq 2\exp\left(-\frac{\epsilon^2 F(S_k)}{3F(S_k)/\kappa}\right) = 2\exp\left(\frac{-\kappa\epsilon^2}{3}\right) \end{aligned}$$

Recall that $\kappa = 3\log(2n^{r+1}/\delta)/\epsilon^2$. Note that there are at most n^r sets in a matroid of rank r. Taking a union bound over all n^r subsets yields that Algorithm 5 with probability at least $1 - \delta$ returns a sparsifier for F over the matroid. Similar to Lemma 7.3.2, $\sum_{i \in D} p_i \leq n \cdot (\max_{i \in D} |B(f_i)|)$, and having $\kappa = 3\log(2n^{r+1}/\delta)/\epsilon^2$ gives

$$\mathbb{E}[\text{size}(\boldsymbol{w})] \leq \sum_i \kappa p_i \leq O\left(\frac{rn}{\epsilon^2} \cdot \max_{i \in D} |B(f_i)|\right).$$

□

7.7.3 Proof of Theorem 7.4.2

Proof. The proof is almost the same as that of Theorem 3.5 (Theorem 3.5 in [10]). Therefore, we only explain modifications we need to handle a matroid constraint.

In the algorithm used in Theorem 3.5, given a monotone modular function $f : 2^E \to \mathbb{R}_+$, a monotone submodular function $g : 2^E \to \mathbb{R}_+$, and a threshold $c \in \mathbb{R}_+$, we iteratively solve

the following problem:

$$\begin{array}{ll} \text{minimize} & f(X), \\ \text{subject to} & g(X) \geq c. \end{array} \tag{7.15}$$

We say that an algorithm for solving (7.15) a (σ, ρ)-*bicriterion algorithm* if it outputs $X \subseteq E$ such that $f(X) \leq \sigma f(X^*)$ and $g(X) \geq \rho c$, where X^* is the optimal solution. It is shown in [10] that a (σ, ρ)-bicriterion algorithm for constant σ and ρ leads to an $O(\sqrt{n} \log n)$-approximation algorithm for maximizing $g(X)/f(X)$.

If we have an additional matroid constraint $\mathcal{M} = (E, \mathcal{I})$, we need a bicriterion algorithm for the following problem:

$$\begin{array}{ll} \text{minimize} & f(X), \\ \text{subject to} & g(X) \geq c, \\ & X \in \mathcal{I}. \end{array} \tag{7.16}$$

To solve (7.16), we consider the following problem.

$$\begin{array}{ll} \text{maximize} & g(X), \\ \text{subject to} & f(X) \leq d, \\ & X \in \mathcal{I}. \end{array} \tag{7.17}$$

This problem is a monotone submodular function maximization problem subject to an intersection of a matroid constraint and a knapsack constraint (recall that f is modular), and is known to admit α-approximation for some constant α [51]. Then by computing an α-approximate solution X for every d of the form 2^i, and take the minimum d such that $g(X) \geq \alpha \cdot c$, we obtain a $(1, \alpha)$-bicriterion approximation to (7.16), as desired. □

7.7.4 Proof of Theorem 7.5.2

Proof. It follows from the fact that $F^L(\boldsymbol{x})$ is a linear combination of $F(S)$, $S \subseteq E$. More precisely, for a decomposable submodular function $F = \sum_{i \in D} f_i$, its Lovász extension is

$$F^L(\boldsymbol{x}) = \sum_{j=0}^{n} \sum_{i \in D} (\boldsymbol{x}_{\pi(j)} - \boldsymbol{x}_{\pi(j+1)}) f_i(S_j^\pi) \tag{7.18}$$

$$= \sum_{j=0}^{n} (\boldsymbol{x}_{\pi(j)} - \boldsymbol{x}_{\pi(j+1)}) \sum_{i \in D} f_i(S_j^\pi) \tag{7.19}$$

$$= \sum_{j=0}^{n} (\boldsymbol{x}_{\pi(j)} - \boldsymbol{x}_{\pi(j+1)}) F(S_j^\pi) \tag{7.20}$$

Now since our sparsifier approximates $F(S)$ for all subsets $S \subseteq E$ we have

$$(1-\epsilon)F'(S_j^\pi) \leq F(S_j^\pi) \leq (1+\epsilon)F'(S_j^\pi) \quad 0 \leq j \leq n \tag{7.21}$$

Finally, (7.20) and (7.21) yield the following

$$(1-\epsilon)F'^L(\boldsymbol{w}) \leq F^L(\boldsymbol{x}) \leq (1+\epsilon)F'^L(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in [0,1]^n.$$

□

Chapter 8

Submodular optimization under privacy

8.1 Introduction

The need for efficient optimization methods that guarantee the privacy of individuals is widespread across many applications concerning sensitive data about individuals, e.g., medical data, web search query data, salary data, social networks. Let us motivate privacy concerns by an example.

Example 8.1.1 (Feature Selection [138, 171])**.** A sensitive dataset $D = \{(\mathbf{x}_i, C_i)\}_{i=1}^n$ consists of a feature vector $\mathbf{x}_i = (\mathbf{x}_i(1), \ldots, \mathbf{x}_i(m))$ associated to each individual i together with a binary class label C_i. The objective is to select a small (e.g., size at most k) subset $S \subseteq [m]$ of features that can provide a good classifier for C. One particular example for this setting is determining collection of features such as height, weight, and age that are most relevant in predicting if an individual is likely to have a particular disease such as diabetes and HIV. One approach to address the feature selection problem, due to [138], is based on maximizing a submodular function which captures the mutual information between a subset of features and the class label of interest. Here, it is important that the selection of relevant features does not compromise the privacy of any individual who has contributed to the training dataset.

Differential privacy is a rigorous notion of privacy that allows statistical analysis of sensitive data while providing strong privacy guarantees. Basically, differential privacy requires that computations be insensitive to changes in any particular individual's record. A dataset is a collection of records from some domain, and two datasets are *neighboring* if they differ in a single record. Simply put, the requirement for differential privacy is that the computation behaves nearly identically on two neighboring datasets; Formally, for $\epsilon, \delta \in \mathbb{R}_+$, we say that a randomized computation M is (ϵ, δ)-*differentially private* if for any neighboring

datasets $D \sim D'$, and for any set of outcomes $S \subseteq \text{range}(M)$,

$$\Pr[M(D) \in S] \leq \exp(\epsilon) \Pr[M(D') \in S] + \delta.$$

When $\delta = 0$, we say M is *ϵ-differentially private*. Differentially private algorithms must be calibrated to the *sensitivity* of the function of interest with respect to small changes in the input dataset.

In this chapter we consider designing a differentially private algorithm for maximizing nonnegative and *monotone* submodular functions in *low-sensitivity* regime. Whilst, a *cardinality* constraint (as in Example 8.1.1) is a natural one to place on a submodular maximization problem, many other problems, e.g., personalized data summarization [170], require the use of more general types of constraints, i.e., *matroid* constraints. The problem of maximizing a submodular function under a matroid constraint is a classical problem [70], with many important special cases, e.g., uniform matroid (the subset selection problem, see Example 8.1.1), partition matroid (submodular welfare/partition problem). We consider the following.

Problem 8.1.2. *Given a sensitive dataset D associated to a monotone submodular function $F_D \colon 2^E \to \mathbb{R}_+$ and a matroid $\mathcal{M} = (E, \mathcal{I})$. Find a subset $S \in \mathcal{I}$ that approximately maximizes F_D in a manner that guarantees differential privacy with respect to the input dataset D.*

Furthermore, we consider a natural generalization of submodular functions, namely, k-submodular functions. k-submodular function maximization allows for a richer problem structure than submodular maximization. For instance, coupled feature selection [199], sensor placement with k kinds of measures [179], and influence maximization with k topics can be expressed as k-submodular function maximization problems. To motivate the privacy concerns, consider the next example. More examples are given in Section 8.5.2.

Example 8.1.3 (Influence Maximization with k Topics)**.** For k topics, a sensitive dataset is a directed graph $G = (V, E)$ with an edge probability $p^i_{u,v}$ for each edge $(u, v) \in E$, representing the strength of influence from u to v on the i-th topic. The goal is to distribute these topics to N vertices of the graph so that we maximize *influence spread*. The problem of maximizing influence spread can be formulated as k-submodular function maximization problem [179]. An example for this setting is in viral marketing where dataset consists of a directed graph where each vertex represents a user and each edge represents the friendship between a pair of users. Given k kinds of products, the objective is to promote products by giving (discounted) items to a selected group of influential people in the hope that large number of product adoptions will occur. Here, besides maximizing the influence spread, it is important to preserve the privacy of individuals in the dataset.

Problem 8.1.4. *Given a sensitive dataset D associated to a monotone k-submodular function $F_D\colon (k+1)^E \to \mathbb{R}_+$ and a matroid $\mathcal{M} = (E, \mathcal{I})$. Find $S = (S_1, \ldots, S_k)$ with $\bigcup_{i\in[k]} S_i \in \mathcal{I}$ that approximately maximizes F_D in a manner that guarantees differential privacy with respect to the input dataset D.*

8.1.1 Overview of our contributions

Submodular Maximization: For maximizing a nonnegative monotone submodular function subject to a matroid constraint, we show that a modification of the *continuous greedy* algorithm [48] yields a good approximation guarantee as well as a good privacy guarantee. Following the same idea, we maximize the so-called *multilinear extension* of the input submodular function in the corresponding *matroid polytope*, denoted by $\mathcal{P}(\mathcal{M})$. However, in order to greedily choose a direction, it requires to have a *discretization* of the matroid polytope. Fortunately, due to [221], an efficient discretization can be achieved. That is, we can *cover* a polytope with a small number of balls in polynomial time. Having these in hand, we prove the following.

Theorem 8.1.5. *Suppose F_D is monotone with sensitivity Δ and $\mathcal{M} = (E, \mathcal{I})$ is a matroid. For every $\epsilon > 0$, there is an $(\epsilon r(\mathcal{M})^2)$-differentaily private algorithm that, with high probability, returns $S \in \mathcal{I}$ with quality at least $(1 - \frac{1}{\mathrm{e}})OPT - O\left(\sqrt{\epsilon} + \frac{\Delta r(\mathcal{M})|E| \ln |E|}{\epsilon^3}\right)$.*

For covering C of $\mathcal{P}(\mathcal{M})$, the algorithm in Theorem 8.1.5 makes $O(r(\mathcal{M})|E||C|)$ queries to the *evaluation oracle*. We point out that C has a size of roughly $|E|^{1/\epsilon^2}$. In Section 8.4, we present an algorithm that makes significantly fewer queries to the evaluation oracle.

Theorem 8.1.6. *Suppose F_D is monotone and has sensitivity Δ and $\mathcal{M} = (E, \mathcal{I})$ is a matroid. For every $\epsilon > 0$, there is an $(\epsilon r(\mathcal{M})^2)$-differentaily private algorithm that, with high probability, returns $S \in \mathcal{I}$ with quality at least $(1-\frac{1}{\mathrm{e}})OPT - O\left(\sqrt{\epsilon} + \frac{\Delta r(\mathcal{M})|E| \ln(|E|/\epsilon)}{\epsilon^3}\right)$. Moreover, this algorithm makes at most $O(r(\mathcal{M})|E|^2 \ln \frac{|E|}{\epsilon})$ queries to the evaluation oracle.*

k-submodular Maximization: To the best of our knowledge, there is no algorithm for maximizing k-submodular functions concerning differential privacy. We study Problem 8.1.4 in Section 8.5. First, we discuss an $(\epsilon r(\mathcal{M}))$-differentially private algorithm that uses the evaluation oracle at most $O(kr(\mathcal{M})|E|)$ times and outputs a solution with quality at least $1/2$ of the optimal one.

Theorem 8.1.7. *Suppose $F_D : (k+1)^E \to \mathbb{R}_+$ is monotone and has sensitivity Δ. For any $\epsilon > 0$, there is an $O(\epsilon r(\mathcal{M}))$-differentially private algorithm that, with high probability, returns a solution $X = (X_1, \ldots, X_k) \in (k+1)^E$ with $\bigcup_{i\in[k]} X_i \in \mathcal{I}$ and $F_D(X) \geq \frac{1}{2}\mathrm{OPT} - O(\frac{\Delta r(\mathcal{M}) \ln |E|}{\epsilon})$ by evaluating F_D at most $O(kr(\mathcal{M})|E|)$ times.*

This $1/2$ approximation ratio is asymptotically tight due to the hardness result in [119]. Applying a sampling technique [168, 171, 179], we propose an algorithm that preserves the

same privacy guarantee and the same quality as before while evaluating F_D almost linear number of times, namely $O\left(k|E|\ln r(\mathcal{M})\ln\frac{r(\mathcal{M})}{\gamma}\right)$. Here, γ is the failure probability of our algorithm.

8.1.2 Related work

Gupta et al. [94] considered an important case of Problem 8.1.2 called the *Combinatorial Public Projects* (CPP problem). The CPP problem was introduced by Papadimitriou, Schapira, and Singer [180] and is as follows. For a data set $D = (x_1, \ldots, x_n)$, each individual x_i submits a *private* non-decreasing and submodular valuation function $F_{x_i}\colon 2^E \to [0,1]$. Our goal is to select a subset $S \subseteq E$ of size k to maximize function F_D that takes the particular form $F_D(S) = \frac{1}{n}\sum_{i=1}^{n} F_{x_i}(S)$. Note that in this setting, the sensitivity can be always bounded from above by $\frac{1}{n}$. Gupta et al. showed the following.

Theorem 8.1.8 ([94])**.** *For any $\delta \leq 1/2$, there is an (ϵ,δ)-differentially private algorithm for the CPP problem under cardinality constraint that, with high probability, returns a solution $S \subseteq E$ of size k with quality at least $(1-\frac{1}{\mathrm{e}})\mathrm{OPT} - O(\frac{k\ln(\mathrm{e}/\delta)\ln|E|}{\epsilon})$.*

There are many cases which do not fall into the CPP framework. For some problems, including feature selection via mutual information (Example 8.1.1), the submodular function F_D of interest depends on the dataset D in ways much more complicated than averaging functions associated to each individual. Unfortunately, the privacy analysis of Theorem 8.1.8 heavily relies on the assumption that the input function $F_D = \frac{1}{n}\sum_{i=1}^{n} F_{x_i}(S)$ is the average of F_{x_i}'s, and does not directly generalize to arbitrary submodular functions. Using a *composition theorem* for differentially private mechanisms, Mitrovic et al. [171] proved the following

Theorem 8.1.9 ([171])**.** *Suppose F_D is monotone and has sensitivity Δ. For any $\epsilon > 0$, there is a $(k\epsilon)$-differentially private algorithm that, with high probability, returns $S \subseteq E$ of size k with quality at least $\left(1-\frac{1}{\mathrm{e}}\right)\mathrm{OPT} - O\left(\frac{\Delta k\ln|E|}{\epsilon}\right)$.*

In the same work [171], authors considered matroid constraints and more generally *p-extendable* constraints.

Theorem 8.1.10 ([171])**.** *Suppose F_D is monotone with sensitivity Δ and let $\mathcal{M} = (E,\mathcal{I})$ be a matroid. Then for any $\epsilon > 0$, there is an $(\epsilon r(\mathcal{M}))$-differentially private algorithm that, with high probability, returns a solution $S \in \mathcal{I}$ with quality at least $\frac{1}{2}\mathrm{OPT} - O\left(\frac{\Delta r(\mathcal{M})\ln|E|}{\epsilon}\right)$.*

k-submodular Maximization: The terminology for k-submodular functions was first introduced in [117] while the concept has been studied previously in [54]. Note for $k = 1$ the notion of k-submodularity is the same as submodularity. For $k = 2$, this notion is known as *bisubmodularity*. Bisubmodularity arises in bicooperative games [27] as well as variants

of sensor placement problems and coupled feature selection problems [199]. For unconstrained nonnegative k-submodular maximization, [217] proposed a $\max\{1/3, 1/(1+a)\}$-approximation algorithm where $a = \max\{1, \sqrt{(k-1)/4}\}$. The approximation ratio was improved to $1/2$ in [119]. They also provided $k/(2k-1)$-approximation for maximization of monotone k-submodular functions. The problem of maximizing a monotone k-submodular function was considered in [179] subject to different constraints. They gave a $1/2$-approximation algorithm for total size constraint, i.e., $|\bigcup_{i\in[k]} X_i| \leq N$, and $1/3$-approximation algorithm for individual size constraints, i.e., $|X_i| \leq N_i$ for $i = 1, \ldots, k$. [195] proved that $1/2$-approximation can be achieved for matroid constraint, i.e., $\bigcup_{i\in[k]} X_i \in \mathcal{I}$.

