Promise Constraint Satisfaction Problems

Thesis submitted for the degree of
Doctor of Philosophy

Tamio-Vesa Nakajima
Trinity College, Oxford

Trinity Term, 2025

To my mother and Laura

Declaration

I certify that I wrote this thesis by myself, with feedback from my supervisor. The results in this thesis are based on work done in collaboration with others, in particular the following peer-reviewed works.

[CKKNŽ24] Lorenzo Ciardo, Marcin Kozik, Andrei A. Krokhin, Tamio-Vesa Nakajima and Stanislav Živný. '1-in-3 vs. Not-All-Equal: Dichotomy of a broken promise'. In: *LICS*. ACM, 2024, 24:1–24:12. DOI: 10.1145/3661814.3662069.

[CKKNŽ25] Lorenzo Ciardo, Marcin Kozik, Andrei A. Krokhin, Tamio-Vesa Nakajima and Stanislav Živný. '1-in-3 vs. Not-All-Equal: Dichotomy of a Broken Promise'. In: *ACM Trans. Comput. Log.* 26.2 (2025), 10:1–10:22. DOI: 10.1145/3719007.

[HMNŽ24] Johan Håstad, Björn Martinsson, Tamio-Vesa Nakajima and Stanislav Živný. 'A Logarithmic Approximation of Linearly-Ordered Colourings'. In: *APPROX/RANDOM*. Vol. 317. LIPIcs. LZI, 2024, 7:1–7:6. DOI: 10.4230/lipics.approx/random.2024.7.

[NŽ22] Tamio-Vesa Nakajima and Stanislav Živný. 'Linearly Ordered Colourings of Hypergraphs'. In: *ICALP*. Vol. 229. LIPIcs. LZI, 2022, 128:1–128:18. DOI: 10.4230/lipics.icalp.2022.128.

[NŽ23a] Tamio-Vesa Nakajima and Stanislav Živný. 'Boolean symmetric vs. functional PCSP dichotomy'. In: *LICS*. IEEE, 2023, pp. 1–12. DOI: 10.1109/lics56636.2023.10175746.

[NŽ23b] Tamio-Vesa Nakajima and Stanislav Živný. 'Linearly Ordered Colourings of Hypergraphs'. In: *ACM Trans. Comput. Theory* 14.3-4 (2023), pp. 1–19. DOI: 10.1145/3570909.

[NŽ24c] Tamio-Vesa Nakajima and Stanislav Živný. 'On the Complexity of Symmetric vs. Functional PCSPs'. In: *ACM Trans. Algorithms* 20.4 (2024), 33:1–33:29. DOI: 10.1145/3673655.

[NŽ25a] Tamio-Vesa Nakajima and Stanislav Živný. *A Dichotomy for Maximum PCSPs on Graphs*. 2025. arXiv: 2406.20069 [cs.DS].

[NŽ25b] Tamio-Vesa Nakajima and Stanislav Živný. 'Maximum And- vs. Even-SAT'. In: *APPROX/RANDOM*. Vol. 353. LIPIcs. LZI, 2025, 3:1–3:8. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2025.3.

[NŽ25c] Tamio-Vesa Nakajima and Stanislav Živný. 'Maximum Bipartite vs. Triangle-Free Subgraph'. In: *ICALP*. Vol. 334. LIPIcs. LZI, 2025, 121:1–121:16. DOI: 10.4230/lipics.icalp.2025.121.

The precise distribution of papers to chapters will be specified at the beginning of each chapter. I have contributed significantly to all of these publications; some particularities

are worth specifying. The work [HMNŽ24] is a merger of work produced with my supervisor (distributed in preprint form as [NŽ24a]), with the independent work of Håstad and Martinsson. The work [NŽ25a] is a preprint full version of [NŽ25c].

Acknowledgements

First, I want to thank my supervisor Standa Živný for his incredible support throughout my DPhil. Without the support of such a good supervisor, I would not have finished this degree. I also want to thank Andrew Ker, with whom I took my first steps in research, and who advised me to pursue a DPhil with Standa.

In the same breath, I want to thank the other CSP researchers at Oxford, especially Standa's other students and postdocs (Alex Brandts, Tom Orton, Jakub Opršal, Lorenzo Ciardo, Caterina Viola, Silvia Butti, Karolina Okrasa, George Osipov and Zephyr Verwimp) for making Oxford a wonderful environment to do research in. Your friendship helped me see the joy of research.

I want to thank all of my coauthors for very pleasant and productive collaborations: Standa Živný, Marcin Kozik, Andrei Krokhin, Lorenzo Ciardo, Gianluca Tasinato, Marcin Wrochna, Jakub Opršal, Marek Filakovský, Johan Håstad, Björn Martinsson, Zephyr Verwimp, Karolina Okrasa and Benjamin Bedert. Thank you for your invaluable insights and ideas, and for making writing papers fun.

I want to acknowledge all my friends and family, without whose support this endeavour would have been impossible. Whether Oxford colleagues, friends from Beiuș or former Romanian Olympiad rivals — each one of you contributed to my well-being and productivity throughout this process, and for this I thank you.

In particular, I want to thank Dani Posdărăscu and Andrei Heidelbacher, who first opened my eyes to the world of competitive programming. I also want to thank Andrei-Costin Constantinescu for all the delightful discussions about TCS and mathematics we have had, long into the night. I want to thank Mihaela Cismaru and Oana Petrescu for keeping me company online. I want to thank the friends I made in Oxford, Maria Crăciun, Raluca Vasiu, Tudor Caba, Teo Reu, Ruxandra Nanu, Clara Dima, Otilia Cășuneanu, Mihaela Stoian, Maria Tașcă, Iulia Vatau, Alexandra Ondrea, Șerban Cercelescu, Reo Barborică, Bianca Pasca, Yundi Li, Sandra Kiefer, and all of the others: your constant companionship was what made life in Oxford beautiful.

I want to acknowledge my father, my sister and my grandparents for their constant encouragement and support throughout this process.

Most of all, I want to thank my mother, and Laura Marin. To my mother: I would not be here if you didn't inspire in me an obsession with mathematics. To Laura: thinking of you has given me someone to fight for in the last years.

Abstract

We explore the complexity of various approximations of constraint satisfaction problems. In the constraint satisfaction problem (CSP), we are given a set of variables, a domain for the variables, and a set of constraints on those variables, and must construct an assignment of values from the domain to the variables so that all constraints are satisfied.

We begin with several results for crisp promise constraint satisfaction problems (PCSPs). The PCSP is a qualitative approximation of the CSP, in which we must find a weak solution to a CSP instance that admits a strong solution.

We first give positive results for approximate linearly ordered (LO) colouring of hypergraphs, culminating in the following: given an LO 2-colourable 3-uniform hypergraph with n vertices, we can find an LO $\log_2(n)$-colouring. We complement this result with a hardness result for linearly ordered colouring: we show that finding an LO ℓ-colouring of an LO k-colourable r-uniform hypergraph is NP-hard whenever $r \geq \ell - k + 4$.

For rainbow-free hypergraph PCSPs, we give several complexity classifications: whenever $\mathbf{A}, \mathbf{B}$ have one ternary, symmetric, rainbow-free relation, we classify PCSP($\mathbf{1\text{-}in\text{-}3}, \mathbf{B}$), PCSP($\mathbf{A}, \mathbf{NAE}$) and, provided $\mathbf{A}, \mathbf{B}$ are (in an appropriate sense) bipartite, PCSP($\mathbf{A}, \mathbf{B}$).

We move on to *functional* PCSPs. For relational structures $\mathbf{A} \to \mathbf{B}$ where $\mathbf{B}$ is functional and $\mathbf{A}$ is symmetric, under appropriate technical conditions we classify the complexity of PCSP($\mathbf{A}, \mathbf{B}$).

We conclude our discussion of crisp PCSPs by showing that some well-known algorithms, namely the *Affine Integer Programming relaxation* (Brakensiek and Guruswami, SICOMP 2018) and the *combined Basic Linear Programming and Affine Integer Programming relaxation* (Brakensiek, Guruswami, Wrochna and Živný, SICOMP 2020) are equivalent for a surprisingly large fragment of PCSPs (including e.g. symmetric PCSPs).

We then continue with a discussion of maximum PCSPs (maxPCSPs). These are a combined qualitative and quantitative approximation of the CSP. We first give a full classification of maxPCSPs with one Boolean, binary relation, alongside several generalisations. We then give a dichotomy result for maxPCSPs on graphs, conditional on the Unique Games Conjecture (Khot, STOC 2002). Finally, we discuss the maximum bipartite versus triangle-free subgraph problem, giving a nontrivial algorithmic result.

Contents

Declaration i

Acknowledgements iii

Abstract v

1. Introduction 1
   1.1. Promise CSPs . . . 4
   1.2. Maximum PCSPs . . . 8

2. Preliminaries 13
   2.1. Miscellaneous notations . . . 13
   2.2. Graphs, hypergraphs and relational structures . . . 13
      2.2.1. Basic definitions . . . 13
      2.2.2. Homomorphisms, sums, products . . . 14
   2.3. Minions and PCSPs . . . 15
   2.4. Further algebraic hardness . . . 19

I. Crisp Promise Constraint Satisfaction Problems 27

3. Algorithms for Non-Constant Regimes 29
   3.1. Introduction . . . 29
   3.2. PCSP $\left(\mathbf{LO}_2, \mathbf{LO}_{O(\sqrt{n})}\right)$ . . . 32
   3.3. PCSP $\left(\mathbf{LO}_2, \mathbf{LO}_{O(\sqrt{n/\log n})}\right)$ . . . 36
   3.4. PCSP $\left(\mathbf{LO}_2, \mathbf{LO}_{O(\sqrt[3]{n})}\right)$ . . . 39
   3.5. PCSP $\left(\mathbf{LO}_2, \mathbf{LO}_{\log_2 n}\right)$ . . . 42

4. Hardness of Linearly Ordered Colouring 49
   4.1. Introduction . . . 49
      4.1.1. Algebraic preliminaries . . . 51
   4.2. Equivalence of Conjecture 4.1 and Conjecture 4.4 . . . 52
   4.3. First hardness result . . . 54
   4.4. Second hardness result . . . 57
      4.4.1. Minion relations . . . 58
      4.4.2. Hardness for large arity . . . 64

5. Rainbow-Free Structures 67
   5.1. Introduction . . . 67
   5.2. Tractability . . . 70

5.3. Hardness of Theorem 5.6 . . . 71
5.4. Hardness of Theorem 5.7 . . . 79
5.4.1. Proof for directed cycles . . . 80
5.4.2. Proof of reduction . . . 83

6. Functional Structures 89
6.1. Introduction . . . 89
6.2. Deductions . . . 92
6.2.1. Additivity . . . 93
6.2.2. Dependency . . . 94
6.2.3. Formal system . . . 96
6.2.4. Super-connectedness . . . 100
6.2.5. Proofs of super-connectedness . . . 102
6.3. Proof of dichotomy . . . 106
6.3.1. Tractability . . . 107
6.3.2. Hardness . . . 110

7. Equivalencies Between Classical Algorithms 113
7.1. Introduction . . . 113
7.2. Proof of main theorem . . . 117
7.3. Proof of characterisations . . . 119

II. Maximum Promise Constraint Satisfaction Problems 121

8. A Dichotomy for Boolean, Binary maxPCSPs 123
8.1. Introduction . . . 123
8.2. Tractability of promise max-and . . . 127
8.3. Applications of promise max-and . . . 128

9. A Dichotomy for maxPCSPs on Graphs 131
9.1. Introduction . . . 131
9.2. Algorithm for maxPCSP(K_2, K_3) . . . 134
9.2.1. Goemans-Williamson algorithm . . . 134
9.2.2. Frieze-Jerrum rounding; our algorithm . . . 135
9.2.3. Derandomisation . . . 140
9.3. Hardness for maxPCSP(K_2, G_3) . . . 144
9.4. Hardness for maxPCSP(C_{2k+1}, K_ℓ) . . . 149
9.4.1. Proof of Proposition 9.23 . . . 150
9.4.2. Reduction . . . 153
9.5. Algorithm for maxPCSP(K_2, G_3) . . . 156

Bibliography 161

1. Introduction

In computational complexity, the broad goal is as follows: given some computational problem, determine whether the problem is (in some sense) easy, or if it is (in some sense) hard. This area began with the study of *decidability*. In this case, a problem is easy if it can be solved by any computational procedure whatsoever (appropriately formalised in some way), and it is hard if this is not the case. This first approach has some nice properties:

1. The classification into hard and easy problems is *complete*, i.e. by construction every problem is either hard or easy.
2. It is possible to *unconditionally* prove the hardness of problems in this setting [Tur37; Chu36]. (Indeed, proving tractability is usually done by simply exhibiting an algorithm; whereas proving hardness can be much more difficult.)
3. For most natural models of computation, the notion of decidability does not change. Indeed, the *Church-Turing thesis* [Tur04; Chu36] asserts that all reasonable models of computation are equivalent (to either Turing machines or λ-calculus respectively).

With this being established, it becomes reasonable to consider the question of *efficiency*. That some problem or another *can* be solved is good to know; however, in practice we also want our computation to take a reasonable time and to use a reasonable amount of space. Now, we are interested in the study of computational complexity where the easy problems are *efficiently* computable functions. How does one formalise the notion of "efficiently computable"? Cobham [Cob65] and Edmonds [Edm65] first proposed that an "efficient algorithm" is one that runs in polynomial time. This has the nice property that it is invariant for most natural models of computation (similarly to Item 3 above). On the other hand, the other nice properties from before are no longer conserved. In particular, it is quite difficult to unconditionally prove hardness for this notion of efficiency — the famous P vs. NP Millennium problem essentially asks us to prove that e.g. 3-SAT is not solvable in polynomial time.[1]

Thus, we will change the notion of hardness to make proving hardness easier. Note that many natural computational problems ask us to find a solution to some problem, where the solution is easy to check but hard to find (a priori). Take for example solving systems of linear equations, finding colourings of graphs, or solving Sudokus. The class of all such computational problems is called NP, i.e. the class of all problems admitting a polynomial-time verifier for their solutions, or equivalently the class of problems that can be solved in polynomial time while allowing nondeterministic guessing [Kar72]. We will take as a convention that a problem is hard if it is as hard as any problem within NP (i.e. it is NP-hard), in the sense that any problem within NP can be reduced to it in polynomial time. The famous Cook-Levin theorem [Coo71; Lev73] showed that 3-SAT is NP-hard — hence,

[1] Recall that 3-SAT is just Boolean satisfiability of CNF formulae where each clause has at most 3 literals, e.g. checking if $(x \vee y \vee \neg z) \wedge (y \vee z \vee t)$ is satisfiable.

it is sufficient, for this standard of hardness, to show that a problem is at least as difficult as 3-SAT. (For this easy versus hard dichotomy to formally make sense, one should in principle show that P ≠ NP — no one has been able to prove this so far, so we will assume this.) Thus (modulo the assumption P ≠ NP), the "P is easy, NP-hard is hard" convention has many of the pleasant properties of the "decidable is easy, undecidable is hard" convention. The main issue is Item 1: assuming P ≠ NP, there exist problems in NP that are neither solvable in polynomial time, nor NP-hard!

Such problems are called *NP-intermediate*. No problem has been unconditionally proved to be NP-intermediate yet (indeed, this would imply that P ≠ NP). However, that these exist under the necessary assumption P ≠ NP was proved by Ladner [Lad75].

How can we recover this nice property? A natural approach is to consider a *structured subset* of the entire universe of possible computational problems. Perhaps the structure will allow us to prove a *dichotomy result*, i.e. a classification of all problems in question into hard and easy problems. Furthermore, perhaps the structure will allow us to derive *structural reasons* for tractability and/or hardness.

One such structured subset of problems is the family of *constraint satisfaction problems*, or CSPs for short. In a constraint satisfaction problem, one is given a set of variables X, a domain D, and a set of constraints C. Each constraint is applied to a certain tuple of variables, and specifies the values which that tuple is allowed to take. The goal is to assign the variables from X values from D so that each constraint is satisfied. As stated, this problem is clearly NP-hard (indeed, it is at least as hard as 3-SAT, since 3-SAT is clearly a CSP), so we will need to constrain the problem a bit more. Indeed, for a *relational structure* $\mathbf{A}$ (which we will define formally in Chapter 2 — for now, imagine that it is a domain for all the variables, together with a list of allowed constraint relations), we let $\mathrm{CSP}(\mathbf{A})$ denote the constraint satisfaction problem where the domain is the domain of $\mathbf{A}$, and where each constraint is taken from the allowed constraint relations of $\mathbf{A}$. In other words, we have parametrised the CSP by the *constraint relations that we allow to appear in each instance*. We call such a CSP a *fixed-template CSP* if we want to emphasize the fact that the allowed constraint relations are fixed — however, in general, when we say CSP we will mean a fixed-template CSP.

Let us begin by remarking that CSPs encompass a very wide variety of problems, in many different areas of theoretical computer science. For example, k-colouring of graphs is a fixed-template CSP, as is solving systems of linear equations, and various variants of Boolean satisfiability such as 2-SAT, or (co)-Horn SAT. (Each of these is a different fixed-template CSP.) Indeed, *every computational problem* is equivalent to a fixed-template CSP (perhaps on an infinite domain) up to polynomial time Turing reductions [BG08]. Thus, results about CSPs are not mere theoretical curiosities (no matter the theoretical interest and beauty they may present): they have broad application.

In particular, let us consider the case of Boolean CSPs — in other words, CSPs where the domain is $\{0, 1\}$ or, equivalently, $\{\mathit{true}, \mathit{false}\}$. In this case, *Schaefer's dichotomy theorem* [Sch78] gives us a version of Item 1. Indeed, consider some problem $\mathrm{CSP}(\mathbf{A})$. $\mathrm{CSP}(\mathbf{A})$ is tractable if all of the constraint relations in $\mathbf{A}$ fall into one of the following cases:

Affine. They are conjunctions of modulo-2 equations.

Horn. They are conjunctions of Horn clauses.[2]

Co-Horn. They are conjunctions of co-Horn clauses.

Bijunctive. They are conjunctions of disjunctive clauses of size at most 2.

0-valid. They are satisfied by setting all variables to 0.

1-valid. They are satisfied by setting all variables to 1.

If CSP(**A**) falls into none of the previous cases, then CSP(**A**) is NP-hard. (Observe that it is not sufficient that each constraint relation falls into one of the classes. Rather, it is necessary that there exists a class into which *all* the relations fall.) A similar dichotomy also holds for CSPs where **A** must be a graph [HN90]. (For graphs, excluding trivial CSPs — i.e. CSPs where the answer is always yes, or where the answer is only yes on empty instances — the only tractable CSPs are equivalent to graph 2-colouring.)

Constraint satisfaction problems have a rich theory. Indeed, it turns out that the complexity of a (finite domain) CSP is uniquely determined by the symmetries of the solution space of the problem: the study of CSPs through this algebraic invariant is known as the *algebraic approach to CSPs* [JCG97; Jea98; BJ01; BJK05], which we will now informally describe. For the discussion that follows, we will consider only *finite domain* CSPs. Fix some constraint satisfaction problem CSP(**A**). Then, informally, a *polymorphism* (which we will define formally in Chapter 2) is a function which takes in several values from the domain of **A**, and outputs a value from the domain of **A**, so that if this function is applied component-wise to several solutions of an instance of CSP(**A**), the output is yet another solution for that instance. For example, if CSP(**A**) encodes modulo-2 equations, then

$$(x, y, z) \mapsto x + y + z \mod 2$$

is a (ternary) polymorphism. To see that this is the case, consider any particular constraint, say $a + b + c + d \equiv 1 \bmod 2$. Suppose we have three solutions that satisfy this constraint, i.e. we have (a, b, c, d), (a', b', c', d') and (a'', b'', c'', d'') such that

$$a + b + c + d \equiv a' + b' + c' + d' \equiv a'' + b'' + c'' + d'' \equiv 1 \mod 2$$

Then, we observe that

$$(a + a' + a'') + (b + b' + b'') + (c + c' + c'') + (d + d' + d'') \equiv 1 \mod 2$$

In other words, combining the three solutions pointwise using $(x, y, z) \mapsto x + y + z \bmod 2$ gives rise to another solution. Similarly, for 0/1-valid CSPs the functions $x \mapsto 0$ and $x \mapsto 1$ respectively are polymorphisms; for Horn and co-Horn SAT the functions min and max respectively are polymorphisms; and for bijunctive CSPs the function maj (which on input (x, y, z) returns the most frequent element among x, y, z) is a polymorphism. Indeed, the algorithms for these problems can be generalised to *any problem which admits these polymorphisms*. Hence, we see that the tractability part of Schaefer's dichotomy theorem can be shown using polymorphisms.

To prove the hardness part of Schaefer's dichotomy theorem, we need one further observation. Let Pol(**A**) denote the set of polymorphisms of **A**. It can be shown that, if

[2] Recall that a *Horn clause* is a disjunctive clause where there is at most one un-negated literal; and a *co-Horn* clause is a disjunctive clause where there is at most one negated literal.

$\mathrm{Pol}(\mathbf{A}) \subseteq \mathrm{Pol}(\mathbf{B})$, then $\mathrm{CSP}(\mathbf{B})$ reduces to $\mathrm{CSP}(\mathbf{A})$. Indeed, the reduction is a gadget reduction: each relation in $\mathbf{B}$ can be defined by conjunction and existential quantification from the relations in $\mathbf{A}$.

Next, we observe that the set of polymorphisms of a CSP forms an algebraic structure known as a *clone* (i.e. it is a set of functions which are closed under composition, permuting inputs and introducing dummy variables, which contains all projections). It turns out that Boolean clones (i.e. clones containing functions on two-element domains) are fully and explicitly classified; the lattice of Boolean clones under inclusion is known as *Post's lattice* [Pos41]. The fact that $\mathrm{Pol}(\mathbf{A}) \subseteq \mathrm{Pol}(\mathbf{B})$ implies that $\mathrm{CSP}(\mathbf{B})$ reduces to $\mathrm{CSP}(\mathbf{A})$ means that the set of tractable CSPs corresponds to an join-closed subset of Post's lattice, and the set of NP-hard CSPs to a downwards-closed subset of Post's lattice. Thus, inspecting the lattice, we see that the maximal clone *below* all of the tractable cases is the clone containing only functions that consider one fixed input and either return it, or return its negation. It suffices to prove that any CSP that admits only these functions as polymorphisms must be NP-hard — since all of these problems are equivalent, we might as well consider the *maximal* problem of this kind, i.e. the problem that admits as relations all those relations *invariant* under the specified polymorphisms. It turns out that this particular problem includes the relation e.g. for ternary positive not-all-equal SAT, which is a classical NP-hard problem. This implies Schaefer's dichotomy theorem.

Can this result be generalised? This was an open question for many years, posed by Feder and Vardi [FV93]: they conjecture that every fixed-template, finite-domain CSP must be either solvable in polynomial time, or NP-hard. After decades of effort, the Feder-Vardi dichotomy was proved in 2017 independently by Bulatov and Zhuk.

Theorem 1.1 (CSP dichotomy theorem [Bul17; Zhu20]). If $\mathrm{CSP}(\mathbf{A})$ is a fixed-template, finite domain CSP, then $\mathrm{CSP}(\mathbf{A})$ is either solvable in polynomial time or NP-hard.

This result was proved using precisely the algebraic approach mentioned above. While incredible in scope and depth, it leaves many questions to be answered. In particular, it leaves open questions relating to *approximation*, which we will discuss in the section that follows.

1.1. Promise CSPs

One of the most classical examples of CSP is the *graph k-colouring problem*. In this problem, we are given a graph G, and must create a k-colouring, i.e. an assignment of one of k labels to each vertex, so that no edge connects two vertices with the same label. We denote this problem by $\mathrm{CSP}(K_k)$. Famously, for every $k \geq 3$ this problem is NP-hard, whereas for $k = 1, 2$ it is tractable [Kar72]. Recall also that the *chromatic number* of a graph G, denoted by $\chi(G)$, is the smallest number k so that G is k-colourable.

Before CSPs were even studied as a topic in and of themselves, Garey and Johnson [GJ76] asked whether it is possible to approximate $\chi(G)$. Unfortunately, it is NP-hard to approximate this number to within $n^{1-\varepsilon}$ for an input graph with n vertices [Zuc07]. Thus, let us

consider an easier setting: supposing we are promised that $\chi(G)$ is at most some fixed k, and that the input graph has n vertices — is it possible then to find some $f(n,k)$-colouring?

It was conjectured by Garey and Johnson [GJ76] that, for any fixed $3 \leq k \leq \ell$, finding an ℓ-colouring of a k-colourable graph is NP-hard. This conjecture is widely believed by the community, and it also follows [GS20] as a consequence of the *d-to-1 conjecture with perfect completeness* [Kho02]. Furthermore, it has been proved unconditionally that finding a 5-colouring of a 3-colourable graph [BKO19; BBKO21] (and indeed even finding a $\left(\binom{k}{\lfloor k/2 \rfloor} - 1\right)$-colouring of a k-colourable graph for $k \geq 4$ [KOWŽ23]) is NP-hard. On the tractability side, it was proved by Wigderson in the 1980's that finding an $O(\sqrt{n})$-colouring of a 3-colourable graph is possible in polynomial time [Wig82]. This was steadily improved to the current state of the art, due to Kawarabayashi, Thorup and Yoneda [KTY24]: given a 3-colourable graph, it is possible to find an $O(n^{0.19747})$-colouring. We underline the immense gap between the best known upper and lower bounds for finding colourings for 3-colourable graphs: it is NP-hard to find a 5-colouring (and the complexity of finding a 6-colouring is not known); whereas it is possible to find an $O(n^{0.19747})$- colouring in polynomial time.

This vast gap shows us that we understand the complexity of *approximating* CSPs much less than solving them exactly. How are we to properly improve this situation? One reasonable approach is to place the problem of approximate graph colouring within its proper context, solving other strongly related (potentially simpler to classify) problems, strengthening our tools until we solve the problem we wanted to solve initially. In other words: rather than try to "jump" to the most difficult problem, let us build the "rungs in the ladder" to this problem step by step. This is one of the motivations for studying *promise* CSPs (PCSPs), which we now discuss.

Informally (formal definitions will be given in Chapter 2), PCSPs are defined as follows. Suppose $\mathrm{CSP}(\mathbf{A})$ and $\mathrm{CSP}(\mathbf{B})$ are two CSPs, where a solution to $\mathrm{CSP}(\mathbf{A})$ can be turned element-by-element into a solution to $\mathrm{CSP}(\mathbf{B})$ — i.e. $\mathrm{CSP}(\mathbf{A})$ is *harder* than $\mathrm{CSP}(\mathbf{B})$.[3] Then, $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is defined as follows: given a solvable instance of $\mathrm{CSP}(\mathbf{A})$, find a solution to that instance *interpreted as an instance of* $\mathrm{CSP}(\mathbf{B})$. This problem is clearly no harder than $\mathrm{CSP}(\mathbf{A})$ or $\mathrm{CSP}(\mathbf{B})$, hence we are in particular interested in the case when both of these problems are NP-hard. Observe that this is not formally a decision problem — we are given some instance, and we must create some solution, not merely decide if one exists. We have not discussed this distinction in the case of CSPs, since it turns out that for CSPs *finding* a solution is no harder than *merely deciding* that it exists [BJK05; Coh04]. For PCSPs, the appropriate corresponding decision problem is the following: given an instance of $\mathrm{CSP}(\mathbf{A})$, output YES if it is solvable, and NO if it is not even solvable when interpreted as an instance of $\mathrm{CSP}(\mathbf{B})$. With these definitions, the decision version immediately reduces to the search version — however, in contrast to the CSP case, it is *not yet known* if they are equivalent. This technical wrinkle has not yet proved to be particularly significant in practice: for most problems (and indeed for every case we will deal with in this thesis), we can prove tractability for the (harder) search version, and hardness for the (easier) decision version. (See, however [Lar25].)

[3] To be precise, for this to be possible, the admissible constraint relations in $\mathrm{CSP}(\mathbf{A})$ have to be paired up in some specified way with the admissible constraint relations of $\mathrm{CSP}(\mathbf{B})$, with these pairings respecting arities. This is so that we can interpret an instance of $\mathrm{CSP}(\mathbf{A})$ as one of $\mathrm{CSP}(\mathbf{B})$: each constraint relation from $\mathbf{A}$ in the instance is interpreted as the corresponding constraint relation in $\mathbf{B}$.

Thus, for example, recalling that $\mathrm{CSP}(K_k)$ denotes k-colouring, $\mathrm{PCSP}(K_k, K_\ell)$ is the problem of finding an ℓ-colouring of a k-colourable graph. However, this is far from the only case that is of interest. Indeed, other problems in this framework are interesting for a variety of reasons. We will quickly go through some of these problems, in particular some of the problems that we will explore later on in this thesis.

Let us begin with *approximate linearly-ordered (LO) colouring*. Given a hypergraph $H = (V, E)$, a *linearly-ordered colouring* of H is an assignment $c : V \to \{0, 1, \ldots\}$ of non-negative integers to the vertices of H, so that for every edge $\{x_1, \ldots, x_r\} \in E$, there exists a unique maximum among $\{c(x_1), \ldots c(x_r)\}$.[4] Consider the case of 3-uniform hypergraphs (i.e. every edge has cardinality 3), and the case of LO 2-colouring (i.e. the image of c is $\{0, 1\}$). Observe that in this case, every edge must be assigned exactly one 1 and exactly two 0s — in other words, this problem is identical to *monotone 1-in-3 SAT*. In the *approximate LO colouring problem*, we are given an n-vertex LO 2-colourable 3-uniform hypergraph, and we must find an $f(n)$-colouring of this hypergraph for some f. For approximate graph colouring, all tractability results so far have been for the case $f(n) = O(n^c)$ for some fixed c. In contrast to this, for approximate LO colouring a solution is possible even for $f(n) = \log_2(n)!$[5] This result uses the *linear structure* of monotone 1-in-3 SAT. Observe that the constraint "precisely one of x, y, z are 1, the others are 0" is equivalent to "$x, y, z \in \{0, 1\}$ and $x + y + z = 1$". This is what we will exploit to prove our tractability result. This is as opposed to, for example, graph colouring — which has no linear structure except in the (tractable) case of 2-colouring. This result also has some interesting connections with infinite-domain constraint satisfaction, cf. Remark 3.28. We will describe this result, along with several results that lead to it, in Chapter 3.

Next, in Chapter 4, we will explore the hardness side of this problem. Barto, Batistelli and Berg conjectured that approximate LO colouring with a constant number of colours on 3-uniform hypergraphs is NP-hard; we restate this conjecture to be precise.

Conjecture 1.2 ([BBB21]). Fix $2 \leq k \leq \ell$. The following problem is NP-hard: given a 3-uniform hypergraph, output YES if it is LO k-colourable, and NO if it is not even LO ℓ-colourable.