8.2 Preliminaries

Multilinear extension. The *multilinear extension* $f\colon [0,1]^E \to \mathbb{R}$ of a set function $F\colon 2^E \to \mathbb{R}$ is

$$f(\mathbf{x}) = \sum_{S\subseteq E} F(S) \prod_{e\in S} \mathbf{x}(e) \prod_{e\notin S} (1 - \mathbf{x}(e)).$$

There is a probabilistic interpretation of the multilinear extension. Given $\mathbf{x} \in [0,1]^E$ we can define X to be the random subset of E in which each element $e \in E$ is included independently with probability $\mathbf{x}(e)$ and is not included with probability $1-\mathbf{x}(e)$. We write $X \sim \mathbf{x}$ to denote that X is a random subset sampled this way from $\mathbf{x}$. Then we can simply write f as

$$f(\mathbf{x}) = \mathbb{E}_{X\sim\mathbf{x}}[F(X)].$$

Observe that for all $S \subseteq E$ we have $f(\mathbf{1}_S) = F(S)$. The following is well known:

Proposition 8.2.1 ([48])**.** *Let $f\colon [0,1]^E \to \mathbb{R}$ be the multilinear extension of a monotone submodular function $F\colon 2^E \to \mathbb{R}$. Then*

1. *f is monotone, meaning $\frac{\partial f}{\partial \mathbf{x}(e)} \geq 0$. Hence, $\nabla f(\mathbf{x}) = (\frac{\partial f}{\partial \mathbf{x}(1)}, \ldots, \frac{\partial f}{\partial \mathbf{x}(n)})$ is a nonnegative vector.*
2. *f is concave along any direction $\mathbf{d} \geq \mathbf{0}$.*

k-submodular functions. Given a natural number $k \geq 1$, a function $F\colon (k+1)^E \to \mathbb{R}_+$ defined on k-tuples of pairwise disjoint subsets of E is called *k-submodular* if for all k-tuples $S = (S_1, \ldots, S_k)$ and $T = (T_1, \ldots, T_k)$ of pairwise disjoint subsets of E,

$$F(S) + F(T) \geq F(S \sqcap T) + F(S \sqcup T),$$

where we define

$$S \sqcap T = (S_1 \cap T_1, \ldots, S_k \cap T_k),$$
$$S \sqcup T = \left((S_1 \cup T_1) \setminus \left(\bigcup_{i \neq 1} S_i \cup T_i\right), \ldots, (S_k \cup T_k) \setminus \left(\bigcup_{i \neq k} S_i \cup T_i\right)\right).$$

Matroids polytopes and coverings. Recall that a pair $\mathcal{M} = (E, \mathcal{I})$ of a set E and $\mathcal{I} \subseteq 2^E$ is called a *matroid* if

1) $\emptyset \in \mathcal{I}$,

2) $A \in \mathcal{I}$ for any $A \subseteq B \in \mathcal{I}$, and

3) for any $A, B \in \mathcal{I}$ with $|A| < |B|$, there exists $e \in B \setminus A$ such that $A \cup \{e\} \in \mathcal{I}$.

We call a set in $\mathcal{I}$ an *independent set*. The *rank function* $r_{\mathcal{M}} \colon 2^E \to \mathbb{Z}_+$ of $\mathcal{M}$ is

$$r_{\mathcal{M}}(S) = \max\{|I| : I \subseteq S, I \in \mathcal{I}\}.$$

An independent set $S \in \mathcal{I}$ is called a *base* if $r_{\mathcal{M}}(S) = r_{\mathcal{M}}(E)$. We denote the set of all bases by $\mathcal{B}$ and rank of $\mathcal{M}$ by $r(\mathcal{M})$. The *matroid polytope* $\mathcal{P}(\mathcal{M}) \subseteq \mathbb{R}^E$ of $\mathcal{M}$ is $\mathcal{P}(\mathcal{M}) = \text{conv}\{\mathbf{1}_I : I \in \mathcal{I}\}$, where conv denotes the convex hull. Or equivalently [71],

$$\mathcal{P}(\mathcal{M}) = \{\mathbf{x} \geq \mathbf{0} : \mathbf{x}(S) \leq r_{\mathcal{M}}(S)\ \forall S \subseteq E\}.$$

Note that the matroid polytope is *down-monotone*, that is, for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^E$ with $\mathbf{0} \leq \mathbf{x} \leq \mathbf{y}$ and $\mathbf{y} \in \mathcal{P}(\mathcal{M})$ then $\mathbf{x} \in \mathcal{P}(\mathcal{M})$.

Definition 8.2.2 (ρ-covering)**.** *Let $K \subseteq \mathbb{R}^E$ be a set. For $\rho > 0$, a set $C \subseteq K$ of points is called a* ρ-covering *of K if for any $\mathbf{x} \in K$, there exists $\mathbf{y} \in C$ such that $\|\mathbf{x} - \mathbf{y}\| \leq \rho$.*

Theorem 8.2.3 (Theorem 5.5 of [221], paraphrased)**.** *Let $\mathcal{M} = (E, \mathcal{I})$ be a matroid. For every $\epsilon > 0$, we can construct an ϵB-cover C of $\mathcal{P}(\mathcal{M})$ of size $|E|^{O(1/\epsilon^2)}$ in $|E|^{O(1/\epsilon^2)}$ time, where B is the maximum ℓ_2-norm of a point in $\mathcal{P}(\mathcal{M})$.*

8.2.1 Differential privacy

The definition of differential privacy relies on the notion of neighboring datasets. Recall that two datasets are *neighboring* if they differ in a single record. When two datasets D, D' are neighboring, we write $D \sim D'$.

Definition 8.2.4 ([66])**.** *For $\epsilon, \delta \in \mathbb{R}_+$, we say that a randomized computation M is* (ϵ, δ)-differentially private *if for any neighboring datasets $D \sim D'$, and for any set of outcomes $S \subseteq \text{range}(M)$,*

$$\Pr[M(D) \in S] \leq \exp(\epsilon) \Pr[M(D') \in S] + \delta.$$

When $\delta = 0$, we say M is ϵ-differentially private.

In our case, a dataset D consists of *private* submodular functions $F_1, \ldots, F_n \colon 2^E \to [0,1]$. Two datasets D and D' are neighboring if all but one submodular function in those datasets are equal. The submodular function F_D depends on the dataset D in different ways, for example $F_D(S) = \sum_{i=1}^{n} F_i(S)/n$ (CPP problem), or much more complicated ways than averaging functions associated to each individual.

Differentially private algorithms must be calibrated to the sensitivity of the function of interest with respect to small changes in the input dataset, defined formally as follows.

Definition 8.2.5. *The sensitivity of a function $F_D \colon X \to Y$, parameterized by a dataset D, is defined as*

$$\max_{D' : D' \sim D} \max_{x \in X} |F_D(x) - F_{D'}(x)|.$$

A function with sensitivity Δ is called Δ-sensitive.

Composition of differential privacy. Let $\{(\epsilon_i, \delta_i)\}_{i=1}^{k}$ be a sequence of privacy parameters and let M^* be a mechanism that behaves as follows on an input D. In each of rounds $i = 1, \ldots, k$, the algorithm M^* selects an (ϵ_i, δ_i)-differentially private algorithm M_i possibly depending on the previous outcomes $M_1(D), \ldots, M_i(D)$ (but not directly on the sensitive dataset D itself), and releases $M_i(D)$. The output of M^* is informally referred as the *k-fold adaptive composition* of (ϵ_i, δ_i)-differentially private algorithms. For a formal treatment of adaptive composition, see [68, 69]. We have the following guarantee on the differential privacy of the composite algorithm.

Theorem 8.2.6. *[44, 67, 69] The k-fold adaptive composition of k (ϵ_i, δ_i)-differentially private algorithms, with $\epsilon_i \le \epsilon_0$ and $\delta_i \le \delta_0$ for every $1 \le i \le k$, satisfies (ϵ, δ)-differential privacy where*

- *$\epsilon = k\epsilon_0$ and $\delta = k\delta_0$ (the basic composition), or*
- *$\epsilon = \frac{1}{2} k \epsilon_0^2 + \sqrt{2 \ln 1/\delta'} \epsilon_0$ and $\delta = \delta' + k\delta$ for any $\delta' > 0$ (the advanced composition).*

Exponential Mechanism. One particularly general tool that we will use is the *exponential mechanism* of [166]. The exponential mechanism is defined in terms of a *quality function* $q_D \colon \mathcal{R} \to \mathbb{R}$, which is parameterized by a dataset D and maps a candidate result $R \in \mathcal{R}$ to a real-valued score.

Definition 8.2.7 ([166])**.** *Let $\epsilon, \Delta > 0$ and let $q_D \colon \mathcal{R} \to \mathbb{R}$ be a quality score. Then, the* exponential mechanism *$EM(\epsilon, \Delta, q_D)$ outputs $R \in \mathcal{R}$ with probability proportional to* $\exp\left(\frac{\epsilon}{2\Delta} \cdot q_D(R)\right)$.

Theorem 8.2.8 ([166])**.** *Suppose that the quality score $q_D \colon \mathcal{R} \to \mathbb{R}$ is Δ-sensitive. Then, $EM(\epsilon, \Delta, q_D)$ is ϵ-differentially private, and for every $\beta \in (0,1)$ outputs $R \in \mathcal{R}$ with*

$$\Pr\left[q_D(R) \geq \max_{R' \in \mathcal{R}} q_D(R') - \frac{2\Delta}{\epsilon} \ln\left(\frac{|\mathcal{R}|}{\beta}\right)\right] \geq 1 - \beta.$$

8.2.2 Probability distributions

Let P be a probability distribution over a finite set E. For an element $e \in E$, we write $P(e)$ to denote the probability that e is sampled from P.

Let P and Q be two distributions over the same set E. The *total variation distance* and the *Hellinger distance* between P and Q are

$$d_{\mathrm{TV}}(P,Q) = \frac{1}{2}\sum_{e \in E} |P(e) - Q(e)| \quad \text{and}$$
$$h(P,Q) = \frac{1}{\sqrt{2}}\sqrt{\sum_{e \in E}\left(\sqrt{P(e)} - \sqrt{Q(e)}\right)^2},$$

respectively. It is well known that $d_{\mathrm{TV}}(P,Q) \leq \sqrt{2}h(P,Q)$ holds.

For two distributions P and Q, we denote by $P \otimes Q$ their product distribution. The following is well known:

Lemma 8.2.9. *Let $P_1, \ldots, P_n$ and $Q_1, \ldots, Q_n$ be probability distributions over E. Then, we have*

$$h(P_1 \otimes P_2 \otimes \cdots \otimes P_n, Q_1 \otimes Q_2 \otimes \cdots \otimes Q_n)^2 \leq \sum_{i=1}^{n} h(P_i, Q_i)^2.$$

Finally, we use the following result due to Hoeffding in order to bound the error of our sampling step in Section 8.4.

Theorem 8.2.10 (Hoeffding's inequality [115])**.** *Let $X_1, \ldots, X_n$ be independent random variables bounded by the interval $[0,1] : 0 \leq X_i \leq 1$. We define the empirical mean of these variables by $\bar{X} = \frac{1}{n}(X_1 + \cdots + X_n)$. Then*

$$\Pr[\bar{X} - E[\bar{X}] \geq t] \leq \exp(-2nt^2).$$

8.3 Differentially private continuous greedy algorithm

In this section we prove Theorem 8.1.5. Throughout this section, we fix (private) monotone submodular functions $F_1, \ldots, F_n \colon 2^E \to [0,1]$, $\epsilon, \delta > 0$, and a matroid $M = (E, \mathcal{I})$.

Let $\mathbf{x}^* \in \mathcal{P}(\mathcal{M})$ be a maximizer of f_D. We drop the subscript D when it is clear from the context. Our algorithm (Algorithm 7) is a modification of the continuous greedy algorithm [48].

Algorithm 7 Differentially Private Continuous Greedy

Input: Submodular function $F_D \colon 2^E \to [0,1]$, dataset D, matroid $\mathcal{M} = (E, \mathcal{I})$, and $\epsilon > 0$ and $\rho \geq 0$.
Let C_ρ be a ρ-covering of $\mathcal{P}(\mathcal{M})$, and f_D be the multilinear extension of F_D.
$\mathbf{x}_0 \leftarrow \mathbf{0}$, $\epsilon' \leftarrow \frac{\epsilon}{2\Delta}$.
$\alpha \leftarrow \frac{1}{T}$, where $T = r(\mathcal{M})$.
for $t = 1$ to T **do**
    Sample $\mathbf{y} \in C_\rho$ with probability proportional to $\exp(\epsilon' \langle \mathbf{y}, \nabla f_D(\mathbf{x}_{t-1}) \rangle)$.
    Let $\mathbf{y}_{t-1}$ be the sampled vector.
    $\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} + \alpha \mathbf{y}_{t-1}$.
Output: $\mathbf{x}_T$

8.3.1 Approximation guarantee

Lemma 8.3.1. *For every $\mathbf{x}, \mathbf{v} \in [0,1]^E$ with $\|\mathbf{v}\|_2 \leq \rho$ and $\mathbf{x} + \mathbf{v} \in [0,1]^E$, we have $|f(\mathbf{x}) - f(\mathbf{x} + \mathbf{v})| \leq 4\sqrt[4]{|E|}\sqrt{\rho}$.*

Lemma 8.3.2. *Suppose $\mathbf{y} \in [0,1]^E$ satisfies $\|\mathbf{y} - \mathbf{x}^*\|_2 \leq \rho$. Then for any $\mathbf{x} \in [0,1]^E$, we have $\langle \mathbf{y}, \nabla f(\mathbf{x}) \rangle \geq f(\mathbf{x}^*) - f(\mathbf{x}) - C_{8.3.2}\sqrt{\rho}$ for some constant $C_{8.3.2} > 0$.*

Proof. First, we show

$$\langle \mathbf{y}, \nabla f(\mathbf{x}) \rangle \geq f(\mathbf{y}) - f(\mathbf{x}).$$

Let us consider a direction $\mathbf{d} \in [0,1]^E$ such that $\mathbf{d}(e) = \max\{\mathbf{y}(e) - \mathbf{x}(e), 0\}$ for every $e \in E$. Then, we have

$$\begin{aligned} \langle \mathbf{y}, \nabla f(\mathbf{x}) \rangle &\geq \langle \mathbf{d}, \nabla f(\mathbf{x}) \rangle \\ &\geq f(\mathbf{x} + \mathbf{d}) - f(\mathbf{x}) \\ &\geq f(\mathbf{y}) - f(\mathbf{x}), \end{aligned}$$

where the first inequality follows from $\mathbf{y} \geq \mathbf{d}$ and $\nabla f(\mathbf{x}) \geq 0$, the second inequality follows from the concavitity of f along $\mathbf{d}$, and the third inequality follows from $\mathbf{x} + \mathbf{d} \geq \mathbf{y}$ and the monotonicity of f. By Lemma 8.3.1, we have

$$f(\mathbf{y}) \geq f(\mathbf{x}^*) - 4\sqrt[4]{|E|}\sqrt{\rho},$$

which yields the desired result with $C_{8.3.2} = 4\sqrt[4]{|E|}$. □

Theorem 8.3.3. *Suppose F_D is Δ-sensitive and C_ρ is a ρ-covering of $\mathcal{P}(\mathcal{M})$. Then Algorithm 7, with high probability, returns $\mathbf{x}_T \in \mathcal{P}(\mathcal{M})$ such that*

$$f_D(\mathbf{x}_T) \geq \left(1 - \frac{1}{\mathrm{e}}\right) \mathrm{OPT} - O\left(C_{8.3.2}\rho + \frac{\Delta r(\mathcal{M}) \ln |E|}{\epsilon \rho^2}\right)$$

Moreover, the algorithm evaluates f_D at most $O\left(r(\mathcal{M}) \cdot |C_\rho|\right)$ times.

Proof. Clearly Algorithm 7 evaluates f at most $O\left(r(\mathcal{M})|C_\rho|\right)$ times. Observe that the algorithm forms a convex combination of T vertices of the polytope $\mathcal{P}(\mathcal{M})$, each with weight α hence $\mathbf{x}_T \in \mathcal{P}(\mathcal{M})$. In what follows, we focus on the quality of the output of the algorithm. Suppose $\mathbf{y}' \in C_\rho$ with $\|\mathbf{y}' - \mathbf{x}^*\|_2 \leq \rho$. By Theorem 8.2.8, with probability at least $1 - \frac{1}{|E|^2}$, we have

$$\begin{aligned}
\langle \mathbf{y}_t, \nabla f(\mathbf{x}_t)\rangle &\geq \operatorname*{argmax}_{\mathbf{y}\in C_\rho} \langle \mathbf{y}, \nabla f(\mathbf{x}_t)\rangle - \frac{2\Delta}{\epsilon}\ln(|E|^2|C_\rho|) \\
&\geq \langle \mathbf{y}', \nabla f(\mathbf{x}_t)\rangle - \frac{2\Delta}{\epsilon}\ln(|E|^2|C_\rho|) \\
&\overset{\text{By Lemma 8.3.2}}{\geq} f(\mathbf{x}^*) - f(\mathbf{x}_t) - C_{8.3.2}\sqrt{\rho} - \frac{2\Delta}{\epsilon}\ln(|E|^2|C_\rho|)
\end{aligned}$$

By a union bound, with probability at least $1 - \frac{1}{poly(|E|)}$, the above inequality holds for every t. In what follows, we assume this has happened. Further, let us assume that t is a continuous variable in $[0, T]$. We remark that discretization of t in our algorithm introduces error into the approximation guarantee. However, this can be handled by sufficiently large T, say, $r(\mathcal{M})$ as in Algorithm 7, and small step size α [48]. In what follows t is assumed to be continuous and we write $\frac{d\mathbf{x}_t}{dt} = \alpha \mathbf{y}_t$, hence

$$\begin{aligned}
\frac{df(\mathbf{x}_t)}{dt} &= \sum_e \frac{\partial f(\mathbf{x}_t(e))}{\partial \mathbf{x}_t(e)} \frac{d\mathbf{x}_t(e)}{dt} \\
&= \nabla f(\mathbf{x}_t) \cdot \frac{d\mathbf{x}_t}{dt} = \alpha \langle \mathbf{y}_t, \nabla f(\mathbf{x}_t)\rangle \\
&\geq \alpha\Big(f(\mathbf{x}^*) - f(\mathbf{x}_t) - C_{8.3.2}\sqrt{\rho} - \frac{2\Delta}{\epsilon}\ln(|E|^2|C_\rho|)\Big),
\end{aligned}$$

where the first equality follows from the chain rule. Let $\beta = f(\mathbf{x}^*) - C_{8.3.2}\sqrt{\rho} - \frac{2\Delta}{\epsilon}\ln(|E|^2|C_\rho|)$. Solving the following differential equation $\frac{df(\mathbf{x}_t)}{dt} = \alpha(\beta - f(\mathbf{x}_t))$ with $f(\mathbf{x}_0) = 0$ gives us $f(\mathbf{x}_t) = \beta(1 - \mathrm{e}^{-\alpha t})$. For $\alpha = \frac{1}{T}, t = T$ we obtain

$$\begin{aligned}
f(\mathbf{x}_T) &= \beta(1 - \mathrm{e}^{-1}) \\
&= \left(1 - \frac{1}{\mathrm{e}}\right) f(\mathbf{x}^*) - O\left(C_{8.3.2}\sqrt{\rho} + \frac{2\Delta}{\epsilon}\ln(|E|^2|C_\rho|)\right) \\
&= \left(1 - \frac{1}{\mathrm{e}}\right) f(\mathbf{x}^*) - O\left(C_{8.3.2}\sqrt{\rho} + \frac{\Delta}{\epsilon}(\ln|E| + \ln|E|^{\left(\frac{B}{\rho}\right)^2})\right) \\
&= \left(1 - \frac{1}{\mathrm{e}}\right) f(\mathbf{x}^*) - O\left(C_{8.3.2}\sqrt{\rho} + \frac{\Delta}{\epsilon}\left(\frac{B}{\rho}\right)^2 \ln|E|\right) \qquad (B^2 \leq r(\mathcal{M})) \\
&\geq \left(1 - \frac{1}{\mathrm{e}}\right) f(\mathbf{x}^*) - O\left(C_{8.3.2}\sqrt{\rho} + \frac{\Delta r(\mathcal{M})\ln|E|}{\epsilon\rho^2}\right) \qquad \square
\end{aligned}$$

Remark 8.3.4. *As already pointed out in the proof of Theorem 8.3.3, the discretization of t introduces error into the approximation guarantee yielding* $(1 - 1/\mathrm{e} - 1/\mathrm{poly}(|E|))\mathrm{OPT}$.

However, this can be shaved off to $(1-1/\mathrm{e})\mathrm{OPT}$ *by sufficiently large* T *[48]. Moreover, evaluating* f *(even approximately) is expensive. To achieve the nearly optimal approximation guarantees, the evaluation error needs to be very small and in a lot of cases, the error needs to be* $O(1/|E|)$ *times the function value. As a result, a single evaluation of the multilinear extension* f *requires* $\Omega(|E|)$ *evaluations of* F *(see [72] for recent improvement). Therefore, our algorithm requires* $O(r(\mathcal{M})|E||C_\rho|)$ *evaluation of* F*.*

Remark 8.3.5. *From a fractional solution* $\mathbf{x}^*$*, we can obtain an integral solution* $\mathbf{s} \in \{0,1\}^E$ *such that* $f(\mathbf{s}) \geq f(\mathbf{x}^*)$*. Such an integer solution corresponds to a vertex of* $\mathcal{P}(\mathcal{M})$ *and hence a discrete solution* $S \in \mathcal{I}$*. This can be done using the so-called* swap rounding *[50].*

8.3.2 Privacy analysis

Theorem 8.3.6. *Algorithm 7 preserves* $O(\epsilon r(\mathcal{M})^2)$*-differential privacy.*

Proof. Let D and D' be two neighboring datasets and $F_D, F_{D'}$ be their associated functions. For a fixed $\mathbf{y}_t \in C_\rho$, we consider the relative probability of Algorithm 7 (denoted by M) choosing $\mathbf{y}_t$ at time step t given multilinear extensions of F_D and $F_{D'}$. Let $M_t(f_D \mid \mathbf{x}_t)$ denote the output of M at time step t given dataset D and point $\mathbf{x}_t$. Similarly, $M_t(f_{D'} \mid \mathbf{x}_t)$ denotes the output of M at time step t given dataset D' and point $\mathbf{x}_t$. Further, write $d_{\mathbf{y}} = \langle \mathbf{y}, \nabla f_D(\mathbf{x}_t)\rangle$ and $d'_{\mathbf{y}} = \langle \mathbf{y}, \nabla f_{D'}(\mathbf{x}_t)\rangle$. We have

$$\begin{aligned}\frac{\Pr[M_t(f_D \mid \mathbf{x}_t) = \mathbf{y}_t]}{\Pr[M_t(f_{D'} \mid \mathbf{x}_t) = \mathbf{y}_t]} &= \frac{\exp(\epsilon' \cdot d_{\mathbf{y}_t}) / \sum_{\mathbf{y}\in C_\rho} \exp(\epsilon' \cdot d_{\mathbf{y}})}{\exp(\epsilon' \cdot d'_{\mathbf{y}_t}) / \sum_{\mathbf{y}\in C_\rho} \exp(\epsilon' \cdot d'_{\mathbf{y}})} \\ &= \frac{\exp(\epsilon' \cdot d_{\mathbf{y}_t})}{\exp(\epsilon' \cdot d'_{\mathbf{y}_t})} \cdot \frac{\sum_{\mathbf{y}\in C_\rho} \exp(\epsilon' \cdot d'_{\mathbf{y}})}{\sum_{\mathbf{y}\in C_\rho} \exp(\epsilon' \cdot d_{\mathbf{y}})}.\end{aligned}$$

For the first factor, we have

$$\begin{aligned}\frac{\exp(\epsilon' \cdot d_{\mathbf{y}_t})}{\exp(\epsilon' \cdot d'_{\mathbf{y}_t})} &= \exp(\epsilon'(d_{\mathbf{y}_t} - d'_{\mathbf{y}_t})) \\ &= \exp(\epsilon'(\langle \mathbf{y}_t, \nabla f_D(\mathbf{x}_t) - \nabla f_{D'}(\mathbf{x}_t)\rangle)) \\ &\leq \exp(\epsilon' \|\mathbf{y}_t\|_1 \|\nabla f_D(\mathbf{x}_t) - \nabla f_{D'}(\mathbf{x}_t)\|_\infty) \\ &= \exp\left(\epsilon' \sum_{e\in E} \mathbf{y}_t(e) \cdot \left(\max_{e\in E} \mathop{\mathbb{E}}_{R\sim\mathbf{x}_t} \left[F_D(R\cup\{e\}) - F_D(R) - F_{D'}(R\cup\{e\}) + F_{D'}(R)\right]\right)\right) \\ &\leq \exp(O(\epsilon' \cdot r(\mathcal{M}) \cdot 2\Delta)) = \exp(O(\epsilon \cdot r(\mathcal{M})))\end{aligned}$$

Note that the last inequality holds since $\mathbf{y}_t$ is a member of the matroid polytope $\mathcal{P}(\mathcal{M})$ and by definition we have $\sum_{e\in E} \mathbf{y}_t(e) \leq r_{\mathcal{M}}(E) = r(\mathcal{M})$. Moreover, recall that F_D is Δ-sensitive.

For the second factor, let us write $\beta_{\mathbf{y}} = d'_{\mathbf{y}} - d_{\mathbf{y}}$ to be the *deficit* of the probabilities of choosing direction $\mathbf{y}$ in instances $f_{D'}$ and f_D. Then, we have

$$\begin{aligned}\frac{\sum_{\mathbf{y}\in C_\rho}\exp(\epsilon'\cdot d'_{\mathbf{y}})}{\sum_{\mathbf{y}\in C_\rho}\exp(\epsilon'\cdot d_{\mathbf{y}})} &= \frac{\sum_{\mathbf{y}\in C_\rho}\exp(\epsilon'\cdot\beta_{\mathbf{y}})\exp(\epsilon'\cdot d_{\mathbf{y}})}{\sum_{\mathbf{y}\in C_\rho}\exp(\epsilon'\cdot d_{\mathbf{y}})}\\ &= \mathbb{E}_{\mathbf{y}}[\exp(\epsilon'\cdot\beta_{\mathbf{y}})] \le \exp\big(O(\epsilon'\cdot r(\mathcal{M})\cdot 2\Delta)\big)\\ &= \exp\big(O(\epsilon\cdot r(\mathcal{M}))\big).\end{aligned}$$

The expectation is taken over the probability distribution over $\mathbf{y}$ selected at time t in instance with input D. Recall that we choose $\mathbf{y}$ with probability proportional to $\exp(\epsilon' d_{\mathbf{y}})$. By a union bound, Algorithm 7 preserves $O(\epsilon T r(\mathcal{M})) \le O(\epsilon r(\mathcal{M})^2)$-differential privacy. To obtain an integral solution from a fractional solution, we use swap rounding technique (see Remark 8.3.5) which does not depend on the input function and hence preserves the privacy. □

Note that the privacy factor in the work of [171] is $O(\epsilon r(\mathcal{M}))$. However, our privacy factor is $O(\epsilon r(\mathcal{M})^2)$, this is because we deal with the multilinear extension of a submodular function rather than the function itself (which is different from the previous works).

Theorem 8.3.7 (Formal version of Theorem 8.1.5)**.** *Suppose F_D is Δ-sensitive and Algorithm 7 is instantiated with $\rho = \frac{\epsilon}{|E|^{1/2}}$. Then Algorithm 7 is $(\epsilon r(\mathcal{M})^2)$-differentially private and, with high probability, returns $S \in \mathcal{I}$ with quality at least*

$$F_D(S) \ge \left(1 - \frac{1}{\mathrm{e}}\right)\mathrm{OPT} - O\left(\sqrt{\epsilon} + \frac{\Delta r(\mathcal{M})|E|\ln|E|}{\epsilon^3}\right)$$

Example 8.3.8 (Maximum Coverage)**.** Let $G = (U, V, E)$ be a bipartite graph, and B be a budget constraint. In Maximum Coverage problem, the goal is to find a set S of B vertices in U so that the number of vertices in V incident to some vertex in S is maximized. The edges incident to a vertex $v \in V$ are private information about v. If we instantiate Theorem 8.3.7 on this problem, the privacy factor is ϵB^2 and the additive error is $O(\Delta B|U|\ln(|U|)/\epsilon^3)$, where Δ is the maximum degree of a vertex in V. To have a meaningful privacy bound, we set $\epsilon \ll 1/B^2$, and the additive error becomes $\Delta B^7|U|\ln(|U|)$. However, OPT could be $\Omega(|V|)$, which is much larger than the additive error when $|V| \gg |U|$. Indeed, by optimizing ρ, we can improve the additive error to $O(\Delta B^3|U|\ln(|U|))$, which will be more practical.

8.4 Improving the query complexity

In this section, we improve the number of evaluations of F from $O(r(\mathcal{M})|E|^{1+(\frac{r(\mathcal{M})}{\epsilon})^2})$ to $O(r(\mathcal{M})|E|^2\ln\frac{|E|}{\epsilon})$. In Algorithm 7, in order to choose a point with probability proportional to $\exp(\langle\mathbf{y}, \nabla f(\mathbf{x})\rangle)$, it requires to compute $Z = \sum_{\mathbf{z}\in C_\rho}\exp(\langle\mathbf{z}, \nabla f(\mathbf{x})\rangle)$. This summation needs

evaluating $(\langle \mathbf{z}, \nabla f(\mathbf{x})\rangle)$ for all $\mathbf{z}$ in C_ρ. One way of improving the query complexity of this step is as follows. Partition C_ρ into a number of layers such that points in each layer are almost the same in terms of the inner product $\langle \cdot, \nabla f(\mathbf{x})\rangle$. Now, instead of choosing a point in C_ρ, we carefully select a layer with some probability (i.e., proportional to its size and *quality* of points in it) and then choose a point from that layer uniformly at random. Of course, to estimate the size of each layer, we need to sample a sufficiently large number of points from C_ρ.

Definition 8.4.1 (layer)**.** *For a point $\mathbf{x} \in C_\rho$ and $\mu > 0$, let the i-th layer to be $\mathcal{L}^{\mathbf{x}}_{\mu,i} = \{\mathbf{z} \in C_\rho \mid (1+\mu)^{i-1} \leq \exp(\langle \mathbf{z}, \nabla f(\mathbf{x})\rangle) < (1+\mu)^i\}$, for $1 \leq i \leq k$, where*

$$k = \left\lceil \log_{1+\mu} \left(\frac{\max\limits_{\mathbf{y} \in C_\rho} \exp(\langle \mathbf{y}, \nabla f(\mathbf{x})\rangle)}{\min\limits_{\mathbf{y} \in C_\rho} \exp(\langle \mathbf{y}, \nabla f(\mathbf{x})\rangle)} \right) \right\rceil.$$

For a layer $\mathcal{L}^{\mathbf{x}}_{\mu,i}$ let $|\mathcal{L}^{\mathbf{x}}_{\mu,i}|$ denote the number of points in it, and define $\tilde{Z} \in \mathbb{R}$ and $\tilde{Z}_i \in \mathbb{R}$ for each $i \in [k]$ as follows:

$$\tilde{Z} = \sum_{i \in [k]} |\mathcal{L}^{\mathbf{x}}_{\mu,i}|(1+\mu)^{i-1} \qquad \text{and} \qquad \tilde{Z}_i = |\mathcal{L}^{\mathbf{x}}_{\mu,i}|(1+\mu)^{i-1}.$$

Then, a layer $\mathcal{L}^{\mathbf{x}}_{\mu,i}$ is chosen with probability $\frac{\tilde{Z}_i}{\tilde{Z}}$. Note that we do not want to spend time computing the exact value of $|\mathcal{L}^{\mathbf{x}}_{\mu,i}|$ for every layer, instead, we are interested in efficiently estimating these values. By Hoeffding's inequality [115], to estimate $|\mathcal{L}^{\mathbf{x}}_{\mu,i}|/|C_\rho|$ with additive error of λ with probability at least $1-\theta$, it suffices to sample $\Theta(\ln(1/\theta)/\lambda^2)$ points from C_ρ. Hence, by a union bound, if we want to estimate $|\mathcal{L}^{\mathbf{x}}_{\mu,i}|/|C_\rho|$ with additive error of λ for all $i = 1, \ldots, k$ with probability at least $1-\theta$, it suffices to sample $\Theta(\ln(k/\theta)/\lambda^2)$ points from C_ρ.

Corollary 8.4.2. *Let C_ρ be a ρ-covering of $\mathcal{P}(\mathcal{M})$ and $\mathbf{x}_t$ be a point in $\mathcal{P}(\mathcal{M})$. Algorithm 8 estimates $|\mathcal{L}^{\mathbf{x}_t}_{\mu,i}|/|C_\rho|$ with an additive error $\lambda_{8.4.2}$ with probability at least $1-\theta_{8.4.2}$.*

Lemma 8.4.3 (Analogous to Theorem 8.2.8)**.** *At each time step t, Algorithm 8 returns $\mathbf{y}_{t-1}$ such that for every $\beta \in (0,1)$ and $\xi = \ln\left(\frac{|C_\rho|(1+k\lambda|C_\rho|)(1+\mu)^{\epsilon'}}{\beta}\right)$ we have*

$$\Pr\left[\langle \mathbf{y}_{t-1}, \nabla f(\mathbf{x}_{t-1})\rangle \geq \max_{\mathbf{z} \in C_\rho} \langle \mathbf{z}, \nabla f(\mathbf{x}_{t-1})\rangle - \frac{2\Delta}{\epsilon}\xi\right] \geq 1-\beta.$$

Theorem 8.4.4. *Suppose F_D is Δ-sensitive and C_ρ is a ρ-covering of $\mathcal{P}(\mathcal{M})$. Then Algorithm 8, with high probability (depending on $\theta_{8.4.2}$), returns $\mathbf{x}_T \in \mathcal{P}(\mathcal{M})$ such that*

$$f(\mathbf{x}_T) \geq \left(1-\frac{1}{e}\right)\mathrm{OPT} - O\left(C_{8.3.2}\sqrt{\rho} + \ln(1+\mu) + \left(\frac{\Delta r(\mathcal{M})}{\epsilon\rho^2}\right)(\ln|E| + \ln(k\lambda_{8.4.2}))\right)$$

Algorithm 8 Improved Differentially Private Continuous Greedy Algorithm

Input: Submodular function $F_D \colon 2^E \to [0,1]$, dataset D, a matroid $\mathcal{M} = (E, \mathcal{I})$, and $\epsilon, \mu, \rho, \lambda, \theta > 0$.
Let C_ρ be a ρ-covering of $\mathcal{P}(\mathcal{M})$, and f_D be the multilinear extension of F_D.
$\mathbf{x}(0) \leftarrow \mathbf{0}$, $\epsilon' \leftarrow \frac{\epsilon}{2\Delta}$.
for $t = 1$ to $T = r(\mathcal{M})$ **do**
    $C'_\rho \leftarrow$ Sample $\Theta(\ln(k/\theta)/\lambda^2)$ points from C_ρ uniformly at random.
    Define $\mathcal{L}_{\mu,i}^{\mathbf{x}_{t-1}}$ as in Definition 8.4.1, and estimate each $|\mathcal{L}_{\mu,i}^{\mathbf{x}_{t-1}}|$ using C'_ρ.
    Let $\tilde{L}_{\mu,i}^{\mathbf{x}_{t-1}}$ denote the estimated value.
    Set $\tilde{Z}_i \leftarrow \tilde{L}_{\mu,i}^{\mathbf{x}_{t-1}}(1+\mu)^{\epsilon'(i-1)}$ and $\tilde{Z} \leftarrow \sum_{i \in [k]} \tilde{L}_{\mu,i}^{\mathbf{x}_{t-1}}(1+\mu)^{\epsilon'(i-1)}$
    Let $\mathcal{L}$ be the chosen layer $\mathcal{L}_{\mu,i}^{\mathbf{x}_{t-1}}$ with probability proportional to $\frac{\tilde{Z}_i}{\tilde{Z}}$.
    Let $\mathbf{y}_{t-1}$ be a point sampled uniformly at random from $\mathcal{L}$.
    $\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} + \alpha \mathbf{y}_{t-1}$.
Outout: $\mathbf{x}_T$

Theorem 8.4.5. *Algorithm 8 preserves $O\left(\epsilon r(\mathcal{M})^2\right)$-differential privacy.*

Theorem 8.4.6 (Formal version of Theorem 8.1.6)**.** *Suppose F_D is Δ-sensitive and Algorithm 8 is instantiated with $\rho = \frac{\epsilon}{|E|^{1/2}}, \mu = \mathrm{e}^\epsilon, \lambda_{8.4.2} = 1/\sqrt{|E|}, \theta_{8.4.2} = 1/|E|^2$. Then Algorithm 8 is $(\epsilon r(\mathcal{M})^2)$-differentially private and, with high probability, returns $S \in \mathcal{I}$ with quality at least*

$$F_D(S) \geq \left(1 - \frac{1}{e}\right) \mathrm{OPT} - O\left(\sqrt{\epsilon} + \frac{\Delta r(\mathcal{M}) |E| \ln(\frac{|E|}{\epsilon})}{\epsilon^3}\right).$$

Moreover, it evaluates F_D at most $O(r(\mathcal{M})|E|^2 \ln(\frac{|E|}{\epsilon}))$ times.