First, we discuss some connections between Conjecture 1.2 and the *daisy chain terms* of Brady [Bra22]. Next, we give a hardness proof for certain regimes of approximate linearly ordered colouring, using the algebraic machinery of [BBKO21]. This proof also uses an auxiliary notion that may be of independent interest: the edge co-colouring of a graph. In an *edge co-colouring*, the edges are coloured so that disjoint edges are coloured differently (as opposed to *adjacent* edges in edge colouring). Unfortunately, these results only cover r-uniform hypergraphs for $r \geq 5$, thus not affecting Conjecture 1.2. (Outside of the results presented in this thesis, exciting new progress has been achieved [FNOTW24; NVWŽ25; KV25].)

[4] In general we will allow vertices to appear multiple times within edges — i.e. the edges of H are *multisets*. The cardinality of a multiset is given by the total number of elements, taking into account repeated elements. This choice will be discussed more in Chapter 2.

[5] We remark that this problem has since been shown to be tractable even for $f(n) = (0.999)\log_2(n)$, using advanced ideas from additive combinatorics [BNOŽ25].

Barto, Batistelli and Berg [BBB21] first considered adding and removing *rainbow tuples* from ternary relational structures. In their classification, they consider ternary symmetric relations, and divide their tuples into *rainbow tuples* (i.e. tuples (x, y, z) where $|\{x, y, z\}| = 3$), and *non-rainbow tuples* (i.e. tuples (x, y, z) where $|\{x, y, z\}| \leq 2$). In particular, they classify $\mathrm{PCSP}(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ where $\mathbf{B}$ contains one ternary symmetric relation on a ternary domain without rainbow tuples (i.e. it is *rainbow-free*), and classify all the cases with rainbow tuples except for LO 3-colouring. In Chapter 5, we complete the former classification to *all* $\mathbf{B}$ with one ternary, symmetric, rainbow-free relation. Furthermore, we also classify $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ where both $\mathbf{A}, \mathbf{B}$ have one ternary, symmetric, rainbow-free relation, so long as both $\mathbf{A}$ and $\mathbf{B}$ are (in an appropriate sense) *bipartite*.

In particular, these classifications imply that approximate *rainbow-free* LO colouring (i.e. the same problem as approximate LO colouring, but disallowing rainbow tuples) is NP-hard for any fixed number of colours. This case is, in some sense, the central example of such a PCSP — all the hardness proofs for $\mathrm{PCSP}(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ in this chapter can be seen as a generalisation of it. Its hardness would also be implied by the hardness of approximate LO colouring for constant numbers of colours. Thus, the results in this chapter can be seen as partial steps towards resolving Conjecture 1.2. They crucially depend on the *order properties* of LO colouring, exploiting an inductive argument to "make progress" in this order.

We next turn to what we call *functional* PCSPs, in Chapter 6. In a functional relation, if we are told all but the last element in any tuple, the last element is uniquely determined. For symmetric relations, an equivalent definition is that no two tuples differ in exactly one element. We are able to classify the cases corresponding to ternary and quaternary symmetric and functional relational structures, as well as the Boolean symmetric and functional case, among others. The behaviour of these PCSPs has a distinct flavour compared to the previous results. Whereas those exploited ordering, these exploit a *monoidal structure* implicit in the problem. While this structure always exists in our cases, when the problem is tractable the structure happens to also be an Abelian group in a useful way — the tractability then follows by solving systems of linear equations over that group!

Finally, in Chapter 7, we show that two classical algorithms for PCSPs, namely the *affine integer programming relaxation (AIP)* [BG21] and the *combined basic linear and affine integer programming relaxation (BLP+AIP)* [BGWŽ20], turn out to be equivalent for a large class of computational problems (in particular, for PCSPs with one symmetric relation). We interpret this as evidence for the following conjecture, which states that AIP is a universal algorithm for a certain class of PCSPs. (In the following conjecture, $\mathbf{A} \to \mathbf{B}$ should be read "$\mathbf{A}$ maps homomorphically to $\mathbf{B}$"; this notation will be defined formally in Chapter 2.)

Conjecture 1.3 ([NŽ24c]). For every structure $\mathbf{B}$ with $\mathbf{1\text{-}in\text{-}3} \to \mathbf{B}$, $\mathrm{PCSP}(\mathbf{1\text{-}in\text{-}3}, \mathbf{B})$ is either NP-hard or solvable in polynomial time by AIP.

Let us conclude our introductory discussion on PCSPs by taking stock of these results. I believe that the tractability result makes good progress on understanding the usefulness of promises with linear structure such as LO 2-colourability. Compared to 3-colourability of graphs, this promise seems to be more algorithmically exploitable — I thus hope that this result will open up exciting new areas of algorithms for PCSPs. (Indeed, see e.g. [BNOŽ25].) As for the hardness results, I believe that the hardness results for rainbow-free structures

and functional structures in particular highlight two orthogonal combinatorial strategies that are useful for proving hardness in fragments of PCSPs, i.e. inductive order-based strategies, and monoidal/group-theoretic strategies. These particular results show these strategies in their "elemental" form — future work on the hardness of PCSPs will likely require combining these strategies alongside others in highly nontrivial ways. I believe that these pure examples will be useful to that end.

1.2. Maximum PCSPs

So far, we have dealt mainly with the *qualitative* approximation of CSPs. In other words, we are trying to solve a *qualitatively* easier version of a CSP. This is, of course, not the only type of approximation: one can also consider *quantitative* approximation, in which we try to solve as many constraints as possible.

This problem is quite well-understood. For example, *valued CSPs*, i.e. CSPs where each constraint is replaced by a *payoff function*, and the goal is to maximise the payoff, are fully classified [KKR17; KO15; TŽ16]. The gist of this line of work is that, after finding a sufficiently "feasible" sub-instance by using a generic (tractable) CSP solver, the optimal assignment can be found using linear programming. Similarly, we can consider *maxCSPs*, i.e. CSPs where we are tasked with satisfying as many constraints as possible, but without being guaranteed that the input instance is truly solvable. In this case, it was proved [Rag08; RS09] that (assuming the *Unique Games Conjecture (UGC)* of Khot [Kho02]) *semidefinite programming (SDP)* is the best possible approximation algorithm.

Observe that all of the classifications for valued and maximum CSPs were completed years before the classifications for crisp CSPs. At a methodological level, this indicates that, if we intend to classify PCSPs, a *maximisation* version may be a good step towards that classification. (Of course, we should carefully choose which such problems we consider, so that their complexity tells us something about the complexity of PCSPs.) In general, our goals for this class of problems are the following.

- It should have a natural complexity relation to PCSPs.
- Fully classifying it should *not* require fully classifying PCSPs.
- Fully classifying it should *not* require understanding the exact approximability ratio of maxCSPs. (This is because this ratio is only understood in the sense that it can be computed within ε in *double exponential time* in $1/\varepsilon$ [Rag08; RS09]. Heuristically, this means that this approximation ratio of maxCSPs is not well-enough understood for a discrete version of it to be easily classifiable.)
- The classification should not immediately follow from the results of [Rag08; RS09]. (Or else the problem is, in some sense, already solved.)

Thus, we will define maxPCSP($\mathbf{A}, \mathbf{B}$) as follows. Given an instance of CSP($\mathbf{A}$)[6] which admits an assignment that satisfies ρ of the constraints, we want to find an assignment to this

[6] We allow constraints to appear more than once. In particular, when discussing (directed) graphs that are input instances to maxPCSPs, we implicitly allow multiple edges.

instance *interpreted as an instance of* CSP(**B**)[7] which satisfies at least ρ of the constraints. Let us now check that this satisfies all of the conditions we have enumerated above.

- maxPCSP(**A**, **B**) includes every instance of (the search version of) PCSP(**A**, **B**). Hence it is at least as difficult as PCSP(**A**, **B**) — appropriate structural classifications for the complexity of maxPCSP(**A**, **B**) can hence serve as a guide for classifying PCSP(**A**, **B**).
- Since we do not allow "crisp" constraints (i.e. constraints that must be solved in any feasible assignment), maxPCSPs as we have defined them cannot encode PCSPs.
- Since we are *not* allowing arbitrary payoff functions, maxPCSPs in our sense cannot encode approximating maxCSPs with some particular approximation ratio.
- The algorithmic side of the classification *does not* follow from [Rag08; RS09], even if one generalises these results to the promise setting. This is because we solve the *search version*, and require outputting an assignment with the same value as the promised one. While in principle generalising [Rag08; RS09] to the promise setting may prove to us that SDP solves the *decision* version of any finite domain maxPCSP whenever it is not UGC-hard, the techniques of [Rag08; RS09] seem to necessitate some loss in value when explicitly finding an assignment. Furthermore, even if it was possible to generalise these techniques, deciding which templates they apply to would not be immediate, since again the optimal approximation ratio in [Rag08; RS09] can only be found approximately with a double exponential algorithm.[8]

With this problem defined, let us now discuss results. One of the first cases one can consider is the Boolean case, i.e. when the domain of the problem contains two elements. As a toy example, we can consider the case where our problem admits only *one, binary, constraint relation*. While this case is extremely restricted, it still shows some interesting behaviour. In particular, there are (essentially) three problems: max-cut, max-dicut, and max-dicut vs. max-cut. Recall that a *cut* in a graph is a way of splitting the vertices of the graph into two parts so that as many edges as possible go from one side to the other. A *dicut* is the directed analogue of this object: it is a way of splitting the vertices of a *directed* graph into two parts so that as many edges as possible go from *one particular side to the other side* (in the right direction). It is well-known that finding the largest cut or dicut is NP-hard [Hås01; TSSW00] — indeed it is even NP-hard to do this approximately better than some constant factor.

In contrast to this, the remaining promise problem (max-dicut vs. max-cut, in which we are given a directed graph G that admits a ρ-edge dicut, and must find a ρ-edge cut ignoring directions in G) can be solved in polynomial time, using linear programming! This result, together with a generalisation to higher arities, will be described within Chapter 8. The algorithm is based on a combination of local reasoning with linear programming. Furthermore, by rounding this relaxation in a different way, we can solve the following seemingly unrelated problem: given a directed graph with a ρ-edge dicut, find a ρ-edge acyclic (not necessarily induced) subgraph. (This subgraph will surely exist, since the edges that form the dicut give rise to such an acyclic subgraph.)

[7] We assume that **A** and **B** are chosen such that PCSP(**A**, **B**) is a valid problem — then this reinterpretation makes sense.

[8] We find it interesting that this is another example of a search/decision phenomenon in approximation, where the search and the decision are not obviously equivalent.

With this warm up out of the way, in Chapter 9 we move on to a general classification of all maximum PCSPs on graphs. We first explicitly define the problem maxPCSP(G, H) for the case where G, H are graphs. In this problem, we are given a graph X. Suppose that X admits a ρ-edge subgraph that maps homomorphically to G. Then, the goal is to find a ρ-edge subgraph that maps homomorphically to H.[9] Observe that maxPCSP(G, H) is at least as hard as PCSP(G, H). Hence, the following conjecture, widely believed by the community, is relevant.

Conjecture 1.4 ([BG21]). If $G \to H$ are undirected, loopless, nonempty graphs, then PCSP(G, H) is tractable if and only if G is bipartite; otherwise it is NP-hard.

Hence, we would only expect maxPCSP(G, H) to be tractable when G is bipartite (ignoring trivial cases like graphs with loops, or without edges). We will show that this is the case (assuming the Unique Games Conjecture). Furthermore, we will show that even G being bipartite is not sufficient — indeed, we show the following.

Theorem 1.5. Let $G \to H$ be two loopless, nonempty graphs. Then maxPCSP(G, H) can be solved in polynomial time if G is bipartite and H contains a triangle; otherwise it is NP-hard assuming the Unique Games Conjecture.

We continue with a discussion on the *maximum bipartite vs. triangle-free subgraph* problem. Consider first the max-cut problem. It is known [KKMO07] that this problem cannot be approximated better than ≈ 0.878 under the Unique Games Conjecture, and that this approximation ratio is achieved by the Goemans-Williamson algorithm [GW95]. Observe that max-cut is equivalent to finding a maximum bipartite subgraph of a given input graph. It is reasonable to ask what happens if we relax the output requirement of such a problem. For example, suppose we are given a graph G that admits a ρ-edge bipartite subgraph. What if we are merely required to output a large *triangle-free* subgraph, rather than a truly bipartite one? It turns out that this problem is easier than max-cut, in the sense that it is possible to approximate with approximation ratio 0.8823.

Let us conclude with an evaluation of the significance of these results. I believe that maximum PCSPs give rise to a fascinating blend of quantitative and qualitative approximation. On the tractability side, they are an interesting laboratory for better understanding how to round linear programs and semidefinite programs. (As opposed to maxCSPs, where the optimal approximation ratio is often a "random" constant, and therefore the exact optimality of a rounding scheme can be hard to show, here we are mostly interested in exact solvability, and we can therefore get crisp results. One can compare, for example, the result on max-dicut vs. max-cut with the current state of the art for max-dicut. For max-dicut, the optimal approximation ratio is known up to some small error by using extensive numerical methods [BHPZ23]. For max-dicut vs. max-cut, our algorithm is a simple rounding of a linear program.) On the hardness side, they serve as a good "trial run" for hardness proofs for PCSPs. Indeed, hardness proofs for PCSPs are usually based on

[9] Note that this is completely equivalent to the definition given before that revolved around satisfying as many constraints as possible — all that we must observe is that, when interpreting a graph G as a CSP template, some graph X is a satisfiable instance of CSP(G) if and only if X maps homomorphically to G.

"decoding" a polymorphism to a set of its coordinates of bounded size, in a way consistent (in an appropriate sense) with arbitrarily permuting *and merging* its coordinates. In max-PCSPs, in some sense we only need to do this up to permuting coordinates, *not merging*. To conclude, I believe that the dichotomy result for graphs also shows that interesting, crisp classification results are possible in this framework, and I hope that more work is done on it.

2. Preliminaries

2.1. Miscellaneous notations

We let $[n] = \{1, \dots, n\}$ for any $n \in \mathbb{N}$.

We use the *Iverson bracket*, i.e. for any statement ϕ, we let $[\phi] = 1$ if ϕ is true, and $[\phi] = 0$ if ϕ is false. For example, $[2 \text{ is even}] = 1$ and $[3 \text{ is even}] = 0$.

We use standard asymptotic notation, namely $O, o, \Omega, \omega, \Theta$. Furthermore, we use $\tilde{O}, \tilde{o}, \tilde{\Omega}, \tilde{\omega}, \tilde{\Theta}$ to denote these same notions, but up to a poly-logarithmic factor.

We use common notations for set theory (e.g. $\cup$ for union etc.). The only nonstandard set-theoretic notations we use are $A+x$ for $A \cup \{x\}$ and $A-x$ for $A \setminus \{x\}$. Furthermore, when applying a function to an explicit set (e.g. apply f to $\{1, 2, 3\}$), we will elide the parentheses for f (e.g. $f\{1, 2, 3\}$). Also, we write $\binom{A}{k}$ to denote the set of subsets of A of size k, and

$$\binom{A}{\leq n} = \bigcup_{k \leq n} \binom{A}{k}.$$

We define the *Hamming weight* of a 0–1 vector as being the number of nonzero elements contained within it. The *Hamming distance* between two tuples $(x_1, \dots, x_n)$ and $(y_1, \dots, y_n)$ is the number of indices $i \in [n]$ for which $x_i \neq y_i$.

We will also use *Markov's inequality*, which we now restate.

Theorem 2.1 (Markov's inequality). Suppose X is a random variable that takes real, non-negative values. Then, for any $a > 0$,

$$\Pr[X \geq a] \leq \frac{\mathbb{E}[X]}{a}.$$

2.2. Graphs, hypergraphs and relational structures

2.2.1. Basic definitions

A *signature* σ is a set of *relation symbols* $\{R_1, R_2, \dots\}$, where each relation symbol R_i is equipped with an *arity* $\mathrm{ar}(R_i) \in \mathbb{N}$. For a given signature $\sigma = \{R_1, \dots, R_k\}$, a *relational structure* is a tuple $\mathbf{A} = (A; R_1^{\mathbf{A}}, \dots, R_k^{\mathbf{A}})$, where the set A is the *domain*, *vertex set* or *universe* of the relational structure, and $R_1^{\mathbf{A}} \in A^{\mathrm{ar}(R_1)}, \dots, R_k^{\mathbf{A}} \in A^{\mathrm{ar}(R_k)}$ are the relations. Two relational structures with the same signature are called *similar*. We say that a relational

structure is *finite* if its domain is finite, and it has finite signature — we will generally assume all relational structures to be finite.

We call a relation *symmetric* if it is stable under arbitrarily permuting the elements in its tuples. We call a relational structure *empty* if all its relations are empty.

If the relational structure has exactly one symmetric r-ary relation, then we call it an *r-uniform hypergraph*. (In particular, this means that we explicitly allow the hyperedges of our hypergraphs to contain *repeated vertices*. This is analogous to allowing loops in graphs. While this choice is non-standard in combinatorics, it has appeared before in the PCSP literature, e.g. [BBB21].) In particular, if it has one binary symmetric relation, we call it a *graph*. If it has exactly one binary relation that need not be symmetric, we call it a *directed graph* or *digraph*.

In general we denote relational structures with bold uppercase letters e.g. $\mathbf{A}, \mathbf{B}, \ldots$, and the universe of such relational structures by $A, B, \ldots$. For the case of r-uniform hypergraphs, we usually use the letter $\mathbf{H}$, and let V denote the universe and E the edge relation. For the case of graphs and digraphs, due to the strong convention from graph theory, we sometimes use non-bold uppercase letters such as $G, H, \ldots$, and let V denote the vertex set and E the edge relation.

Suppose $\mathbf{A} = (A; R_1^{\mathbf{A}}, \ldots, R_k^{\mathbf{A}})$ is a relational structure, and let $B \subseteq A$. Then $\mathbf{B} = \mathbf{A}|_B$, the *substructure of* $\mathbf{A}$ *induced by* B is the relational structure $(B; R_1^{\mathbf{B}}, \ldots, R_k^{\mathbf{B}})$, similar to $\mathbf{A}$, and where

$$R_i^{\mathbf{B}} = R_i^{\mathbf{A}} \cap B^{\mathrm{ar}(R_i)}.$$

Furthermore, for $S \subseteq A$, we define $\mathbf{A} - S = \mathbf{A}|_{A \setminus S}$.

Consider the case of r-uniform hypergraphs, including graphs. Let $\mathbf{H} = (V, E)$ be an r-uniform hypergraph. The *open neighbourhood* $N_{\mathbf{H}}(v)$ of a vertex v is given by the set of vertices $w \neq v$ which appear in an edge together with v. The *closed neighbourhood* $N_{\mathbf{H}}[v]$ is defined by $N_{\mathbf{H}}[v] = N_{\mathbf{H}}(v) + v$. The *degree* of v is given by $|N_{\mathbf{H}}(v)|$.

Recall the notation K_n for the *n-vertex clique*, i.e. $K_n = ([n], \{(i, j) \in [n]^2 \mid i \neq j\})$, and the notation C_n for the *n-vertex cycle*, i.e. $C_n = ([n], \{(1,2), (2,1), (2,3), (3,2), \ldots, (n-1, n), (n, n-1), (n, 1), (1, n)\})$. Also, recall that $\mathbf{1\text{-}in\text{-}3} = (\{0,1\}, \{(1,0,0), (0,1,0), (0,0,1)\})$ and $\mathbf{NAE} = (\{0,1\}, \{0,1\}^3 \setminus \{(0,0,0), (1,1,1)\})$.

2.2.2. Homomorphisms, sums, products

For all the definitions in this subsection, assume that all relational structures have some fixed signature $\sigma = \{R_1, \ldots, R_k\}$. In other words, all relational structures mentioned are similar. Let r_i denote $\mathrm{ar}(R_i)$ for $i \in [k]$.

Consider two relational structures $\mathbf{A} = (A; R_1^{\mathbf{A}}, \ldots, R_k^{\mathbf{A}})$ and $\mathbf{B} = (B; R_1^{\mathbf{B}}, \ldots, R_k^{\mathbf{B}})$. A *homomorphism from* $\mathbf{A}$ *to* $\mathbf{B}$, denoted by $h : \mathbf{A} \to \mathbf{B}$, is a function $h : A \to B$ such that, for any $i \in [k]$ and for any $(a_1, \ldots, a_{r_i}) \in R_i^{\mathbf{A}}$, we have $(h(a_1), \ldots, h(a_{r_i})) \in R_i^{\mathbf{B}}$. We denote the mere existence of a homomorphism by $\mathbf{A} \to \mathbf{B}$. Crucially, homomorphisms compose, i.e. if $h : \mathbf{A} \to \mathbf{B}$ and $g : \mathbf{B} \to \mathbf{C}$, then $g \circ h : \mathbf{A} \to \mathbf{C}$.

For any two relational structures $\mathbf{A}, \mathbf{B}$ as above, we define the (cartesian, direct, tensor, categorical) product $\mathbf{A} \times \mathbf{B} = (A \times B; R_1^{\mathbf{A}\times\mathbf{B}}, \dots, R_k^{\mathbf{A}\times\mathbf{B}})$, where

$$R_i^{\mathbf{A}\times\mathbf{B}} = \{((a_1, b_1), \dots, (a_{r_i}, b_{r_i})) \mid (a_1, \dots, a_{r_i}) \in R_i^{\mathbf{A}}, (b_1, \dots, b_{r_i}) \in R_i^{\mathbf{B}}\}.$$

Note that $\mathbf{X} \to (\mathbf{A} \times \mathbf{B})$ if and only if $\mathbf{X} \to \mathbf{A}$ and $\mathbf{X} \to \mathbf{B}$. We write $\mathbf{A}^n$ for the n-fold product of $\mathbf{A}$ with itself.

For any two relational structures $\mathbf{A}, \mathbf{B}$ as above, where A and B are disjoint, we define $\mathbf{A} + \mathbf{B} = (A \cup B; R_1^{\mathbf{A}+\mathbf{B}}, \dots, R_k^{\mathbf{A}+\mathbf{B}})$ by $R_i^{\mathbf{A}+\mathbf{B}} = R_i^{\mathbf{A}} \cup R_i^{\mathbf{B}}$ for $i \in [k]$. Note that $(\mathbf{A} + \mathbf{B}) \to \mathbf{X}$ if and only if $\mathbf{A} \to \mathbf{X}$ and $\mathbf{B} \to \mathbf{X}$. Furthermore, we say that $\mathbf{X}$ is *connected* if $\mathbf{X} = \mathbf{A}+\mathbf{B}$ implies that either $\mathbf{A}$ or $\mathbf{B}$ has empty domain. Hence every relational structure $\mathbf{X}$ can be written as a sum of connected relational structures, which we call its *connected components*.

2.3. Minions and PCSPs

In this section we will recount key elements of the *algebraic approach to PCSPs*, following [BBKO21]. First, let us define PCSPs.

Definition 2.2. Consider two similar relational structures $\mathbf{A} \to \mathbf{B}$. The problem $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ has two versions: a *search version* and a *decision version*. In both, the input is a relational structure $\mathbf{X}$, similar to $\mathbf{A}$ and $\mathbf{B}$. In the search version, we are promised $\mathbf{X} \to \mathbf{A}$, and must find a homomorphism $h : \mathbf{X} \to \mathbf{B}$. In the decision version, we must answer YES if $\mathbf{X} \to \mathbf{A}$, and NO if $\mathbf{X} \not\to \mathbf{B}$.
We define $\mathrm{CSP}(\mathbf{A}) = \mathrm{PCSP}(\mathbf{A}, \mathbf{A})$. Furthermore, we call a pair $(\mathbf{A}, \mathbf{B})$ with $\mathbf{A} \to \mathbf{B}$ a *PCSP template* or simply a *template*.

Remark 2.3. Equivalently, in $\mathrm{CSP}(\mathbf{A})$ we may see the relations within $\mathbf{A}$ as a *constraint language*, i.e. a set of permissible constraint relations. (In this encoding, the constraint corresponding to a constraint relation is satisfied by the tuples contained within that relation.) Then, in $\mathrm{CSP}(\mathbf{A})$, an instance consists of a set of variables and a set of constraints from the constraint language, applied to the variables. The goal is to find an assignment of values from the domain of $\mathbf{A}$ to the variables so that all constraints are satisfied. In this interpretation, the domain of $\mathbf{X}$ is the set of variables, and the relations of $\mathbf{X}$ encode the tuples of variables to which the constraints are applied.
Similarly, in $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$, we see the relations within $(\mathbf{A}, \mathbf{B})$ as permissible constraint relations — only now, each constraint has a strong version (i.e. the version in $\mathbf{A}$), and a weak version (i.e. the version in $\mathbf{B}$). The fact that $\mathbf{A} \to \mathbf{B}$ enforces that a solution to the strong version of the constraints can be transformed entry-by-entry into a solution to the weak version of the constraints. The goal is now to find a solution that satisfies all the weak versions of constraints, under the assumption that there exists a solution that satisfies all the strong versions of the constraints.
Thus, when constructing instances of $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ with $\mathbf{A} = (A; R_1^{\mathbf{A}}, \dots, R_t^{\mathbf{A}})$ and $\mathbf{B} = (B; R_1^{\mathbf{B}}, \dots, R_t^{\mathbf{B}})$, we will speak of imposing constraints $R_1, \dots, R_t$ on a set of variables. A solution to such an instance *in the language of* $\mathbf{A}$ *or* $\mathbf{B}$ *respectively* is an assignment of values from A or B respectively to the variables such that all the constraints are satisfied, if R_i is interpreted as $R_i^{\mathbf{A}}$ or $R_i^{\mathbf{B}}$ respectively.

The decision version reduces to the search version. To see why, suppose we have an algorithm $\mathcal{A}$ for the search version of PCSP($\mathbf{A}, \mathbf{B}$), which runs in time $T(n)$ for an input instance of size n.[1] Suppose we are given an instance $\mathbf{X}$ of the decision version of the problem, of size n. Then, try to run $\mathcal{A}$ on $\mathbf{X}$, stopping the execution if it takes more than $T(n)$ time. We output YES if $\mathcal{A}$ returns a homomorphism from $\mathbf{X}$ to $\mathbf{B}$, and NO if it does not, or if the run of $\mathcal{A}$ is stopped. To see why this is correct, first suppose $\mathbf{X}$ is a YES instance. Thus $\mathbf{X} \to \mathbf{A}$, and $\mathcal{A}$ finds a homomorphism from $\mathbf{X}$ to $\mathbf{B}$ within $T(n)$ time. Thus we correctly output YES in this case. Conversely, suppose we output YES. In this case, our algorithm has found a homomorphism $\mathbf{X} \to \mathbf{B}$, and thus it is safe to output YES.

In general however, the exact relationship between the two problems is not known. Throughout, all algorithmic results will hold for the search version, and all hardness results for the decision version.

Let us first quickly go through some elementary reductions that occur between PCSPs.

Lemma 2.4 (Homomorphic Relaxation). Suppose $\mathbf{A} \to \mathbf{A}' \to \mathbf{B}' \to \mathbf{B}$. Then PCSP($\mathbf{A}, \mathbf{B}$) reduces to PCSP($\mathbf{A}', \mathbf{B}'$).

Proof. Observe that (in the decision version) the YES and NO instances respectively of PCSP($\mathbf{A}, \mathbf{B}$) are YES and NO instances of PCSP($\mathbf{A}', \mathbf{B}'$), hence an algorithm for PCSP($\mathbf{A}', \mathbf{B}'$) also solves PCSP($\mathbf{A}, \mathbf{B}$) without modifying the input instance. This is also the case in the search version, though in that reduction the output of the algorithm must be "piped through" the homomorphism from $\mathbf{B}'$ to $\mathbf{B}$. □

Corollary 2.5 (Homomorphic Sandwich). If $\mathbf{A} \to \mathbf{C} \to \mathbf{B}$ and CSP($\mathbf{C}$) is tractable, then PCSP($\mathbf{A}, \mathbf{B}$) is also tractable.

Next, we define a useful notion of gadget reduction, namely *primitive-positive (pp) definitions*.

Definition 2.6. Consider some signature $\sigma = \{R_1, \dots, R_k\}$. Let r_i denote $\mathrm{ar}(R_i)$ for $i \in [k]$. A primitive-positive definition is a formula $\phi(x_1, \dots, x_r)$ of the form

$$\exists y_1 \cdots \exists y_s \cdot \left(\bigwedge_{j \in [t]} R_{i_j}(z_1^{(j)}, \dots, z_{r_{i_j}}^{(j)}) \right) \wedge \left(\bigwedge_{j \in [u]} p^{(j)} = q^{(j)} \right)$$

where $z_1^{(j)}, \dots, z_{r_{i_j}}^{(j)}$ and $p^{(j)}, q^{(j)}$ are taken from among $x_1, \dots, x_r, y_1, \dots, y_s$. Consider some relational structure $\mathbf{A} = (A; R_1^{\mathbf{A}}, \dots, R_k^{\mathbf{A}})$ with signature σ. Then, ϕ interpreted in $\mathbf{A}$ is a relation S of arity r, which admits a tuple $(x_1, \dots, x_r) \in A^r$ if and only if there exist $y_1, \dots, y_s \in A$ such that $(z_1^{(j)}, \dots, z_{r_{i_j}}^{(j)}) \in R_{i_j}^{\mathbf{A}}$ for all $j \in [t]$, and also $p^{(j)} = q^{(j)}$ for all $j \in [u]$.
By abuse of language, we will also say that ϕ interpreted in $\mathbf{A}$ is the relational structure $\mathbf{A}' = (A; R^{\mathbf{A}'})$, where R is a relation symbol with arity r, and $R^{\mathbf{A}'} = S$.

[1] Formally, there are some restrictions on the function T, namely that it be time constructible. Since we will only apply this reduction in the context of polynomial-time solvability, this is a mild restriction.

The key fact about pp-definitions is that they can be efficiently implemented as a reduction between PCSPs. Hence, suppose there exists some pp-formula ϕ such that ϕ interpreted in $\mathbf{A}$ is $\mathbf{A}'$ and ϕ interpreted in $\mathbf{B}$ is $\mathbf{B}'$. Then $\mathrm{PCSP}(\mathbf{A}', \mathbf{B}')$ reduces to $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$. The reduction is simple: replace every occurrence of the relation symbol R in the instance of $\mathrm{PCSP}(\mathbf{A}', \mathbf{B}')$ by ϕ. To deal with the equality constraints (which are not in the signature of $\mathbf{A}, \mathbf{B}$), repeatedly identify any pair of variables set equal in the instance, until no equality constraints remain.[2]

The algebraic approach to PCSPs is based on the fact that certain *algebraic invariants* determine the complexity of a PCSP. In particular, the key algebraic invariant is the *polymorphism*. Polymorphisms form a mathematical structure known as a *minion*, which we now define.