8.5 k-submodular function maximization

In this section, we study a natural generalization of submodular functions, namely k-submodular functions. Associate $(S_1, \ldots, S_k) \in (k+1)^E$ with $\mathbf{s} \in \{0, 1, \ldots, k\}^E$ by $S_i = \{e \in E \mid \mathbf{s}(e) = i\}$ for $i \in [k]$ and define the *support* of $\mathbf{s}$ as $\mathrm{supp}(\mathbf{s}) = \{e \in E \mid \mathbf{s}(e) \neq 0\}$. Let $\preceq$ be a partial ordering on $(k+1)^E$ such that, for $\mathbf{s} = (S_1, \ldots, S_k)$ and $\mathbf{t} = (T_1, \ldots, T_k)$ in $(k+1)^E$, $\mathbf{s} \preceq \mathbf{t}$ if $S_i \subseteq T_i$ for every $i \in [k]$. We say that a function $F \colon (k+1)^E \to \mathbb{R}_+$ is *monotone* if $F(\mathbf{s}) \leq F(\mathbf{t})$ holds for every $\mathbf{s} \preceq \mathbf{t}$. Define the *marginal gain* of adding $e \notin \bigcup_{\ell \in [k]} S_\ell$ to the i-th set of $\mathbf{s} \in (k+1)^E$ to be

$$\Delta_{e,i} F(\mathbf{s}) = F(S_1, \ldots, S_{i-1}, S_i \cup \{e\}, S_{i+1}, \ldots, S_k) - F(S_1, \ldots, S_k).$$

The monotonicity of F is equivalent to $\Delta_{e,i} F(\mathbf{s}) \geq 0$ for any $\mathbf{s} = (S_1, \ldots, S_k)$ and $e \notin \bigcup_{\ell \in [k]} S_\ell$ and $i \in [k]$.

Algorithm 9 Differentially private k-submodular maximization with a matroid constraint

Input: monotone k-submodular functions $F_D\colon (k+1)^E \to [0,1]$, a matroid $\mathcal{M} = (E, \mathcal{I})$, and $\epsilon > 0$.
$\mathbf{x} \leftarrow \mathbf{0}$, $\epsilon' \leftarrow \frac{\epsilon}{2\Delta}$
for $t = 1$ to $r(\mathcal{M})$ **do**
    Let $\Lambda(\mathbf{x}) = \{e \in E \setminus \mathrm{supp}(\mathbf{x}) \mid \mathrm{supp}(\mathbf{x}) \cup \{e\} \in \mathcal{I}\}$
    Choose $e \in \Lambda(\mathbf{x})$ and $i \in [k]$ with probability proportional to $\exp(\epsilon' \Delta_{e,i} F_D(\mathbf{x}))$.
    $\mathbf{x}(e) \leftarrow i$.
Output: $\mathbf{x}$

Our goal is maximizing a monotone k-submodular function under matroid constraints. That is, given a monotone k-submodular function $F_D\colon (k+1)^E \to \mathbb{R}_+$ and a matroid $\mathcal{M} = (E, \mathcal{I})$, we want to solve the following problem.

$$\max_{\mathbf{x} \in (k+1)^E} F_D(\mathbf{x}) \quad \text{subject to } \bigcup_{i \in [k]} X_i \in \mathcal{I}$$

The following are known due to [195]. They may have appeared in other literature that we are not aware of.

Lemma 8.5.1 ([195])**.** *For any maximal optimal solution* $\mathbf{o}$ *we have* $|\mathrm{supp}(\mathbf{o})| = r(\mathcal{M})$.

Lemma 8.5.2 ([195])**.** *Suppose* $A \in \mathcal{I}$ *and* $B \in \mathcal{B}$ *(recall* $\mathcal{B}$ *denotes the set of bases) satisfy* $A \subseteq B$*. Then, for any* $e \notin A$ *satisfying* $A \cup \{e\} \in \mathcal{I}$*, there exists* $e' \in B \setminus A$ *such that* $B \setminus \{e'\} \cup \{e\} \in \mathcal{B}$.

Having Lemma 8.5.1, our algorithm runs in $r(\mathcal{M})$ iterations and at each iteration chooses an element e with probability proportional to $\exp(\epsilon' \Delta_{e,i} F_D(\mathbf{x}))$ and adds e to $\mathrm{supp}(\mathbf{x})$. The analysis for the approximation guarantee is similar to the ones in [119, 179, 195, 217] and relies on Theorem 8.2.8.

Theorem 8.5.3. *Suppose* F_D *has sensitivity* Δ*. Then Algorithm 9, with high probability, returns* $\mathbf{x} \in (k+1)^E$ *such that* $\mathrm{supp}(\mathbf{x}) \in \mathcal{B}$ *and* $F_D(\mathbf{x}) \geq \frac{1}{2}\mathrm{OPT} - O(\frac{\Delta r(\mathcal{M}) \ln |E|}{\epsilon})$.

The privacy guarantee follows immediately from the ϵ-differential privacy of the exponential mechanism, together with Theorem 8.2.6.

Theorem 8.5.4. *Algorithm 9 preserves* $O(\epsilon r(\mathcal{M}))$*-differential privacy. It also provides* $(\frac{1}{2} r(\mathcal{M}) \epsilon^2 + \sqrt{2 \ln 1/\delta'}\epsilon, \delta')$*-differential privacy for every* $\delta' > 0$.

Clearly, Algorithm 9 evaluates F_D at most $O(k|E|r(\mathcal{M}))$ times. Next theorem summarizes the results of this section.

Theorem 8.5.5. *Suppose* F_D *has sensitivity* Δ*. Then Algorithm 9, with high probability, outputs a solution* $\mathbf{x} \in (k+1)^E$ *such that* $\mathrm{supp}(\mathbf{x})$ *is a base of* $\mathcal{M}$ *and* $F_D(\mathbf{x}) \geq \frac{1}{2}\mathrm{OPT} - O(\frac{\Delta r(\mathcal{M}) \ln |E|}{\epsilon})$ *by evaluating* F_D *at most* $O(k|E|r(\mathcal{M}))$ *times. Moreover, this algorithm preserves* $O(r(\mathcal{M})\epsilon)$*-differential privacy.*

Algorithm 10 Improved differentially private k-submodular maximization with a matroid constraint

Input: monotone k-submodular functions $F_D\colon (k+1)^E \to [0,1]$, a matroid $\mathcal{M} = (E, \mathcal{I})$, $\epsilon > 0$, and a failure probability $\gamma > 0$.
$\mathbf{x} \leftarrow \mathbf{0}$, $\epsilon' \leftarrow \frac{\epsilon}{2\Delta}$
for $t = 1$ to $r(\mathcal{M})$ **do**
    $R \leftarrow$ a random subset of size $\min\{\frac{|E|-t+1}{r(\mathcal{M})-t+1} \log \frac{r(\mathcal{M})}{\gamma}, |E|\}$ uniformly sampled from $E \setminus \mathrm{supp}(\mathbf{x})$.
    Choose $e \in R$ with $\mathrm{supp}(\mathbf{x}) \cup \{e\} \in \mathcal{I}$ and $i \in [k]$ with probability proportional to $\exp(\epsilon' \Delta_{e,i} F_D(\mathbf{x}))$.
    $\mathbf{x}(e) \leftarrow i$.
Output: $\mathbf{x}$

8.5.1 Improving the query complexity

By applying a sampling technique [168, 179], we improve the number of evaluations of F from $O(k|E|r(\mathcal{M}))$ to $O(k|E| \ln r(\mathcal{M}) \ln \frac{r(\mathcal{M})}{\gamma})$, where $\gamma > 0$ is a failure probability. Hence, even when $r(\mathcal{M})$ is as large as $|E|$, the number of function evaluations is almost linear in $|E|$. The main difference from Algorithm 9 is that we sample a sufficiently large subset R of E, and then greedily assign a value only looking at elements in R.

Theorem 8.5.6. *Suppose F_D has sensitivity Δ. Then Algorithm 10, with probability at least $1-\gamma$, outputs a solution with quality at least $\frac{1}{2}\mathrm{OPT} - O\left(\frac{\Delta r(\mathcal{M}) \ln \frac{|E|}{\gamma}}{\epsilon}\right)$ by evaluating F_D at most $O\left(k|E| \ln r(\mathcal{M}) \ln \frac{r(\mathcal{M})}{\gamma}\right)$ times.*

Similar to Theorem 8.5.4 and using the composition Theorem 8.2.6, Algorithm 10 preserves $O(\epsilon r(\mathcal{M}))$-differential privacy. It also provides $O\left(\frac{1}{2} r(\mathcal{M}) \epsilon^2 + \sqrt{2 \ln 1/\delta'}\epsilon, \delta'\right)$-differential privacy for every $\delta' > 0$. In summary, we have

Theorem 8.5.7. *Suppose F_D has sensitivity Δ. Then, with probability at least $1-\gamma$, Algorithm 10 returns a solution $\mathbf{x} \in (k+1)^E$ such that $\mathrm{supp}(\mathbf{x}) \in \mathcal{B}$ and $F_D(\mathbf{x}) \geq \frac{1}{2}\mathrm{OPT} - O\left(\frac{\Delta r(\mathcal{M}) \ln \frac{|E|}{\gamma}}{\epsilon}\right)$ by evaluating F_D at most $O\left(k|E| \ln r(\mathcal{M}) \ln \frac{r(\mathcal{M})}{\gamma}\right)$ times. Moreover, this algorithm preserves $O(\epsilon r(\mathcal{M}))$-differential privacy.*

8.5.2 Motivating examples

Example 8.5.8. Suppose that we have m ad slots and k ad agencies, and we want to allocate at most $B(\leq m)$ slots to the ad agencies. Each ad agency i has a "influence graph" G_i, which is a bipartite graph (U, V, E_i), where U and V correspond to ad slots and users, respectively, and an edge $uv \in E_i$ indicates that if the ad agency i takes the ad slot u (and put an ad there), the user v will be influenced by the ad. The goal is to maximize the number of influenced people (each person will be counted multiple times if he/she is influenced by

multiple ad agencies), based on which we get revenue from the ad agencies. This problem can be modeled as k-submodular function maximization under a cardinality constraint (a special case of matroid constraints), and edges incident to a user v in $G_1, \dots, G_k$ are sensitive data about v.

Example 8.5.9. Another example comes from (a variant of) facility location. Suppose that we have a set E of n lands, and we want to provide k resources (e.g., gas and electricity) to all the lands by opening up facilities at some of the lands. For each resource type i and lands $e, e' \in E$, we have a cost $c_i(e, e')$ of sending the resource of type i from e to e'. For a set $S \subseteq E$, let $c_i(e, S) = \min_{e' \in S} c_i(e, e')$, which is the cost of sending a resource of type i to e when we open up facilities of type i at lands in S. Assume we cannot open two or more facilities in the same land. Then, the goal is to find disjoint sets $S_1, \dots, S_k$ with $\sum_i |S_i| <= B$ for some fixed B that maximize $\sum_e \sum_i (C - c_i(e, S_i))$, where C is a large number so that the objective function is always non-negative. This problem can be modeled as k-submodular function maximization under a cardinality constraint, and the costs $c_i(e, \cdot)$ are sensitive data about e.

8.6 Missing proofs from section 8.3

Proof of Lemma 8.3.1. We have

$$\begin{aligned}
&|f(\mathbf{x}) - f(\mathbf{x}+\mathbf{v})| \\
&= \left| \sum_{S \subseteq E} F(S) \Big(\prod_{e \in S} \mathbf{x}(e) \prod_{e \notin S} (1 - \mathbf{x}(e)) - \prod_{e \in S} (\mathbf{x}(e) + \mathbf{v}(e)) \prod_{e \notin S} (1 - \mathbf{x}(e) - \mathbf{v}(e)) \Big) \right| \\
&\leq \sum_{S \subseteq E} \left| \prod_{e \in S} \mathbf{x}(e) \prod_{e \notin S} (1 - \mathbf{x}(e)) - \prod_{e \in S} (\mathbf{x}(e) + \mathbf{v}(e)) \prod_{e \notin S} (1 - \mathbf{x}(e) - \mathbf{v}(e)) \right|. \qquad (8.1)
\end{aligned}$$

Now, we define probability distributions $\{P_e\}_{e \in E}$ and $\{Q_e\}_{e \in E}$ over $\{0, 1\}$ so that $P_e(1) = \mathbf{x}(e)$ and $Q_e(1) = \mathbf{x}(e) + \mathbf{v}(e)$, respectively, for every $e \in E$. Note that

$$\begin{aligned}
g(\mathbf{x}(e)) &= h(P_e, Q_e)^2 \\
&= \left(\sqrt{\mathbf{x}(e)} - \sqrt{\mathbf{x}(e) + \mathbf{v}(e)} \right)^2 + \left(\sqrt{1 - \mathbf{x}(e)} - \sqrt{1 - \mathbf{x}(e) - \mathbf{v}(e)} \right)^2
\end{aligned}$$

is a convex function with domain $\mathbf{x}(e) \in [0, 1 - \mathbf{v}(e)]$. The maximum value for this function happens at $\mathbf{x}(e) = 0$ and $\mathbf{x}(e) = 1 - \mathbf{v}(e)$. Further its minimum is at $\mathbf{x}(e) = [1 - \mathbf{v}(e)]/2$.

$$\begin{aligned} h(P_e, Q_e)^2 &= g(\mathbf{x}(e)) \\ &\leq g(0) \\ &= g(1 - \mathbf{v}(e)) \\ &= 2 - 2\sqrt{1 - \mathbf{v}(e)} \\ &\leq \mathbf{v}(e)^2 + \mathbf{v}(e) \\ &\leq 2\mathbf{v}(e) && \text{(for } \mathbf{v}(e) \in [0, 1]\text{)} \end{aligned}$$

Letting $P = \bigotimes_{e \in E} P_e$ and $Q = \bigotimes_{e \in E} Q_e$, we have

$$\begin{aligned} (8.1) &\leq 2 \cdot d_{\mathrm{TV}}(P, Q) \\ &= 2\sqrt{2} \cdot h(P, Q) \\ &\leq 2\sqrt{2} \sqrt{\sum_{e \in E} h(P_e, Q_e)^2} && \text{(By Lemma 8.2.9)} \\ &= 2\sqrt{2} \sqrt{\sum_{e \in E} 2\mathbf{v}(e)} \\ &= 4\sqrt{|\mathbf{v}|_1} \\ &\leq 4\sqrt{\sqrt{|E|} \|\mathbf{v}\|_2} \\ &\leq 4\sqrt[4]{|E|}\sqrt{\rho} && \square \end{aligned}$$

8.7 Missing proofs from section 8.4

8.7.1 Proof of Lemma 8.4.3

Proof of Lemma 8.4.3. Let $OPT = \max_{\mathbf{z} \in C_\rho} \langle \mathbf{z}, \nabla f(\mathbf{x}_{t-1}) \rangle$ and $q_t(\mathbf{z}) = \langle \mathbf{z}, \nabla f(\mathbf{x}_{t-1}) \rangle$ for every $\mathbf{z} \in \mathcal{P}(\mathcal{M})$. Further, let $\mathbf{y}_t$ be the output of the algorithm and $\tilde{L}_{\mu,i}^{\mathbf{x}_{t-1}}$ denote the

estimated size of the i-th layer.

$$\begin{aligned}
\Pr\left[q(\mathbf{y}_t) \leq OPT - \frac{2\Delta}{\epsilon}\xi\right] &\leq \frac{\Pr[q(\mathbf{y}_t) \geq OPT - \frac{2\Delta}{\epsilon}\xi]}{\Pr[q(\mathbf{y}_t) = OPT]} \\
&\leq \frac{\exp\left[\epsilon'\left(OPT - \frac{2\Delta}{\epsilon}\xi + \ln(1+\mu)\right)\right]}{\sum_{j=1}^{k} \tilde{L}_{\mu,j}^{\mathbf{x}_{t-1}}(1+\mu)^{\epsilon'(j-1)}} \times \frac{\sum_{j=1}^{k} |\mathcal{L}_{\mu,j}^{\mathbf{x}_{t-1}}|(1+\mu)^{\epsilon'(j-1)}}{\exp(\epsilon' OPT)} \\
&= \frac{\exp\left[\epsilon'\left(OPT - \frac{2\Delta}{\epsilon}\xi + \ln(1+\mu)\right)\right]}{\exp(\epsilon' OPT)} \times \frac{\sum_{j=1}^{k} |\mathcal{L}_{\mu,j}^{\mathbf{x}_{t-1}}|(1+\mu)^{\epsilon'(j-1)}}{\sum_{j=1}^{k} \tilde{L}_{\mu,j}^{\mathbf{x}_{t-1}}(1+\mu)^{\epsilon'(j-1)}}
\end{aligned}$$

Consider the first term,

$$\begin{aligned}
\frac{\exp\left[\epsilon'\left(OPT - \frac{2\Delta}{\epsilon}\xi + \ln(1+\mu)\right)\right]}{\exp(\epsilon' OPT)} &= \exp\left[\epsilon'\left(-\frac{2\Delta}{\epsilon}\xi + \ln(1+\mu)\right)\right] \\
&= \exp(-\xi)\exp\left(\epsilon'\ln(1+\mu)\right) \\
&= \exp(-\xi)(1+\mu)^{\epsilon'}
\end{aligned}$$

Consider the second term. By Corollary 8.4.2, the algorithm estimates $|\mathcal{L}_{\mu,j}^{\mathbf{x}_{t-1}}|/|C_\rho|$ within additive error $\lambda_{8.4.2}$ with probability at least $1 - \theta_{8.4.2} = 1 - \beta$. Therefore,

$$\begin{aligned}
\frac{\sum_{j=1}^{k} |\mathcal{L}_{\mu,j}^{\mathbf{x}_{t-1}}|(1+\mu)^{\epsilon'(j-1)}}{\sum_{j=1}^{k} \tilde{L}_{\mu,j}^{\mathbf{x}_{t-1}}(1+\mu)^{\epsilon'(j-1)}} &\leq \frac{\sum_{j=1}^{k} (\tilde{L}_{\mu,j}^{\mathbf{x}_{t-1}} + \lambda_{8.4.2}|C_\rho|)(1+\mu)^{\epsilon'(j-1)}}{\sum_{j=1}^{k} \tilde{L}_{\mu,j}^{\mathbf{x}_{t-1}}(1+\mu)^{\epsilon'(j-1)}} \\
&\leq 1 + \frac{\sum_{j=1}^{k} (\lambda_{8.4.2}|C_\rho|)(1+\mu)^{\epsilon'(j-1)}}{\sum_{j=1}^{k} \tilde{L}_{\mu,j}^{\mathbf{x}_{t-1}}(1+\mu)^{\epsilon'(j-1)}} \\
&\leq 1 + \sum_{j=1}^{k} \frac{\lambda_{8.4.2}|C_\rho|}{\tilde{L}_{\mu,j}^{\mathbf{x}_{t-1}}} \\
&\leq 1 + k\lambda_{8.4.2}|C_\rho|
\end{aligned}$$

Therefore, putting both upper bounds together yields

$$\Pr\left[q(\mathbf{y}_t) \leq OPT - \frac{2\Delta}{\epsilon}\xi\right] \leq \exp(-\xi)(1+\mu)^{\epsilon'}(1 + k\lambda_{8.4.2}|C_\rho|)$$

As there are at most $|C_\rho|$ outputs with quality $OPT - \frac{2\Delta}{\epsilon}\xi$ their cumulative probability is at most

$$\begin{aligned}|C_\rho|(1+k\lambda_{8.4.2}|C_\rho|)(1+\mu)^{\epsilon'}\exp(-\xi) &= \frac{|C_\rho|(1+k\lambda_{8.4.2}|C_\rho|)(1+\mu)^{\epsilon'}\beta}{|C_\rho|(1+k\lambda_{8.4.2}|C_\rho|)(1+\mu)^{\epsilon'}} \\ &= \beta. \end{aligned}$$

□

8.7.2 Proof of Theorem 8.4.4

Proof of Theorem 8.4.4. Suppose $\mathbf{y}' \in C_\rho$ with $\|\mathbf{y}'-\mathbf{x}^*\|_2 \le \rho$. Let $\beta = \frac{1}{|E|^2}$. By Lemma 8.4.3, with probability at least $1 - \frac{1}{|E|^2}$, we have

$$\begin{aligned}\langle \mathbf{y}_t, \nabla f(\mathbf{x}_t)\rangle &\ge \operatorname*{argmax}_{\mathbf{y}\in C_\rho}\langle \mathbf{y}, \nabla f(\mathbf{x}_t)\rangle - \frac{2\Delta}{\epsilon}\xi \\ &\ge \langle \mathbf{y}', \nabla f(\mathbf{x}_t)\rangle - \frac{2\Delta}{\epsilon}\xi \\ &\ge f(\mathbf{x}^*) - f(\mathbf{x}_t) - C_{8.3.2}\sqrt{\rho} - \frac{2\Delta}{\epsilon}\xi \qquad \text{(by Lemma 8.3.2)}\end{aligned}$$

By a union bound, with probability at least $1 - \frac{1}{\text{poly}(|E|)}$, the above inequality holds for every t. In what follows, we assume this has happened. As in the proof of Theorem 8.3.3, suppose t is a continuous variable and define $\frac{d\mathbf{x}_t}{dt} = \alpha \mathbf{y}_t$.

$$\begin{aligned}\frac{df(\mathbf{x}_t)}{dt} &= \sum_e \frac{\partial f(\mathbf{x}_t(e))}{\partial \mathbf{x}_t(e)}\frac{d\mathbf{x}_t(e)}{dt} \\ &= \nabla f(\mathbf{x}_t)\cdot\frac{d\mathbf{x}_t}{dt} \\ &= \alpha\langle \mathbf{y}_t, \nabla f(\mathbf{x}_t)\rangle \\ &\ge \alpha\left(f(\mathbf{x}^*) - f(\mathbf{x}_t) - C_{8.3.2}\sqrt{\rho} - \frac{2\Delta}{\epsilon}\xi\right),\end{aligned}$$

Solving the differential equation with $f(\mathbf{x}_0) = 0$ gives us

$$f(\mathbf{x}_t) = (1-\mathrm{e}^{-\alpha t})\left(f(\mathbf{x}^*) - C_{8.3.2}\sqrt{\rho} - \frac{2\Delta}{\epsilon}\xi\right).$$

For $\alpha = \frac{1}{T}$ and $t = T$ we obtain

$$\begin{aligned}f(\mathbf{x}_T) &= (1-\mathrm{e}^{-1})\left(f(\mathbf{x}^*) - C_{8.3.2}\sqrt{\rho} - \frac{2\Delta}{\epsilon}\xi\right) \\ &= f(\mathbf{x}^*)(1-\mathrm{e}^{-1}) - O\left(C_{8.3.2}\sqrt{\rho} + \frac{2\Delta}{\epsilon}\xi\right).\end{aligned}$$

Recall that $\xi = \ln\Big(\Big[|C_\rho|(1+k\lambda_{8.4.2}|C_\rho|)(1+\mu)^{\epsilon'}\Big]/\beta\Big)$ and $\beta = 1/|E|^2$. Next we give an upper bound for the error term.

$$\begin{aligned}
O\left(C_{8.3.2}\sqrt{\rho}+\frac{2\Delta}{\epsilon}\xi\right) &= O\Big(C_{8.3.2}\sqrt{\rho}+\frac{2\Delta}{\epsilon}\ln(|E|^2|C_\rho|)+\frac{2\Delta}{\epsilon}\ln(1+\mu)^{\epsilon'}+\frac{2\Delta}{\epsilon}\ln(k\lambda_{8.4.2}|C_\rho|)\Big)\\
&= O\Big(C_{8.3.2}\sqrt{\rho}+\ln(1+\mu)+\frac{2\Delta}{\epsilon}\Big[\ln(|E|^2|C_\rho|)+\ln(k\lambda_{8.4.2}|C_\rho|)\Big]\Big)\\
&= O\Big(C_{8.3.2}\sqrt{\rho}+\ln(1+\mu)+\frac{\Delta}{\epsilon}(\frac{B}{\rho})^2(\ln|E|+\ln(k\lambda_{8.4.2}))\Big)
\end{aligned}$$

Note that by letting $\mu = \mathrm{e}^\epsilon - 1$ we get $\ln(1+\mu) = \epsilon$. Moreover, we get $k \le \frac{r(\mathcal{M})}{\epsilon}$. □

8.7.3 Proof of Theorem 8.4.5

Proof of Theorem 8.4.5. Let M denote Algorithm 8. Let D and D' be two neighboring datasets and F_D and $F_{D'}$ be their associated functions. Suppose $C'_\rho(D,t)$ denotes the set of sampled points at time step t given dataset D. Similarly, $C'_\rho(D',t)$ denotes set of sampled points at time step t given dataset D'. Samples are drawn uniformly at random and independent from the input function. Hence, Line 5 of M is 0-differentially private. Therefore, we assume $C'_\rho(D,t) = C'_\rho(D',t) = S_t$ for every time step t. Define k, k' as follow:

$$k = \left\lceil \log_{1+\mu}\left(\frac{\max\limits_{\mathbf{y}\in C_\rho}\exp(\langle \mathbf{y}, \nabla f_D(\mathbf{x})\rangle)}{\min\limits_{\mathbf{y}\in C_\rho}\exp(\langle \mathbf{y}, \nabla f_D(\mathbf{x})\rangle)}\right)\right\rceil$$

$$k' = \left\lceil \log_{1+\mu}\left(\frac{\max\limits_{\mathbf{y}\in C_\rho}\exp(\langle \mathbf{y}, \nabla f_{D'}(\mathbf{x})\rangle)}{\min\limits_{\mathbf{y}\in C_\rho}\exp(\langle \mathbf{y}, \nabla f_{D'}(\mathbf{x})\rangle)}\right)\right\rceil$$

Note that the layers might be different. Let us use $\mathcal{L}_i(D)$ and $\mathcal{L}_i(D')$ for the i-th layer given dataset D and D', respectively. Further, $\tilde{L}_i(D)$ and $\tilde{L}_i(D')$ denote the estimated size of the i-th layer.

For a fixed $\mathbf{y} \in C_\rho$, we consider the relative probability of M choosing $\mathbf{y}$ at time step t given multilinear extensions of F_D and $F_{D'}$. Let $M_t(f_D \mid \mathbf{x}_t)$ denote the output of M at time step t given dataset D and point $\mathbf{x}_t$. Similarly, $M_t(f_{D'} \mid \mathbf{x}_t)$ denote the output of M at time step t given dataset D' and point $\mathbf{x}_t$. Further, write $d_{\mathbf{y}} = \langle \mathbf{y}, \nabla f_D(\mathbf{x}_t)\rangle$ and $d'_{\mathbf{y}} = \langle \mathbf{y}, \nabla f_{D'}(\mathbf{x}_t)\rangle$.

Suppose $\mathbf{y} \in \mathcal{L}_i(D)$ given dataset D, and $\mathbf{y} \in \mathcal{L}_{i'}(D')$ given dataset D'. Then, we have

$$\frac{\Pr[M_t(f_D \mid \mathbf{x}_t) = \mathbf{y}]}{\Pr[M_t(f_{D'} \mid \mathbf{x}_t) = \mathbf{y}]} = \frac{\Pr[\mathbf{y} \in S_t \mid D]}{\Pr[\mathbf{y} \in S_t \mid D']} \times \frac{\frac{|\tilde{L}_i(D)|(1+\mu)^{\epsilon'(i-1)}}{|\tilde{L}_i(D)|}}{\frac{|\tilde{L}_{i'}(D')|(1+\mu)^{\epsilon'(i'-1)}}{|\tilde{L}_{i'}(D')|}} \times \frac{\sum_{j=1}^{k'} \tilde{L}_j(D') \exp(\epsilon' \cdot (1+\mu)^{j-1})}{\sum_{j=1}^{k} \tilde{L}_j(D) \exp(\epsilon' \cdot (1+\mu)^{j-1})}$$

$$= \frac{(1+\mu)^{\epsilon'(i-1)}}{(1+\mu)^{\epsilon'(i'-1)}} \times \frac{\sum_{j=1}^{k'} \tilde{L}_j(D') \exp(\epsilon' \cdot (1+\mu)^{j-1})}{\sum_{j=1}^{k} \tilde{L}_j(D) \exp(\epsilon' \cdot (1+\mu)^{j-1})} \tag{8.2}$$

The second equality holds since points are sampled uniformly at random from C_ρ in Line 5.

Lemma 8.7.1. *Let D, D' be neighboring datasets and F be Δ-sensitive. Suppose $\mathbf{z} \in C_\rho$ is a point in $\mathcal{L}_j(D)$. Then*

$$(1+\mu)^{\epsilon'(j-1)} \exp(-\frac{\epsilon r(\mathcal{M})}{2}) \leq \exp(\epsilon' \langle \mathbf{z}, \nabla f_{D'}(\mathbf{x}_t) \rangle) < (1+\mu)^{\epsilon' j} \exp(\frac{\epsilon r(\mathcal{M})}{2})$$

Proof. Since $\mathbf{z} \in C_\rho$ is a point in $\mathcal{L}_j(D)$, then $(1+\mu)^{j-1} \leq \exp(\langle \mathbf{z}, \nabla f_D(\mathbf{x}) \rangle) < (1+\mu)^j$. Since F_D is Δ-sensitive hence f_D is $\Delta r(\mathcal{M})$-sensitive (recall the proof of Theorem 8.3.6). Therefore,

$$\exp(\langle \mathbf{z}, \nabla f_{D'}(\mathbf{x}_t) \rangle) \leq \exp(\langle \mathbf{z}, \nabla f_D(\mathbf{x}_t) \rangle + \Delta r(\mathcal{M})) < (1+\mu)^j \exp(\Delta r(\mathcal{M})) \tag{8.3}$$

$$(1+\mu)^{j-1} \exp(-\Delta r(\mathcal{M})) \leq \exp(\langle \mathbf{z}, \nabla f_D(\mathbf{x}_t) \rangle - \Delta r(\mathcal{M}))) \leq \exp(\langle \mathbf{z}, \nabla f_{D'}(\mathbf{x}_t) \rangle) \tag{8.4}$$

$$(8.3), (8.4) \Rightarrow (1+\mu)^{j-1} \exp(-\Delta r(\mathcal{M})) \leq \exp(\langle \mathbf{z}, \nabla f_{D'}(\mathbf{x}_t) \rangle) < (1+\mu)^j \exp(\Delta r(\mathcal{M})) \tag{8.5}$$

$$(8.5) \Rightarrow (1+\mu)^{\epsilon'(j-1)} \exp(-\frac{\epsilon r(\mathcal{M})}{2}) \leq \exp(\epsilon' \langle \mathbf{z}, \nabla f_{D'}(\mathbf{x}_t) \rangle) < (1+\mu)^{\epsilon' j} \exp(\frac{\epsilon r(\mathcal{M})}{2}) \tag{8.6}$$

□

The interpretation of (8.5) is that if a point $\mathbf{z} \in S_t$ appears in layer $\mathcal{L}_j(D)$ then it can be in any of the layers $\mathcal{L}_p(D')$ for

$$(j-1) + \log_{1+\mu}[\exp(-\Delta r(\mathcal{M}))] \leq p < j + \log_{1+\mu}[\exp(\Delta r(\mathcal{M}))]^1.$$

[1]Note that in low-sensitivity regime, where $\Delta \ll r(\mathcal{M})$, we have $j - 1 \leq p < j$.

In a sense, the same argument in Claim 8.7.2 shows that $\lfloor \frac{k'}{k} \rfloor = 1$. Now, we are ready to provide an upper bound for (8.2).

Consider the first term $\frac{(1+\mu)^{\epsilon'(i-1)}}{(1+\mu)^{\epsilon'(i'-1)}}$. Recall that $\mathbf{y} \in \mathcal{L}_i(D)$ given dataset D, and $\mathbf{y} \in \mathcal{L}_{i'}(D')$ given dataset D'. By Lemma 8.7.1, we have

$$(1+\mu)^{\epsilon'(i-1)} \exp(-\frac{\epsilon r(\mathcal{M})}{2}) \leq \exp(\epsilon' \langle \mathbf{z}, \nabla f_{D'}(\mathbf{x}_t) \rangle).$$

Therefore,

$$\begin{aligned}
\frac{(1+\mu)^{\epsilon'(i-1)}}{(1+\mu)^{\epsilon'(i'-1)}} &\leq \frac{(1+\mu)^{\epsilon'(i-1)}}{(1+\mu)^{\epsilon'(i-1)} \exp(-\frac{\epsilon r(\mathcal{M})}{2})} \\
&= \exp(\frac{\epsilon r(\mathcal{M})}{2})
\end{aligned}$$

Now, we provide an upper bound for the second term of (8.2):

$$\begin{aligned}
\frac{\sum_{j=1}^{k'} \tilde{L}_j(D') \exp(\epsilon' \cdot (1+\mu)^{j-1})}{\sum_{j=1}^{k} \tilde{L}_j(D) \exp(\epsilon' \cdot (1+\mu)^{j-1})} &\leq \frac{\sum_{j=1}^{k} \tilde{L}_j(D) \exp(\epsilon r(\mathcal{M})) \exp(\epsilon' \cdot (1+\mu)^{j-1})}{\sum_{j=1}^{k} \tilde{L}_j(D) \exp(\epsilon' \cdot (1+\mu)^{j-1})} \\
&= \frac{[\exp(\epsilon r(\mathcal{M}))] \sum_{j=1}^{k} \tilde{L}_j(D) \exp(\epsilon' \cdot (1+\mu)^{j-1})}{\sum_{j=1}^{k} \tilde{L}_j(D) \exp(\epsilon' \cdot (1+\mu)^{j-1})} \\
&\leq \exp(\epsilon r(\mathcal{M}))
\end{aligned}$$

By a union bound and composition Theorem 8.2.6, Algorithm 8 preserves $O(\epsilon Tr(\mathcal{M})) \leq O(\epsilon r(\mathcal{M})^2)$-differential privacy. The heart of the above inequality is that, given the set of sample points, the layers defined for both instances are almost identical. □

Claim 8.7.2. $\frac{k'}{k} \leq 1 + \frac{2\Delta r(\mathcal{M})}{k \ln(1+\mu)}$.