Definition 2.7 (Minion). A *minion* is a set $\mathcal{M}$, where every element $f \in \mathcal{M}$ has an arity $\mathrm{ar}(f) \in \mathbb{N}$. We let $\mathcal{M}^{(n)} = \{f \in \mathcal{M} \mid \mathrm{ar}(f) = n\}$. On this set, for each $\pi : [n] \to [m]$, there is a corresponding operation $(-^\pi) : \mathcal{M}^{(n)} \to \mathcal{M}^{(m)}$, called a *minor map*, such that the following are true:

1. If $\pi : [n] \to [m]$ and $\sigma : [m] \to [k]$, then for any $f \in \mathcal{M}^{(n)}$ we have $f^{\sigma\circ\pi} = (f^\pi)^\sigma$.
2. Letting $\mathrm{id} : [n] \to [n]$ denote the identity function, for any $f \in \mathcal{M}^{(n)}$ we have $f^{\mathrm{id}} = f$.

We will often denote the relation $f^\pi = g$ by $f \xrightarrow{\pi} g$.

Now, we define maps between minions.

Definition 2.8 (Minion homomorphisms). A *minion homomorphism* $\xi : \mathcal{M} \to \mathcal{N}$ is a map from $\mathcal{M}$ to $\mathcal{N}$ that respects arities and minor maps. In other words, for any $f \in \mathcal{M}$ we have $\mathrm{ar}(\xi(f)) = \mathrm{ar}(f)$, and for every $\pi : [n] \to [m]$ and $f \in \mathcal{M}^{(n)}$ we have

$$\xi(f)^\pi = \xi(f^\pi).$$

In particular, observe that if $\mathcal{M} \subseteq \mathcal{N}$ then $\mathcal{M} \to \mathcal{N}$ — the homomorphism is given by the inclusion of $\mathcal{M}$ into $\mathcal{N}$.

Now, let us define polymorphisms.

Definition 2.9 (Polymorphisms). Suppose $\mathbf{A}, \mathbf{B}$ are relational structures with signature $\{R_1, \ldots, R_k\}$, where r_i is the arity of R_i for $i \in [k]$. A *polymorphism* from $\mathbf{A} = (A; R_1^{\mathbf{A}}, \ldots, R_k^{\mathbf{A}})$ to $\mathbf{B} = (B; R_1^{\mathbf{B}}, \ldots, R_k^{\mathbf{B}})$ is a homomorphism $f : \mathbf{A}^n \to \mathbf{B}$. In more detail, a polymorphism is a function $f : A^n \to B$ such that the following holds: For every $i \in [k]$, and for every matrix $M \in A^{r_i \times n}$ with r_i rows and n columns, such that every column of M belongs to $R_i^{\mathbf{A}}$, if we apply f to the rows of M, then the resulting tuple is an element of $R_i^{\mathbf{B}}$. For any polymorphism $f : \mathbf{A}^n \to \mathbf{B}$, we let $\mathrm{ar}(f) = n$ denote the arity of f. We let $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ denote the set of polymorphisms from $\mathbf{A}$ to $\mathbf{B}$.

[2] This entire discussion can be generalised to the case where $(\mathbf{A}', \mathbf{B}')$ have signatures with more than one relation (cf. [BBKO21]), however for our purposes this is sufficient.

Example 2.10. Let us consider what conditions a 4-ary polymorphism f from **1-in-3** to **NAE** must satisfy. Consider any matrix with 4 columns and 3 rows, where every column is a tuple of **1-in-3**, e.g.

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Then, applying f to every row of this matrix must yield a tuple of **NAE**. In other words, $f(1,1,0,0)$, $f(0,0,0,1)$ and $f(0,0,1,0)$ are not all equal.

Note that if f is a polymorphism, then any function made from reordering the inputs of f, merging them, or introducing dummy inputs, remains a polymorphism. In fact, for every n-ary polymorphism f and every function $\pi : [n] \to [m]$, the function

$$(x_1, \dots, x_m) \mapsto f(x_{\pi(1)}, \dots, x_{\pi(n)})$$

is an m-ary polymorphism. We denote this new polymorphism by f^π. Hence, we see that the set of polymorphisms from $\mathbf{A}$ to $\mathbf{B}$ in fact form a minion.

The useful thing about polymorphisms is that their minion structure determines the complexity of the corresponding PCSP. In more detail,

Theorem 2.11 ([BBKO21, Theorem 3.1]). If $\mathrm{Pol}(\mathbf{A}, \mathbf{B}) \to \mathrm{Pol}(\mathbf{A}', \mathbf{B}')$, then $\mathrm{PCSP}(\mathbf{A}', \mathbf{B}')$ reduces to $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ in logarithmic space.

Unfortunately it is not always easy to decide whether there exists a minion homomorphism between two minions. Indeed, as formulated above such a homomorphism contains an infinite amount of information, and hence its existence (even between polymorphism minions of finite relational structures) is not a priori decidable. Luckily, there exists a relational structure, called the *free structure*, that helps us decide this.

Definition 2.12 (Free structure). Let $\mathcal{M}$ be a minion and $\mathbf{A} = (A; R_1^{\mathbf{A}}, \dots, R_k^{\mathbf{A}})$ be a relational structure with signature $\{R_1, \dots, R_k\}$. Let r_i denote $\mathrm{ar}(R_i)$ for $i \in [k]$. Without loss of generality, suppose $A = [a]$. Then $\mathbf{F}_{\mathcal{M}}(\mathbf{A})$, the *free structure of* $\mathcal{M}$ *generated by* $\mathbf{A}$, is a relational structure similar to $\mathbf{A}$, defined as follows. The domain of $\mathbf{F}_{\mathcal{M}}(\mathbf{A})$ is $\mathcal{M}^{(a)}$, and its relations are $R_1^{\mathbf{F}_{\mathcal{M}}(\mathbf{A})}, \dots, R_k^{\mathbf{F}_{\mathcal{M}}(\mathbf{A})}$, defined as follows. To construct the relation $R_i^{\mathbf{F}_{\mathcal{M}}(\mathbf{A})}$, take the relation $R_i^{\mathbf{A}}$ and arrange its tuples into the columns of a matrix $M \in A^{r_i \times |R_i^{\mathbf{A}}|}$ with r_i rows and $|R_i^{\mathbf{A}}|$ columns. Then, let $\pi_1, \dots, \pi_{r_i} : |R_i^{\mathbf{A}}| \to [a]$ be the functions reading off the rows of M; in other words, $\pi_x(y) = M_{xy}$. Now, $R_i^{\mathbf{F}_{\mathcal{M}}(\mathbf{A})}$ admits a tuple $(f_1, \dots, f_{r_i})$ if and only if there exists $g \in \mathcal{M}^{(|R_i^{\mathbf{A}}|)}$ such that $f_1 = g^{\pi_1}, \dots, f_{r_i} = g^{\pi_{r_i}}$.

Example 2.13. For any minion $\mathcal{M}$, the free structure $\mathbf{F}_{\mathcal{M}}(\textbf{1-in-3})$ is a 3-uniform hypergraph, whose vertex set is $\mathcal{M}^{(2)}$, and where $(f_1, f_2, f_3) \in \left(\mathcal{M}^{(2)}\right)^3$ is an edge if and only if there exists $g \in \mathcal{M}^{(3)}$ such that

$$\begin{aligned} f_1(x, y) &= g(y, x, x) \\ f_2(x, y) &= g(x, y, x) \\ f_3(x, y) &= g(x, x, y). \end{aligned}$$

It is not a coincidence that every column of the matrix formed by the inputs of g has exactly one y and two xs, and indeed every such tuple appears as a column.

Example 2.14. For any minion $\mathcal{M}$, the free structure $\mathbf{F}_{\mathcal{M}}(\textbf{NAE})$ is a 3-uniform hypergraph, whose vertex set is $\mathcal{M}^{(2)}$, and where $(f_1, f_2, f_3) \in \left(\mathcal{M}^{(2)}\right)^3$ is an edge if and only if there exists $g \in \mathcal{M}^{(6)}$ such that

$$\begin{aligned} f_1(x, y) &= g(y, x, x, x, y, y) \\ f_2(x, y) &= g(x, y, x, y, x, y) \\ f_3(x, y) &= g(x, x, y, y, y, x). \end{aligned}$$

Again, it is not a coincidence that every column of the matrix formed by the inputs of g does not contain only xs or only ys, and that every such tuple appears as a column.

Free structures precisely capture minion homomorphisms for polymorphism minions, as is shown in the following theorem.

Theorem 2.15 ([BBKO21, Lemma 4.4]). For any minion $\mathcal{M}$ and any similar relational structures $\mathbf{A} \to \mathbf{B}$, we have that $\mathbf{F}_{\mathcal{M}}(\mathbf{A}) \to \mathbf{B}$ if and only if $\mathcal{M} \to \mathrm{Pol}(\mathbf{A}, \mathbf{B})$.

Hence, in particular, deciding whether $\mathrm{Pol}(\mathbf{A}, \mathbf{B}) \to \mathrm{Pol}(\mathbf{A}', \mathbf{B}')$ reduces to a finite instance of $\mathrm{CSP}(\mathbf{B}')$.

We will also need the following simple hardness condition, an easy corollary of [BBKO21, Corollary 5.2].

Theorem 2.16 (Easy corollary of [BBKO21, Corollary 5.2]). Suppose $\mathrm{Pol}^{(3)}(\mathbf{A}, \mathbf{B})$ contains only essentially unary, non-constant functions (i.e. functions that depend on exactly one input). Then $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is NP-hard.

2.4. Further algebraic hardness

In this section, we prove certain algebraic hardness conditions for PCSPs. Some of these have been proved before (e.g. [BBKO21, Theorem 5.21]), and in any case these proofs are very standard. However we reprove them because of two factors: one of our proofs will require a slightly altered version of [BBKO21, Theorem 5.21]. Another one of our proofs will require a "multilayered Unique Games Conjecture". Here the reduction from the

nonlayered version to the layered version is identical to the reduction from the nonlayered PCP to the layered PCP, as done in e.g. [DGKR05; BWŽ21] — however, it is possible to do all of these proofs "in one go", and we will do this to simplify the general presentation. Our presentation will be slightly different to [BBKO21; BWŽ21], however the end goal is the same.

We first define what an $(\ell+1)$-player 1-round projective game is — the associated computational problem is often called the *label cover* problem, however we will consistently use the "game" terminology.

Definition 2.17. An $(\ell+1)$-player 1-round projective game is a computational problem given by a set of variables X, partitioned into *layers* $X = X_0 \cup \cdots \cup X_\ell$, a sequence of domains $D_0, \ldots, D_\ell$, and constraints. Each constraint is given by a sequence of variables $x_0 \in X_0, \ldots, x_\ell \in X_\ell$, and a family of functions $\boldsymbol{\pi} = (\pi_{ij})_{0 \le i < j \le \ell}$ where $\pi_{ik} = \pi_{jk} \circ \pi_{ij}$ for $0 \le i < j < k \le \ell$. An *assignment* for this problem is a function c which takes $X_i \to D_i$ for $i \in \{0, \ldots, \ell\}$. A constraint given by $x_0, \ldots, x_\ell$ and $\boldsymbol{\pi} = (\pi_{ij})_{0 \le i < j \le \ell}$ is *strongly satisfied* if $\pi_{ij}(c(x_i)) = c(x_j)$ for all $0 \le i < j \le \ell$. It is *weakly satisfied* if $\pi_{ij}(c(x_i)) = c(x_j)$ for *at least one* pair (i, j) with $0 \le i < j \le \ell$. We say that the *strong/weak* value of c is the proportion of constraints strongly/weakly satisfied by c. The strong/weak value of the game is the maximum strong/weak value of any assignment.
We say that the game is *unique* (and drop the "projective" qualifier) if $D_0 = \cdots = D_\ell = D$, and all functions in all constraints are permutations. We say that the game is $(\ell+1)$*-regular* (e.g. bi-regular, etc.) if there exists a sequence $d_0, \ldots, d_\ell$ so that, for each $i \in \{0, \ldots, \ell\}$, each variable $x \in X_i$ belongs to exactly d_i constraints.

Often we will assume, without loss of generality, that the domains are of form $[N]$ for some natural number N.

Observe that for the case $\ell = 1$, i.e. 2-player 1-round projective games, the strong and weak value of any assignment — and hence of any instance — coincide. Hence, for 2-player 1-round projective (unique) games, we will simply speak of the value of an assignment and/or of an instance.

A key fact is that deciding whether the value of a 2-player 1-round projective game is 1 or not even ε is NP-hard. This follows from the PCP theorem (see [ALMSS98; AS98], or also [Din07]), combined with Raz's parallel repetition theorem [Raz95].

Theorem 2.18 ([ALMSS98; AS98; Raz95]). For any $\varepsilon \in (0, 1)$, there exists some finite D_0, D_1 such that the following problem is NP-hard: Given a 2-player 1-round projective game with domains D_0, D_1, decide if the value is 1, or not even ε. This is even the case if the instance is assumed to be bi-regular [DRS05].

Next, let us recall the Unique Games Conjecture (UGC) of Khot [Kho02].

Conjecture 2.19 ([Kho02]). For any $\varepsilon \in (0, 1)$, there exists some finite D such that the following problem is NP-hard: Given a 2-player 1-round unique game with domain D, decide whether the value is at least $1 - \varepsilon$, or not even ε. This is even the case if the instance is assumed to be bi-regular [Kho10].

We will now show how to reduce finding the value of a 2-player 1-round projective game to finding the value of an $(\ell+1)$-player 1-round projective game. This reduction and most of its analysis is (essentially) identical to a reduction in [BWŽ21], which leverages layered label covers similarly to [DGKR05]. The proof in [BWŽ21] does not deal with cases with imperfect completeness (i.e. value < 1). We will need this case, hence we will also explicitly need to analyse the completeness of the reduction — this is quite easy, and the proof appears in [NŽ25c] (on which Chapter 9 is based).

Theorem 2.20 ([BWŽ21; NŽ25c]). Fix ℓ, and suppose we are given a bi-regular 2-player 1-round projective game with finite domains of size at most N. There exists an algorithm which, in polynomial time, generates an $(\ell+1)$-player 1-round projective game with domains of size at most N^ℓ such that the following holds.

Completeness. If the input game has value at least $1-\varepsilon$, then the output game has strong value at least $(1-\varepsilon)^\ell$.

Soundness. If the output game has weak value at least ε, the input game has value at least $\varepsilon / \binom{\ell+1}{2}$.

Uniqueness. If the input game is unique, then the output game is also unique.

Before moving on to the proof, let us define some simple notation that will be useful: if $f : A \to B$ and $g : C \to D$ then we define $(f \times g) : (A \times C) \to (B \times D)$ by $(f \times g)(a, c) = (f(a), g(c))$. In particular, if f, g are bijective, then $f \times g$ is also bijective, and indeed $(f \times g)^{-1} = f^{-1} \times g^{-1}$. Furthermore, $(f \times g) \circ (f' \times g') = (f \circ f') \times (g \circ g')$.[3] Also, let $\mathrm{id}_A : A \to A$ denote the identity map on A.

Proof. We will first describe the reduction, then prove completeness, soundness, and uniqueness.

Reduction. Suppose the instance is given by variable layers X and Y, domains D_X, D_Y, and constraints E, where a typical constraint is of form (x, y, π), with $x \in X, y \in Y, \pi : D_X \to D_Y$. Furthermore, suppose that d_X, d_Y are such that, for every variable $x \in X, y \in Y$, x is contained in precisely d_X constraints, and y is contained in precisely d_Y constraints.

We will output an instance with variable layers $X_0, \dots, X_\ell$, domains $D_0, \dots, D_\ell$, and constraint set E'. We set

$$X_i = X^{\ell-i} \times Y^i,$$
$$D_i = D_X^{\ell-i} \times D_Y^i.$$

Now, let us construct our constraint set. Consider any ℓ-tuple of constraints

$$(x_1, y_1, \pi_1), \dots, (x_\ell, y_\ell, \pi_\ell) \in E.$$

[3] This is analogous to the "mixed product property" for Kronecker products, i.e. $(A \otimes B)(A' \otimes B') = (AA') \otimes (BB')$.

Each such tuple will lead to one constraint in E', on variables

$$\begin{aligned}(x_1, \dots, x_{\ell-1}, x_\ell) &\in X_0,\\ (x_1, \dots, x_{\ell-1}, y_\ell) &\in X_1,\\ &\vdots\\ (y_1, \dots, y_{\ell-1}, y_\ell) &\in X_\ell,\end{aligned}$$

and where the family of functions $\boldsymbol{\pi} = (\pi_{ij})_{0 \le i < j \le \ell}$ is defined as follows:

$$\pi_{ij} = \underbrace{\mathrm{id}_X \times \cdots \times \mathrm{id}_X}_{\ell - j \text{ times}} \times \pi_{\ell-j+1} \times \cdots \times \pi_{\ell-i} \times \underbrace{\mathrm{id}_Y \times \times \cdots \times \mathrm{id}_Y}_{i \text{ times}}.$$

In words, π_{ij} acts as the identity on all coordinates of its input, except those in the index range $\ell-j+1, \dots, \ell-i$, where it acts as $\pi_{\ell-j+1}, \dots, \pi_{\ell-i}$ respectively. Note that, by definition, $\pi_{ik} = \pi_{jk} \circ \pi_{ij}$ for all $0 \le i < j < k \le \ell$.

Completeness. Suppose that the original instance had an assignment c with value $1-\varepsilon$. We will show that there exists an assignment c' such that a $(1-\varepsilon)^\ell$-fraction of all constraints in the output instance are strongly satisfied. The assignment c' will map the variable

$$(x_1, \dots, x_{\ell-i}, y_{\ell-i+1}, \dots, y_\ell) \in X_i$$

to the value $(c(x_1), \dots, c(x_{\ell-i}), c(y_{\ell-i+1}), \dots, c(y_\ell))$. Now, select a constraint uniformly at random. Such a constraint is determined by an ℓ-tuple of constraints

$$(x_1, y_1, \pi_1), \dots, (x_\ell, y_\ell, \pi_\ell) \in E.$$

Since we selected the constraint uniformly at random, this ℓ-tuple of constraints is also selected uniformly at random. Hence, with probability at least $(1-\varepsilon)^\ell$, all of these constraints are satisfied by c, i.e. $c(y_i) = \pi_i(c(x_i))$ for $i \in [\ell]$. We will show that in this case the outputted constraint is strongly satisfied — this is sufficient for completeness.

Indeed, consider π_{ij}, and observe that

$$\begin{aligned}&\pi_{ij}(c'(x_1, \dots, x_{\ell-i}, y_{\ell-i+1}, \dots, y_\ell))\\ &\qquad = \pi_{ij}(c(x_1), \dots, c(x_{\ell-i}), c(y_{\ell-i+1}), \dots, c(y_\ell))\\ &\quad = (c(x_1), \dots, c(x_{\ell-j}), \pi_{\ell-j+1}(c(x_{\ell-j+1})), \dots, \pi_{\ell-i}(c(x_{\ell-i})), c(y_{\ell-i+1}), \dots, c(y_\ell))\\ &\qquad = (c(x_1), \dots, c(x_{\ell-j}), c(y_{\ell-j+1}), \dots, c(y_\ell))\\ &\qquad = c'(x_1, \dots, x_{\ell-j}, y_{\ell-j+1}, \dots, y_\ell).\end{aligned}$$

Hence c' strongly satisfies this constraint, as required.

Soundness. Suppose that the output instance has weak value at least ε. In other words, there exists an assignment c such that, for an ε-fraction of all constraints, there exist two layers between which c satisfies the constraint. Since there are $\binom{\ell+1}{2}$ pairs of layers, this implies that there exist $0 \le a < b \le \ell$ such that, for an $\varepsilon/\binom{\ell+1}{2}$-fraction of constraints, the following holds. If the constraint is given by variables $(x_0, \dots, x_\ell)$ and functions

$\boldsymbol{\pi} = (\pi_{ij})_{0 \le i < j \le \ell}$, then $c(x_b) = \pi_{ab}(x_a)$. If this is the case for some constraint, we say that it is satisfied between X_a and X_b.

Observe that, for $(z_1, \dots, z_\ell) \in X_0 \cup \dots \cup X_\ell$, we have that $c(z_1, \dots, z_\ell)$ is a tuple of ℓ values in D. Let $c(z_1, \dots, z_\ell) = (c^{(1)}(z_1, \dots, z_\ell), \dots, c^{(\ell)}(z_1, \dots, z_\ell))$. Pick some arbitrary $k \in \{\ell - b + 1, \dots, \ell - a\}$. Split X_a, X_b into equivalence classes, where two variable $(z_1, \dots, z_\ell), (z'_1, \dots, z'_\ell) \in X_a \cup X_b$ are equivalent if and only if $z_i = z'_i$ for $t \in [\ell] - k$. Note that, by the bi-regularity of the original instance, and by construction, for any two equivalence classes, one in X_a and one in X_b, there are either no constraints or some positive, constant number of constraints between them. Hence, since a $\varepsilon / \binom{\ell+1}{2}$-fraction of the constraints are satisfied between X_a and X_b, there exists some equivalence class in X_a and some equivalence class in X_b between which there exists a nonzero number of constraints, and an $\varepsilon / \binom{\ell+1}{2}$-fraction of the constraints are satisfied.

Now, let us consider what this means. These equivalence classes are determined by some constants $z_1, \dots, z_{k-1}, z_{k+1}, \dots, z_\ell, z'_1, \dots, z'_{k-1}, z'_{k+1}, \dots, z'_\ell \in X \cup Y$, and contain variables of form $(z_1, \dots, z_{k-1}, x, z_{k+1}, \dots, z_\ell) \in X_a$ and $(z'_1, \dots, z'_{k-1}, y, z'_{k+1}, \dots, z'_\ell) \in X_b$. Since there are constraints between these two equivalence classes, either $z_i = z'_i$ (if $z_i, z'_i \in X$ or $z_i, z'_i \in Y$), or there exists a constraint between z_i and z'_i in the original instance (if $z_i \in X$ and $z'_i \in Y$). By construction, such variables exist for every $x \in X, y \in Y$ — and furthermore, the constraints between these equivalence classes are in one-to-one relation with the constraints between x and y in the original instance. Furthermore, if we take x and y so that there exists a constraint $(x, y, \pi) \in E$, then

$$\pi_{ab}(c(z_1, \dots, z_{k-1}, x, z_{k+1}, \dots, z_\ell)) = c(z'_1, \dots, z'_{k-1}, y, z'_{k+1}, \dots, z'_\ell)$$

only if

$$\pi(c^{(k)}(z_1, \dots, z_{k-1}, x, z_{k+1}, \dots, z_\ell)) = c^{(k)}(z'_1, \dots, z'_{k-1}, y, z'_{k+1}, \dots, z'_\ell).$$

Hence, since the former happens for an $\varepsilon / \binom{\ell+1}{2}$-fraction of all constraints between the equivalence classes, i.e. an $\varepsilon / \binom{\ell+1}{2}$-fraction of all (x, y) for which there exists a constraint $(x, y, \pi) \in E$, we see that the assignment

$$c'(x) = c^{(k)}(z_1, \dots, z_{k-1}, x, z_{k+1}, \dots, z_\ell)$$
$$c'(y) = c^{(k)}(z'_1, \dots, z'_{k-1}, y, z'_{k+1}, \dots, z'_\ell)$$

satisfies an $\varepsilon / \binom{\ell+1}{2}$-fraction of the constraints of the original instance, as required.

Uniqueness. Note that a generic constraint function π_{ij} in the outputted instance is of form

$$\mathrm{id}_X \times \dots \times \mathrm{id}_X \times \pi_{\ell-j+1} \times \dots \times \pi_{\ell-i} \times \mathrm{id}_Y \times \times \dots \times \mathrm{id}_Y,$$

where $\pi_{\ell-j+1}, \dots, \pi_{\ell-i}$ are constraint functions from the input instance. But, if all the functions from the input instance are bijective, then clearly this output function is also bijective (recall that if f, g are bijective then $f \times g$ is bijective). □

This, together with the fact that $(1 - \varepsilon)^\ell \to 1$ and $\varepsilon / \binom{\ell+1}{2} \to 0$ for $\varepsilon \to 0$, and the fact that we can take domains to be of form $[N]$ without changing the complexity of the problem, immediately implies the following two corollaries.

Corollary 2.21. For any $\ell \in \mathbb{N}$, $\ell > 0$, $\varepsilon \in (0,1)$, there exists some $N_0, \ldots, N_\ell$ such that the following problem is NP-hard: Given an $(\ell+1)$-player 1-round projective game with domains $[N_0], \ldots, [N_\ell]$, decide if the strong value is 1, or the weak value is not even ε.

Corollary 2.22. Assuming UGC, for any $\ell \in \mathbb{N}$, $\ell > 0$, $\varepsilon \in (0,1)$, there exists some N such that the following problem is NP-hard: Given an $(\ell+1)$-player 1-round unique game with domain $[N]$, decide if the strong value is at least $1-\varepsilon$, or the weak value is not even ε.

Finally, we will need the following relatively simple corollary of Corollary 2.21. This is very similar to [BBKO21, Lemma 5.11, Theorem 5.21], and is proved using very standard techniques (e.g. using the long code/polymorphism gadget to solve label cover problems). Nevertheless, it is incomparable with that result — in this version, one only needs to prove that the "selection functions" (i.e. I_i below) work on polymorphisms of arity at most N. The fact that this alteration is possible was observed in [NŽ23b]. We will only use this extra power in one proof in Chapter 4; in all other cases [BBKO21, Lemma 5.11, Theorem 5.21] is sufficient. This theorem could also be proved in a more elegant manner using more theory in the style of [BBKO21], however we take the direct route since this is all we need.

Theorem 2.23. Fix C, k. There exists some N such that the following holds. Consider some $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$, and let $\mathcal{M} = \mathrm{Pol}(\mathbf{A}, \mathbf{B})$. Cover $\mathcal{M}$ by k sets $\mathcal{M}_1, \ldots, \mathcal{M}_k$. For every $i \in [k]$, suppose there exists a mapping I_i that takes $f \in \mathcal{M}_i$ of *arity* $n \leq N$ to a set $I_i(f) \subseteq [n]$ of size at most C. Suppose that for every $f, g \in \mathcal{M}_i$ *where* $\mathrm{ar}(f) \leq N, \mathrm{ar}(g) \leq N$ and π such that $f \xrightarrow{\pi} g$ we have that $\pi(I_i(f)) \cap I_i(g) \neq \emptyset$. Then $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is NP-hard.

For the proof that follows, we let $\mathrm{PCSP}_=(\mathbf{A}, \mathbf{B})$ denote the problem $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$, but augmenting both $\mathbf{A}$ and $\mathbf{B}$ with an equality relation in their respective domains. Note that the two problems are equivalent up to poly-time reductions: $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is clearly no more difficult than $\mathrm{PCSP}_=(\mathbf{A}, \mathbf{B})$. To reduce $\mathrm{PCSP}_=(\mathbf{A}, \mathbf{B})$ to $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$, simply repeatedly replace pairs of variables bound by an $=$ by one fresh variable. This is sufficient for the decision version of the problem; for the search version, undo the merges to get a solution to the original instance. (Indeed this reduction works even in log-space, using the fact that undirected reachability can be decided in logarithmic space [Rei08]; however we will not use this fact.)

Proof. Let $\mathbf{A} = (A; R_1^{\mathbf{A}}, \ldots, R_t^{\mathbf{A}})$ and $\mathbf{B} = (B; R_1^{\mathbf{B}}, \ldots, R_t^{\mathbf{B}})$. Suppose $r_i = \mathrm{ar}(R_i)$ for $i \in [t]$. Suppose that Corollary 2.21, for $\ell = k$ and $\varepsilon = 1/C^2$, gives us domain sizes $N_0, \ldots, N_k$. Then we set $N = \max(N_0, \ldots, N_k)$, and we reduce from the problem of Corollary 2.21 for $(k+1)$-player 1-round games with $\varepsilon = 1/C^2$ and domain sizes $N_0, \ldots, N_k$ to $\mathrm{PCSP}_=(\mathbf{A}, \mathbf{B})$.

Reduction. Suppose the input $(k+1)$-player 1-round projective game has domain $X = X_0 \cup \ldots \cup X_k$, and constraint set E. For every variable $x \in X_i$, we introduce in the output instance a *long code block* $\boldsymbol{x} = \{(a_1, \ldots, a_{N_i}) \mid a_1, \ldots, a_{N_i} \in A\}$. Furthermore, for every $j \in [t]$ and every matrix $M \in A^{r_j \times N_i}$ with r_j rows and N_i columns, such that every column is an element of $R_j^{\mathbf{A}}$, letting $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{r_j}$ be the rows of this matrix, we introduce the constraint $R_j(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{r_j})$ within the long code block.

Next, consider the set of constraints E. Consider some constraint on variables $x_0, \dots, x_k$, given by the family of functions $\boldsymbol{\pi} = (\pi_{ij})_{0 \le i < j \le k}$. We introduce equality constraints between long code blocks based on this constraint. In particular, for every $0 \le i < j \le k$ we do the following. For every $a_1, \dots, a_{N_j} \in A$, introduce an equality constraint between the variable given by $(a_1, \dots, a_{N_j})$ in the long code block of x_j, and the variable given by $(a_{\pi_{ij}(1)}, \dots, a_{\pi_{ij}(N_i)})$ in the long code block of x_i.

Let us remark on the semantics of this reduction. Suppose that c is a solution to this instance in the constraint language of $\mathbf{C} \in \{\mathbf{A}, \mathbf{B}\}$. Consider some long-code block $\boldsymbol{x} = \{(a_1, \dots, a_{N_i}) \mid a_1, \dots, a_{N_i} \in A\}$, corresponding to a variable x in the original instance. We observe that all the constraints within the long code block essentially impose the fact that the map $(a_1, \dots a_{N_i}) \mapsto c(a_1, \dots, a_{N_i})$ — where this latter value refers to the value assigned to the variable corresponding to $(a_1, \dots, a_{N_i})$ within the long code block — is a polymorphism from $\mathbf{A}$ to $\mathbf{C}$. For any variable x, and for some fixed solution c, we let f_x denote this associated polymorphism.

Next, consider a constraint on $x_0, \dots, x_k$, given by the family of functions $\boldsymbol{\pi} = (\pi_{ij})_{0 \le i < j \le k}$. The equality constraints essentially impose the fact that, for every $0 \le i < j \le k$, we have $f_{x_i}^{\pi_{ij}} = f_{x_j}$.

Completeness. Suppose that c is an assignment for the input instance which strongly satisfies all constraints. We claim then that there exists some solution to the output instance in the constraint language of $\mathbf{A}$. Equivalently, we must assign each variable $x \in X_0 \cup \dots \cup X_k$ a polymorphism f_x from $\mathbf{A}$ to $\mathbf{A}$ of appropriate arity, so that, for every constraint $x_0, \dots, x_k$ given by the family of functions $\boldsymbol{\pi} = (\pi_{ij})_{0 \le i < j \le k}$, we have $f_{x_i}^{\pi_{ij}} = f_{x_j}$. This is easy to do: for $x \in X_i$, let

$$f_x(a_1, \dots, a_{N_i}) = a_{c(x)}.$$

It is easy to check that such functions are always polymorphisms, and that they satisfy the equality constraints.