Proof.

$$\begin{aligned}
k' &= \log_{1+\mu}\left(\frac{\max\limits_{\mathbf{y}\in C_\rho}\exp(\langle \mathbf{y}, \nabla f_{D'}(\mathbf{x})\rangle)}{\min\limits_{\mathbf{y}\in C_\rho}\exp(\langle \mathbf{y}, \nabla f_{D'}(\mathbf{x})\rangle)}\right)\\
&\leq \log_{1+\mu}\left(\frac{\max\limits_{\mathbf{y}\in C_\rho}\exp(\langle \mathbf{y}, \nabla f_{D}(\mathbf{x})\rangle + \Delta r(\mathcal{M}))}{\min\limits_{\mathbf{y}\in C_\rho}\exp(\langle \mathbf{y}, \nabla f_{D}(\mathbf{x})\rangle - \Delta r(\mathcal{M}))}\right)\\
&\leq \log_{1+\mu}\left(\frac{\max\limits_{\mathbf{y}\in C_\rho}\exp(\langle \mathbf{y}, \nabla f_{D}(\mathbf{x})\rangle)}{\min\limits_{\mathbf{y}\in C_\rho}\exp(\langle \mathbf{y}, \nabla f_{D}(\mathbf{x})\rangle)}\right)\\
&\quad + \log_{1+\mu}\exp(2\Delta r(\mathcal{M}))\\
&= k + \frac{2\Delta r(\mathcal{M})}{\ln(1+\mu)}\\
&= k\left(1 + \frac{2\Delta r(\mathcal{M})}{k\ln(1+\mu)}\right).
\end{aligned}$$

□

8.8 Missing proofs from section 8.5

8.8.1 Proof of Theorem 8.5.3

Proof of Theorem 8.5.3. Consider the j-th iteration of the algorithm. Let $(e^{(j)}, i^{(j)})$ be the pair chosen in this iteration. Further, let $\mathbf{o}$ be the optimal solution and $\mathbf{x}^{(j)}$ be the solution after the j-th iteration. Note that $|\mathrm{supp}(\mathbf{x}^{(j)})| = j$ for $j \in [r(\mathcal{M})]$. We define a sequence of vectors $\mathbf{o}^{(0)} = \mathbf{o}, \mathbf{o}^{(1)}, \ldots, \mathbf{o}^{r(\mathcal{M})}$, as in [119, 179, 195, 217], such that

1. $\mathbf{x}^{(j)} \prec \mathbf{o}^{(j)}$ for all $0 \leq j \leq r(\mathcal{M}) - 1$,
2. $\mathbf{x}^{(r(\mathcal{M}))} = \mathbf{o}^{(r(\mathcal{M}))}$,
3. $O^{(j)} := \mathrm{supp}(\mathbf{o}^{(j)}) \in \mathcal{B}$ for all $0 \leq j \leq r(\mathcal{M})$.

For the sake of completeness, let us describe how to obtain $\mathbf{o}^{(j)}$ from $\mathbf{o}^{(j-1)}$ assuming $\mathbf{x}^{(j-1)} \prec \mathbf{o}^{(j-1)}$ and $O^{(j-1)} \in \mathcal{B}$. Let $X^{(j)} = \mathrm{supp}(\mathbf{x}^{(j)})$. $\mathbf{x}^{(j-1)} \prec \mathbf{o}^{(j-1)}$ implies that $X^{(j-1)} \subsetneq O^{(j-1)}$ and $e^{(j)}$ is chosen to satisfy $X^{(j-1)} \cup \{e^{(j)}\} \in \mathcal{I}$. By Lemma 8.5.2, there exists $e' \in O^{(j-1)} \setminus X^{(j-1)}$ such that $O^{(j-1)} \setminus \{e'\} \cup \{e^{(j)}\} \in \mathcal{B}$.

Now let $o^{(j)} = e'$ and define $\mathbf{o}^{(j-1/2)}$ as the vector obtained by assigning 0 to the $o^{(j)}$-th element of $\mathbf{o}^{(j-1)}$. We then define $\mathbf{o}^{(j)}$ as the vector obtained from $\mathbf{o}^{(j-1/2)}$ by assigning $i^{(j)}$ to the $e^{(j)}$-th element. Therefore, for vector $\mathbf{o}^{(j)}$ we have $O^{(j)} \in \mathcal{B}$ and $\mathbf{x}^{(j)} \prec \mathbf{o}^{(j)}$.

By Theorem 2 in [195], if we always selected $(e^{(j)}, i^{(j)})$ with $e^{(j)} \in \Lambda(\mathbf{x}), i \in [k]$ and maximum $\Delta_{e,i} f(\mathbf{x})$, we would have

$$F(\mathbf{x}^{(j)}) - F(\mathbf{x}^{(j-1)}) \geq F(\mathbf{o}^{(j-1)}) - F(\mathbf{o}^{(j)}).$$

Instead we use the exponential mechanism which, by Theorem 8.2.8, selects $(e^{(j)}, i^{(j)})$ within $\frac{2\Delta}{\epsilon} \ln \frac{|\Lambda(\mathbf{x}^{(j)})|}{\beta}$ from the optimal choice with probability at least $1-\beta$. Therefore,

$$F(\mathbf{x}^{(j)}) - F(\mathbf{x}^{(j-1)}) \geq F(\mathbf{o}^{(j-1)}) - F(\mathbf{o}^{(j)}) - \frac{2\Delta}{\epsilon} \ln \frac{|\Lambda(\mathbf{x}^{(j)})|}{\beta}$$

with probability at least $1-\beta$. Given this, one can derive the following:

$$\begin{aligned}
F(\mathbf{o}) - F(\mathbf{x}^{(r(\mathcal{M}))}) &= \sum_{j=1}^{r(\mathcal{M})} F(\mathbf{o}^{(j-1)}) - F(\mathbf{o}^{(j)}) \\
&\leq \sum_{j=1}^{r(\mathcal{M})} \left(F(\mathbf{x}^{(j-1)}) - F(\mathbf{x}^{(j)}) + \frac{2\Delta}{\epsilon} \ln \frac{|\Lambda(\mathbf{x}^{(j)})|}{\beta} \right) \\
&= F(\mathbf{x}^{(r(\mathcal{M}))}) - F(\mathbf{0}) + r(\mathcal{M}) \left(\frac{2\Delta}{\epsilon} \ln \frac{|\Lambda(\mathbf{x}^{(j)})|}{\beta} \right) \\
&= F(\mathbf{x}^{(r(\mathcal{M}))}) + r(\mathcal{M}) \left(\frac{2\Delta}{\epsilon} \ln \frac{|\Lambda(\mathbf{x}^{(j)})|}{\beta} \right),
\end{aligned}$$

which means Algorithm 9 returns $\mathbf{x} = \mathbf{x}^{(r(\mathcal{M}))}$ with quality at least $\frac{1}{2}\mathrm{OPT} - r(\mathcal{M})(\frac{2\Delta}{\epsilon} \ln \frac{|\Lambda(\mathbf{x}^{(j)})|}{\beta})$ with probability at least $1 - r(\mathcal{M})\beta$. Having $\beta = \frac{1}{|E|^2}$, $|\Lambda(\mathbf{x}^{(j)})| \leq |E|$ gives us

$$F(\mathbf{x}) \geq \frac{1}{2}\mathrm{OPT} - O\left(\frac{\Delta r(\mathcal{M}) \ln |E|}{\epsilon} \right). \qquad \square$$

8.8.2 Proof of Theorem 8.5.6

Proof of Theorem 8.5.6. Let $R^{(j)}$ be R in the j-th iteration, $\mathbf{o}$ be the optimal solution and $\mathbf{x}^{(j)}$ be the solution after the j-th iteration. Further, let $X^{(j)} = \mathrm{supp}(\mathbf{x}^{(j)})$, $O^{(j)} = \mathrm{supp}(\mathbf{o}^{(j)})$, and

$$\Lambda(\mathbf{x})^{(j)} = \{e \in E \setminus \mathrm{supp}(\mathbf{x}^{(j)}) \mid \mathrm{supp}(\mathbf{x}^{(j)}) \cup \{e\} \in \mathcal{I}\}$$

We iteratively define $\mathbf{o}^{(0)} = \mathbf{o}, \mathbf{o}^{(1)}, \ldots, \mathbf{o}^{r(\mathcal{M})}$ as follows. If $R^{(j)} \cap \Lambda(\mathbf{x})^{(j)} = \emptyset$, then we regard that the algorithm failed. Else we proceed as follows. By Lemma 8.5.2, for any $e^{(j)} \in R^{(j)} \cap \Lambda(\mathbf{x})^{(j)}$, there exists e' such that $e' \in O^{(j-1)} \setminus X^{(j-1)}$ and $O^{(j-1)} \setminus \{e'\} \cup \{e^{(j)}\} \in \mathcal{B}$. Now let $o^{(j)} = e'$ and define $\mathbf{o}^{(j-1/2)}$ as the vector obtained by assigning 0 to the $o^{(j)}$-th element of $\mathbf{o}^{(j-1)}$. We then define $\mathbf{o}^{(j)}$ as the vector obtained from $\mathbf{o}^{(j-1/2)}$ by assigning $i^{(j)}$ to the $e^{(j)}$-th element. Therefore, for vector $\mathbf{o}^{(j)}$ we have $O^{(j)} \in \mathcal{B}$ and $\mathbf{x}^{(j)} \prec \mathbf{o}^{(j)}$.

If the algorithm does not fail and $\mathbf{o}^{(0)} = \mathbf{o}, \mathbf{o}^{(1)}, \ldots, \mathbf{o}^{r(\mathcal{M})}$ are well defined, or in other words, if $R^{(j)} \cap \Lambda(\mathbf{x})^{(j)}$ is not empty for every $j \in [r(\mathcal{M})]$, then the rest of the analysis is completely the same as in Theorem 8.5.3, and we achieve an approximation ratio of (roughly) 1/2. Hence, it suffices to show that $R^{(j)} \cap \Lambda(\mathbf{x})^{(j)}$ is not empty with a high probability.

Lemma 8.8.1. *With probability at least* $1 - \frac{\gamma}{r(\mathcal{M})}$*, we have* $R^{(j)} \cap \Lambda(\mathbf{x})^{(j)} \neq \emptyset$ *for every* $j \in [r(\mathcal{M})]$*.*

Analogous to the analysis in Theorem 8.5.3, for every time step $0 \leq j \leq r(\mathcal{M})$, with probability at least $1 - \frac{\gamma}{r(\mathcal{M})}$ we have

$$F(\mathbf{x}^{(j)}) - F(\mathbf{x}^{(j-1)}) \geq F(\mathbf{o}^{(j-1)}) - F(\mathbf{o}^{(j)}) - \frac{2\Delta}{\epsilon} \ln(\frac{r(\mathcal{M})|\Lambda(\mathbf{x}^{(j)})|}{\gamma}).$$

By a union bound over $j \in [r(\mathcal{M})]$, with probability at least $1 - \gamma$, it follows that

$$F(\mathbf{x}) \geq \frac{1}{2}\mathrm{OPT} - O\left(\frac{\Delta r(\mathcal{M}) \ln(|E|/\gamma)}{\epsilon}\right).$$

Applying a similar argument as in [179], the number of evaluations of f is at most

$$\begin{aligned} k \sum_{t=1}^{r(\mathcal{M})} \frac{|E| - t + 1}{r(\mathcal{M}) - t + 1} \ln \frac{r(\mathcal{M})}{\gamma} &= k \sum_{t=1}^{r(\mathcal{M})} \frac{|E| - r(\mathcal{M}) + t}{t} \log \frac{r(\mathcal{M})}{\gamma} \\ &= O\left(k|E| \ln r(\mathcal{M}) \ln \frac{r(\mathcal{M})}{\gamma}\right) \end{aligned}$$

□

Proof of Lemma 8.8.1.

$$\begin{aligned} \Pr[R^{(j)} \cap \Lambda(\mathbf{x})^{(j)} = \emptyset] &= \left(1 - \frac{r(\mathcal{M}) - \mathrm{supp}(\mathbf{x}^{(j)})}{|E \setminus \mathrm{supp}(\mathbf{x}^{(j)})|}\right)^{|R^{(j)}|} \\ &\leq \exp\left(-\frac{r(\mathcal{M}) - j + 1}{|E| - j + 1} \frac{|E| - j + 1}{r(\mathcal{M}) - j + 1} \ln \frac{r(\mathcal{M})}{\gamma}\right) \\ &= \exp\left(-\ln \frac{r(\mathcal{M})}{\gamma}\right) = \frac{\gamma}{r(\mathcal{M})} \end{aligned}$$

□

Chapter 9

Conclusion

In this thesis we focused on the CSPs and related problems arising from them that span over various research areas in mathematics, computer science, and machine learning. Our ultimate goal is to fully understand the power and limitations of universal algebraic techniques for theses problems and provide a unifying algorithmic framework that deals/solves such problems. While we have achieved this goal to some extend and taken important steps towards this goal, there still are several intriguing and challenging questions that need to be addressed. Here we provide a list of such problems and propose them as future directions.

Combinatorial IMPs. The study of CSP-related Ideal Membership Problems is in its infancy, and pretty much all research directions are open. These include expanding the range of tractable IMPs. A number of candidates for such expansions are readily available from the existing results about the CSP. There are however several questions that seem to be more intriguing; they mainly concern with relationship between the IMP, CSP and other problems.

The first one is how the tractability of the IMP can be used in applications such as Nullstellensatz and SOS proofs. The several results we obtain here barely scratch the surface. Establishing connections of this kind seem important, because it would allow for using a much larger toolbox than the usual Gröbner Basis. Note also that constructing an explicit Gröbner Basis beyond Boolean case is getting very hard very quickly; such techniques may be not very useful in more general cases.

One of the principal techniques in solving the CSP is constraint propagation, that is, the study how local interaction between constraints may tighten them, and even sometimes refute the existence of a solution. In some cases such as IMPs with the dual-discriminator polymorphism, computing the S-polynomial, and therefore constructing a Gröbner Basis is equivalent to establishing so-called $(2,3)$-consistency. This however is not the case in general. On the other hand, constraint propagation is done through some very simple pp-definitions, and so Theorem 2.2.4 (1) and its proof imply that there likely are some parallel constructions with polynomials and ideals.

The main tool for solving restricted degree problems $\mathrm{IMP}_d(\Gamma)$ is constructing a d-truncated Gröbner Basis, in which the degrees of polynomials are also bounded by d. It is interesting what effect restricting the degree of polynomials in a generating set has on the properties of the underlying CSP. More precisely, if $\mathcal{I}(\mathcal{P})$ is the ideal corresponding to a CSP instance $\mathcal{P}$, and $\mathcal{I}_d(\mathcal{P})$ is the ideal generated by the truncated Gröbner Basis, then $\mathcal{I}_d(\mathcal{P})$ can be translated back, to a less constrained CSP $\mathcal{P}'$. What is the connection between $\mathcal{P}$ and $\mathcal{P}'$? For example, every instance $\mathcal{P}$ of $\mathrm{CSP}(\Gamma)$ where Γ is Boolean and has a semilattice polymorphism, then $\mathcal{P}$ is equivalent to a Horn- or AntiHorn-Satisfiability. Restricting the degrees of polynomials in $\mathcal{I}_d(\mathcal{P})$ is apparently equivalent to removing all clauses of length more than d in $\mathcal{P}$.

Approximation of VCSPs and MinHOM. Here we established several positive results for (constant) approximability of the MinHOM problem for digraphs and gave a complete classification of the approximable cases for graphs. We conjecture, see Conjecture 6.1.6, that the class of DAT-free digraphs (i.e., digraphs with bounded width) is the right class of digraphs that establishes the dichotomy for approximation of the MinHOM. Hence, a major open problem is to verify this conjecture. Another natural question is whether our approximation factors can be improved, in particular, if they can be independent of the size of H. Another interesting question would be studying the MinHOM problem for hypergraphs i.e. relational structures. While the focus of our work is on digraphs, one can consider the minimum cost homomorphism problem for hypergraphs or equivalently VCSPs. That is, characterizing hypergraphs $\mathbb{H}$ for which MinHOM($\mathbb{H}$) admits a constant factor approximation.

Strong inapproximability results are known for special cases of MinHOM. It is a classical result that Vertex Cover has a 2-approximation algorithm and inapproximability results are also known. The authors of [63] proved that it is **NP**-hard to approximate Vertex Cover within factor 1.3606. Later, the factor was improved to $(\sqrt{2}-\epsilon)$ for any $\epsilon > 0$ [132]. Moreover, assuming UGC, Vertex Cover cannot be approximated within any constant factor better than 2. While such inapproximability of special cases of the MinHOM are known, to the best of our knowledge, there is no work that studies inapporixmability of MinHOM in a unifying framework. We therefore propose the following problems: what are the classes of digraphs for which MinHOM is **APX**-complete? Is there an universal constant δ such that the inapproximability of the MinHOM is bounded away from 1 by at least δ for all digraphs H for which MinHOM is **APX**-complete? Answering similar questions for hypergraphs is an interesting research direction.

Sparsification and differential privacy for VCSPs. We presented sparsification and differentially private algorithms for a very important class of VCSPs, namely submodular functions, that appear in numerous applications in machine learning and data mining. We

believe our sparsification technique will be useful in studying sparsification of VCSPs and characterizing the tractable cases. Thus, an important step towards understanding sparsification of VCSPs is to understand the limitations (e.g., in terms of time complexity) and effectiveness of the importance sampling technique.

Our approach to design differentially private algorithms for submodular functions is rather ad hoc and it is perhaps unlikely to extend it to general VCSPs. Hence, in order to address the privacy issue for VCSPs, a new way of thinking is required. The main tool for exact solvability of VCSPs and their approximation are the so-called basic LP and basic SDP. These two are among the most fundamental and powerful tools in algorithmic design. An important problem that is required to be addressed here is solving LPs and SDPs under differential privacy. Note that, not only is this problem of great interest because of applications of LPs and SDPs, but also is the key step towards achieving a full understanding of VCSPs for which we can design efficient differentially private algorithms. Hsu et al. [116] provide a differentially private algorithm for solving LPs in restricted settings. To the best of our knowledge, no differentially private algorithm is known for solving SDPs.

Bibliography

[1] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 5–14, 2012.

[2] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Spectral sparsification in dynamic graph streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, (APPROX/RANDOM)*, pages 1–10, 2013.

[3] Alexandr Andoni, Jiecao Chen, Robert Krauthgamer, Bo Qin, David P. Woodruff, and Qin Zhang. On sketching quadratic forms. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 311–319, 2016.

[4] Sanjeev Arora, László Babai, Jacques Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *J. Comput. Syst. Sci.*, 54(2):317–331, 1997.

[5] Albert Atserias, Andrei A. Bulatov, and Víctor Dalmau. On the power of k - consistency. In *Proceedings of the 34th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 279–290, 2007.

[6] Albert Atserias and Joanna Ochremiak. Proof complexity meets algebra. *ACM Trans. Comput. Log.*, 20(1):1:1–1:46, 2019.

[7] Per Austrin. Towards sharp inapproximability for any 2-csp. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 307–317, 2007.

[8] Kyriakos Axiotis, Adam Karczmarz, Anish Mukherjee, Piotr Sankowski, and Adrian Vladu. Decomposable submodular function minimization via maximum flow. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 446–456, 2021.

[9] Guillaume Bagan, Arnaud Durand, Emmanuel Filiot, and Olivier Gauwin. Efficient enumeration for conjunctive queries over x-underbar structures. In *Proceedings of the 24th International Workshop Computer Science Logic (CSL), 19th Annual Conference of the EACSL*, pages 80–94, 2010.

[10] Wenruo Bai, Rishabh K. Iyer, Kai Wei, and Jeff A. Bilmes. Algorithms for optimizing the ratio of submodular functions. In *Proceedings of the 33nd International Conference on Machine Learning (ICML)*, pages 2751–2759, 2016.

[11] Ramakrishna Bairi, Rishabh K. Iyer, Ganesh Ramakrishnan, and Jeff A. Bilmes. Summarization of multi-document topic hierarchies using submodular mixtures. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 553–563, 2015.

[12] Nikhil Bansal, Ola Svensson, and Luca Trevisan. New notions and constructions of sparsification for graphs and hypergraphs. In *Proceedings of the 60th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 910–928, 2019.

[13] Amotz Bar-Noy, Mihir Bellare, Magnús M. Halldórsson, Hadas Shachnai, and Tami Tamir. On chromatic sums and distributed resource allocation. *Inf. Comput.*, 140(2):183–202, 1998.

[14] Libor Barto. The dichotomy for conservative constraint satisfaction problems revisited. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 301–310, 2011.

[15] Libor Barto. Constraint satisfaction problem and universal algebra. *ACM SIGLOG News*, 1(2):14–24, 2014.

[16] Libor Barto. The constraint satisfaction problem and universal algebra. *Bull. Symb. Log.*, 21(3):319–337, 2015.

[17] Libor Barto and Marcin Kozik. Constraint satisfaction problems solvable by local consistency methods. *J. ACM*, 61(1):3:1–3:19, 2014.

[18] Libor Barto, Andrei A. Krokhin, and Ross Willard. Polymorphisms, and how to use them. In *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 1–44. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[19] Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-Ramanujan sparsifiers. *SIAM J. Comput.*, 41(6):1704–1721, 2012.

[20] Paul Beame, Russell Impagliazzo, Jan Krajícek, Toniann Pitassi, and Pavel Pudlák. Lower bound on hilbert's nullstellensatz and propositional proofs. In *Proceedings of the 35th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 794–806. IEEE Computer Society, 1994.

[21] Thomas Becker and Volker Weispfenning. Gröbner bases. In *Gröbner Bases*, pages 187–242. Springer, 1993.

[22] András A. Benczúr and David R. Karger. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 47–55, 1996.

[23] András A. Benczúr and David R. Karger. Randomized approximation schemes for cuts and flows in capacitated graphs. *SIAM J. Comput.*, 44(2):290–319, 2015.

[24] Arpitha P. Bharathi and Monaldo Mastrolilli. Ideal membership problem and a majority polymorphism over the ternary domain. In *Proceedings of the 45th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 13:1–13:13, 2020.

[25] Arpitha P. Bharathi and Monaldo Mastrolilli. Ideal membership problem for boolean minority. *CoRR*, abs/2006.16422, 2020.

[26] Arpitha P. Bharathi and Monaldo Mastrolilli. Ideal membership problem for boolean minority and dual discriminator. In *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 16:1–16:20, 2021.

[27] Jesús M Bilbao, Julio R Fernández, Nieves Jiménez, and Jorge J López. A survey of bicooperative games. In *Pareto Optimality, Game Theory And Equilibria*, pages 187–216. Springer, 2008.

[28] V.G. Bodnarchuk, L.A. Kaluzhnin, V.N. Kotov, and B.A. Romov. Galois theory for post algebras. i. *Kibernetika*, 3:1–10, 1969.

[29] Richard C. Brewster, Tomás Feder, Pavol Hell, Jing Huang, and Gary MacGillivray. Near-unanimity functions and varieties of reflexive graphs. *SIAM J. Discrete Math.*, 22(3):938–960, 2008.

[30] Bruno Buchberger. Bruno buchbergerÃ¢s phd thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. *Journal of Symbolic Computation*, 41(3):475 – 511, 2006. Logic, Mathematics and Computer Science: Interactions in honor of Bruno Buchberger (60th birthday).

[31] Andrei A. Bulatov. Tractable conservative constraint satisfaction problems. In *Proceedings of the 18th IEEE Symposium on Logic in Computer Science (LICS)*, page 321, 2003.

[32] Andrei A. Bulatov. H-coloring dichotomy revisited. *Theor. Comput. Sci.*, 349(1):31–39, 2005.

[33] Andrei A. Bulatov. Complexity of conservative constraint satisfaction problems. *ACM Trans. Comput. Log.*, 12(4):24:1–24:66, 2011.

[34] Andrei A. Bulatov. Conservative constraint satisfaction re-revisited. *J. Comput. Syst. Sci.*, 82(2):347–356, 2016.

[35] Andrei A. Bulatov. A dichotomy theorem for nonuniform csps. In *Proceedings of the 58th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 319–330, 2017.

[36] Andrei A. Bulatov and Víctor Dalmau. A simple algorithm for mal'tsev constraints. *SIAM J. Comput.*, 36(1):16–27, 2006.

[37] Andrei A. Bulatov and Víctor Dalmau. Towards a dichotomy theorem for the counting constraint satisfaction problem. *Inf. Comput.*, 205(5):651–678, 2007.

[38] Andrei A. Bulatov and Peter Jeavons. An algebraic approach to multi-sorted constraints. In *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming (CP)*, pages 183–198, 2003.

[39] Andrei A. Bulatov, Peter Jeavons, and Andrei A. Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM J. Comput.*, 34(3):720–742, 2005.

[40] Andrei A. Bulatov, Andrei A. Krokhin, and Benoît Larose. Dualities for constraint satisfaction problems. In *Complexity of Constraints - An Overview of Current Research Themes [Result of a Dagstuhl Seminar]*, volume 5250 of *Lecture Notes in Computer Science*, pages 93–124. Springer, 2008.

[41] Andrei A. Bulatov and Akbar Rafiey. On the complexity of csp-based ideal membership problems. *CoRR*, abs/2011.03700, 2020.

[42] Andrei A. Bulatov and Akbar Rafiey. The ideal membership problem and abelian groups. In *Proceedings of the 39th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 18:1–18:16, 2022.

[43] Andrei A. Bulatov and Akbar Rafiey. On the complexity of csp-based ideal membership problems. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, page 436–449, 2022.

[44] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Proceedings of the 14th International Conference on Theory of Cryptography (TCC)*, pages 635–658, 2016.

[45] S. Burris and H.P. Sankappanavar. *A course in universal algebra*, volume 78 of *Graduate Texts in Mathematics*. Springer-Verlag, New York-Berlin, 1981.

[46] Samuel R. Buss and Toniann Pitassi. Good degree bounds on nullstellensatz refutations of the induction principle. In *Proceedings of the 11th Annual IEEE Conference on Computational Complexity (CCC)*, pages 233–242, 1996.

[47] Silvia Butti and Stanislav Zivný. Sparsification of binary csps. *SIAM J. Discret. Math.*, 34(1):825–842, 2020.

[48] Gruia Călinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.*, 40(6):1740–1766, 2011.

[49] Anamay Chaturvedi, Huy Le Nguyen, and Lydia Zakynthinou. Differentially private decomposable submodular maximization. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*, pages 6984–6992, 2021.

[50] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 575–584, 2010.

[51] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. *SIAM J. Comput.*, 43(6):1831–1879, 2014.

[52] Yu Chen, Sanjeev Khanna, and Ansh Nagda. Near-linear size hypergraph cut sparsifiers. In *Proceedings of the 61st IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 61–72, 2020.

[53] Matthew Clegg, Jeff Edmonds, and Russell Impagliazzo. Using the groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 174–183, 1996.

[54] David A. Cohen, Martin C. Cooper, Peter Jeavons, and Andrei A. Krokhin. The complexity of soft constraint satisfaction. *Artif. Intell.*, 170(11):983–1016, 2006.

[55] David A. Cohen, Martin C. Cooper, Peter G. Jeavons, Andrei A. Krokhin, Robert Powell, and Stanislav Zivny. Binarisation for valued constraint satisfaction problems. *SIAM J. Discrete Math.*, 31(4):2279–2300, 2017.

[56] Michael B. Cohen, Jonathan Kelner, John Peebles, Richard Peng, Anup B. Rao, Aaron Sidford, and Adrian Vladu. Almost-linear-time algorithms for markov chains and new spectral primitives for directed graphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 410–419, 2017.

[57] Michael B. Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 181–190, 2015.

[58] Martin C. Cooper, David A. Cohen, and Peter Jeavons. Characterising tractable constraints. *Artif. Intell.*, 65(2):347–361, 1994.

[59] David Cox, John Little, and Donal OShea. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*. Springer Science & Business Media, 2013.

[60] Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity classifications of boolean constraint satisfaction problems*, volume 7 of *SIAM monographs on discrete mathematics and applications*. SIAM, 2001.

[61] Víctor Dalmau, Andrei A. Krokhin, and Rajsekar Manokaran. Towards a characterization of constant-factor approximable finite-valued csps. *J. Comput. Syst. Sci.*, 97:14–27, 2018.

[62] Alicia Dickenstein, Noaï Fitchas, Marc Giusti, and Carmen Sessa. The membership problem for unmixed polynomial ideals is solvable in single exponential time. *Discret. Appl. Math.*, 33(1-3):73–94, 1991.

[63] Irit Dinur and Samuel Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162(1):439–485, 2005.

[64] Shahar Dobzinski and Michael Schapira. An improved approximation algorithm for combinatorial auctions with submodular bidders. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1064–1073, 2006.

[65] Delbert Dueck and Brendan J. Frey. Non-metric affinity propagation for unsupervised image categorization. In *Proceedings of the IEEE 11th International Conference on Computer Vision ICCV*, pages 1–8, 2007.

[66] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Proceedings of the 25th Annual International Conference on the Theory and Applications of Cryptographic Technique (EUROCRYPT)*, pages 486–503, 2006.

[67] Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 371–380, 2009.

[68] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[69] Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. Boosting and differential privacy. In *Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 51–60, 2010.

[70] Jack Edmonds. Matroids and the greedy algorithm. *Math. Program.*, 1(1):127–136, 1971.

[71] Jack Edmonds. Submodular functions, matroids, and certain polyhedra. In *Combinatorial Optimization - Eureka, You Shrink!*, pages 11–26, 2001.

[72] Alina Ene and Huy L. Nguyen. Towards nearly-linear time algorithms for submodular maximization with a matroid constraint. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 54:1–54:14, 2019.

[73] Alina Ene, Huy L. Nguyen, and László A. Végh. Decomposable submodular function minimization: Discrete and continuous. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS)*, pages 2870–2880, 2017.

[74] Alina Ene, Jan Vondrák, and Yi Wu. Local distribution and the symmetry gap: Approximability of multiway partitioning problems. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 306–325, 2013.

[75] Jean-Charles Faugère, Patrizia M. Gianni, Daniel Lazard, and Teo Mora. Efficient computation of zero-dimensional gröbner bases by change of ordering. *J. Symb. Comput.*, 16(4):329–344, 1993.

[76] Tomás Feder, Pavol Hell, and Jing Huang. Bi-arc graphs and the complexity of list homomorphisms. *Journal of Graph Theory*, 42(1):61–80, 2003.

[77] Tomás Feder, Pavol Hell, Peter Jonsson, Andrei A. Krokhin, and Gustav Nordh. Retractions to pseudoforests. *SIAM J. Discrete Math.*, 24(1):101–112, 2010.

[78] Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1998.

[79] Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1998.

[80] Uriel Feige. On maximizing welfare when utility functions are subadditive. *SIAM J. Comput.*, 39(1):122–142, 2009.

[81] Uriel Feige and Jan Vondrák. Approximation algorithms for allocation problems: Improving the factor of 1 - 1/e. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 667–676, 2006.

[82] Arnold Filtser and Robert Krauthgamer. Sparsification of two-variable valued constraint satisfaction problems. *SIAM J. Discret. Math.*, 31(2):1263–1276, 2017.

[83] Alexander Fix, Thorsten Joachims, Sung Min Park, and Ramin Zabih. Structured learning of sum-of-submodular higher order energy functions. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3104–3111, 2013.

[84] D. Geiger. Closed systems of function and predicates. *Pacific Journal of Mathematics*, pages 95–100, 1968.

[85] Krzysztof Giaro, Robert Janczewski, Marek Kubale, and Michal Malafiejski. A 27/26-approximation algorithm for the chromatic sum coloring of bipartite graphs. In *Proceedings of the Approximation Algorithms for Combinatorial Optimization, 5th International Workshop (APPROX)*, pages 135–145, 2002.

[86] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.

[87] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.

[88] Ryan Gomes and Andreas Krause. Budgeted nonparametric learning from data streams. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 391–398, 2010.

[89] João Gouveia, Monique Laurent, Pablo A. Parrilo, and Rekha R. Thomas. A new semidefinite programming hierarchy for cycles in binary matroids and cuts in graphs. *Math. Program.*, 133(1-2):203–225, 2012.

[90] João Gouveia, Pablo A. Parrilo, and Rekha R. Thomas. Theta bodies for polynomial ideals. *SIAM J. Optim.*, 20(4):2097–2118, 2010.

[91] Dima Grigoriev. Tseitin's tautologies and lower bounds for nullstellensatz proofs. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 648–652, 1998.

[92] Joshua A. Grochow and Toniann Pitassi. Circuit complexity, proof complexity, and polynomial identity testing: The ideal proof system. *J. ACM*, 65(6):37:1–37:59, 2018.

[93] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Comb.*, 1(2):169–197, 1981.

[94] Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. Differentially private combinatorial optimization. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1106–1125, 2010.

[95] Gregory Z. Gutin, Pavol Hell, Arash Rafiey, and Anders Yeo. A dichotomy for minimum cost graph homomorphisms. *Eur. J. Comb.*, 29(4):900–911, 2008.

[96] Gregory Z. Gutin, Arash Rafiey, and Anders Yeo. Minimum cost homomorphisms to semicomplete bipartite digraphs. *SIAM J. Discrete Math.*, 22(4):1624–1639, 2008.

[97] Gregory Z. Gutin, Arash Rafiey, Anders Yeo, and Michael Tso. Level of repair analysis and minimum cost homomorphisms of graphs. *Discrete Applied Mathematics*, 154(6):881–889, 2006.

[98] Wolfgang Gutjahr, Emo Welzl, and Gerhard J. Woeginger. Polynomial graph-colorings. *Discrete Applied Mathematics*, 35(1):29–45, 1992.

[99] Magnús M. Halldórsson, Guy Kortsarz, and Hadas Shachnai. Minimizing average completion of dedicated tasks and interval graphs. In *Proceedings of the Approximation, Randomization and Combinatorial Optimization: Algorithms and Techniques (APPROX/RANDOM)*, pages 114–126, 2001.

[100] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

[101] Pavol Hell, Jing Huang, Ross M. McConnell, and Arash Rafiey. Interval-like graphs and digraphs. In *Proceedings of the 43rd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 68:1–68:13, 2018.

[102] Pavol Hell, Jing Huang, Ross M McConnell, and Arash Rafiey. Min-orderable digraphs. *SIAM Journal on Discrete Mathematics*, 34(3):1710–1724, 2020.

[103] Pavol Hell, Monaldo Mastrolilli, Mayssam Mohammadi Nevisi, and Arash Rafiey. Approximation of minimum cost homomorphisms. In *Proceedings of the 20th Annual European Symposium on Algorithms Algorithms (ESA)*, pages 587–598, 2012.