Soundness. Suppose that the output instance has a solution in the constraint language of $\mathbf{B}$ — we must now prove that the original instance had weak value at least $1/C^2$. The fact that the output instance has a solution in the constraint language of $\mathbf{B}$ is equivalent to the fact that, for every variable $x \in X_i$ for $i \in \{0, \dots, k\}$, we can assign a polymorphism $f_x : \mathbf{A}^{N_i} \to \mathbf{B}$, such that the equality constraints mentioned above are satisfied. We will randomly construct an assignment c for the original instance, so that an expected $1/C^2$-fraction of the constraints are weakly satisfied. To do this, recall that we can split $\mathcal{M} = \mathrm{Pol}(\mathbf{A}, \mathbf{B})$ into k parts $\mathcal{M}_1, \dots, \mathcal{M}_k$, so that certain mappings $I_1, \dots, I_k$ exist that satisfy certain properties. For each $f_x \in \mathrm{Pol}(\mathbf{A}, \mathbf{B})$, consider the minimum u_x so that $f_x \in \mathcal{M}_{u_x}$. Recall that $I_{u_x}(f_x) \subseteq [N_i]$ is a set of size at most C. Select $c(x)$ uniformly and independently at random from the set $I_{u_x}(f_x)$.

Now, let us prove that this assignment weakly satisfies an expected $1/C^2$-fraction of the constraints. It is sufficient to prove that each constraint is weakly satisfied with probability at least $1/C^2$. Consider some constraint on variables $x_0, \dots, x_k$, given by the family of functions $\boldsymbol{\pi} = (\pi_{ij})_{0 \le i < j \le k}$. By the pigeonhole principle, there exists $0 \le i < j \le \ell$ such that $u = u_{x_i} = u_{x_j}$. Hence, $c(x_i)$ is selected uniformly and independently at random from $I_u(f_{x_i})$,

and $c(x_j)$ is selected uniformly and independently at random from $I_u(f_{x_j})$. Since $f_{x_i}^{\pi_{ij}} = f_{x_j}$ due to the equality constraints, by assumption we have that $\pi_{ij}(I_u(f_{x_i})) \cap I_u(f_{x_j}) \neq \emptyset$; suppose that the intersection contains the element e. Then, since $|I_u(f_{x_i})| \leq C, |I_u(f_{x_j})| \leq C$, we have that $\pi_{ij}(c(x_i)) = c(x_j) = e$ with probability at least $1/C^2$. Hence, the conclusion follows. □

Part I.

Crisp Promise Constraint Satisfaction Problems

3. Algorithms for Non-Constant Regimes

In this chapter, we will describe new algorithms for *non-constant regime PCSPs*: PCSPs where the variable domains and the constraints depend on the size of the instance. We will focus on algorithms for *Approximate Linearly-Ordered Colouring*. Given a linearly-ordered 2-colourable 3-uniform hypergraph with n vertices, we will give algorithms that produce a linearly-ordered $f(n)$-colouring, where $f(n)$ goes from $O(\sqrt{n})$, to $O(\sqrt{n/\log n})$, to $O(\sqrt[3]{n})$, and finally to $\log_2(n)$. This chapter will present results from the following previously published works.

[HMNŽ24] Johan Håstad, Björn Martinsson, Tamio-Vesa Nakajima and Stanislav Živný. 'A Logarithmic Approximation of Linearly-Ordered Colourings'. In: *APPROX/RANDOM*. Vol. 317. LIPIcs. LZI, 2024, 7:1–7:6. DOI: 10.4230/lipics.approx/random.2024.7.

[NŽ22] Tamio-Vesa Nakajima and Stanislav Živný. 'Linearly Ordered Colourings of Hypergraphs'. In: *ICALP*. Vol. 229. LIPIcs. LZI, 2022, 128:1–128:18. DOI: 10.4230/lipics.icalp.2022.128.

[NŽ23b] Tamio-Vesa Nakajima and Stanislav Živný. 'Linearly Ordered Colourings of Hypergraphs'. In: *ACM Trans. Comput. Theory* 14.3-4 (2023), pp. 1–19. DOI: 10.1145/3570909.

The works [NŽ22; NŽ23b] were produced in collaboration with my supervisor; furthermore, the work [HMNŽ24] is the result of a merger of work produced with my supervisor (distributed in preprint form as [NŽ24a]), with the independent work of Håstad and Martinsson.

3.1. Introduction

As mentioned above, the primary problem we will discuss is the approximate linearly-ordered colouring problem, which we now define formally. First, we recall the definition of a Linearly Ordered (LO) colouring.

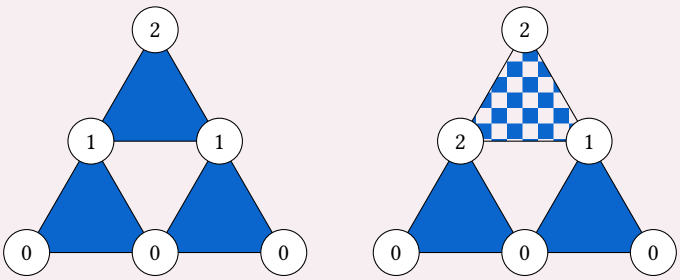
Definition 3.1. Given an r-uniform hypergraph $\mathbf{H} = (V, E)$ with vertex set V and edge set $E \subseteq V^r$, an LO k-colouring of $\mathbf{H}$ is a function $c : V \to \{0, \dots, k-1\}$ such that, for every $(x_1, \dots, x_r) \in E$, the sequence $c(x_1), \dots, c(x_r)$ contains a unique maximum element — that is, the largest element in the sequence appears with multiplicity exactly one.
Equivalently, we let $\mathbf{LO}_k^r$ be a relational structure with one r-ary relation, on domain $\{0, \dots, k-1\}$, whose one relation is given by

$$\{(x_1, \dots, x_r) \mid \exists i \cdot \forall j \cdot (i \neq j) \Rightarrow (x_i > x_j)\}.$$

Then an LO k-colouring of an r-uniform hypergraph $\mathbf{H}$ is a homomorphism $\mathbf{H} \to \mathbf{LO}_k^r$.
Within this chapter, we will generally assume $r = 3$ if it is omitted i.e. $\mathbf{LO}_k = \mathbf{LO}_k^3$.

This notion of colouring has appeared previously under the name *Unique Maximum Colouring* [CKP13], and is similar to the notion of *Conflict Free (CF) Colouring* [Smo10; ELRS03] (which requires that *some* element appear with multiplicity exactly one, not necessarily the maximum element). We note that there has been some work on approximate CF colouring [NVWŽ25]. We use the "Linearly Ordered" terminology for consistency with the promise LO colouring literature, which begins with [BBB21].

Example 3.2. The assignment on the left is an LO colouring of a hypergraph. The assignment on the right is *not* an LO colouring of a hypergraph, due to the checkerboard edge.



Remark 3.3. While LO colouring is of theoretical interest by itself (as we shall see in this chapter as well as others), the idea appears in some practical scenarios as well. For example, in the conflict-free colouring literature [Smo10; CKP13; ELRS03], an example that often appears is that of cellphone tower frequency assignment. Suppose that there are several cellphone towers, each tower t broadcasting at frequency $\nu(t)$. Suppose that we are a cellphone operator, and we have several towers $t_1, \dots, t_r$ in range. To communicate with some tower, say t_i, its frequency must be unique — i.e. for all $j \neq i$ we must have $\nu(t_i) \neq \nu(t_j)$. This motivates conflict free colouring; however, in practice it would be useful to be able to easily determine *which* frequency is the one we want to communicate on. One easy way to do this is to ensure that the tower t_i with the maximum frequency $\nu(t_i)$ can be communicated with. This then is exactly an LO colouring, where the cellphone towers are the vertices, and the edges correspond to the possible sets of cellphone towers that can be in range simultaneously.

We now move on to approximate LO colouring.

Definition 3.4. In the r-uniform k-vs.-$f(n)$ approximate LO colouring problem, we are given an n-vertex r-uniform hypergraph $\mathbf{H} = (V, E)$ that admits an LO k-colouring. We must then compute an LO $\max(k, \lfloor f(n) \rfloor)$-colouring of $\mathbf{H}$. We denote this problem by $\mathrm{PCSP}\left(\mathbf{LO}^r_k, \mathbf{LO}^r_{f(n)}\right)$.[1]

In this chapter, we will prove the following theorem.

Theorem 3.5. The following problems have polynomial time algorithms:

(i) $\mathrm{PCSP}\left(\mathbf{LO}_2, \mathbf{LO}_{O(\sqrt{n})}\right)$ [NŽ22].

(ii) $\mathrm{PCSP}\left(\mathbf{LO}_2, \mathbf{LO}_{O(\sqrt{n/\log n})}\right)$ (simplification of [NŽ22]).

(iii) $\mathrm{PCSP}\left(\mathbf{LO}_2, \mathbf{LO}_{O(\sqrt[3]{n})}\right)$ (simplification of [NŽ23b]).

(iv) $\mathrm{PCSP}\left(\mathbf{LO}_2, \mathbf{LO}_{\log_2(n)}\right)$ [HMNŽ24].

(We prove simplified versions of results in [NŽ22; NŽ23b], worse by a factor of the form $(\log n/\log\log n)^{O(1)}$.[2] This is since we have found some errors in the proofs presented in [NŽ22; NŽ23b].)

We will prove this theorem in multiple parts in the following sections (Sections 3.2 to 3.5).

First we will prove an easy lemma that is implicit within [HMNŽ24; NŽ22; NŽ23b]. This has to do with how we can combine two LO colourings for subhypergraphs of our hypergraph to create a full LO colouring.

Lemma 3.6 (LO colouring combination). Suppose $\mathbf{H} = (V, E)$ is a 3-uniform hypergraph, and let $V = A \cup B$ be a partition of V. Suppose every edge $(x, y, z) \in E$ contains exactly 0, 2 or 3 elements in A (and hence exactly 0, 1 or 3 elements in B). For $X \in \{A, B\}$, let $c_X : X \to \{0, \dots, k_X - 1\}$ be an LO k_X-colouring of $\mathbf{H}|_X$.[3] Then $c : V \to \{0, \dots, k_A + k_B - 1\}$, given by

$$c(v) = \begin{cases} c_A(v), & v \in A \\ c_B(v) + k_A, & v \in B. \end{cases},$$

is an LO $(k_A + k_B)$-colouring of $\mathbf{H}$.

Proof. Consider any edge $(x, y, z) \in E$. We must show that $c(x, y, z) = (c(x), c(y), c(z))$ has a unique maximum element. There are several cases, depending on the number of vertices among x, y, z that belong to A.

[1] This involves slight abuse of notation, since $\mathbf{LO}^r_{f(n)}$ depends on the size of the instance, and also $f(n)$ might not be an integer. We have the $\max(k, -)$ in order to deal with trivial cases where $f(n) < k$ for small values of n — otherwise these cases would be ill-posed.

[2] In particular, Theorem 3.5 (ii) is worse than [NŽ22] by $\sqrt{\log n/\log\log n}$, and Theorem 3.5 (iii) is worse than [NŽ23b] by $\sqrt[3]{\log n/\log\log n}$.

[3] Recall that $\mathbf{H}|_X = (X, E \cap X^3)$, and that $\mathbf{H} - X = \mathbf{H}|_{(V \setminus X)}$.

No elements in A. In this case $c(x,y,z) = c_B(x,y,z) + k_A$, which must have a unique maximum element since (x,y,z) is an edge of $\mathbf{H}|_B$ and c_B is an LO colouring.

One element in A. This case is impossible by our assumption on A and B.

Two elements in A. Suppose without loss of generality that $x, y \in A, z \in B$. Then $c(x,y,z) = (c_A(x), c_A(y), c_B(z) + k_A)$ — and $c_B(z) + k_A$ is the unique maximum, since it is at least k_A, whereas $c_A(x), c_A(y)$ are strictly less than k_A.

All elements in A. This case is analogous to the first case, but based on the fact that c_A is an LO colouring of $\mathbf{H}|_A$.

Thus our conclusion follows in all cases, and c is an LO $(k_A + k_B)$-colouring. □

3.2. PCSP $\left(\mathbf{LO}_2, \mathbf{LO}_{O(\sqrt{n})}\right)$

In this section we prove Theorem 3.5 (i), namely that there exists an algorithm for finding an LO $O(\sqrt{n})$-colouring for an LO 2-colourable 3-uniform hypergraph that works in polynomial time. This algorithm is very similar in spirit to Wigderson's algorithm [Wig82]. Recall that Wigderson's algorithm, if given a 3-colourable graph G, outputs an $O(\sqrt{n})$ colouring in the following way. If there exists a vertex with degree $\geq \sqrt{n}$, say v, then $N_G(v)$ must induce a 2-colourable graph. Thus find a 2-colouring of $N_G(v)$, eliminate $N_G(v)$ from the graph, and colour recursively. Otherwise, if every vertex has degree $< \sqrt{n}$, there exists a $\lceil\sqrt{n}\rceil$-colouring immediately: one can repeatedly extract independent sets of size about $\sqrt{n}$ from the graph, colouring each with a different colour. Our algorithm follows the same strategy: if the hypergraph is dense, we can utilise local structure to get a colouring of a small part of the hypergraph; if the hypergraph is sparse, we globally find an independent set and use it. The difference lies in how the combinatorics of graph colouring differs from that of LO colouring; in particular we select a "high-degree edge" (what this means precisely will become clear later), rather than a high-degree vertex like Wigderson's algorithm.

We begin by considering the following algorithm which extends a partial LO 2-colouring into a larger LO 2-colouring, such that the domain of the larger LO 2-colouring intersects every edge in 0, 1 or 3 vertices.

Lemma 3.7. Algorithm 1 is correct, and runs in polynomial time.

Proof. First observe that the while loop in Algorithm 1 runs at most $|V|$ times, and each iteration does polynomial work. So the entire algorithm runs in polynomial time. For correctness, we will show the required facts about our output in order.

1. As we never remove elements from A, we return a superset of the original input A.
2. The A we return intersects each edge in 0, 1 or 3 vertices — otherwise we could never escape the while loop in the algorithm.
3. In fact, c is an LO 2-colouring of $\mathbf{H}|_A$ at the beginning and end of each iteration. This follows from the fact that $c = c^*|_A$ at these times — since c^* is a global LO 2-colouring.

Algorithm 1 Extension algorithm

▷ $\mathbf{H}$ is a 3-uniform hypergraph; c is an LO 2-colouring of $\mathbf{H}|_A$.
▷ Suppose we return (A', c'). In all cases:
1. $A' \supseteq A$,
2. A' intersects every edge of $\mathbf{H}$ in 0, 1 or 3 vertices,

Furthermore, if there exists some LO 2-colouring $c^* : V \to \{0,1\}$ of $\mathbf{H}$, where $c^*|_A = c$, then:
3. c' is an LO 2-colouring of $\mathbf{H}|_{A'}$,
4. $c' = c^*|_{A'}$.

procedure EXTEND($\mathbf{H} = (V, E), A \subseteq V, c : A \to \{0,1\}$)
  while an edge $(x, y, z) \in E$ intersects A in 2 vertices **do**
    Suppose $x, y \in A, z \notin A$
    $A := A + z$
    $c(z) := 1 - c(x) - c(y)$
  return (A, c)

4. To see that $c = c^*|_A$ at the beginning and end of each iteration, observe that as c^* is an LO 2-colouring, $c^*(x) + c^*(y) + c^*(z) = 1$ for each edge $(x, y, z) \in E$. Hence extending c at z by $c(z) = 1 - c(x) - c(y)$ preserves compatibility with c^*.

Hence the algorithm is correct. □

We will also need a simple algorithm to find an independent set in a sparse hypergraph. This is quite straightforward, however we need to define what "sparse" means somewhat carefully.

Definition 3.8. A 3-uniform hypergraph $\mathbf{H} = (V, E)$ is Δ-sparse if, for every edge $(x, y, z) \in E$, there exists some vertex $v \in \{x, y, z\}$ such that $\deg_{\mathbf{H}}(v) \leq \Delta$.

Algorithm 2 Greedy sparse independent set algorithm

▷ $\mathbf{H}$ is an n-vertex 3-uniform hypergraph that is Δ-sparse.
▷ We return an independent set of size at least $n/(\Delta + 1)$.
procedure GREEDYIS($\mathbf{H} = (V, E)$)
  $I := \emptyset$
  while there exists an edge $(x, y, z) \in E$ **do**
    $v := \arg\min_{v \in \{x,y,z\}} \deg_{\mathbf{H}}(v)$
    $I := I + v$
    $\mathbf{H} = (V, E) := \mathbf{H} - N_{\mathbf{H}}[x]$
  return $I \cup V$

Lemma 3.9. Algorithm 2 is correct and runs in polynomial time.

Proof. At the beginning and end of each iteration of the algorithm, we have that the current value of $\mathbf{H}$ contains only vertices that do not belong to the closed neighbourhood of I *in*

the original hypergraph. This insures that the vertex v added to I will (i) increase the size of I by at least one, and (ii) keep I an independent set. Thus we return an independent set in polynomial time.

To see why the independent set has size at least $n/(\Delta+1)$, observe that in every iteration the size of $N_{\mathbf{H}}[x]$ must be at most $\Delta+1$, by the fact that $\mathbf{H}$ is Δ-sparse. (Indeed, note that every subhypergraph of a Δ-sparse hypergraph must also be Δ-sparse, so this property is maintained as we restrict $\mathbf{H}$ further and further.) Thus with every increase of I within the loop, $|V|$ decreases by at most $\Delta+1$. Since at the end we return the current independent set together with all the remaining vertices, and we started with n vertices, we observe that we return an independent set of size at least $n/(\Delta+1)$. □

Algorithms 1 and 2 together with Lemma 3.6 are almost all that we need in order to provide an algorithm for $O(\sqrt{n})$ approximation. We need one more small technical lemma. In future algorithms this will be replaced by more heavy machinery; however in this simple case we will do the analysis explicitly.

Lemma 3.10. For $x \geq 1$, we have $2 + 4\sqrt{x - \sqrt{x}} \leq 4\sqrt{x}$.

Proof. Let $f(y) = y - \sqrt{y^2 - y}$. It is sufficient to prove that for $y \geq 1$ we have $f(y) \geq \frac{1}{2}$. Observe that

$$f(y) = y - \sqrt{y^2-y} = \frac{(y-\sqrt{y^2-y})(y+\sqrt{y^2-y})}{y+\sqrt{y^2-y}} = \frac{y^2-y^2+y}{y+\sqrt{y^2-y}}$$
$$= \frac{y}{y+\sqrt{y^2-y}} = \frac{1}{1+\sqrt{1-1/y}} \geq \frac{1}{1+\sqrt{1-0}} = \frac{1}{2},$$

for all $y \geq 1$, as required. □

Theorem 3.11. Algorithm 3 is correct and works in polynomial time.

Proof. First we show that Algorithm 3 produces an LO colouring. If we enter the first if block, we trivially return an LO colouring. Next, suppose we enter the second if block. We observe that I is an independent set, and hence c is an LO 1-colouring of $\mathbf{H}|_I$, since this hypergraph is a collection of independent vertices. Furthermore, the recursive call on $\mathbf{H}'$ gives an LO colouring of $\mathbf{H}'$ — hence by Lemma 3.6, c'' is an LO colouring. Finally, suppose we enter the final else block; we must show soundness (that we always return an LO colouring), and completeness (that we always return something). For soundness, observe that since c, c' are LO colourings of $\mathbf{H}|_S, \mathbf{H} - S$, it follows by Lemma 3.6 that c'' is an LO colouring. For completeness, observe that since c^* is an LO 2-colouring, for some $v \in \{x, y, z\}$ we have $c^*(v) = 1$. Therefore, it follows that $c^*(u) = 0$ for $u \in N_{\mathbf{H}}(v)$. Thus $c^*|_{N_{\mathbf{H}}[v]} = c$, and hence for this choice of v, the call of $\textsc{extend}(c_0, N_{\mathbf{H}}[v])$ succeeds, and c is an LO 2-colouring. Thus for this choice of v we return something.

Now, let us show that we use the advertised number of colours — indeed, we will prove that we use at most $4\sqrt{n}$ colours. This is obviously true for $n = 0$; we prove the rest by

Algorithm 3 Algorithm for PCSP $\left(\mathbf{LO}_2, \mathbf{LO}_{O(\sqrt{n})}\right)$

▷ $\mathbf{H}$ is an n-vertex 3-uniform hypergraph that admits an LO 2-colouring c^*.
▷ We return an LO $O(\sqrt{n})$-colouring.
procedure COLOUR1($\mathbf{H} = (V, E)$)
    $\Delta \coloneqq \lfloor \sqrt{n} \rfloor - 1$
    if $n = 0$ **then**
        return the empty colouring
    else if $\mathbf{H}$ is Δ-sparse **then**
        $I \coloneqq$ GREEDYIS($\mathbf{H}$)
        Let the LO 1-colouring $c : I \to \{0\}$ be defined by $c(v) = 0$
        $\mathbf{H}' \coloneqq \mathbf{H} - I$
        $c' \coloneqq$ COLOUR1($\mathbf{H}'$)
        Let c'' be the colouring made by applying Lemma 3.6 to c and c'
        return c''
    else
        As $\mathbf{H}$ is not Δ-sparse, there exists $(x, y, z) \in E$ with $\min_{v \in \{x,y,z\}} \deg_{\mathbf{H}}(v) > \Delta$
        for $v \in \{x, y, z\}$ **do**
            Let $c_0 : N_{\mathbf{H}}[v] \to \{0, 1\}$ be defined by $c_0(v) = 1$, $c_0(u) = 0$ for $u \neq v$
            $(c, S) \coloneqq$ EXTEND($c_0, N_{\mathbf{H}}[v]$)
            if c is an LO 2-colouring of $\mathbf{H}|_S$ **then**
                $\mathbf{H}' \coloneqq \mathbf{H} - S$
                $c' \coloneqq$ COLOUR1($\mathbf{H}'$)
                Let c'' be the colouring made by applying Lemma 3.6 to c and c'
                return c''

induction. Suppose this is true for all hypergraphs with $< n$ vertices, and consider the behaviour of the algorithm on an n-vertex hypergraph. If we enter the second if block, the independent set must have $n/(\Delta+1) = n/\lfloor\sqrt{n}\rfloor \geq \sqrt{n}$ vertices. Hence, inductively c' uses at most $4\sqrt{n-\sqrt{n}}$ colours. Thus we return a colouring with at most $1 + 4\sqrt{n-\sqrt{n}}$ colours, which by Lemma 3.10 is at most $4\sqrt{n}$ colours. On the other hand, suppose that we enter the final else block. Suppose we have chosen a vertex v that leads to termination. Then $\deg_H(v) > \Delta$, hence $|N_h[v]| \geq \sqrt{n}$. The call of EXTEND must therefore return an S with $|S| \geq \sqrt{n}$; hence, inductively c' uses at most $4\sqrt{n-\sqrt{n}}$ colours. Thus we obtain a colouring with at most $2 + 4\sqrt{n-\sqrt{n}}$ colours, which again by Lemma 3.10 is at most $4\sqrt{n}$.

Finally, the algorithm must run in polynomial time. This is because each recursive call uses up at least one of the polynomially many colours we can use; and furthermore each call runs in polynomial time. □

3.3. PCSP $\left(\mathbf{LO}_2, \mathbf{LO}_{O(\sqrt{n/\log n})}\right)$

In this section we prove Theorem 3.5 (ii), namely that there exists an algorithm for the PCSP $\left(\mathbf{LO}_2, \mathbf{LO}_{O(\sqrt{n/\log n})}\right)$ problem that works in polynomial time. This algorithm follows the progression from approximate graph colouring: After Wigderson's algorithm, the algorithm of Berger and Rompel [BR90] slightly improved performance by selecting multiple high-degree vertices at once, rather than only one. Similarly, the algorithm in this section is very close to the one in the previous section, but selects multiple edges at once. First, let us show a simple technical bound.

Lemma 3.12. Consider any algorithm that, on an instance of size n, uses $O(1)$ of some resource (e.g. time, memory, colours, etc.), and then does a recursive call on an instance of size $n - f(n)$, where $f(n) = \Theta(n^\alpha (\log n)^\beta)$ for $\alpha < 1$, terminating when the instance size is $O(1)$. The algorithm will then use $O(n^{1-\alpha}(\log n)^{-\beta})$ of that resource overall.

Proof. We can assume n to be a power of two without loss of generality. For large enough n, how much of our resource do we use until we halve the instance size? Observe that each iteration reduces the instance size by at least $\min_{n/2 \leq t \leq n} f(t) = \Omega(n^\alpha (\log n)^\beta)$. Thus we require at most $n/2$ divided by this, i.e. $O(n^{1-\alpha}(\log n)^{-\beta})$ steps to halve our instance size. It follows that we use the same amount of our resource (up to a constant factor) to halve the instance size.

Thus, on an instance of size n, we use $O(n^{1-\alpha}(\log n)^{-\beta})$ of our resource, and recurse to one instance of size $n/2$. By the *Master theorem* [CLRS22], we find that we use at most $O(n^{1-\alpha}(\log n)^{-\beta})$ of our resource overall. □

We are now ready to give the main algorithm from this section.

Theorem 3.13. Algorithm 4 is correct and runs in polynomial time.

Algorithm 4 Algorithm for PCSP $\left(\mathbf{LO}_2, \mathbf{LO}_{O(\sqrt{n/\log n})}\right)$

▷ $\mathbf{H}$ is an n-vertex 3-uniform hypergraph that admits an LO 2-colouring c^*.
▷ We return an LO $O(\sqrt{n/\log n})$-colouring
procedure COLOUR2($\mathbf{H} = (V, E)$)
    if $n = 0$ **then**
        return an empty colouring
    $\Delta := \lfloor \sqrt{n/\log n} \rfloor$
    $k := \lceil \log_3 n \rceil$
    $S_0 := \varnothing$
    Let c_0 be the empty colouring
    $\mathbf{H}_0 := \mathbf{H}$
    for $t_1, \ldots, t_k \in \{0, 1, 2\}$ **do**
        for $i = 1, \ldots, k$ **do**
            if $\mathbf{H}_{i-1}$ has no edges **then**
                $\mathbf{H}_i := \mathbf{H}_{i-1}$
                $S_i := S_{i-1}$
                $c_i := c_{i-1}$
            else
                Let (v_i^0, v_i^1, v_i^2) be the edge of $\mathbf{H}_{i-1}$ where $\min_t \deg_{\mathbf{H}_{i-1}}(v_i^t)$ is maximised. In case there are multiple such edges, break ties in a consistent way (i.e. for a fixed value of $\mathbf{H}_{i-1}$ choose the same edge (v_i^0, v_i^1, v_i^2)).
                $S'_{i-1} := S_{i-1} \cup N_{\mathbf{H}}[v_i^{t_i}]$
                Let $c'_{i-1} : S'_{i-1} \to \{0, 1\}$ be given by

$$c'_{i-1}(u) = \begin{cases} c_{i-1}(u), & u \in S_{i-1} \\ 1, & u = v_i^{t_i} \\ 0, & \text{otherwise} \end{cases}$$

                $(S_i, c_i) := \text{EXTEND}(\mathbf{H}, S'_{i-1}, c'_{i-1})$
                $\mathbf{H}_i := \mathbf{H} - S_i$
        if c_k is an LO colouring of $\mathbf{H}|_{S_k}$ **then**
            $I := \varnothing$
            if $\mathbf{H}_k$ is Δ-sparse **then**
                $I := \text{GREEDYIS}(\mathbf{H}_k)$
            Let $c : I \to \{0\}$ be given by $c(u) = 0$
            $c' := \text{COLOUR2}(\mathbf{H} - (S_k \cup I))$
            Combine c with c' using Lemma 3.6 to get a colouring of $\mathbf{H} - S_k$, and then with c_k using Lemma 3.6 again to get c''
            return c''

Proof. We inductively prove the correctness of our algorithm in three steps: first completeness (i.e. the algorithm always returns something), next soundness (i.e. the algorithm returns an LO colouring), and finally we prove a bound on the number of colours used. These being proved, the fact that the algorithm runs in polynomial time is immediate, since each iteration must consume at least one colour, and each iteration runs in polynomial time. Note also that the base case of $n = 0$ is obviously correct.

Completeness. There exists some choice of $t_1, \dots, t_k$ such that for our choices of v_i^t (for values of i where these are defined), we have $c^*(v_i^{t_i}) = 1$. This is because c^* is an LO 2-colouring of $\mathbf{H}$, and thus so is its restriction to $\mathbf{H}_{i-1}$. (This crucially depends on breaking ties in a consistent way in Line 18: due to this we can be sure that, for any given choice of $t_1, \dots, t_{i-1}$, the hypergraph $\mathbf{H}_{i-1}$ is uniquely defined.)

We show that when we are at this choice, we must return something. We prove inductively that $c_i = c^*|_{S_i}$. This is clearly true initially (i.e. when $S_0 = \emptyset$). So assume it is true for c_{i-1} and S_{i-1}. If $\mathbf{H}_{i-1}$ has no edges, then $S_i = S_{i-1}$ and $c_i = c_{i-1}$, so the induction works immediately. Assume therefore that $\mathbf{H}_{i-1}$ has edges. We then note that $c'_{i-1} = c^*|_{S'_{i-1}}$. This is because $c^*(v_i^{t_i}) = 1$ by assumption, and thus $c^*(u) = 0$ for $u \in N_{\mathbf{H}}(v_i^{t_i})$ as c^* is an LO 2-colouring — thus the new values in c'_{i-1} coincide with those in c^*, and the old ones do by inductive hypothesis. By the properties of Algorithm 1, we get that $c_i = c^*|_{S_i}$ as desired.

Since $c_k = c^*|_{S_k}$, it follows easily that c_k is an LO colouring of $\mathbf{H}|_{S_k}$. So in this case we enter the if block containing the return statement. Thus we must return something in this case.

Soundness. Suppose we are in an iteration of the outer for loop when we return something. Thus we can assume that c_k is an LO colouring of $\mathbf{H}|_{S_k}$. Furthermore since S_k is an output of Algorithm 1 (or $\emptyset$), it must intersect every edge in 0, 1 or 3 vertices.

Next, I is an independent set of $\mathbf{H}_k$: it is (trivially) thus initially, and if we set I to $\textsc{greedyis}(\mathbf{H}_k)$, it remains an independent set of $\mathbf{H}_k$.

Furthermore c is an LO colouring of $\mathbf{H}|_I$ trivially, and c' is an LO colouring of $\mathbf{H} - (S_k \cup I)$ inductively. By these properties, we can combine c with c' using Lemma 3.6 and then with c_k using Lemma 3.6 again, to get a proper LO colouring of $\mathbf{H}$. So we always return an LO colouring.

Colour bound. We will show that in all cases we colour at least $\Omega(\sqrt{n \log n})$ vertices in each iteration. Since in each iteration we use at most $3 = O(1)$ colours (at most 2 from c_k, and at most 1 from c), the desired bound on the number of colours follows from Lemma 3.12.

There are two cases, depending on whether $\mathbf{H}_k$ is Δ-sparse or not.

$\mathbf{H}_k$ is Δ-sparse. In this case, by the correctness of Algorithm 2, we see that $I \cup S_k$ has at least $\Omega(\sqrt{n \log n})$ vertices. Hence we colour at least this many vertices in this iteration.

$\mathbf{H_k}$ is not Δ-sparse. It follows that $\mathbf{H}_k$ contains some edge (x, y, z), where the degree of all of x, y, z within $\mathbf{H}_k$ is at least Δ. Since $\mathbf{H}_k$ is a subhypergraph of $\mathbf{H}_i$ for $0 \leq i \leq k$, we have that $\mathbf{H}_i$ also contains some such edge. Thus, $\mathbf{H}_{i-1}$ has at least one edge; and when we select (v_i^0, v_i^1, v_i^2), we must have that $\min_t \deg_{\mathbf{H}_{i-1}}(v_i^t) \geq \Delta$. But observe that S_i is larger than S_{i-1} by at least $\deg_{\mathbf{H}_{i-1}}(v_i^{t_i})$. So S_i grows by at least Δ. It follows that S_k has size at least $\Omega(\Delta \log n) = \Omega(\sqrt{n \log n})$. Hence in this case we also colour at least this many vertices in this iteration.

This completes the proof. □

3.4. PCSP $\left(\mathbf{LO}_2, \mathbf{LO}_{O(\sqrt[3]{n})}\right)$

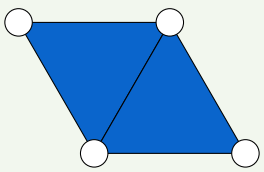
In this section we will give an algorithm for PCSP $\left(\mathbf{LO}_2, \mathbf{LO}_{O(\sqrt[3]{n})}\right)$, proving Theorem 3.5 (iii). In the previous section, our algorithm dealt with sparse hypergraphs using a greedy independent set algorithm, and with dense hypergraphs by using a relatively complex combinatorial approach. This is inspired by the way things are done in approximate graph colouring (i.e. [Wig82; BR90]). However there is a much superior method for dealing with dense hypergraphs, which uses the linear nature of LO 2-colouring: each edge (x, y, z) must satisfy $x + y + z = 1$ in an LO 2-colouring. This will also simplify the sparse case — the somewhat fiddly "Δ-sparse" definition can be replaced by simpler notions. Some of the sparse case will thus be similar, but simpler than the sparse case in the previous section. We will also simplify the presentation from [NŽ23b], removing its notion of "progress", which is based on the notion of progress from [Blu94].

To exploit the linear nature of LO 2-colouring, we will use the following algorithm as a black box.

Theorem 3.14 ([KSTW00, Theorem 2.12, (2)]). Given a satisfiable system of linear equations over $\mathbb{Z}_2$ that has a solution with at least ρ variables set to 1, a solution that has at least $\frac{1}{2}\rho$ variables set to 1 can be found in polynomial time.

In order to build our algorithm, we will apply a useful simplifying assumption: we will assume that our input hypergraph is *linear* (unrelated to the linearity discussed before), a property which we now define.

Definition 3.15 (Linear 3-uniform hypergraph). A 3-uniform hypergraph $\mathbf{H} = (V, E)$ is *linear* if any two edges meet in at most 1 vertex. Equivalently, it is linear if it does not contain a copy of the following hypergraph as a (not necessarily induced) subhypergraph.



It turns out that, for approximate LO colouring, it is sufficient to consider only linear hypergraphs.

Lemma 3.16. Suppose there exists a polynomial-time algorithm $\mathcal{A}$ for PCSP $\left(\mathbf{LO}_2, \mathbf{LO}_{f(n)}\right)$ that works on *linear* hypergraphs. Then there exists a polynomial-time algorithm for PCSP $\left(\mathbf{LO}_2, \mathbf{LO}_{f(n)}\right)$ that works for (not necessarily linear) hypergraphs.

Algorithm 5 LO colouring conditional on an algorithm for linear hypergraphs

```
▷ H is a 3-uniform hypergraph that admits an LO 2-colouring. 𝒜 is a polynomial-time
  algorithm for PCSP (LO_2, LO_{f(n)}) that assumes its input to be linear.
▷ We return an LO f(n)-colouring of H.
procedure conditionalColour(H = (V, E), 𝒜)
    if H is linear then
        return 𝒜(H)
    else
        Since H is not linear, there exist two edges (u, v, x), (u, v, y) ∈ E
        Create H' = (V', E') by replacing x, y in H by a fresh vertex z
        c := conditionalColour(H', 𝒜)
        Define h : V → V' by h(x) = h(y) = z and h(u) = u for u ∉ {x, y}
        return c ∘ h
```

Proof. Consider Algorithm 5 — if we show it is correct the lemma follows. Suppose we give it an LO 2-colourable hypergraph $\mathbf{H} = (V, E)$ (with an unknown colouring $c^* : V \to \{0, 1\}$), and a polynomial-time algorithm $\mathcal{A}$ for PCSP $\left(\mathbf{LO}_2, \mathbf{LO}_{f(n)}\right)$ that works on linear hypergraphs. If we enter the first case, then our algorithm returns an appropriate output in polynomial time. On the other hand, in the second case, we observe that as

$$c^*(u) + c^*(v) + c^*(x) = c^*(u) + c^*(v) + c^*(y) = 1,$$

it follows that $c^*(x) = c^*(y)$. Hence the hypergraph $\mathbf{H}'$ passed to the recursive call remains LO 2-colourable. Furthermore, it is easy to check that h is a homomorphism from $\mathbf{H}$ to $\mathbf{H}'$. Thus by precomposing an LO $f(n)$-colouring of $\mathbf{H}'$ with h, we get an LO $f(n)$-colouring of $\mathbf{H}$. It follows that if we eventually return an output (i.e. we do not enter an infinite loop), we must return a correct output. Termination and time complexity follow from the fact that each recursive call reduces the size of the input hypergraph by at least 1, so we must terminate within polynomially many iterations — and furthermore each iteration runs in polynomial time. □

With this in hand, we can immediately give the algorithm for this case, in Algorithm 6. We will assume that the input hypergraph is linear.

To prove the correctness of our algorithm, we will give a few lemmata first.

Lemma 3.17. Suppose $\mathbf{H} = (V, E)$ is an n-vertex 3-uniform hypergraph where $|E| \leq n\Delta$. Then $\mathbf{H}$ has $\Omega(n)$ vertices with degree at most $O(\Delta)$.

Algorithm 6 Algorithm for PCSP $\left(\mathbf{LO}_2, \mathbf{LO}_{O(\sqrt[3]{n})}\right)$

```
▷ H is an n-vertex linear 3-uniform hypergraph that admits an LO 2-colouring c*.
▷ We will return an LO O(∛n)-colouring.
procedure COLOUR3(H = (V, E))
    if n = 0 then
        return the empty colouring.
    Δ := ∛n
    if |E| ≤ nΔ then
        Let I be the result of greedily finding an independent set in H, i.e. add the
        minimum degree vertex to I, removing its neighbours from H
        Let c : I → {0} be a constant function
        c' := COLOUR3(H − I)
        Let c'' be the result of applying Lemma 3.6 to c, c'
        return c''
    else
        Write down a system of linear equations A on variable set x_v for v ∈ V, with
        equations x_a + x_b + x_c ≡ 0 mod 2 for (x, y, z) ∈ E
        Let x̂_v for v ∈ V be the result of applying the algorithm of Theorem 3.14 to A
        Let S = {v | x̂_v = 1}
        Let c : S → {0} be a constant function
        c' := COLOUR3(H − S)
        Let c'' be the result of applying Lemma 3.6 to c, c'
        return c''
```

Proof. Suppose we randomly select a vertex v from $\mathbf{H}$. The expected degree of such a vertex at most $O(|E|/n) = O(\Delta)$. By Markov's inequality, with probability at least $1/2$ the vertex has degree at most $O(\Delta)$. Thus there are at least $\Omega(n)$ vertices with degree at most $O(\Delta)$. □

Lemma 3.18. Suppose $\mathbf{H} = (V, E)$ is an n-vertex, linear hypergraph. Then any LO 2-colouring of $\mathbf{H}$ must colour at least $\sqrt{|E|}$ vertices to 0.

Proof. Consider any LO 2-colouring $c : V \to \{0, 1\}$. Let $A = c^{-1}(0)$, and consider any pair $(u, v) \in A^2$. By linearity of $\mathbf{H}$ there is at most one edge in E that contains both u and v. Hence $|E| \leq |A|^2$, or equivalently $|A| \geq \sqrt{|E|}$, as required. □

We are now ready to prove the correctness of Algorithm 6.

Theorem 3.19. Algorithm 6 is correct, and runs in polynomial time.

Proof. We first prove that we always return an LO colouring, then we prove that we use the advertised number of colours. This second fact implies that we run in polynomial time, since we use up a colour each iteration, and each iteration runs in polynomial time. We prove this inductively — the base case $n = 0$ is obvious.

Soundness. Consider the two cases of the algorithm. In the first case, we find an independent set I, we create a trivially valid colouring of $\mathbf{H}|_I$, namely c, and then we recursively colour $\mathbf{H} - I$, creating c'. Since I intersects each edge in 0 or 1 vertices, we can indeed apply Lemma 3.6 to create the valid LO colouring c''.

In the second case, we create a set of vertices S which (by definition) intersects each edge of $\mathbf{H}$ in an *even* number of vertices. In other words, S intersects each edge in 0 or 2 vertices. Observe therefore that $\mathbf{H}|_S$ contains no edges at all, and thus $c : S \to \{0\}$ is a trivially valid LO colouring of $\mathbf{H}|_S$. Furthermore, since c' is a recursively found valid LO colouring of $\mathbf{H} - S$, the fact that S intersects each edge in 0 or 2 vertices means that we can indeed apply Lemma 3.6 to create the valid LO colouring c''.

Hence in all cases we return a valid LO colouring.

Colour bound. Again, consider the two cases of the algorithm. We will show that in both cases we colour $\Omega(n^{2/3})$ vertices in each iteration. By Lemma 3.12, it follows that we use at most $O(\sqrt[3]{n})$ colours.

Sparse case. By Lemma 3.17, we see that $\mathbf{H}$ has at least $\Omega(n)$ vertices with degree at most $O(\Delta)$. Hence, the greedy independent set I must have size

$$\Omega(n/\Delta) = \Omega(n/\sqrt[3]{n}) = \Omega(n^{2/3}).$$

Hence, in this case we colour the right number of vertices in this step.

Dense case. By Lemma 3.18, any LO colouring (and in particular c^*) must set at least $\sqrt{|E|} \geq \sqrt{n\Delta}$ vertices to 0. Observe that $x^*_v = 1 - c^*(v)$ is a solution to the system of linear equations A, which sets at least $\sqrt{n\Delta}$ variables to 1. It follows that the solution given by $\widehat{x_v}$ for $v \in V$ must set at least $\frac{1}{2}\sqrt{n\Delta} = \Omega(\sqrt{n\Delta})$ variables to 1. Hence $|S| = \Omega(\sqrt{n\Delta})$, which by our choice of Δ means $|S| = \Omega(n^{2/3})$. So in this case also we coloured the required number of vertices in this step.

Noting that the algorithm always returns something completes the proof. □

Corollary 3.20. PCSP $\left(\mathbf{LO}_2, \mathbf{LO}_{O(\sqrt[3]{n})}\right)$ can be solved in polynomial time.

Proof. Lemma 3.16 together with Theorem 3.19 imply this result. □

3.5. PCSP $\left(\mathbf{LO}_2, \mathbf{LO}_{\log_2 n}\right)$

In this section we prove Theorem 3.5 (iv). This algorithm extensively uses the system of linear equations from the previous algorithm, together with a generalisation of the observation that we only need to consider linear hypergraphs. This algorithm is also, surprisingly, both exponentially better and much simpler than the previous algorithms.

We mention in passing that concurrently with this algorithm, a different algorithm was proposed by Louis, Newman and Ray [LNR24]. Their algorithm uses SDP and produces a colouring using $\tilde{O}(\sqrt[5]{n})$ colours (where $\tilde{O}$ ignores poly-logarithmic factors). Furthermore,

our algorithm can be improved by applying an appropriate *strong sparsification* to the input, leading to a $(0.999)\log_2 n$ colouring [BNOŽ25].

The presentation given here will follow the original preprint version [NŽ24a] more than the final published version [HMNŽ24]. We will first need a simple procedure for finding a modulo 2 linear combination of vectors with Hamming weight at least as large as that of the random one. This algorithm is in fact identical to the algorithm from [KSTW00] i.e. the algorithm of Theorem 3.14, and the analysis is quite similar. We give it for completeness.

Algorithm 7 Algorithm to find a modulo 2 combination of vectors with large Hamming weight

▷ Suppose $r_1, \dots, r_d$ taken uniformly at random from $\{0,1\}$.
▷ We return the expected Hamming weight of $x^0 + \sum_{i=1}^d r_i x^i$.
procedure FINDEXPECTATION($x^0, \dots, x^d \in \mathbb{Z}_2^n$)
    $a := 0$
    for $t = 1, \dots, n$ **do**
        if $x_t^i = 1$ for some $i \in [d]$ **then**
            $a := a + \frac{1}{2}$
        else
            $a := a + x_t^0$
    return a
▷ Suppose $r_1, \dots, r_d$ are taken uniformly at random from $\{0,1\}$, and suppose ρ is the expected Hamming weight of $\sum_{i=1}^d r_i x^i$. Then we return $a_1, \dots, a_d$ such that the Hamming weight of $\sum_{i=1}^d a_i x^i$ is at least ρ.
procedure FINDCOMBINATION($x^1, \dots, x^d \in \mathbb{Z}_2^n$)
    $x^0 := \vec{0}$
    for $i = 1, \dots, n$ **do**
        if FINDEXPECTATION$(x^0, x^{i+1}, \dots, x^d) \leq$ FINDEXPECTATION$(x^0 + x^i, x^{i+1}, \dots, x^d)$ **then**
            $a_i := 1$
            $x^0 := x^0 + x^i$
        else
            $a_i := 0$
    return $(a_1, \dots, a_n)$

Lemma 3.21. Algorithm 7 is correct and runs in polynomial time.

Proof. First note that both procedures clearly run in polynomial time. Thus we need only show correctness. We show in particular the correctness of the first procedure: the correctness of the second procedure follows immediately by the method of conditional expectations.

To see why the first procedure is correct, note that by linearity of expectation, the expected Hamming weight of $x^0 + \sum_{i=1}^d r_i x^i$ is equal to the sums of the expectations of $b_t = x_t^0 + \sum_{i=1}^d r_i x_t^i$ for $t \in [n]$ (where the sum is taken modulo 2). Now, this is in fact a Bernoulli random variable. If there exists some i such that $x_t^i = 1$, then r_i contributes to the distribution and we can see that b_t takes values 0 and 1 equiprobably; thus it has expectation $\frac{1}{2}$. On

the other hand if $x_t^i = 0$ for all $i \in [d]$, then $b_t = x_t^0$ by definition and has expectation x_t^0. It follows that the variable a accumulates the expectations of b_t for $t \in [n]$, and thus returning it is correct. □

We next define a convenient notation for the system of equations used in Algorithm 6.

Definition 3.22. Suppose $\mathbf{H} = (V, E)$ is a 3-uniform hypergraph. We define $A(\mathbf{H})$ to be the system of equations whose variables are $x_v \in \{0, 1\}$ for $v \in V$, and whose equations are $x_a + x_b + x_c \equiv 0 \bmod 2$ for $(a, b, c) \in E$.

The important thing about this system of equations is that it is a *relaxation* of LO 2-colouring: every LO 2-colouring is a solution to this system of equations (after flipping 0 and 1). The following property then follows.

Lemma 3.23. If the variable x_u in $A = A(\mathbf{H})$ is set to 0 in all solutions to A, then for any LO 2-colouring c of $\mathbf{H}$ we have $c(u) = 1$.

Proof. Note that setting $x_u = 1 - c(u)$ gives rise to a solution of A. This is because these evaluations satisfy $x_v \in \{0, 1\}$ for $v \in V$ and $x_a + x_b + x_c = 2$ for $(a, b, c) \in E$ — thus of course they satisfy $x_a + x_b + x_c \equiv 0 \bmod 2$. In particular, since $x_u = 0$ by assumption, we have $c(u) = 1$ as required. □

It follows that *all variables fixed to 0 in all solutions of $A(\mathbf{H})$ can be merged!* Thus, for the purposes of approximate LO colouring, we can assume that at most one variable in $A(\mathbf{H})$ is fixed to 0. This is quite similar to the observation that we can force our hypergraph to be linear.[4] For completeness we write out this observation similarly to Lemma 3.16.

Lemma 3.24. Suppose there exists a polynomial-time algorithm $\mathcal{A}$ for PCSP $\left(\mathbf{LO}_2, \mathbf{LO}_{f(n)}\right)$ that works on hypergraphs $\mathbf{H}$ where at most one variable of $A = A(\mathbf{H})$ is fixed to 0. Then there exists a polynomial-time algorithm for PCSP $\left(\mathbf{LO}_2, \mathbf{LO}_{f(n)}\right)$ that works for arbitrary hypergraphs.

Proof. If we show Algorithm 8 is correct and runs in polynomial time then our lemma follows. Note first that the initial checks can be done in polynomial time, using Gaussian elimination. Note that, by Lemma 3.23, all vertices in Z are forced to be 1 in any LO 2-colouring. In particular, they are coloured the same in c^* — thus $\mathbf{H}'$ remains LO 2-colourable, and the call of $\mathcal{A}$ succeeds. Finally, $h : \mathbf{H} \to \mathbf{H}'$ is easily seen to be a homomorphism, so $c \circ h$ is an appropriate LO colouring of $\mathbf{H}$. □

With this in hand, we can present our final algorithm. In what follows, observe that a solution to $A(\mathbf{H})$ can be seen as an $|V|$-dimensional vector in $\mathbb{Z}_2^{|V|}$.

We also need a very simple lemma.

[4] This kind of reasoning can be generalised to *strong sparsification* [BNOŽ25].

Algorithm 8 LO colouring conditional on an algorithm for hypergraphs with assumptions of Lemma 3.24

▷ $\mathbf{H}$ is a 3-uniform hypergraph that admits an LO 2-colouring c^*. $\mathcal{A}$ is a polynomial-time algorithm for PCSP $(\mathbf{LO}_2, \mathbf{LO}_{f(n)})$ with extra assumptions as in Lemma 3.24.
▷ We return an LO $f(n)$-colouring of $\mathbf{H}$.
procedure conditionalColour2($\mathbf{H} = (V, E), \mathcal{A}$)
    Write out $A = A(\mathbf{H})$
    For every variable x_u of A, check if there exists a solution setting $x_u = 1$.
    Let $Z \subseteq V$ be the set of vertices u for which no such solution exists.
    Create $\mathbf{H}' = (V', E')$ by replacing all vertices in Z by a fresh vertex w.
    $c := \mathcal{A}(\mathbf{H}')$.
    Define $h : V \to V'$ by $h(Z) = w$ and $h(t) = t$ for $t \notin Z$.
    return $c \circ h$

Algorithm 9 Algorithm for PCSP $(\mathbf{LO}_2, \mathbf{LO}_{\log_2 n})$

▷ $\mathbf{H} = (V, E)$ is an n-vertex, LO 2-colourable, 3-uniform hypergraph that admits an LO 2-colouring.
▷ We assume $A(\mathbf{H})$ has at most one variable fixed to 0. We will return an LO $\max(2, \lfloor \log_2 n \rfloor)$-colouring.
procedure colour4($\mathbf{H} = (V, E)$)
    if $n \leq 8$ **then**
        return an LO 2-colouring found by brute force
    Write down $A = A(\mathbf{H})$
    Let $x^1, \ldots, x^d$ be a basis to the space of solutions of A
    $(a_1, \ldots, a_d) :=$ findCombination$(x^1, \ldots, x^d)$
    Let $\widehat{x} = \sum_{i=1}^{d} a_i x^i$, where the sum is modulo 2
    Let $S = \{v \mid \widehat{x_v} = 1\}$
    Let $c : S \to \{0\}$ be a constant function
    $c' :=$ colour4$(\mathbf{H}|_{(V \setminus S)})$
    Let c'' be the result of applying Lemma 3.6 to c, c'
    return c''

Lemma 3.25. Suppose $T(1) = \cdots = T(8) = 2$, and for $n > 8$ we have

$$T(n) = 1 + T\left(\left\lceil \frac{n}{2} \right\rceil\right).$$

Then $T(n) \leq \max(2, \log_2 n)$.

Proof. Firstly, it is easy to see by induction that T must be non-decreasing. Fix $n > 8$ (the result is clearly true for $n \leq 8$) and let N be the minimum integer such that $2^N \geq n$. We show that $T(2^N) \leq \max(2, \log_2 n)$ — a fortiori this is true for $T(n)$, since T is non-decreasing. Observe that $T(2^N) = 1 + T(2^{N-1}) = \cdots = (N-3) + T(2^3) = (N-3) + 2$. But $2^{N-1} \leq n$, hence $N - 1 \leq \log_2(n)$. Thus $T(2^N) = N - 1 \leq \log_2(n)$. □

Theorem 3.26. Algorithm 9 is correct and works in polynomial time.

Proof. We first prove that we indeed return an LO colouring, and then that it uses at most $\max(2, \lfloor \log_2(n) \rfloor)$ colours. Since we use at least one colour in every iteration, it follows that we use at most $\log_2(n)$ iterations, and since every iteration clearly runs in polynomial time, the algorithm overall runs in polynomial time.

Why do we return an LO colouring? If $n \leq 8$ the we obviously return an LO colouring, so assume $n > 8$. Observe that the set S intersects each edge in 0 or 2 vertices. Hence c is trivially an LO colouring of $\mathbf{H}|_S$, and indeed S and $V \setminus S$ satisfy the requirements of Lemma 3.6. So we can apply it to c, c' to make c''. (Observe furthermore that if $A(\mathbf{H})$ has at most one variable fixed to 0, then $A(\mathbf{H} - S)$ also does as well. This is because every solution to $A(\mathbf{H})$ is also a solution to $A(\mathbf{H} - S)$, ignoring the variables corresponding to S.)

Now let us analyse the number of colours we use. For how many $v \in V$ do we have $x_v^i = 0$ for all $i \in [d]$? Note that for any such v, we have that x_v must be equal to 0 in all solutions to A. Hence, by assumption, there is at most one such vertex. It follows therefore (by linearity of expectation) that the expected Hamming weight of $\sum_{i=1}^{d} a_i x^i$ is at least $\frac{n-1}{2}$. Hence it follows that the Hamming weight of $\widehat{x}$ is at least $\frac{n-1}{2}$, and thus $|S| \geq \left\lceil \frac{n-1}{2} \right\rceil$. Thus, from an instance of size n we recurse to an instance of size at most $n - \left\lceil \frac{n-1}{2} \right\rceil = \left\lceil \frac{n}{2} \right\rceil$. Letting $T(n)$ be the obvious recursive upper bound on the number of colours used on a hypergraph with *at most* n vertices and applying Lemma 3.25, we get that we use at most $\max(2, \lfloor \log_2 n \rfloor)$ colours. □

Corollary 3.27. $\mathrm{PCSP}(\mathbf{LO}_2, \mathbf{LO}_{\log_2 n})$ can be solved in polynomial time

Proof. Lemma 3.24 together with Theorem 3.26 imply this result. □

Remark 3.28. This algorithm has a certain relation to algorithms for *temporal CSPs* [BK10]. (One way of thinking about temporal CSPs is that they are constraint satisfaction problems where we output a total preorder of the input variables as our answer, and where each constraint cares only about the order induced on the variables it constrains. Equivalently, the domain of the CSP is $\mathbb{Q}$ and every relation is first-order definable from $<$.) In such problems, the algorithm is often of the form "determine which variables can be put first in the output order, eliminate them and recurse". Indeed, the obvious algorithm for *rainbow-free LO colouring on infinite domain* is very similar to the one we present above. This problem is $\mathrm{CSP}(\mathbf{X})$ where $\mathbf{X}$ has domain $\mathbb{Q}$, and one ternary relation R given by

$$R(x, y, z) \coloneqq (x = y < z) \vee (y = z < x) \vee (z = x < y).$$

Proceeding analogously to Algorithm 9, this problem can be solved as follows: solve a system of modulo 2 linear equations to find the variables equal to the minimum value, remove those variables and recurse. Algorithm 9 does the same thing, but attempting to set as many colours as possible in each step. (Indeed, the algorithm from Section 3.4 is also similar, but it also colours from large colours to small colours.) There is a more general interplay between infinite domain CSPs and promise CSPs, which has recently become more studied [PRSS25; Mot25]. On the other hand, this connection between optimisation versions of algorithms for infinite domain CSPs, and PCSPs on non-constant domains, has not been much explored.

4. Hardness of Linearly Ordered Colouring

In this chapter, we will give hardness results for Approximate Linearly Ordered Colouring in the constant regime. That is, for fixed values of r, k, ℓ, we will prove that, given an r-uniform hypergraph that admits an LO k-colouring, it is NP-hard to find an LO ℓ-colouring. In particular, we prove that this is the case when $r \geq \ell - k + 4$. This chapter will present results from the following previously published works.

[NŽ22] Tamio-Vesa Nakajima and Stanislav Živný. 'Linearly Ordered Colourings of Hypergraphs'. In: *ICALP*. Vol. 229. LIPIcs. LZI, 2022, 128:1–128:18. DOI: 10.4230/lipics.icalp.2022.128.

[NŽ23b] Tamio-Vesa Nakajima and Stanislav Živný. 'Linearly Ordered Colourings of Hypergraphs'. In: *ACM Trans. Comput. Theory* 14.3-4 (2023), pp. 1–19. DOI: 10.1145/3570909.

The works [NŽ22; NŽ23b] were produced in collaboration with my supervisor. This chapter will also discuss the results from [FNOTW24; NVWŽ25] as part of a review of the literature, but without claiming or presenting the proofs.

4.1. Introduction

In view of the extended introduction to LO colouring in Chapter 3, in this chapter we will give a more expedited introduction to the hardness of approximating the LO chromatic number, assuming knowledge of the notations in Chapter 3.

In the context of PCSPs, the first mention of LO colouring appeared in [BBB21]: in that work all problems $\mathrm{PCSP}(\mathbf{LO}_2^3, \mathbf{B})$, where $\mathbf{B}$ is a ternary, symmetric structure on a domain of size 3, are classified, save for the case $\mathbf{B} = \mathbf{LO}_3^3$. The problem $\mathrm{PCSP}(\mathbf{LO}_2^3, \mathbf{LO}_3^3)$ was conjectured to be hard — indeed, more generally the following was conjectured.

Conjecture 4.1 ([BBB21, Conjecture 27]). For every $2 \leq k < \ell$, $\mathrm{PCSP}(\mathbf{LO}_k^3, \mathbf{LO}_\ell^3)$ is NP-hard.

This conjecture may seem somewhat unrelated to the greater PCSP literature — however its truth is connected to a central concept from the CSP literature, *daisy chain terms* [Bra22].[1] We generalise these terms to the case of minions, by dropping the idempotency requirement. (Recall that a polymorphism f is called *idempotent* if $f(x, \ldots, x) = x$ for all x in the domain of the polymorphism.)

[1] I thank Zarathustra 'Zeb' Brady for telling me that these terms have a name.

Definition 4.2 (Daisy chain terms [Bra22], without idempotency). A sequence $f_1, f_2, \ldots \in \mathcal{M}^{(3)}$ is called a sequence of *daisy chain terms* if

- the sequence is periodic,
- $f_i(y,x,x) = f_i(x,x,y) = f_{i+1}(x,y,x)$.

Why are these terms important? It turns out that they give a good hardness condition for CSPs.

Theorem 4.3 (Easy consequence of [Bra22, Corollary 4.1.11]). If $\mathrm{Pol}(\mathbf{A})$ fails to admit daisy chain terms, then $\mathrm{CSP}(\mathbf{A})$ is NP-hard.

It is natural (though perhaps optimistic) to conjecture that the lack of daisy chain terms continues to indicate hardness even for PCSPs.

Conjecture 4.4. For relational structures $\mathbf{A} \to \mathbf{B}$, $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is NP-hard whenever $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ does not admit daisy chain terms.

Surprisingly, Conjecture 4.1 and Conjecture 4.4 are equivalent!

Theorem 4.5. Conjecture 4.1 is equivalent to Conjecture 4.4.

We will prove this fact in Section 4.2.

With this motivation in hand, it let us survey the known results. The first hardness result for this problem is the following.

Theorem 4.6 ([NŽ22]). $\mathrm{PCSP}(\mathbf{LO}^r_k, \mathbf{LO}^r_{k+1})$ is NP-hard for $r \geq 5$ and $k \geq 2$.

We will present the proof from [NŽ22] in Section 4.3.

This was then quickly generalised to the following result.

Theorem 4.7 ([NŽ23b]). $\mathrm{PCSP}(\mathbf{LO}^r_k, \mathbf{LO}^r_\ell)$ is NP-hard for $r \geq \ell - k + 4$ and $2 \leq k \leq \ell$.

We will prove this fact in Section 4.4.

Since the publication of these results, two further results have appeared. The first uses *Algebraic Topology* in the style of [KOWŽ23] to prove hardness.

Theorem 4.8 ([FNOTW24]). $\mathrm{PCSP}(\mathbf{LO}^3_3, \mathbf{LO}^3_4)$ is NP-hard.

Next, using topology in the style of [DRS05] (i.e. as a black box to show lower bounds on the chromatic number of Kneser Graphs [Lov78]), one can prove a very strong hardness result.

Theorem 4.9 ([NVWŽ25]). $\mathrm{PCSP}(\mathbf{LO}_k^r, \mathbf{LO}_\ell^r)$ is NP-hard for $r \geq 4$ and $3 \leq k \leq \ell$.

Informally, this theorem does "twice as much" as Theorem 4.7. Nonetheless, Theorem 4.7 is not subsumed, since it proves results even for the case $k = 2$. Furthermore, hardness for $k = 2, \ell = 3, r = 3$ has been announced [KV25]; however this result also does not subsume ours (it does not imply, e.g., the NP-hardness of $\mathrm{PCSP}(\mathbf{LO}_2^6, \mathbf{LO}_4^6)$. To see a summary of the known and open cases, see Table 4.1.

To conclude this introduction, we remark on the central role LO colouring seems to be taking in the field of PCSPs. The problem has (i) interesting algorithmic content, (ii) connections to the algebraic approach for CSPs, (iii) connections to topology, (iv) easy to state open problems. The only other problem with a similar range seems to be approximate graph colouring, the "giant" in the field.

4.1.1. Algebraic preliminaries

In this section we restate some of the algebraic theory of PCSPs [BBKO21], but specialised to the case of linearly ordered colouring.