[104] Pavol Hell and Jaroslav Nesetril. On the complexity of H-coloring. *J. Comb. Theory, Ser. B*, 48(1):92–110, 1990.

[105] Pavol Hell and Jaroslav Nesetril. *Graphs and homomorphisms*. Oxford University Press, 2004.

[106] Pavol Hell and Mayssam Mohammadi Nevisi. Minimum cost homomorphisms with constrained costs. In *Proceedings of the Computing and Combinatorics - 22nd International Conference (COCOON)*, pages 194–206, 2016.

[107] Pavol Hell, Akbar Rafiey, and Arash Rafiey. Bi-arc digraphs and conservative polymorphisms. *arXiv preprint arXiv:1608.03368*, 2016.

[108] Pavol Hell and Arash Rafiey. The dichotomy of list homomorphisms for digraphs. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1703–1713, 2011.

[109] Pavol Hell and Arash Rafiey. The dichotomy of minimum cost homomorphism problems for digraphs. *SIAM J. Discrete Math.*, 26(4):1597–1608, 2012.

[110] Pavol Hell and Arash Rafiey. Monotone proper interval digraphs and min-max orderings. *SIAM J. Discrete Math.*, 26(4):1576–1596, 2012.

[111] Grete Hermann. Die frage der endlich vielen schritte in der theorie der polynomideale. *Mathematische Annalen*, 95(1):736–788, 1926.

[112] David Hilbert. Über die darstellung definiter formen als summe von formenquadraten. *Mathematische Annalen*, 32(3):342–350, 1888.

[113] David Hilbert. Ueber die theorie der algebraischen formen. *Mathematische annalen*, 36(4):473–534, 1890.

[114] David Hilbert. Über die vollen invariantensysteme. *Mathematische annalen*, 42(3):313–373, 1893.

[115] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[116] Justin Hsu, Aaron Roth, Tim Roughgarden, and Jonathan R. Ullman. Privately solving linear programs. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 612–624, 2014.

[117] Anna Huber and Vladimir Kolmogorov. Towards minimizing k-submodular functions. In *Proceedings of the 2nd International Symposium on Combinatorial Optimization (ISCO)*, pages 451–462, 2012.

[118] Pawel M. Idziak, Petar Markovic, Ralph McKenzie, Matthew Valeriote, and Ross Willard. Tractability and learnability arising from algebras with few subpowers. *SIAM J. Comput.*, 39(7):3023–3037, 2010.

[119] Satoru Iwata, Shin-ichi Tanigawa, and Yuichi Yoshida. Improved approximation algorithms for k-submodular function maximization. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 404–413, 2016.

[120] Klaus Jansen. Approximation results for the optimum cost chromatic partition problem. *J. Algorithms*, 34(1):54–89, 2000.

[121] Peter Jeavons, David A. Cohen, and Marc Gyssens. Closure properties of constraints. *J. ACM*, 44(4):527–548, 1997.

[122] Christopher Jefferson, Peter Jeavons, Martin J. Green, and M. R. C. van Dongen. Representing and solving finite-domain constraint problems using systems of polynomials. *Annals of Mathematics and Artificial Intelligence*, 67(3):359–382, Mar 2013.

[123] Tao Jiang and Douglas B West. Coloring of trees with minimum sum of colors. *Journal of Graph Theory*, 32(4):354–358, 1999.

[124] Peter Jonsson and Gustav Nordh. Introduction to the maximum solution problem. In *Complexity of Constraints - An Overview of Current Research Themes [Result of a Dagstuhl Seminar]*., pages 255–282, 2008.

[125] Michael Kapralov, Robert Krauthgamer, Jakab Tardos, and Yuichi Yoshida. Spectral hypergraph sparsifiers of nearly linear size. In *Proceedings of the 62nd IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1159–1170, 2021.

[126] Michael Kapralov, Robert Krauthgamer, Jakab Tardos, and Yuichi Yoshida. Towards tight bounds for spectral sparsification of hypergraphs. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 598–611, 2021.

[127] David R. Karger and Matthew S. Levine. Random sampling in residual graphs. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 63–66, 2002.

[128] David Kempe, Jon M. Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 137–146, 2003.

[129] Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P. Williamson. The approximability of constraint satisfaction problems. *SIAM J. Comput.*, 30(6):1863–1920, 2000.

[130] Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P. Williamson. The approximability of constraint satisfaction problems. *SIAM J. Comput.*, 30(6):1863–1920, 2000.

[131] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? In *Proceedings of the 45th Symposium on Foundations of Computer Science (FOCS)*, pages 146–154, 2004.

[132] Subhash Khot, Dor Minzer, and Muli Safra. On independent sets, 2-to-2 games, and grassmann graphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 576–589, 2017.

[133] Dmitry Kogan and Robert Krauthgamer. Sketching cuts in graphs and hypergraphs. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 367–376, 2015.

[134] Pushmeet Kohli, Lubor Ladicky, and Philip H. S. Torr. Robust higher order potentials for enforcing label consistency. *Int. J. Comput. Vis.*, 82(3):302–324, 2009.

[135] Phokion G. Kolaitis and Moshe Y. Vardi. Conjunctive-query containment and constraint satisfaction. In *Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 205–213, 1998.

[136] Vladimir Kolmogorov, Andrei A. Krokhin, and Michal Rolinek. The complexity of general-valued csps. In *Proceedings of the IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1246–1258, 2015.

[137] Andreas Krause and Daniel Golovin. Submodular function maximization. *Tractability*, 3:71–104, 2014.

[138] Andreas Krause and Carlos Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence (UAI)*, pages 324–331, 2005.

[139] Andreas Krause and Carlos Guestrin. Near-optimal observation selection using submodular functions. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI)*, pages 1650–1654, 2007.

[140] Leo G. Kroon, Arunabha Sen, Haiyong Deng, and Asim Roy. The optimal cost chromatic partition problem for trees and interval graphs. In *Proceedings of the Graph-Theoretic Concepts in Computer Science, 22nd International Workshop (WG)*, pages 279–292, 1996.

[141] Ewa Kubicka and Allen J. Schwenk. An introduction to chromatic sums. In *Computer Trends in the 1990s - Proceedings of the 1989 ACM 17th Annual Computer Science Conference, Louisville, Kentucky, USA, February 21-23, 1989*, pages 39–45, 1989.

[142] Amit Kumar, Rajsekar Manokaran, Madhur Tulsiani, and Nisheeth K. Vishnoi. On lp-based approximability for strict csps. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1560–1573, 2011.

[143] J. Kunegis. Konect: the koblenz network collection. In *Proceedings of the 22nd International World Wide Web Conference (WWW), Companion Volume*, pages 1343–1350, 2013.

[144] Benoît Larose. Algebra and the complexity of digraph csps: a survey. In *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 267–285. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[145] Monique Laurent. Semidefinite representations for finite varieties. *Math. Program.*, 109(1):1–26, 2007.

[146] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in õ(vrank) iterations and faster algorithms for maximum flow. In *Proceedings of the 55th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 424–433, 2014.

[147] Yin Tat Lee and He Sun. Constructing linear-sized spectral sparsification in almost-linear time. *SIAM J. Comput.*, 47(6):2315–2336, 2018.

[148] Michael Lewin, Dror Livnat, and Uri Zwick. Improved rounding techniques for the MAX 2-sat and MAX DI-CUT problems. In *Proceedings of the 9th International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 67–82, 2002.

[149] Mu Li, Gary L. Miller, and Richard Peng. Iterative row sampling. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 127–136, 2013.

[150] Pan Li and Olgica Milenkovic. Inhomogeneous hypergraph clustering with applications. In *Proceedings of the 31st Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 2308–2318, 2017.

[151] Pan Li and Olgica Milenkovic. Submodular hypergraphs: p-laplacians, cheeger inequalities and spectral clustering. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 3020–3029, 2018.

[152] Hui Lin and Jeff A. Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 510–520, 2011.

[153] Erik M. Lindgren, Shanshan Wu, and Alexandros G. Dimakis. Leveraging sparsity for efficient submodular data summarization. In *Proceedings of the 30th Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 3414–3422, 2016.

[154] László Lovász. On the shannon capacity of a graph. *IEEE Trans. Inf. Theory*, 25(1):1–7, 1979.

[155] László Lovász. Semidefinite programs and combinatorial optimization. In *Recent advances in algorithms and combinatorics*, pages 137–194. Springer, 2003.

[156] Gary MacGillivray and Jacobus Swarts. The c_k-extended graft construction. *Discrete Applied Mathematics*, 159(12):1293–1301, 2011.

[157] Aleksander Madry. Fast approximation algorithms for cut-based problems in undirected graphs. In *Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 245–254, 2010.

[158] Michael W. Mahoney. Randomized algorithms for matrices and data. *Found. Trends Mach. Learn.*, 3(2):123–224, 2011.

[159] Konstantin Makarychev and Yury Makarychev. Approximation algorithms for csps. In *The Constraint Satisfaction Problem: Complexity and Approximability*, pages 287–325. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[160] Dániel Marx. Graph colouring problems and their applications in scheduling. *Periodica Polytechnica Electrical Engineering (Archives)*, 48(1-2):11–16, 2004.

[161] Monaldo Mastrolilli. The complexity of the ideal membership problem for constrained problems over the boolean domain. *ACM Trans. Algorithms*, 17(4):32:1–32:29, 2021.

[162] Monaldo Mastrolilli and Arash Rafiey. On the approximation of minimum cost homomorphism to bipartite graphs. *Discrete Applied Mathematics*, 161(4-5):670–676, 2013.

[163] Ernst W. Mayr. Membership in plynomial ideals over Q is exponential space complete. In *Proceedings of the 6th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 400–406, 1989.

[164] Ernst W Mayr and Albert R Meyer. The complexity of the word problems for commutative semigroups and polynomial ideals. *Advances in mathematics*, 46(3):305–329, 1982.

[165] Ralph N McKenzie, George F McNulty, and Walter F Taylor. *Algebras, lattices, varieties*, volume 383. American Mathematical Soc., 2018.

[166] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 94–103, 2007.

[167] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. Fast constrained submodular maximization: Personalized data summarization. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 1358–1367, 2016.

[168] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1812–1818, 2015.

[169] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. Distributed submodular maximization. *J. Mach. Learn. Res.*, 17:238:1–238:44, 2016.

[170] Baharan Mirzasoleiman, Morteza Zadimoghaddam, and Amin Karbasi. Fast distributed submodular cover: Public-private data summarization. In *Proceedings of the 30th Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 3594–3602, 2016.

[171] Marko Mitrovic, Mark Bun, Andreas Krause, and Amin Karbasi. Differentially private submodular maximization: Data summarization in disguise. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 2478–2487, 2017.

[172] Marko Mitrovic, Ehsan Kazemi, Morteza Zadimoghaddam, and Amin Karbasi. Data summarization at scale: A two-stage submodular approach. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 3593–3602, 2018.

[173] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[174] T.S. Motzkin. The arithmetic-geometric inequality, inequalities (proc. sympos. wright-patterson air force base, ohio, 1965), 1967.

[175] George L. Nemhauser and Laurence A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Math. Oper. Res.*, 3(3):177–188, 1978.

[176] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Math. Program.*, 14(1):265–294, 1978.

[177] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Math. Program.*, 14(1):265–294, 1978.

[178] Ryan O'Donnell. SOS is not obviously automatizable, even approximately. In *Proceedings of the 2017 ACM Conference on Innovations in Theoretical Computer Scienc (ITCS)*, pages 59:1–59:10, 2017.

[179] Naoto Ohsaka and Yuichi Yoshida. Monotone k-submodular function maximization with size constraints. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 694–702, 2015.

[180] Christos H. Papadimitriou, Michael Schapira, and Yaron Singer. On the hardness of being truthful. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 250–259, 2008.

[181] Dona Papert. Congruence relations in semi-lattices. *J. London Math. Soc.*, 39:723–729, 1964.

[182] Shameem Puthiya Parambath, Nicolas Usunier, and Yves Grandvalet. A coverage-based approach to recommendation diversity on similarity graph. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys)*, pages 15–22, 2016.

[183] Pablo A Parrilo. An explicit construction of distinguished representations of polynomials nonnegative over finite sets. *IfA AUT02-02, ETH Zürich*, 2002.

[184] Pablo A Parrilo. Exploiting algebraic structure in sum of squares programs. In *Positive polynomials in control*, pages 181–194. Springer, 2005.

[185] George M Phillips. *Interpolation and approximation by polynomials*, volume 14. Springer Science & Business Media, 2003.

[186] EL Post. The two-valued iterative systems of mathematical logic. number 5 in annals of math. *Studies. Princeton Univ. Press*, 1941.

[187] Akbar Rafiey, Arash Rafiey, and Thiago Santos. Toward a dichotomy for approximation of h-coloring. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 91:1–91:16, 2019.

[188] Akbar Rafiey and Yuichi Yoshida. Fast and private submodular and k-submodular functions maximization with matroid constraints. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 7887–7897, 2020.

[189] Akbar Rafiey and Yuichi Yoshida. Sparsification of decomposable submodular functions. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI)*, pages 10336–10344, 2022.

[190] Prasad Raghavendra. Optimal algorithms and inapproximability results for every csp? In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 245–254, 2008.

[191] Prasad Raghavendra and David Steurer. How to round any CSP. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 586–594, 2009.

[192] Prasad Raghavendra and Benjamin Weitz. On the bit complexity of sum-of-squares proofs. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming, (ICALP)*, pages 80:1–80:13, 2017.

[193] Alexander A. Razborov. Lower bounds for the polynomial calculus. *Comput. Complex.*, 7(4):291–324, 1998.

[194] Fred Richman. Constructive aspects of noetherian rings. *Proceedings of the American Mathematical Society*, 44(2):436–441, 1974.

[195] Shinsaku Sakaue. On maximizing a monotone k-submodular function subject to a matroid constraint. *Discrete Optimization*, 23:105–113, 2017.

[196] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC)*, pages 216–226, 1978.

[197] Abraham Seidenberg. Constructions in algebra. *Transactions of the American Mathematical Society*, 197:273–313, 1974.

[198] Ishant Shanu, Chetan Arora, and Parag Singla. Min norm point algorithm for higher order MRF-MAP inference. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5365–5374, 2016.

[199] Ajit P. Singh, Andrew Guillory, and Jeff A. Bilmes. On bisubmodular maximization. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1055–1063, 2012.

[200] Tasuku Soma and Yuichi Yoshida. Spectral sparsification of hypergraphs. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete (SODA)*, pages 2570–2581, 2019.

[201] Daniel A. Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM J. Comput.*, 40(4):981–1025, 2011.

[202] Aravind Srinivasan. Improved approximations of packing and covering problems. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing (STOC)*, pages 268–276, 1995.

[203] Peter Stobbe and Andreas Krause. Efficient minimization of decomposable submodular functions. In *Proceedings of the 24th Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 2208–2216, 2010.

[204] Matthew J. Streeter and Daniel Golovin. An online algorithm for maximizing submodular functions. In *Proceedings of the 22nd Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 1577–1584, 2008.

[205] Ágnes Szendrei. Clones in universal algebra. *Les presses de L'universite de Montreal*, 1986.

[206] Rustem Takhanov. A dichotomy theorem for the general minimum cost homomorphism problem. In *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 657–668, 2010.

[207] Johan Thapper and Stanislav Zivný. The complexity of finite-valued csps. *J. ACM*, 63(4):37:1–37:33, 2016.

[208] Johan Thapper and Stanislav Zivný. The limits of SDP relaxations for general-valued csps. *ACM Trans. Comput. Theory*, 10(3):12:1–12:22, 2018.

[209] Sebastian Tschiatschek, Rishabh K. Iyer, Haochen Wei, and Jeff A. Bilmes. Learning mixtures of submodular functions for image collection summarization. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 1413–1421, 2014.

[210] Hannes Uppman. The complexity of three-element min-sol and conservative min-cost-hom. In *Proceedings of the 40th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 804–815, 2013.

[211] Hannes Uppman. Computational complexity of the extended minimum cost homomorphism problem on three-element domains. In *Proceedings of the 31st International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 651–662, 2014.

[212] Marc R.C. van Dongen. *Constraints, Varieties, and Algorithms*. PhD thesis, Department of Computer Science, University College, Cork, Ireland, 2002.

[213] Nate Veldt, Austin R. Benson, and Jon M. Kleinberg. Hypergraph cuts with general splitting functions. *CoRR*, abs/2001.02817, 2020.

[214] Nate Veldt, Austin R. Benson, and Jon M. Kleinberg. Minimizing localized ratio cut objectives in hypergraphs. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1708–1718, 2020.

[215] Sara Vicente, Vladimir Kolmogorov, and Carsten Rother. Joint optimization of segmentation and appearance models. In *ICCV*, pages 755–762, 2009.

[216] Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 67–74, 2008.

[217] Justin Ward and Stanislav Zivny. Maximizing bisubmodular and k-submodular functions. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1468–1481, 2014.

[218] Benjamin Weitz. *Polynomial proof systems, effective derivations, and their applications in the sum-of-squares hierarchy*. PhD thesis, UC Berkeley, 2017.

[219] Angelika Wiegele. *Nonlinear optimization techniques applied to combinatorial optimization problems*. na, 2006.

[220] Yuichi Yoshida. Nonlinear Laplacian for digraphs and its applications to network analysis. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining (WSDM)*, pages 483–492, 2016.

[221] Yuichi Yoshida. Cheeger inequalities for submodular transformations. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2582–2601, 2019.

[222] Dmitriy Zhuk. A proof of CSP dichotomy conjecture. In *Proceedings of the 58th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 331–342, 2017.