Definition 4.10. An n-ary polymorphism of $(\mathbf{LO}_k^r, \mathbf{LO}_\ell^r)$ is a function $f : \{0, \ldots, k-1\}^n \to \{0, \ldots, \ell - 1\}$ so that, given any matrix $A \in \{0, \ldots, k-1\}^{r \times n}$ where each column has a unique maximum element, whose rows are $a_1, \ldots, a_r \in \{0, \ldots, k-1\}^n$ — there exists a unique maximum element in the sequence $f(a_1), \ldots, f(a_r)$. For such an f and a function $\pi : [n] \to [m]$, the minor of f by π, denoted by f^π, is given by

$$f^\pi(x_1, \ldots, x_m) = f(x_{\pi(1)}, \ldots, x_{\pi(n)}).$$

In particular this function is an m-ary polymorphism of $(\mathbf{LO}_k^r, \mathbf{LO}_\ell^r)$.
For the special case $k = 2$, we observe that $\{0,1\}^n$ is isomorphic to $2^{[n]}$, by taking a set to its characteristic vector and back. Thus, we can also say that an n-ary polymorphism of $(\mathbf{LO}_2^r, \mathbf{LO}_\ell^r)$ is a function $f : 2^{[n]} \to \{0, \ldots, \ell - 1\}$, such that for any partition $A_1, \ldots, A_r$ of $[n]$, we have that the sequence $f(A_1), \ldots, f(A_r)$ contains a unique maximum. Furthermore, the minor of f by π is given by

$$f^\pi = f \circ \pi^{-1},$$

where

$$\pi^{-1}(S) = \{x \mid \pi(x) \in S\}.$$

We use the notation $\mathcal{M}_{k,\ell}^r$ for $\mathrm{Pol}(\mathbf{LO}_k^r, \mathbf{LO}_\ell^r)$, eliding the r for the case $r = 3$. Hence, e.g. $\mathcal{M}_{2,100} = \mathcal{M}_{2,100}^3 = \mathrm{Pol}(\mathbf{LO}_2^3, \mathbf{LO}_{100}^3)$.

We next define the free structure for the special case of $\mathbf{LO}_2^r$.

Case	Complexity
$r \geq 4, 3 \leq k \leq \ell$	NP-hard [NVWŽ25].
$r \geq 3, r \neq 4, \ell = k+1, k \geq 2$	NP-hard, by [KV25] with Theorem 2.11 and Lemma 4.15. Subsumes [FNOTW24].
$k = 2, \ell \geq k, r \geq \ell + 2$	NP-hard [NŽ23b].
All other cases	Open. The (computationally) hardest open cases are $(r,k,\ell) = (4,2,3)$ and $(r,k,\ell) = (3,2,4)$.

Table 4.1.: Current hardness results for $\mathrm{PCSP}(\mathbf{LO}^r_k, \mathbf{LO}^r_\ell)$.

Definition 4.11. Given a minion $\mathcal{M}$, the *free structure of* $\mathcal{M}$ *generated by* $\mathbf{LO}^r_2$, denoted by $\mathbf{F}_{\mathcal{M}}(\mathbf{LO}^r_2)$, is a relational structure with domain $\mathcal{M}^{(2)}$, and with one r-ary relation. A tuple $(f_1, \dots, f_r)$ belongs to that relation if and only if there exists some $g \in \mathcal{M}^{(r)}$ such that

$$\begin{array}{ccccccc} f_1(x,y) & = & g(y, & x, & \cdots & x) \\ f_2(x,y) & = & g(x, & y, & \cdots & x) \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ f_r(x,y) & = & g(x, & x, & \cdots & y) \end{array}$$

The key fact about free structures is the following.

Theorem 4.12 (Restatement and specialisation of Theorem 2.15). $\mathcal{M} \to \mathcal{M}^r_{2,k}$ if and only if $\mathbf{F}_{\mathcal{M}}(\mathbf{LO}^r_2) \to \mathbf{LO}^r_k$.
Equivalently, $\mathcal{M} \to \mathcal{M}^r_{2,k}$ if and only if there exists an assignment $\omega : \mathcal{M}^{(2)} \to \{0, \dots, k-1\}$ such that, for any $g \in \mathcal{M}^{(r)}$ and $f_1, \dots, f_r$ taken as in Definition 4.11, the sequence $\omega(f_1), \omega(f_2), \dots, \omega(f_r)$ contains a unique maximum.

Recall also the important fact, stated in the preliminaries but restated here.

Theorem 2.11 ([BBKO21, Theorem 3.1]). If $\mathrm{Pol}(\mathbf{A}, \mathbf{B}) \to \mathrm{Pol}(\mathbf{A}', \mathbf{B}')$, then $\mathrm{PCSP}(\mathbf{A}', \mathbf{B}')$ reduces to $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ in logarithmic space.

4.2. Equivalence of Conjecture 4.1 and Conjecture 4.4

In this section we prove Theorem 4.5. We can see this section as a warm up for applying the theory from Section 4.1.1. We first begin with a general characterisation of which 3-uniform hypergraphs are LO colourable (with any finite number of colours). An analogous characterisation for graphs would be "a graph G can be coloured (with some number of colours) if and only if G admits no loops". For LO colouring, the characterisation is somewhat more complicated.

Lemma 4.13. A finite 3-uniform hypergraph $\mathbf{H} = (V, E)$ can be LO coloured with a finite number of colours if and only if there do not exist vertices $v_0, \dots, v_{k-1}, v_k = v_0$ such that $(v_{i-1}, v_i, v_i) \in E$ for all $i \in [k]$.

Proof. First, observe that if such vertices exist, no LO colouring is possible. This is because any prospective LO colouring $c : V \to \mathbb{N}$ must have $c(v_k) < c(v_{k-1}) < \cdots < c(v_0) = c(v_k)$, which is impossible. We must now prove that if no such vertices exist, then we can find an LO colouring. To do this, construct a graph G on the vertex set V, where (u, v) is an edge if and only if $(u, u, v) \in E$. The graph G is acyclic (since a cycle in G is gives rise to precisely the configuration of vertices that we assumed does not exist), so it has a topological sort — suppose that $u_1, \ldots, u_n$ is a topological sort of the vertices of G. We claim that $c(u_i) = i$ is an LO colouring of $\mathbf{H}$. Indeed, for any edge $(u, v, w) \in E$ where u, v, w are all distinct, the colours given to u, v, w are also distinct, and hence the edge is properly LO coloured. On the other hand, for any edge $(u, u, v) \in E$, there is an edge (u, v) in G, and hence $c(u) < c(v)$ as required. □

The existence of such a characterisation is the fundamental reason why we can create a simple condition to express "hardness by reducing from approximate LO colouring". Something similar happens in the case of graph colouring — in that case, a problem is hard by reducing from approximate graph colouring if and only if it does not admit a Siggers polymorphism [BBKO21], i.e. a 6-ary polymorphism f that satisfies $f(a, a, b, b, c, c) = f(b, c, a, c, a, b)$.

The proof of Theorem 4.5 rests on the following lemma.

Lemma 4.14. Consider some *finite* minion $\mathcal{M}$. $\mathcal{M}$ fails to admit daisy chain terms if and only if $\mathcal{M} \to \mathcal{M}_{2,k}$ for some finite k.

Proof. We will prove the equivalence backwards. First, note that by Theorem 4.12, $\mathcal{M} \to \mathcal{M}_{2,k}$ for some k if and only if $\mathbf{F}_{\mathcal{M}}(\mathbf{LO}_2^3)$ is LO colourable with a finite number of colours. Next — as $\mathcal{M}$ is finite, and hence $\mathbf{F}_{\mathcal{M}}(\mathbf{LO}_2^3)$ is finite — by Lemma 4.13 $\mathbf{F}_{\mathcal{M}}(\mathbf{LO}_2^3)$ is LO colourable with a finite number of colours if and only if it does not admit a set of vertices $f_0, \ldots, f_{\ell-1}, f_\ell = f_0$ such that (f_{i-1}, f_i, f_i) is an edge of $\mathbf{F}_{\mathcal{M}}(\mathbf{LO}_2^3)$ for all $i \in [\ell]$.

We must now show that this final condition is equivalent to $\mathcal{M}$ admitting daisy chain terms. This is really a matter of unpacking definitions. This condition is satisfied if and only if there exist elements $g_1, \ldots, g_\ell \in \mathcal{M}^{(3)}$ so that $g_i(y, x, x) = g_i(x, x, y) = f_i(x, y)$ and $g_i(x, y, x) = f_{i-1}(x, y)$. But putting all of these facts together, this is equivalent to there existing terms $g_0, \ldots, g_{\ell-1}, g_\ell = g_0 \in \mathcal{M}^{(3)}$ with $g_i(y, x, x) = g_i(x, x, y) = f_i = g_{i+1}(x, y, x)$ — which is identical to the definition of daisy chain terms. □

We will also need the following fact, which is easy to establish, and which we will use in future sections.

Lemma 4.15. For any $2 \leq k \leq \ell$, $r \geq 3$, $\mathcal{M}^r_{k+1,\ell+1} \to \mathcal{M}^r_{k,\ell}$.

Proof. We will establish a stronger claim, that $\mathcal{M}^r_{k+1,\ell+1} \subseteq \mathcal{M}^r_{k,\ell}$. Indeed consider any $f \in (\mathcal{M}^r_{k+1,\ell+1})^{(n)}$, and consider any matrix of inputs $A \in \{0, \ldots, k-1\}^{r \times n}$, with rows $a_1, \ldots, a_r \in \{0, \ldots, k-1\}^n$, whose columns all have a unique maximum. We must show that $f(a_1), \ldots, f(a_r) \in \{0, \ldots, \ell-1\}$ and that the sequence $f(a_1), \ldots, f(a_r)$ has a unique

maximum. The latter claim is immediate, as f is a polymorphism of $(\mathbf{LO}^r_{k+1}, \mathbf{LO}^r_{\ell+1})$. So we only need to show that $f(a_i) \in \{0, \dots, \ell - 1\}$.

Indeed consider any sequence $x_1, \dots, x_n \in \{0, \dots, k-1\}$. We will show that $f(x_1, \dots, x_n) \in \{0, \dots, \ell - 1\}$ i.e. $f(x_1, \dots, x_n) < \ell$. Consider the matrix that has $r - 1$ rows equal to $x_1, \dots, x_n$, and where the last row contains $x_1 + 1, \dots, x_n + 1$. All of the elements of the matrix have values between 0 and k, and each column of the matrix has a unique maximum — hence, as f is a polymorphism, the sequence containing $r - 1$ copies of $f(x_1, \dots, x_n)$ and one copy of $f(x_1+1, \dots, x_n+1)$ has a unique maximum. But this means that $f(x_1, \dots, x_n) < f(x_1 + 1, \dots, x_n + 1) \leq \ell$, as required. □

Proof of Theorem 4.5. First, suppose that Conjecture 4.1 is true. Then consider any Pol(**A**, **B**) that does not admit daisy chain terms. By Lemma 4.14, for some finite k we have $\mathcal{M} \to \mathcal{M}_{2,k}$, and hence by Theorem 2.11, PCSP($\mathbf{LO}^3_2, \mathbf{LO}^3_k$) reduces to PCSP(**A**, **B**). By assumption the former problem is NP-hard, and hence the latter one is as well.

Conversely suppose that Conjecture 4.4 is true, and fix $2 \leq k < \ell$. By Lemma 4.15,

$$\mathcal{M}_{k,\ell} \to \mathcal{M}_{k-1,\ell-1} \to \cdots \to \mathcal{M}_{2,\ell-k+2}.$$

Hence PCSP($\mathbf{LO}^3_2, \mathbf{LO}^3_{\ell-k+2}$) reduces to PCSP($\mathbf{LO}^3_k, \mathbf{LO}^3_\ell$) — and thus it is sufficient to show hardness in the case $k = 2$. In this case, by Lemma 4.14 (and as, obviously, $\mathcal{M}_{2,\ell} \to \mathcal{M}_{2,\ell}$), $\mathcal{M}_{2,\ell}$ fails to admit daisy chain terms, and by assumption PCSP($\mathbf{LO}^3_2, \mathbf{LO}^3_\ell$) is NP-hard. □

4.3. First hardness result

In this chapter, we present the proof from [NŽ22] for Theorem 4.6, which we now restate

Theorem 4.6 ([NŽ22]). PCSP($\mathbf{LO}^r_k, \mathbf{LO}^r_{k+1}$) is NP-hard for $r \geq 5$ and $k \geq 2$.

The general strategy of the proof follows three steps: (i) show that PCSP($\mathbf{LO}^r_2, \mathbf{LO}^r_3$) is NP-hard for some large enough r; (ii) show that PCSP($\mathbf{LO}^r_2, \mathbf{LO}^r_3$) is no easier than PCSP($\mathbf{LO}^{r+1}_2, \mathbf{LO}^{r+1}_3$) for $r \geq 5$; and (iii) use Lemma 4.15 to complete the proof.

Let us begin with the first step: showing that PCSP($\mathbf{LO}^r_2, \mathbf{LO}^r_3$) is NP-hard for large enough r. How shall we do this? We observe that CSP($\mathbf{LO}^r_2$) is actually just the *1-in-r exact hitting set* problem: given an r-uniform hypergraph, in 1-in-r exact hitting set we are asked to find a subset of vertices that intersects each edge in exactly one vertex. Naturally, PCSP($\mathbf{LO}^r_2, \mathbf{LO}^r_3$) can be seen as an approximation version of this problem; so we would expect that we could convert hardness of approximation for 1-in-r exact hitting set to hardness of PCSP($\mathbf{LO}^r_2, \mathbf{LO}^r_3$). This is indeed the case, as we now show.

Definition 4.16. The α-gap 1-in-r exact hitting set problem is defined as follows. We are given an r-uniform hypergraph **H**. We define the value of that hypergraph as the maximum ρ such that there exists a subset of the vertices of **H** that intersects a ρ-fraction of the edges of **H** in exactly 1 vertex. We are then asked to output YES if the value of **H** is 1, and NO if it is not even α.

Theorem 4.17 ([GT05, Theorem 10, Theorem 12, Lemma 13]). For any ε, for all large enough r, the $(1/(e-\varepsilon))$-gap 1-in-r exact hitting set problem is NP-hard, where e is the base of the natural logarithm.

Corollary 4.18. $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_3^r)$ is NP-hard for all large enough r.

Proof. By Theorem 4.17, the $(1/2)$-gap 1-in-r exact hitting set problem is NP-hard for some large r. For this r we will prove that $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_3^r)$ is NP-hard, by reducing from $(1/2)$-gap 1-in-r exact hitting set. The reduction does not change the input. The soundness of the reduction is immediate (the YES instances of $(1/2)$-gap 1-in-r exact hitting set are identical by definition to those of $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_3^r)$). So we focus on the completeness: Suppose that $\mathbf{H}$ is a 3-uniform hypergraph that admits an LO 3-colouring c. We must show that there exists a subset of vertices of $\mathbf{H}$ that intersects at least half of the edges of $\mathbf{H}$ in exactly one vertex. Consider any edge (x, y, z) of $\mathbf{H}$, and look at the values $c(x), c(y), c(z)$. The maximum of these values is unique, and it must be 1 or 2. It then follows that one of the sets $c^{-1}(1)$ or $c^{-1}(2)$ intersects at least half of the edges in $\mathbf{H}$ in exactly one vertex, as required. □

Having found that $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_3^r)$ is NP-hard for all large enough values of r, we now need to reduce this value step-by-step.

Lemma 4.19. For $r \geq 5$, $\mathcal{M}_{2,3}^r \to \mathcal{M}_{2,3}^{r+1}$.

Proof. We use Theorem 4.12. Take the assignment $\omega : (\mathcal{M}_{2,3}^r)^{(2)} \to \{0, 1, 2\}$ given by $\omega(f) = f(0, 1)$. With this choice, it suffices to show that for any $f \in (\mathcal{M}_{2,3}^r)^{(r+1)}$, among the sequence $f\{1\}, \ldots, f\{r+1\}$ there exists a unique maximum. Recall that the fact that f is a polymorphism is equivalent to the following: For every partition $A_1, \ldots, A_r$ of $[r+1]$, the sequence $f(A_1), \ldots, f(A_r)$ has a unique maximum.

Without loss of generality suppose that $f\{1\} \geq \cdots \geq f\{r+1\}$. We want to show that $f\{1\} > f\{2\}$ — thus suppose that $f\{1\} = f\{2\}$ for contradiction. There are 3 cases depending on $f\{1\}$.

$f\{1\} = 2$. Consider the partition

$$\{1\}, \{2\}, \{3, \ldots, r+1\}, \varnothing, \ldots, \varnothing.$$

This is a valid partition as $r \geq 3$. We have that $f\{1\} = f\{2\} = 2$, and thus there cannot be a unique maximum among

$$f\{1\}, f\{2\}, f\{3, \ldots, r+1\}, f(\varnothing), \ldots, f(\varnothing),$$

which is a contradiction.

$f\{1\} = 1$. In this case $f\{1\}, \dots, f\{r+1\} \leq 1$. Consider the partitions

$$\{1\}, \{2\}, \{3,4\}, \{5\}, \dots, \{r+1\};$$
$$\{1\}, \{2\}, \{3\}, \{4\}, \{5,6\}, \{7\}, \dots, \{r+1\}.$$

These are valid partitions since clearly they contain r sets and $r+1 \geq 6$. Since $f\{1\} = f\{2\} = 1$ and $f\{i\} \leq 1$ for $i \in [r+1]$, the unique maximum must come from $f\{3,4\}$ and $f\{5,6\}$ and these values must both be equal to 2. But this is impossible: consider the partition

$$\{3,4\}, \{5,6\}, \{1,2,7,8,\dots,r+1\}, \varnothing, \dots, \varnothing.$$

Clearly there is no unique maximum, as $f\{3,4\} = f\{5,6\} = 2$.

$f\{1\} = 0$. In this case $f\{1\} = \cdots = f\{r+1\} = 0$. Consider the partitions

$$\{1,2\}, \{3\}, \dots, \{r+1\};$$
$$\{1\}, \{2\}\{3,4\}, \{5\}, \dots, \{r+1\};$$
$$\{1\}, \{2\}, \{3\}, \{4\}, \{5,6\}, \{7\}, \dots, \{r+1\}.$$

These are valid partitions since they have r sets and $r+1 \geq 6$. The unique maximum cannot come from $f\{i\} = 0$, so it must come from among $f\{1,2\}, f\{3,4\}, f\{5,6\}$. Since the unique maximum is taken from the set $\{1,2\}$, by the pigeonhole principle two of these values are equal — without loss of generality suppose $f\{1,2\} = f\{3,4\}$. If $f\{1,2\} = f\{3,4\} = 2$, then we get a contradiction exactly as in the previous case. Thus suppose $f\{1,2\} = f\{3,4\} = 1$. Consider the partition

$$\{1,2\}, \{3,4\}, \{5\}, \dots, \{r+1\}, \varnothing.$$

This is a valid partition since clearly it contains r sets, and $r+1 \geq 4$. Since $f\{1,2\} = f\{3,4\} = 1$ and $f\{i\} = 0$ for $i \in [r+1]$, we have that $f(\varnothing) = 2$ — which is impossible, due to the partition

$$\{1, \dots, r+1\}, \varnothing, \dots, \varnothing,$$

(since of course $r \geq 3$ and thus $\varnothing$ appears at least twice in the partition).

Thus in all cases we get a contradiction, and the result follows. □

Now we can prove Theorem 4.6.

Proof of Theorem 4.6. We first prove that $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_3^r)$ is NP-hard for $r \geq 5$. By Corollary 4.18, there exists some $R \in \mathbb{N}$ so that, for all $r \geq R$, we have that $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_3^r)$ is NP-hard. If $R \leq 5$ then there is nothing to prove, so suppose $R > 5$. For any r such that $5 \leq r < R$, observe that, by Lemma 4.19, we have that

$$\mathcal{M}_{2,3}^r \to \cdots \to \mathcal{M}_{2,3}^R,$$

and hence $\mathcal{M}_{2,3}^r \to \mathcal{M}_{2,3}^R$. Thus by Theorem 2.11, it follows that $\mathrm{PCSP}(\mathbf{LO}_2^R, \mathbf{LO}_3^R)$ reduces to $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_3^r)$, and the latter problem is NP-hard.

Now, we need to extend this result to $\mathrm{PCSP}(\mathbf{LO}_k^r, \mathbf{LO}_{k+1}^r)$. This is a simple matter of applying Lemma 4.15 enough times:

$$\mathcal{M}_{k,k+1}^r \to \mathcal{M}_{k-1,k}^r \to \cdots \mathcal{M}_{2,3}^r.$$

Hence $\mathcal{M}_{k,k+1}^r \to \mathcal{M}_{2,3}^r$ and $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_3^r)$ reduces to $\mathrm{PCSP}(\mathbf{LO}_k^r, \mathbf{LO}_{k+1}^r)$. The latter problem is thus NP-hard as required. □

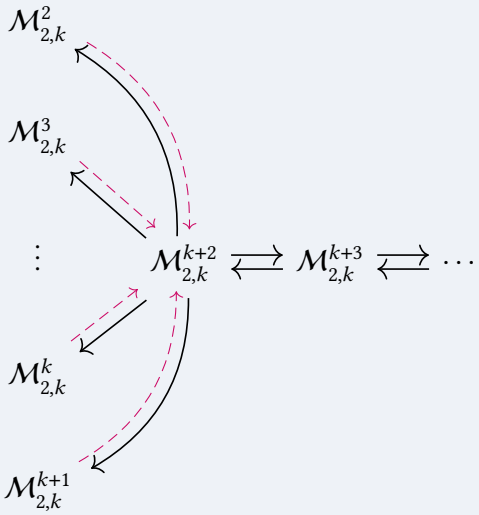
4.4. Second hardness result

In this section, we prove the full hardness result of [NŽ23b], namely Theorem 4.7, which we restate here.

Theorem 4.7 ([NŽ23b]). $\mathrm{PCSP}(\mathbf{LO}_k^r, \mathbf{LO}_\ell^r)$ is NP-hard for $r \geq \ell - k + 4$ and $2 \leq k \leq \ell$.

Indeed, we will also give a better description of the inter-relations between the minions $\mathcal{M}_{2,k}^r$, as described below.

Proposition 4.20. For each $k \geq 3$, the following minion homomorphisms exist between $\mathcal{M}_{2,k}^r$ for various values of r. A dashed arrow indicates the non-existence of a homomorphism.



In particular, for $r \geq k + 2$ and any $r' \geq 2$, we have $\mathcal{M}_{2,k}^r \to \mathcal{M}_{2,k}^{r'}$.

This will be proved in Section 4.4.1 As before, we will initially prove that, for each k, there exists some large r such that $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_k^r)$ is NP-hard.

Proposition 4.21. For each $k \geq 3$, there exists some $r \geq 2$ such that $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_k^r)$ is NP-hard.

This will be proved in Section 4.4.2. Theorem 4.7 now follows as a simple corollary.

Proof of Theorem 4.7. Fix $2 \leq k \leq \ell$ and $r \geq \ell - k + 4$. If $k = \ell$, then $\mathrm{PCSP}(\mathbf{LO}_k^r, \mathbf{LO}_\ell^r) = \mathrm{CSP}(\mathbf{LO}_k^r)$, to which $\mathrm{CSP}(\mathbf{LO}_2^r)$ reduces by applying Lemma 4.15 repeatedly. But $\mathrm{CSP}(\mathbf{LO}_2^r)$ is merely 1-in-r SAT, which is well-known to be NP-hard (for $r \geq 3$) [Sch78]. Thus assume $k < \ell$.

By Proposition 4.21 take r' such that $\mathrm{PCSP}(\mathbf{LO}_2^{r'}, \mathbf{LO}_{\ell-k+2}^{r'})$ is NP-hard. Observe that by Proposition 4.20 and Lemma 4.15, since $r \geq (\ell - k + 2) + 2$,

$$\mathcal{M}_{k,\ell}^r \to \mathcal{M}_{k-1,\ell-1}^r \to \cdots \to \mathcal{M}_{2,\ell-k+2}^r \to \mathcal{M}_{2,\ell-k+2}^{r'},$$

and hence $\mathrm{PCSP}(\mathbf{LO}_2^{r'}, \mathbf{LO}_{\ell-k+2}^{r'})$ — which is NP-hard — reduces to $\mathrm{PCSP}(\mathbf{LO}_k^r, \mathbf{LO}_\ell^r)$, whence the result. □

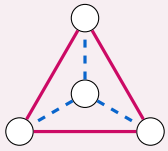
4.4.1. Minion relations

In this chapter we prove Proposition 4.20. Parts of this proof can be seen as generalisations of the proof Lemma 4.19. The main new ingredient is a connection with a new combinatorial object: the (k, ℓ)-edge co-colouring.

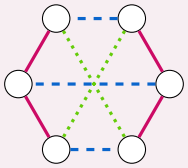
Definition 4.22. A (k, ℓ)-*edge co-colouring* of a graph $G = (V, E)$ is an assignment of k colours to the edges of G, such that any ℓ disjoint edges of G are not all assigned the same colour. Equivalently, no colour class contains ℓ disjoint edges. Two edges are considered disjoint if they do not share any endpoints.

The "co-" prefix indicates that *disjoint* objects are assigned different colours, as opposed to a normal colouring where *incident* objects are assigned different colours.

Example 4.23. The following is a $(2, 2)$-edge co-colouring of K_4. (In what follows edges with different colours will be full/dashed/dotted.)



The following is *not* a $(3, 3)$-edge co-colouring, due to the horizontal edges.



We first prove a result about graphs without disjoint edges.

Lemma 4.24. Suppose G is a graph without a pair of disjoint edges. Then the edges in G form either a triangle or a star.

Proof. This is easy to see via case analysis. However, we will show a proof that is an application of the *Hilton-Milner theorem* [HM67], since we find it an interesting application of this classic area of combinatorics. Recall that an intersecting family is a family of sets which intersect pairwise. A graph whose edges intersect pairwise is identical to an intersecting family $\mathcal{F} \subseteq \binom{V}{2}$.

Recall that the *Erdös-Ko-Rado Theorem* [EKR61] says that an intersecting family $\mathcal{F} \subseteq \binom{[n]}{k}$ has size at most $\binom{n-1}{k-1}$ (provided $2k \leq n$) — this is the same as the size of the "trivial" intersecting family that contains all sets $S \in \binom{[n]}{k}$ where $1 \in S$. The Hilton-Milner theorem bounds the size of the largest *non-trivial* intersecting family; the bound is given by $1 + \binom{n-1}{k-1} - \binom{n-k-1}{k-1}$, again provided $2k \leq n$.

Observe that for $k = 2$, the trivial intersecting families correspond exactly to stars. By the Hilton-Milner theorem, the non-trivial families have size at most

$$1 + \binom{n-1}{1} - \binom{n-2-1}{1} = 1 + (n-1) - (n-3) = 3.$$

Hence, by considering the subgraph induced by its edges, any graph with 4 or more vertices whose edges intersect pairwise is either a star (i.e. trivial), or has at most 3 edges (if it is non-trivial). The 3 edges cannot be disconnected, and cannot form a path of length 3 — they must form either a star or a triangle. If the graph has 3 or fewer vertices then it is either a star or a triangle. □

Next, we give some bounds for $(k, 2)$- and $(k, 3)$-edge co-colourings of cliques.

Lemma 4.25. For $n \geq 3$, K_n admits an $(n-2, 2)$-edge co-colouring, but not an $(n-3, 2)$-edge co-colouring. Furthermore, for $k \geq 3$, K_{10k} does not admit a $(k, 3)$-edge co-colouring.

(These results also follow from the chromatic numbers of Kneser (hyper)graphs [Lov78; AFL86] — indeed, an edge co-colouring of the clique is equivalently a colouring of a Kneser hypergraph!)

Proof. We prove the three required facts separately.

K_n admits an $(n-2, 2)$-edge co-colouring. Suppose that the vertex set of K_n is labelled by $1, \ldots, n$. Assign edge (i, j) colour $\max\{i, j, 3\}$. Observe that this assignment indeed uses $n-2$ colours, namely $3, \ldots, n$. Next, consider the colour classes in this assignment. The colour class for 3 is a triangle $1-2-3-1$; the colour class for $c > 3$ is a star with edges $1-c, 2-c, \ldots, (c-1)-c$. Thus, no colour class has 2 disjoint edges contained within it. Hence this is an $(n-2, 2)$-edge co-colouring.

K_n does not admit a $(n-3,2)$-edge co-colouring. Suppose K_n is the minimal counterexample for this claim. Suppose that there exists some colour class c whose edges form a star of form $x - y_1, \ldots, x - y_k$. Consider the colouring induced on $K_n - x$: since removing x eliminates all edges with colour c, this forms an $(n-4,2)$-edge co-colouring of $K_n - x$. This is impossible by minimality of K_n, so the edges of each colour class are not stars. By Lemma 4.24, each colour class forms a triangle. Thus there are at most $3(n-3)$ edges (three for each colour). Since K_n has $\binom{n}{2}$ edges, we must have

$$\binom{n}{2} \leq 3(n-3),$$

or equivalently

$$n^2 - 7n + 18 = \left(n - \frac{7}{2}\right)^2 + \frac{23}{4} \leq 0,$$

which visibly has no solutions. This contradiction implies our conclusion.

K_{10k} does not admit a $(k,3)$-edge co-colouring. Suppose such a colouring exists. There are $\binom{10k}{2} \geq 45k^2$ edges, and k colours — hence there is a colour class c with at least $45k$ edges. Select an edge e from this colour class, and remove the at most $20k$ edges not disjoint from e from the graph. There are still $25k$ edges left in c, so select e' from among these edges; after removing the edges not disjoint with e', there will still be $5k$ edges left in c. Select e'' from among these edges. We have thus found three disjoint edges all with the same colour, contradicting the fact that the colouring is a $(k,3)$-edge co-colouring. □

We now prove a few facts about the minions $\mathcal{M}^r_{2,k}$ which will help us. In all that follows, it will be useful to create a compact notation for partitions of a set $[n]$. If we write, say, the partition

$$\{x,y\}, \{x',y'\}, \{1\}, \ldots, \overline{\{x\}, \{y\}, \{x'\}, \{y'\}}, \ldots, \{n\},$$

we mean the partition that contains $\{x,y\}$, $\{x',y'\}$, and all singleton sets except the sets $\{x\}, \{y\}, \{x'\}, \{y'\}$; and the meaning of

$$f\{x,y\}, f\{x',y'\}, f\{1\}, \ldots, \overline{f\{x\}, f\{y\}, f\{x'\}, f\{y'\}}, \ldots, f\{n\}$$

is similar.

Lemma 4.26. For $k \geq 2, r \geq 10k+1$, we have $\mathcal{M}^r_{2,k} \to \mathcal{M}^{r+1}_{2,k}$.

Proof. We proceed similarly to Lemma 4.19, taking the same assignment ω. Thus we need to prove that for any $f \in (\mathcal{M}^r_{2,k})^{(r+1)}$, there exists a unique maximum among the sequence $f\{1\}, \ldots, f\{r+1\}$. Construct a graph G on $[r+1]$ which, for $1 \leq x < y \leq r+1$, contains the edge (x,y) if $f\{x,y\}$ is the unique maximum in the sequence

$$f\{x,y\}, f\{1\}, \ldots, \overline{f\{x\}, f\{y\}}, \ldots, f\{r+1\}.$$

Colour the edge (x,y) with $f\{x,y\}$. Observe that $f\{x,y\} > f\{i\}$ for $i \in [r+1] - x - y$. We claim that this is a $(k,3)$-edge co-colouring. It clearly uses k colours at most. Now, consider

three disjoint edges (x, y), (x', y'), (x'', y''). Consider the partition

$$\{x, y\}, \{x', y'\}, \{x'', y''\}, \{1\}, \ldots, \overline{f\{x\}, f\{y\}, f\{x'\}, f\{y'\}, f\{x''\}, f\{y''\}}, \ldots, f\{r+1\}, \emptyset, \emptyset, \emptyset.$$

Observe that the unique maximum image through f must be given by one of

$$f\{x, y\}, f\{x', y'\}, f\{x'', y''\}.$$

This is because, for each $i \in [r+1] - x - y - x' - y' - x'' - y''$, any edge $(x_0, y_0) \in \{(x, y), (x', y'), (x'', y'')\}$ that is not incident to i has $f\{x_0, y_0\} > f\{i\}$ — and furthermore $\emptyset$ is repeated multiple times. Hence, in particular $f\{x, y\}, f\{x', y'\}, f\{x'', y''\}$ are not all equal.

Suppose for contradiction that no unique maximum exists among $f\{1\}, \ldots, f\{r+1\}$. Without loss of generality, suppose $f\{1\} \leq \cdots \leq f\{r\} = f\{r+1\}$. For any $x, y \in [r-1]$, considering the partition containing $\{x, y\}$ and all singletons $\{i\}$ for $i \in [r+1] - x - y$, we see that $f\{x, y\}$ is the unique maximum. Hence (x, y) is an edge in G, and thus G contains K_{r-1} as a subgraph. Thus G contains K_{10k} as a subgraph. This is impossible, since G has a $(k, 3)$-edge co-colouring (cf. Lemma 4.25). □

Lemma 4.27. For $k \geq 2, r \geq k+2$, $\mathcal{M}^r_{2,k} \to \mathcal{M}^{r+2}_{2,k}$.

Proof. The proof is again similar to Lemmas 4.19 and 4.26 — we prove that for $f \in (\mathcal{M}^r_{2,k})^{(r+2)}$ there exists a unique maximum among $f\{1\}, \ldots, f\{r+2\}$. Suppose that this is not the case; hence, without loss of generality, suppose that $f\{1\} \leq \cdots \leq f\{r+1\} = f\{r+2\}$. Depending on the value of $f\{r+2\}$, we have two cases.

- **$f\{r+2\} = 0$.** Construct the complete graph on $[r+2]$, and colour edge $\{x, y\}$ with $f\{x, y\}$. We claim that this colouring is a $(k, 2)$-edge co-colouring. Obviously we use at most k colours — so consider disjoint edges $\{x, y\}, \{z, t\}$. We need to show $f\{x, y\} \neq f\{z, t\}$. Consider the partition

  $$\{x, y\}, \{z, t\}, \{1\}, \ldots, \overline{\{x\}, \{y\}, \{z\}, \{t\}}, \ldots, \{r+2\},$$

  and recall that $f\{1\} = \cdots = f\{r+2\} = 0$ by assumption. Hence the unique maximum image through f is given by one of $f\{x, y\}, f\{z, t\}$, and in particular these two values are distinct. This contradicts Lemma 4.25: any $k+3 \leq r+2$ vertex subgraph of G is a counterexample to Lemma 4.25.

- **$f\{r+2\} > 0$.** Construct the complete graph on $[r]$, and colour edge $\{x, y\}$ with

  $$\max(f\{x, y\}, f\{r+2\}).$$

  We claim that this is a $(k-1, 2)$-edge co-colouring. Indeed, we use at most $k-1$ colours — so consider disjoint edges $\{x, y\}, \{z, t\}$. We again show that $\{x, y\}, \{z, t\}$ are assigned distinct colours. Consider same partition as in the first case. The sequence

  $$f\{x, y\}, f\{z, t\}, f\{1\}, \ldots, \overline{f\{x\}, f\{y\}, f\{z\}, f\{t\}}, \ldots, f\{r+2\}$$

must have a unique maximum. But since $f\{1\} \leq \cdots \leq f\{r+1\} = f\{r+2\}$, the unique maximum must be among $f\{x,y\}, f\{z,t\}$ — and furthermore it is *strictly greater than* $f\{r+2\}$. But this implies that

$$\max(f\{x,y\}, f\{r+2\}) \neq \max(f\{z,t\}, f\{r+2\}),$$

as required. Similarly to the first case, this gives us a counterexample to Lemma 4.25.

Noting that we get a contradiction in all cases completes the proof. □

Lemma 4.28. For $k \geq 2, r \geq 2$, $\mathcal{M}_{2,k}^{r+2} \to \mathcal{M}_{2,k}^{r}$.

Proof. We again proceed along the same lines as Lemmas 4.19, 4.26 and 4.27. Consider $f \in (\mathcal{M}_{2,k+2}^{r+2})^{(r)}$. We show that $f\{1\}, \ldots, f\{r\}$ contains a unique maximum. Indeed, consider the partition

$$\{1\}, \ldots, \{r\}, \emptyset, \emptyset.$$

The images of these sets through f must give rise to a unique maximum. It cannot come from $\emptyset$, so it must come from $\{1\}, \ldots, \{r\}$ — which implies our conclusion. □

For the next two lemmata we want to prove that certain minion homomorphisms *do not* exist. We will use the following simple corollary of Theorem 4.12. Recall that an n-weak near unanimity (or n-WNU) in a minion $\mathcal{M}$ is an element $f \in \mathcal{M}^{(n)}$ such that

$$f(y,x,\ldots,x) = f(x,y,x,\ldots,x) = \cdots = f(x,\ldots,x,y).$$

Corollary 4.29. If $\mathcal{M}$ admits a r-WNU, then $\mathcal{M} \not\to \mathcal{M}_{2,k}^{r}$.

Proof. Consider any $\omega : \mathcal{M}^{(2)} \to \{0,\ldots,k-1\}$, and observe that for an r-WNU $g \in \mathcal{M}^{(r)}$ and $f_1, \ldots, f_r \in \mathcal{M}^{(2)}$ taken as in Definition 4.11, by the definition of a WNU we have that $f_1 = \cdots = f_r$. Hence $\omega(f_1) = \cdots = \omega(f_r)$, and thus by Theorem 4.12 $\mathcal{M} \not\to \mathcal{M}_{2,k}^{r}$. □

Lemma 4.30. For $k \geq 3$, $\mathcal{M}_{2,k}^{k} \not\to \mathcal{M}_{2,k}^{k+1}$.

Proof. Define $f : 2^{[k+1]} \to \{0,\ldots,k-1\}$ by

$$f(S) = \begin{cases} 0, & |S| \leq 1, \\ k-1, & |S| \geq k, \\ \max(1, \max_{x \in S} x - 2), & \text{otherwise.} \end{cases}$$

It is easy to see that f is a $(k+1)$-WNU (this is because all sets of size 1 and of size k respectively are given the same value). If we show that $f \in \mathcal{M}_{2,k}^{k}$, then $\mathcal{M}_{2,k}^{k} \not\to \mathcal{M}_{2,k}^{k+1}$ by Corollary 4.29. To see why f is a polymorphism, consider any partition $A_1, \ldots, A_k$ of $[k+1]$. There are now several cases.

1. Suppose that there exists a part of size k or $k+1$. The image of that part is then the unique maximum: it is $k-1$, whereas all other parts are of size at most 1, and hence have image 0.
2. Suppose that there is exactly one part of size at least 2. Then this part gives rise to the unique maximum: its image is at least 1, whereas all other parts are of size at most 1, and hence have image 0.
3. Suppose that there are 2 or more parts of size at least 2 (and hence no parts with size k or $k+1$). Let A be the part among these which maximises $\max_{x\in A} x$. By the pigeonhole principle, $\max_{x\in A} x \geq 4$, and for all other parts B of size at least 2 we have $\max_{x\in B} x < \max_{x\in A} x$. Hence we see that $f(A)$ is the unique maximum: it is greater than 0 since $\max_{x\in A} x \geq 4$ (and thus is greater than the image of all parts of size at most 1), and it is also greater than $f(B)$ for all other parts B.

Hence f is a polymorphism and the conclusion follows. □

Lemma 4.31. For $k \geq 3$, $\mathcal{M}_{2,k}^{k+1} \not\to \mathcal{M}_{2,k}^{k}$.

Proof. We proceed similarly to Lemma 4.30, creating a k-WNU and applying Corollary 4.29. Define $f : 2^{[k]} \to \{0, \ldots, k-1\}$ by

$$f(S) = \begin{cases} 1, & S = \varnothing \\ 0, & |S| = 1 \\ k-1, & |S| \geq k-1 \\ \max(\max_{x\in S} x - 1, 2), & \text{otherwise.} \end{cases}$$

We will prove $f \in \mathcal{M}_{2,k}^{k+1}$ — so consider any partition $A_1, \ldots, A_{k+1}$ of $[k]$. We must again prove that $f(A_1), \ldots, f(A_k)$ admits a unique maximum element. Consider the following cases.

1. If there exists a part of size $k-1$ or k, its image is the unique maximum — all other parts have size 0 or 1, and so their image is 1 or 0.
2. If the partition contains only parts of size 0 and 1, then the unique maximum is given by the image of $\varnothing$ — indeed in this partition $\varnothing$ appears exactly once, and every singleton appears.
3. If there exists exactly one part with size 2 or higher, then this part gives rise to the unique maximum — its image is at least 2, and all the other images are at most 1.
4. If there are two or more parts with size 2 or higher (and hence no parts with size $k-1$ or k), then the part A which maximises $\max_{x\in A} x$ gives rise to the unique maximum image. This can be shown analogously to Lemma 4.30, as here also $\max_{x\in A} x \geq 4$ by the pigeonhole principle.

Hence f is a polymorphism and the conclusion follows. □

With this body of knowledge, we are now ready to prove Proposition 4.20

Proof of Proposition 4.20. First we show that, for $r \geq k+2$, we have $\mathcal{M}^r_{2,k} \rightleftarrows \mathcal{M}^{r+1}_{2,k}$. For any number $R \geq 10k$ with the same parity as r, we have, by applying Lemma 4.27 then Lemma 4.26 then Lemma 4.28, that

$$\mathcal{M}^r_{2,k} \to \mathcal{M}^{r+2}_{2,k} \to \cdots \to \mathcal{M}^R_{2,k} \to \mathcal{M}^{R+1}_{2,k} \to \mathcal{M}^{R-1}_{2,k} \to \cdots \to \mathcal{M}^{r+1}_{2,k},$$
$$\mathcal{M}^{r+1}_{2,k} \to \mathcal{M}^{r+3}_{2,k} \to \cdots \to \mathcal{M}^{R+1}_{2,k} \to \mathcal{M}^{R+2}_{2,k} \to \mathcal{M}^R_{2,k} \to \cdots \to \mathcal{M}^r_{2,k}.$$

Next, note that $\mathcal{M}^{k+2}_{2,k} \to \mathcal{M}^r_{2,k}$ for $r \leq k+1$ follows by applying Lemma 4.28 sufficiently many times to $\mathcal{M}^{k+2}_{2,k}$ and $\mathcal{M}^{k+3}_{2,k} \rightleftarrows \mathcal{M}^{k+2}_{2,k}$.

Finally, we show that $\mathcal{M}^r_{2,k} \not\to \mathcal{M}^{k+2}_{2,k}$ for $r \leq k+1$; suppose that $\mathcal{M}^r_{2,k} \to \mathcal{M}^{k+2}_{2,k}$ for contradiction. Suppose that $R \in \{k, k+1\}$ is chosen to have the same parity as r, and $R' \in \{k, k+1\}$ is chosen to have the opposite parity as r. Then, applying Lemma 4.28, we have

$$\mathcal{M}^R_{2,k} \to \mathcal{M}^{R-2}_{2,k} \to \cdots \to \mathcal{M}^r_{2,k} \to \mathcal{M}^{k+2}_{2,k} \to \mathcal{M}^{R'}_{2,k}.$$

This contradicts one of Lemmas 4.30 and 4.31. □

4.4.2. Hardness for large arity

In this section we prove Proposition 4.21, namely that for every k there exists some large r such that $\mathrm{PCSP}(\mathbf{LO}^r_2, \mathbf{LO}^r_k)$ is NP-hard. We will use the following hardness theorem from the preliminaries.

Theorem 2.23. Fix C, k. There exists some N such that the following holds. Consider some $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$, and let $\mathcal{M} = \mathrm{Pol}(\mathbf{A}, \mathbf{B})$. Cover $\mathcal{M}$ by k sets $\mathcal{M}_1, \ldots, \mathcal{M}_k$. For every $i \in [k]$, suppose there exists a mapping I_i that takes $f \in \mathcal{M}_i$ of *arity* $n \leq N$ to a set $I_i(f) \subseteq [n]$ of size at most C. Suppose that for every $f, g \in \mathcal{M}_i$ *where* $\mathrm{ar}(f) \leq N, \mathrm{ar}(g) \leq N$ and π such that $f \xrightarrow{\pi} g$ we have that $\pi(I_i(f)) \cap I_i(g) \neq \varnothing$. Then $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is NP-hard.

Recall that, for us, $f^\pi = f \circ \pi^{-1}$.

Proof of Proposition 4.21. Fix k and set $C = 1$. Take N as in Theorem 2.23. Set $r = N + 2$ — we will show that $\mathrm{PCSP}(\mathbf{LO}^r_2, \mathbf{LO}^r_k)$ is NP-hard. To this end we need to split $\mathcal{M}^r_{2,k}$ into k different sets $\mathcal{M}_0, \ldots, \mathcal{M}_{k-1}$ so that various properties are satisfied. We define

$$\mathcal{M}_x = \{f \in \mathcal{M}^r_{2,k} \mid \max_{i \in [\mathrm{ar}(f)]} f\{i\} = x\}.$$

In other words, we split $\mathcal{M}^r_{2,k}$ according to the maximum value of a singleton set.

Next, we observe that for $f \in \mathcal{M}_x$ with $\mathrm{ar}(f) \leq N = r - 2$, the partition

$$\{1\}, \ldots, \{\mathrm{ar}(f)\}, \varnothing, \ldots, \varnothing$$

implies that $f\{1\}, \ldots, f\{\mathrm{ar}(f)\}$ contains a unique maximum. We set $I_x(f)$ to be the singleton that gives rise to this unique maximum. Note that clearly $f(I_x(f)) = x$.

We now must prove that, for $f \xrightarrow{\pi} g$, with $\mathrm{ar}(f), \mathrm{ar}(g) \leq N$, if

$$\max_{i \in [\mathrm{ar}(f)]} f\{i\} = \max_{j \in [\mathrm{ar}(g)]} g\{j\} = x,$$

then $\pi(I_x(f)) \cap I_x(g) \neq \varnothing$.

Suppose this is not the case. Then $\pi^{-1}(I_x(g)) \cap I_x(f) = \varnothing$. Observe that

$$f(\pi^{-1}(I_x(g))) = g(I_x(g)) = x,$$

and of course $f(I_x(f)) = x$. Consider the partition of $[\mathrm{ar}(f)]$ into $r \geq \mathrm{ar}(f) + 2$ sets, containing $\pi^{-1}(I_x(g))$ and all singletons disjoint from $\pi^{-1}(I_x(g))$ (including $I_x(f)$), padded with at least 2 copies of $\varnothing$. This partition contradicts f being a polymorphism, since the images do not contain a unique maximum: the singletons all have value at most x, and this value is attained due to $I_x(f)$; $\pi^{-1}(I_x(g))$ has value x as well; and $\varnothing$ appears at least twice. Hence $\pi(I_x(f)) \cap I_x(g) \neq \varnothing$ and the conditions necessary for Theorem 2.23 are fulfilled. Hence $\mathrm{PCSP}(\mathbf{LO}_2^r, \mathbf{LO}_k^r)$ is NP-hard. □

5. Rainbow-Free Structures

In this chapter, we will give complexity classifications for certain *rainbow-free, ternary PCSPs*: PCSPs with one ternary relation free from *rainbow tuples*. (A rainbow tuple is one containing no repeat elements.) This chapter will present results from the following previously published works.

[CKKNŽ24] Lorenzo Ciardo, Marcin Kozik, Andrei A. Krokhin, Tamio-Vesa Nakajima and Stanislav Živný. '1-in-3 vs. Not-All-Equal: Dichotomy of a broken promise'. In: *LICS*. ACM, 2024, 24:1–24:12. DOI: 10.1145/3661814.3662069.

[CKKNŽ25] Lorenzo Ciardo, Marcin Kozik, Andrei A. Krokhin, Tamio-Vesa Nakajima and Stanislav Živný. '1-in-3 vs. Not-All-Equal: Dichotomy of a Broken Promise'. In: *ACM Trans. Comput. Log.* 26.2 (2025), 10:1–10:22. DOI: 10.1145/3719007.

The works were produced in collaboration with my supervisor and the other co-authors: Lorenzo Ciardo, Marcin Kozik, Andrei Krokhin.

5.1. Introduction

In this chapter we deal exclusively with relational structures $\mathbf{A} = (A, R^{\mathbf{A}})$ with one ternary, symmetric relation.

Definition 5.1. A relational structure $\mathbf{A}$ is called *rainbow-free* if every tuple of its relation contains at least two equal elements.

Example 5.2. Consider

$$\begin{aligned}
\textbf{1-in-3} &= (\{0,1\}, \{(1,0,0),(0,1,0),(0,0,1)\}),\\
\textbf{NAE} &= \left(\{0,1\}, \begin{Bmatrix}(1,0,0), & (0,1,0), & (0,0,1),\\ (0,1,1), & (1,0,1), & (1,1,0)\end{Bmatrix}\right),\\
\mathbf{LO}_3^3 &= (\{0,1,2\}, \{(x_1,x_2,x_3) \mid \exists i \cdot \forall j \cdot (i \neq j) \to x_i > x_j\}).
\end{aligned}$$

1-in-3, **NAE** are rainbow free, whereas $\mathbf{LO}_3^3$ is *not*, because it admits the rainbow tuple $(0,1,2)$ (among others).

We note that the rainbow-free property can be seen as a weakened form of Booleanness. A rainbow-free structure "looks Boolean" if all we see is a single tuple in its relation.

The idea of treating this case separately, as a first step towards determining the complexity of all PCSPs with one ternary, symmetric relation, was inspired by Barto, Batistelli and

Berg [BBB21]. Indeed, they manage to classify PCSP(**1-in-3**, **H**) when **H** is rainbow-free and symmetric on a 3-element domain. (They also classified this problem for **H** *with* rainbow tuples, except for the case $\mathbf{H} = \mathbf{LO}_2^3$.) The first major goal of this line of research was to classify all the remaining cases (i.e. when **H** is rainbow-free but on an arbitrary domain) — this goal was achieved in [CKKNŽ24; CKKNŽ25], and appears below as Theorem 5.6.

Why is the promise **1-in-3** interesting? Many PCSP researchers were drawn to it because of the PCSP template (**1-in-3**, **NAE**). This template was the first proved [BBKO21] to not be *finitely tractable*, i.e. tractable because of a homomorphic sandwich of form **1-in-3** $\to$ **C** $\to$ **NAE** where CSP(**C**) is tractable and **C** is finite. This example first showed that PCSPs can be tractable for fundamentally different reasons than CSPs.

Our line of research can be seen as exploring what happens when **NAE** is replaced by some other rainbow-free relational structure in (**1-in-3**, **NAE**). Symmetry invites us to ask: what if we replace **1-in-3** while keeping **NAE**? A full classification (when **1-in-3** is replaced by a rainbow-free, symmetric template yet again) was achieved in [CKKNŽ24; CKKNŽ25] as well, and this appears as Theorem 5.7 below.

While a significant goal is classifying PCSPs with one ternary, symmetric relation, even the midway point given by classifying PCSPs with one ternary, symmetric, rainbow-free relation seems quite difficult. [CKKNŽ24; CKKNŽ25] classified this case as long as the structures were (in an appropriate sense) *bipartite*. The result is given as Theorem 5.8.

Observe that in a rainbow-free relational structure, all the tuples of the relation are of the form (x, x, y), (x, y, x) or (y, x, x). In other words, the tuples are given essentially by ordered pairs. We see therefore that every rainbow-free relational structure with one ternary relation can be encoded by a directed graph; this observation first appeared in [BBB21]. Precisely this observation is what makes the rainbow-free case more amenable to classification: arbitrary templates have much more freedom, whereas the "digraph structure" of these relational structures makes them much more combinatorially well-behaved.

Definition 5.3. Suppose $\mathbf{G} = (V, E)$ is a directed graph. Then the relational structure $\widehat{\mathbf{G}} = (V, E')$ is given by $E' = \bigcup_{(x,y)\in E} \{(y, x, x), (x, y, x), (x, x, y)\}$.

It is easy to see that every relational structure $\widehat{\mathbf{G}}$ must be rainbow-free; furthermore, every symmetric, rainbow-free structure with one ternary relation is of this form.

Example 5.4. Consider the digraphs

$$\mathbf{G}_1 = (\{0\}, \{(0,0)\}), \qquad \mathbf{G}_2 = (\{0,1\}, \{(0,1),(1,0)\}),$$
$$\mathbf{G}_3 = (\{0,1\}, \{(0,1)\}), \qquad \mathbf{G}_4 = (\{0,1,2\}, \{(0,1),(1,2),(2,0)\}).$$

Then we have that $\widehat{\mathbf{G}_1} = (\{0\}, \{(0,0,0)\})$, $\widehat{\mathbf{G}_3} = \mathbf{1\text{-}in\text{-}3}$, $\widehat{\mathbf{G}_2} = \mathbf{NAE}$, and

$$\widehat{\mathbf{G}_4} = \left(\{0,1,2\}, \begin{Bmatrix} (1,0,0), & (0,1,0), & (0,0,1), \\ (2,1,1), & (1,2,1), & (1,1,2), \\ (0,2,2), & (2,0,2), & (2,2,0) \end{Bmatrix}\right)$$
$$= (\{0,1,2\}, \{(x,y,z) \mid x+y+z \equiv 1 \bmod 3\}).$$

As a warm up, let us prove an elementary property of this construction.

Lemma 5.5. For digraphs $\mathbf{G}, \mathbf{H}$, if $\mathbf{G} \to \mathbf{H}$ then $\widehat{\mathbf{G}} \to \widehat{\mathbf{H}}$.

The converse to this implication also holds, but we will not use this fact.

Proof. Suppose $h : \mathbf{G} \to \mathbf{H}$. We show that $h : \widehat{\mathbf{G}} \to \widehat{\mathbf{H}}$. Consider $(a, b, c) \in R^{\widehat{\mathbf{G}}}$. There must be an edge (u, v) of $\mathbf{G}$ such that $(a, b, c) \in \{(u, u, v), (u, v, u), (v, u, u)\}$. As h is a homomorphism, $(h(u), h(v))$ is an edge of $\mathbf{H}$. So by definition

$$(h(a), h(b), h(c)) \in \{(h(u), h(u), h(v)), (h(u), h(v), h(u)), (h(v), h(u), h(u))\} \subseteq R^{\widehat{\mathbf{H}}},$$

as required. □

With this notation in hand, we will now state the results that we will prove in this chapter.

Theorem 5.6. Let $\multimap \to \mathbf{H}$ be any digraph. Then $\mathrm{PCSP}(\widehat{\multimap}, \widehat{\mathbf{H}})$ is in P if $\mathbf{H}$ contains a directed cycle of length at most 3; otherwise it is NP-hard.

For the following theorem, for any poset $P = (V, \leq)$, let the digraph $\mathbf{L}_P$ be given by $\mathbf{L}_P = (V, \lessdot)$ where $\lessdot$ is the covering relation of P (i.e. $x \lessdot y$ if $x < y$, and, for no z do we have $x < z < y$). Recall also that ω is the poset of the natural numbers under the usual ordering.

Theorem 5.7. Let $\mathbf{G} \to \rightsquigarrow$ be any digraph. Then $\mathrm{PCSP}(\widehat{\mathbf{G}}, \widehat{\rightsquigarrow})$ is in P if $\mathbf{G} \to \mathbf{L}_\omega$, otherwise it is NP-hard.

Theorem 5.8. Let $\mathbf{G} \to \mathbf{H}$ be bipartite, nonempty digraphs. Then $\mathrm{PCSP}(\widehat{\mathbf{G}}, \widehat{\mathbf{H}})$ is in P if (i) $\mathbf{G} \to \mathbf{L}_\omega$ and (ii) $\mathbf{H}$ contains a directed cycle of length 2; otherwise it is NP-hard.[1]

We will prove these results in three parts. First we will prove all our tractability results in Section 5.2; indeed we will show that the tractability result in Theorem 5.8 holds even if $\mathbf{G}, \mathbf{H}$ are not bipartite, and this implies everything else. Then we will show the hardness result of Theorem 5.6 in Section 5.3, and the hardness result of Theorem 5.7 in Section 5.4. It turns out that the hardness result of Theorem 5.8 follows from these other hardness results, as we show now.

[1] When $\mathbf{G}$ or $\mathbf{H}$ is empty $\mathrm{PCSP}(\widehat{\mathbf{G}}, \widehat{\mathbf{H}})$ is trivial: If either of them is empty then $\mathbf{G}$ must be, and hence the answer to an instance $\mathbf{X}$ is YES if $\mathbf{X}$ is empty as well, and otherwise it is NO.

Proof of hardness in Theorem 5.8 assuming Theorem 5.6 and Theorem 5.7. $\mathbf{G}, \mathbf{H}$ are bipartite and nonempty, so

$$\overset{\frown}{\circ\!\!-\!\!\circ} \to \mathbf{G} \to \mathbf{H} \to \overset{\frown}{\circ\!\!\rightarrow\!\!\circ},$$

and hence

$$\widehat{\overset{\frown}{\circ\!\!-\!\!\circ}} \to \widehat{\mathbf{G}} \to \widehat{\mathbf{H}} \to \widehat{\overset{\frown}{\circ\!\!\rightarrow\!\!\circ}}.$$

By homomorphic relaxation (cf. Lemma 2.4), it follows that $\mathrm{PCSP}(\widehat{\overset{\frown}{\circ\!\!-\!\!\circ}}, \widehat{\mathbf{H}})$ and $\mathrm{PCSP}(\widehat{\mathbf{G}}, \widehat{\overset{\frown}{\circ\!\!\rightarrow\!\!\circ}})$ both reduce to $\mathrm{PCSP}(\widehat{\mathbf{G}}, \widehat{\mathbf{H}})$. Thus, if $\mathbf{G} \not\to \mathbf{L}_\omega$ or $\mathbf{H}$ has no directed cycle of length 2 (and hence no directed cycle of length at most 3, since $\mathbf{H}$ is bipartite and hence has no odd cycles), it follows that one of $\mathrm{PCSP}(\widehat{\overset{\frown}{\circ\!\!-\!\!\circ}}, \widehat{\mathbf{H}})$ or $\mathrm{PCSP}(\widehat{\mathbf{G}}, \widehat{\overset{\frown}{\circ\!\!\rightarrow\!\!\circ}})$ is NP-hard, by Theorem 5.6 or Theorem 5.7 respectively. Hence $\mathrm{PCSP}(\widehat{\mathbf{G}}, \widehat{\mathbf{H}})$ is NP-hard. □

5.2. Tractability

Our general tractability result will be the following.

Proposition 5.9. The following PCSPs are solvable in polynomial time.

1. $\mathrm{PCSP}(\widehat{\mathbf{L}_\omega}, \widehat{\circlearrowleft})$,
2. $\mathrm{PCSP}(\widehat{\mathbf{L}_\omega}, \widehat{\overset{\frown}{\circ\!\!\rightarrow\!\!\circ}})$,
3. $\mathrm{PCSP}(\widehat{\mathbf{L}_\omega}, \widehat{\triangle})$.

This can be seen as a generalisation of the fact that $\mathrm{PCSP}(\mathbf{1\text{-}in\text{-}3}, \mathbf{NAE}) = \mathrm{PCSP}(\widehat{\overset{\frown}{\circ\!\!-\!\!\circ}}, \widehat{\overset{\frown}{\circ\!\!\rightarrow\!\!\circ}})$ is solvable in polynomial time [BG21] — indeed, the proof from [BG21] follows by the same homomorphic sandwich (cf. Corollary 2.5), using the same map from that sandwich to **NAE**. Noting that $\widehat{\overset{\frown}{\circ\!\!-\!\!\circ}} \to \widehat{\mathbf{L}_\omega}$ shows us that all the tractability results in Theorems 5.6 to 5.8 follow by homomorphic relaxation.

We will need a very easy lemma.

Lemma 5.10. Define $(a_n)_{n\geq 0}$ by $a_n = \frac{1-(-2)^n}{3}$. Then $a_n \in \mathbb{Z}$, and also

$$a_{n+1} + 2a_n = 1,$$

for all $n \in \mathbb{N}$.

Interestingly, this sequence also appeared in [Bra22, Proposition 4.2.17], in the context of finding *daisy chain terms* in affine algebras. This is not surprising, since this is (unpacking definitions) essentially what we are doing here.

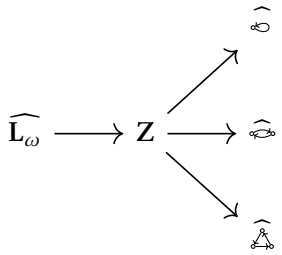
Proof. Note that $-2 \equiv 1 \bmod 3$, so $(-2)^n \equiv 1 \bmod 3$. Hence $1 - (-2)^n \equiv 0 \bmod 3$, and $a_n \in \mathbb{Z}$. Next, observe

$$a_{n+1} + 2a_n = \frac{1-(-2)^{n+1}}{3} + 2\frac{1-(-2)^n}{3} = \frac{1 + 2 - \cancel{(-2)^{n+1}} - \cancel{2(-2)^n}}{3} = 1. \qquad \square$$

Proof of Proposition 5.9. All of these problems can be solved by AIP [BG21], but we explain the algorithm in a simple self-contained way. Define the relational structure

$$\mathbf{Z} = (\mathbb{Z}, \{(x, y, z) \mid x + y + z = 1\}).$$

We first observe that CSP($\mathbf{Z}$) is solvable in polynomial time (indeed solving linear equations with integer coefficients over the integers can be done in polynomial time [Sch98]). We will show that



The result then follows by homomorphic sandwich.

- $\widehat{\mathbf{L}_\omega} \to \mathbf{Z}$. This homomorphism is given by $n \mapsto a_n$, where $(a_n)_{n\geq 0}$ is taken from Lemma 5.10.

- $\mathbf{Z} \to \widehat{\circlearrowleft}$. Take the unique constant map (i.e. $x \mapsto 0$). This is a homomorphism since $(0, 0, 0)$ is a tuple in the relation of $\widehat{\circlearrowleft}$.

- $\mathbf{Z} \to \widehat{\rightrightarrows}$. Take the map given by

$$x \mapsto \begin{cases} 0, & x \leq 0 \\ 1, & x \geq 1 \end{cases}$$

  This is a homomorphism since, for any tuple $(x, y, z) \in \mathbb{Z}^3$ with $x + y + z = 1$, it is not possible for $h(x) = h(y) = h(z) = 0$ (as then $x + y + z \leq 0$), or $h(x) = h(y) = h(z) = 1$ (as then $x + y + z \geq 3$). (This homomorphism appears in the proof that PCSP(**1-in-3**, **NAE**) can be solved in polynomial time [BG21].)

- $\mathbf{Z} \to \widehat{\triangle}$ Take the map $x \mapsto x \bmod 3$. As $\widehat{\triangle} = (\{0, 1, 2\}, \{(x, y, z) \mid x + y + z \equiv 1 \bmod 3\})$ (cf. Example 5.4), this map is a homomorphism: if $x + y + z = 1$ then $(x \bmod 3) + (y \bmod 3) + (z \bmod 3) \equiv 1 \bmod 3$. □

5.3. Hardness of Theorem 5.6

In this section we prove that, whenever $\mathbf{H}$ contains no directed cycle of length 1, 2 or 3, the problem PCSP($\widehat{\rightarrow}, \widehat{\mathbf{H}}$) = PCSP(**1-in-3**, $\widehat{\mathbf{H}}$) is NP-hard. Our proof here can be seen as a generalisation of some of the proofs in [BBB21], in particular the proof that PCSP(**1-in-3**, $\widehat{\mathbf{L}_3}$) is NP-hard (where we have denoted by L_3 the unique linear order on 3 elements). Note that $\widehat{\mathbf{L}_3}$ is denoted by $\mathbf{T}_1$ in [BBB21].

Fix a directed graph $\mathbf{H} = ([D], \prec)$ that contains no directed cycle of length 1, 2 or 3. Define $\succ, \preceq, \succeq, \sim$ as follows:

$$\begin{aligned} a \succ b &= b \prec a, \\ a \preceq b &= (a \prec b \vee a = b), \\ a \succeq b &= (b \prec a \vee a = b), \\ a \sim b &= (a \prec b \vee b \prec a \vee a = b). \end{aligned}$$

Remark 5.11. Why do we denote the edge relation of $\mathbf{H}$ by $\prec$? The proof in this section was first developed for the case $\prec = <$, i.e. $\mathbf{H}$ is the acyclic tournament on D vertices. It was then generalised to arbitrary $\mathbf{H}$ that has no cycle of length 1, 2 or 3. Note that the lack of such cycles implies that (i) $\prec$ is irreflexive, (ii) $\prec$ is antisymmetric, and (iii) $\prec$ has no "positive example" of non-transitivity. By this we mean that, for every $a \prec b \prec c$, while it may be possible that $a \not\prec c$, it is certainly impossible that $c \prec a$. Thus $\prec$ is seen as a very weak order relation. We find that it is useful to consider throughout this proof what the statements mean when $\prec = <$, and how we deal with the lack of full transitivity for $\prec$.

The main tool we will use for hardness is polymorphisms. We will now define what a polymorphism is *for the specific case of* $\mathrm{PCSP}(\textbf{1-in-3}, \widehat{\mathbf{H}})$, where $\mathbf{H} = ([D], \prec)$.

Definition 5.12. An n-ary polymorphism of $(\textbf{1-in-3}, \widehat{\mathbf{H}})$, where $\mathbf{H} = ([D], \prec)$, is a function $f : \{0,1\}^n \to [D]$, such that, for any three vectors $\vec{x}, \vec{y}, \vec{z} \in \{0,1\}^n$ such that, for every $i \in [n]$ exactly one of $\vec{x}_i, \vec{y}_i, \vec{z}_i$ is 1 and the other two are 0, we have that $f(\vec{x}) = f(\vec{y}) \prec f(\vec{z})$, or $f(\vec{y}) = f(\vec{z}) \prec f(\vec{x})$, or $f(\vec{z}) = f(\vec{x}) \prec f(\vec{y})$. (In other words two elements are equal and the third is greater w.r.t $\prec$.)
Equivalently, noting that $\{0,1\}^n \cong 2^{[n]}$ (by taking a set to its characteristic vector and back), we can see polymorphisms in the following way. An n-ary polymorphism of $(\textbf{1-in-3}, \widehat{\mathbf{H}})$ is a function $f : 2^{[n]} \to [D]$ such that, for any partition (A, B, C) of $[n]$, we have $f(A) = f(B) \prec f(C)$, or $f(B) = f(C) \prec f(A)$, or $f(C) = f(A) \prec f(B)$. (This equivalent definition appeared in [BBB21].)
Under this view, if $f : 2^{[n]} \to [D]$ is an n-ary polymorphism and $\pi : [n] \to [m]$, then the minor of f by π is given by $f^\pi = f \circ \pi^{-1}$, where $\pi^{-1} : 2^{[m]} \to 2^{[n]}$ is defined by

$$\pi^{-1}(S) = \{i \in [n] \mid \pi(i) \in S\}.$$

We will use the following two "laws" to deduce facts about $\mathbf{H}$ from its polymorphisms.

Proposition 5.13 (Disjointness Law). Suppose $f : 2^{[n]} \to [D]$ is a polymorphism of $(\textbf{1-in-3}, \mathbf{H})$, and $A, B \subseteq [n]$ are disjoint. Then $f(A) \sim f(B)$.

Proof. Indeed, let $C = [n] \setminus (A \cup B)$. Since (A, B, C) is a partition of $[n]$, as f is a polymorphism we have $f(A) = f(B) \prec f(C)$, or $f(B) = f(C) \prec f(A)$, or $f(C) = f(A) \prec f(B)$. In every case one of $f(A) = f(B)$, $f(A) \prec f(B)$ or $f(A) \succ f(B)$ holds. So $f(A) \sim f(B)$. □

Proposition 5.14 (Union Law). Suppose $f : 2^{[n]} \to [D]$ is a polymorphism of (**1-in-3**, **H**), and consider disjoint $A, B, C \subseteq [n]$. Consider the sequence $f(A), f(B), f(A \cup C), f(B \cup C)$. There exists a way of permuting this sequence so that the resulting sequence $a, b, c, d \in [D]$ satisfies:

(i) $a = b = c = d$, or

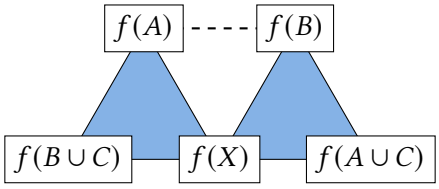
(ii) $c \succ a = b \prec d$, or

(iii) $a = b \prec c \prec d$.

In all cases, $a = b \not\succ c$ and $a = b \not\succ d$. We call the value $a = b$ the *pseudo-minimum* of $f(A), f(B), f(A \cup C), f(B \cup C)$. Thus the pseudo-minimum appears an even number of times.

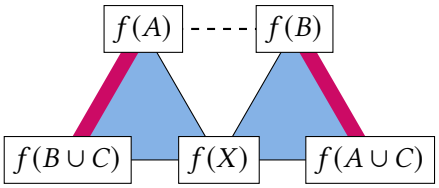
Proof. Note that $a = b \not\succ c$, $a = b \not\succ d$ and the appearance of the pseudo-minimum an even number of times follows from fact that $\prec$ does not have small cycles — provided we prove the classification enumerated above. Thus we focus on proving this classification.

Let $X = [n] \setminus (A \cup B \cup C)$. Observe by the Disjointness Law that $f(A) \sim f(B)$. Apply the fact that f is a polymorphism to $(A, B \cup C, X)$ and $(A \cup C, B, X)$. Now two among $f(A), f(B \cup C), f(X)$ and $f(A \cup C), f(B), f(X)$ respectively are equal, and the third is greater (w.r.t. $\prec$). We visualise this situation in the following figure: filled-in triangles mean that "two of these values are equal, the third is larger", and a dashed line represents $\sim$.



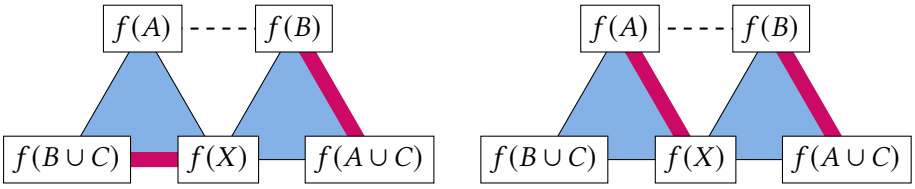
We will consider the cases depending on which values are unique in each triangle. In the diagrams that follow, a thick purple line will denote the equal pair of elements in each triangle.

$f(X)$ is unique in both triangles. In this case we have:



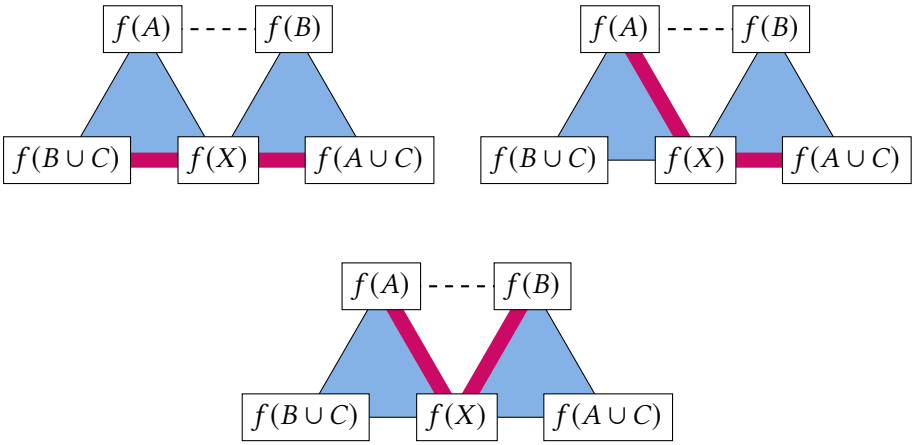
Here we have $f(B \cup C) = f(A) \sim f(B) = f(A \cup C)$. So in this case, either all 4 elements are equal, or they split into two pairs of equal elements, one pair being greater than the other (w.r.t. $\prec$). Thus we are in Case (i) or Case (ii).

$f(X)$ is unique in one triangle. After perhaps swapping A and B, we are in one of the following cases:



In either case $f(B) = f(A \cup C) \prec f(X)$, one of $f(A), f(B \cup C)$ is equal to $f(X)$ and the other is greater (w.r.t $\prec$). So we are in Case (iii).

$f(X)$ is not unique in either triangle. Perhaps swapping A and B, we are in one of the following cases:



So in all cases two of $f(A), f(B), f(A \cup C), f(B \cup C)$ are equal (and equal to $f(X)$), and the other two values are strictly greater (w.r.t. $\prec$). So we are in Case (ii). □

We will use the hardness theorem from the preliminaries, which we restate for the reader's convenience.

Theorem 2.23. Fix C, k. There exists some N such that the following holds. Consider some $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$, and let $\mathcal{M} = \mathrm{Pol}(\mathbf{A}, \mathbf{B})$. Cover $\mathcal{M}$ by k sets $\mathcal{M}_1, \ldots, \mathcal{M}_k$. For every $i \in [k]$, suppose there exists a mapping I_i that takes $f \in \mathcal{M}_i$ of *arity* $n \leq N$ to a set $I_i(f) \subseteq [n]$ of size at most C. Suppose that for every $f, g \in \mathcal{M}_i$ *where* $\mathrm{ar}(f) \leq N, \mathrm{ar}(g) \leq N$ and π such that $f \xrightarrow{\pi} g$ we have that $\pi(I_i(f)) \cap I_i(g) \neq \emptyset$. Then $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is NP-hard.

Recall that for us, $f^\pi = f \circ \pi^{-1}$.

How can we find these sets $I(f)$ for polymorphisms f? It is natural to consider sets with very strong properties, e.g. sets that intersect all sets S where $f(S) = x$ for some fixed x. This idea is implicit in [BBB21]. It is similar to that of a *fixing set* [FKOS19] or of an *avoiding set* [DRS05; NVWŽ25]. This is not enough however (for example it is trivially true for any set if no set has image x). To make the idea work, we add the stipulation that our set contain a subset that has the required value — we call this notion a *hitting set*.

Definition 5.15 (Hitting set). Fix an n-ary polymorphism f of $(\mathbf{1\text{-in-}3}, \widehat{\mathbf{H}})$. We call a set $S \subseteq [n]$ a *hitting set with hitting value* $x \in [D]$, if

1. There exists $T \subseteq S$ with $f(T) = x$.
2. For any $U \subseteq [n]$ with $f(U) = x$, we have $U \cap S \neq \emptyset$.

Having small hitting sets is enough for hardness.

Proposition 5.16. Fix C. Suppose every polymorphism f of $(\mathbf{1\text{-in-}3}, \widehat{\mathbf{H}})$ has a hitting set of size at most C. Then $\mathrm{PCSP}(\mathbf{1\text{-in-}3}, \widehat{\mathbf{H}})$ is NP-hard.

Proof. For every $x \in [D]$, we set $\mathcal{M}_x$ to be the set of polymorphisms of $(\mathbf{1\text{-in-}3}, \widehat{\mathbf{H}})$ that have a hitting set of size at most C, with hitting value x. For every $x \in [D]$ and $f \in \mathcal{M}_x$, we set $I_x(f)$ to be an arbitrarily chosen hitting set of size at most C, with hitting value x.

By assumption $\bigcup_{x \in D} \mathcal{M}_x = \mathrm{Pol}(\mathbf{1\text{-in-}3}, \widehat{\mathbf{H}})$, so we attempt to apply Theorem 2.23. Fix x and consider $f, g \in \mathcal{M}_x$ with $g = f \circ \pi^{-1}$. We want to show that $\pi(I_x(f)) \cap I_x(f \circ \pi^{-1}) \neq \emptyset$. To this end, first note that, as $I_x(f \circ \pi^{-1})$ is a hitting set with value x, there exists $S \subseteq I_x(f \circ \pi^{-1})$ with $f(\pi^{-1}(S)) = (f \circ \pi^{-1})(S) = x$. Now since $I_x(f)$ is a hitting set with value x, it intersects all sets that have image x through f. Hence $I_x(f) \cap \pi^{-1}(S) \neq \emptyset$. In other words there exists some i such that $i \in I_x(f)$ and $i \in \pi^{-1}(S)$. Thus $\pi(i) \in \pi(I_x(f))$, and $\pi(i) \in S$ by definition. Hence

$$\emptyset \neq \pi(I_x(f)) \cap S \subseteq \pi(I_x(f)) \cap I_x(f \circ \pi^{-1})$$

□

How can we ensure that a hitting set exists? It turns out that having large families of sets whose images through f form an ascending chain is sufficient to make a hitting set. This "chain condition" will be our final goal.

Proposition 5.17 (Chain Condition). Fix $f \in \mathrm{Pol}^{(n)}(\mathbf{1\text{-in-}3}, \widehat{\mathbf{H}})$. Suppose we have sets $S_1, \ldots, S_{D+1}$ with $f(S_1) \prec \cdots \prec f(S_{D+1})$, recalling that $[D]$ is the domain of $\mathbf{H}$. Then $\bigcup_{i=1}^{D+1} S_i$ is a hitting set of f.

Proof. We will prove that the existence of $S_1, \ldots, S_{D+1}$ such that $f(S_1) \prec \cdots \prec f(S_{D+1})$ implies that there exist $i \neq j$ such that $f(S_i) \nsim f(S_j)$, by contrapositive. Thus assume that $f(S_i) \sim f(S_j)$ for all $i \neq j$ with $i, j \in [D+1]$. Since $\prec$ has no cycles of length at most 2, the subgraph of $\mathbf{H}$ induced by $f(S_1), \ldots, f(S_{D+1})$ is a tournament. It is well-known that any non-acyclic finite tournament has a cycle of length at most 3 — since $\mathbf{H}$ does not, it follows that the subgraph of $\mathbf{H}$ induced by $f(S_1), \ldots, f(S_{D+1})$ is an acyclic tournament. Hence any walk in this subgraph has at most D vertices (as $\mathbf{H}$ has at most D vertices), and it follows that $f(S_1) \prec \cdots \prec f(S_{D+1})$ cannot be true.

Thus we find that $f(S_i) \nsim f(S_j)$ for some $i \neq j$. By the contrapositive of the Disjointness Law, S_j intersects all sets $T \subseteq [n]$ where $f(T) = f(S_i)$. Thus $\bigcup_{i=1}^{D+1} S_i$ is a hitting set with hitting value $f(S_i)$. (Indeed it also has hitting value $f(S_j)$.) □

Our general goal will therefore be to find a long sequence of small sets with increasing values. In the proofs in [BBB21], the empty set and its value had a special role to play. However, since we want to find ever-growing chains, we will need to find a way to work with polymorphisms "relative" to a "virtual empty set".

Definition 5.18. Fix $f \in \mathrm{Pol}^{(n)}(\textbf{1-in-3}, \widehat{\mathbf{H}})$ and $E \subseteq [n]$. Define $f_E : 2^{[n]\setminus E} \to [D]$ by $f_E(X) = f(E \cup X)$.
We call $f_E(X)$ the *image of X relative to E*. We say that X is *positive relative to E* if $f_E(X) \succ f(E)$, that it is *negative relative to E* if $f_E(X) \prec f(E)$, and that it is *neutral relative to E* if $f_E(X) = f(E)$. Relative nonnegativity and nonpositivity are defined in the obvious way. If E is clear from context we omit "relative to E".

The nice property of $\varnothing$ is that, for any partition S, T, U of $[n]$, $(\varnothing \cup S, \varnothing \cup T, U)$ is a partition of $[n]$. Hence $(f_\varnothing(S), f_\varnothing(T), f(U))$ is a tuple of the relation of $\widehat{\mathbf{H}}$. To imitate this situation, we will want two disjoint virtual empty sets E, F, with equal image. Then for any S, T disjoint and disjoint from E, F, there exists U — namely $U = [n] \setminus (E \cup F \cup S \cup T)$ — such that $(f_E(S), f_F(T), f(U))$ is a tuple of the relation of $\widehat{\mathbf{H}}$.

Lemma 5.19. Take $f \in \mathrm{Pol}^{(n)}(\textbf{1-in-3}, \widehat{\mathbf{H}})$. For $E, F \subseteq [n]$ disjoint with $f(E) = f(F)$, every $X \subseteq [n]$ disjoint from E and F is negative, neutral or positive relative to E.

Proof. By the Disjointness Law, $f_E(X) = f(E \cup X) \sim f(F) = f(E)$. □

The next question is: how do f_E and f_F compare? It turns out that these two maps are the same for sets nonpositive relative to E.

Lemma 5.20. Take $f \in \mathrm{Pol}^{(n)}(\textbf{1-in-3}, \widehat{\mathbf{H}})$. For $E, F \subseteq [n]$ disjoint with $f(E) = f(F)$, and for any $X \subseteq [n]$ disjoint from E, F, if X is nonpositive relative to E, we have $f_E(X) = f_F(X)$. Furthermore, if X is negative, neutral or positive (respectively) relative to E, then it is negative, neutral or positive (respectively) relative to F.

Proof. Suppose X is nonpositive relative to E. Apply the Union Law to sets $E, F, E \cup X, F \cup X$. Consider which value among the images of these four sets through f is the pseudo-minimum.

- **$f(E) = f(F)$ is the pseudo-minimum.** Since $f(E \cup X) = f_E(X) \preceq f(E)$, it must be the case that $f(E) = f(E \cup X)$, and therefore all 4 images are equal. Thus $f_E(X) = f(E \cup X) = f(F \cup X) = f_F(X)$.
- **$f(E) = f(F)$ is not the pseudo-minimum.** Thus, since the pseudo-minimum must appear at least twice, we must have that $f_E(X) = f(E \cup X) = f(F \cup X) = f_F(X)$ is the pseudo-minimum.

So in all cases $f_E(X) = f_F(X)$.

Now, if X is negative or neutral respectively relative to E, then $f_E(X) = f_F(X)$, and as $f(E) = f(F)$, X must be negative or neutral respectively relative to F. On the other hand, if X is positive relative to E, then X cannot be nonpositive relative to F — or else $f(E) \prec f_E(X) = f_F(X) \preceq f(F) = f(E)$, which gives a short cycle in $\mathbf{H}$. □

Next, we determine $f_E(X \cup Y)$ in terms of $f_E(X)$ and $f_E(Y)$ when $f_E(X) = f_E(Y)$.

Lemma 5.21. Take $f \in \mathrm{Pol}^{(n)}(\textbf{1-in-3}, \widehat{\mathbf{H}})$. Consider disjoint E, F with $f(E) = f(F)$, and $X, Y \subseteq [n]$ disjoint from E, F and from each other, with $f_E(X) = f_E(Y)$.

(i) If X, Y are negative relative to E, then either $f_E(X \cup Y) \succ f_E(X)$, or $X \cup Y$ is positive relative to E.

(ii) If X, Y are neutral relative to E, then $X \cup Y$ is neutral relative to E.

Proof. We prove the conclusions of this lemma point by point.

(i) Suppose $f_E(X) = f_E(Y) \prec f(E)$. By Lemma 5.20, we have that $f_F(X) = f_F(Y) = f_E(X) = f_E(Y)$. Apply the Union Law to $F, F \cup X, E \cup Y, E \cup X \cup Y$. We have $f(F) = f(E) \succ f(F \cup X) = f(E \cup Y)$, so the pseudo-minimum must be $f(F \cup X) = f(E \cup Y)$. Then we have two cases according to the Union Law:

- $f(F \cup X) = f(E \cup Y) \prec f(F) \prec f(E \cup X \cup Y)$, or
- $f(F) \succ f(F \cup X) = f(E \cup Y) \prec f(E \cup X \cup Y)$,

which corresponds to the two cases in our conclusion, as $f_E(X \cup Y) = f(E \cup X \cup Y)$.

(ii) Suppose $f_E(X) = f_E(Y) = f(E)$. Apply the Union Law to $F, F \cup X, E \cup Y, E \cup X \cup Y$. Since $f(F) = f(F \cup X) = f(E \cup Y)$ similarly to above, we have $f_E(X \cup Y) = f(E \cup X \cup Y) = f(F) = f(E)$. □

Remark 5.22. The rules in the previous lemma turn out to be all that we need, but more can be said:

- The union of two positive sets is a neutral set.
- The union of a neutral and a positive set is a positive set.
- The union of a negative set with any other set with a different relative value is a negative set (and its relative image is the same as the minimum of the two relative images of the sets, and this minimum is defined).

With this deeper understanding of relative images, we give the most important lemma of this section. This lemma takes some set E, and either immediately generates a hitting set of small size, or finds another small set which has a larger image (w.r.t. $\prec$) — thus contributing to the Chain Condition.

Lemma 5.23. Take $f \in \mathrm{Pol}^{(n)}(\textbf{1-in-3}, \widehat{\mathbf{H}})$ and $E \subseteq [n]$. There either exists a hitting set of size at most $|E| + 2^D$, or some set $E' \subseteq [n]$ with $|E'| \leq |E| + D2^D$ and $f(E) \prec f(E')$.

Proof. If E is a hitting set, then there is nothing left to prove, so suppose that E is not a hitting set. In particular, it is not a hitting set with hitting value $f(E)$, so there exists some F disjoint from E with $f(F) = f(E)$. If at any point we find a set X positive relative to E, with size at most $D2^D$, then we are done immediately — we can take $E' = E \cup X$. Hence assume no such set exists.

Create a partition of $[n] \setminus (E \cup F)$, initially built out of all the singleton subsets of $[n] \setminus (E \cup F)$. (If $E \cup F = [n]$ the partition is empty.) While

(i) all parts P that are negative relative to E have $|P| \leq 2^{D-1}$, and

(ii) there exist parts P, Q, negative relative to E, with $f_E(P) = f_E(Q)$,

merge P with Q, replacing P, Q with $P \cup Q$ in the partition. Note that throughout the run of this algorithm, we only merge parts of size at most 2^{D-1}, and we initially have parts of size 1. So we will never have any parts of size greater than 2^D. Thus, by assumption, *none of the parts are positive relative to E*. Furthermore, since the number of parts decreases after every merge, the procedure must terminate eventually. Let us investigate what happens when it does terminate, depending on the reason for termination. (The following two cases are not disjoint, when they overlap both proofs work.)

Termination due to Reason (i). In this case, there exists a part P_0, negative relative to E, but with $|P_0| > 2^{D-1}$. Construct a tree whose vertices are parts that existed in our partition at some point in time that are subsets of P_0, whose root is P_0, and where the children of a vertex P are the two parts Q, R which were merged to form it. Singleton parts are the leaves in this tree. Consider any vertex P in this tree, with children Q and R. By construction, $f_E(Q) = f_E(R)$, Q, R are disjoint and negative relative to E. So, by Lemma 5.21 (i), we have that either $f_E(Q) \prec f_E(Q \cup R)$, or $Q \cup R$ is positive relative to E. This latter case is impossible by assumption, so we have $f_E(Q) \prec f_E(Q \cup R)$. In other words, the relative image of a vertex in the tree is larger (w.r.t. $\prec$) than that of its children.

Since $|P_0| > 2^{D-1}$, and clearly there are $|P_0|$ leaves in this tree, it follows that there must exist a chain going from the root to the leaves, containing at least $D+1$ vertices; without loss of generality the chain starts at the root, P_0. Thus suppose that the chain is $P_0, P_1, \ldots, P_D$. Now, by the discussion in the previous paragraph, $f_E(P_0) \succ f_E(P_1) \succ \cdots \succ f_E(P_D)$, so by definition $f(E \cup P_0) \succ \cdots \succ f(E \cup P_D)$. By the Chain Condition, $E \cup \bigcup_{i=1}^{D} P_i$ is a hitting set. Since $P_D \subseteq \cdots \subseteq P_0$, this set has at most $|E| + |P_0| \leq |E| + 2^D$ vertices, as required.

Termination due to Reason (ii). In this case, there are no two parts P, Q with $f_E(P) = f_E(Q)$ and P, Q negative relative to E. Suppose $N_1, \ldots, N_t$ are the parts negative relative to E, and $Z_1, \ldots, Z_u$ are the parts neutral relative to E. By the pigeonhole principle $t \leq D$. (As mentioned above, there are no parts that are positive relative to E.) By Lemma 5.19, every part is either neutral, negative or positive, so

$$E, F, N_1, \ldots, N_t, Z_1, \ldots, Z_u$$

form a partition of $[n]$.

Let $N = \bigcup_{i=1}^{t} N_i$ and $Z = \bigcup_{i=1}^{u} Z_i$, and consider the partition $E \cup Z, F, N$ of $[n]$. Note that by Lemma 5.21 (ii) applied inductively, $f(E \cup Z) = f_E(Z) = f(E) = f(F)$, hence $f(E) = f(F) \prec f(N)$. Furthermore, $|N| \leq D2^D$, so we can safely set $E' = N$. □

With this result in hand, proving the required hardness result becomes a simple matter.

Proof of hardness in Theorem 5.6. We apply Proposition 5.16 with

$$C = \max(D^2(D+1)2^{D-1}, (D^2+1)2^D).$$

To find the required hitting set, start with $E_0 = \varnothing$, and repeatedly apply Lemma 5.23, attempting to find a sequence $E_0, E_1, \ldots, E_D \subseteq [n]$ with $f(E_0) \prec \cdots \prec f(E_D)$. There are two cases. First suppose that Lemma 5.23 always provides a set E' of bounded size with $f(E) \prec f(E')$. Then we have that $|E_i| \leq iD2^D$, and thus $\bigcup_{i=0}^{D} E_i$, a hitting set by the Chain Condition, has size at most $D^2(D+1)2^{D-1}$. On the other hand, suppose that Lemma 5.23 provided the set E' when applied $E_0, \ldots, E_{i-1}$, but when given E_i it provided a hitting set of size $|E_i| + 2^D$. Then similarly to before this hitting set has size at most $iD2^D + 2^D \leq (D^2+1)2^D$. Thus we always have the hitting set we need. □

5.4. Hardness of Theorem 5.7

In this section, we will prove that, for any bipartite digraph $\mathbf{G}$ where $\mathbf{G} \not\to \mathbf{L}_\omega$, the problem $\mathrm{PCSP}(\widehat{\mathbf{G}}, \widehat{\rightarrow})$ = $\mathrm{PCSP}(\widehat{\mathbf{G}}, \mathbf{NAE})$ is NP-hard. We will first show that this is the case whenever $\mathbf{G}$ is a directed cycle (i.e. a graph of form $0 \to 1 \to \cdots \to k-1 \to 0$). Recall the notation $\mathbf{D}_k$ for the directed cycle of length k. In other words,

$$\mathbf{D}_k = (\mathbb{Z}_k, \{(i, i+1) \mid i \in \mathbb{Z}_k\}).$$

Note that we have explicitly identified the domain of $\mathbf{D}_k$ with $\mathbb{Z}_k$ — we will extensively use the fact that addition is defined on this domain.

Proposition 5.24. For any $k \geq 1$, the problem $\mathrm{PCSP}(\widehat{\mathbf{D}_{2k}}, \mathbf{NAE})$ is NP-hard.

We prove this fact in Section 5.4.1. The next step involves some more definitions.

Definition 5.25. An *oriented cycle* is a directed graph formed by taking an undirected cycle, and arbitrarily orienting its edges. For example, the digraph $1 \to 2 \to 3 \leftarrow 4 \to 1$ is an oriented cycle. An *oriented path* is a directed graph formed by taking an undirected path, and arbitrarily orienting its edges. For example, the digraph $1 \to 2 \leftarrow 3 \to 4$ is an oriented path.
The *net length* of an oriented cycle or path is the absolute difference between the number of edges going in one direction minus the other. For example, in $1 \to 2 \to 3 \leftarrow 4 \to 1$, the net length is 2, and in the $1 \to 2 \leftarrow 3 \to 4$ the net length is 1.

The following is easy to see, and appears in [HN04]. It gives an alternative characterisation for those graphs $\mathbf{G}$ where $\mathbf{G} \not\to \mathbf{L}_\omega$.

Lemma 5.26. Consider a digraph $\mathbf{G}$. Then $\mathbf{G} \not\to \mathbf{L}_\omega$ if and only if $\mathbf{G}$ contains as a (not necessarily induced) subgraph some oriented cycle with nonzero net length.

Proof. First suppose that $h : \mathbf{G} \to \mathbf{L}_\omega$. In this case, each oriented cycle found within $\mathbf{G}$ must have net length 0. To see why this is the case, note that for any oriented path within $\mathbf{G}$ that starts at u and ends at v, the net length must be $|h(u) - h(v)|$. Thus, starting from some node u in the oriented cycle and following the cycle until we reach u again, we see that the net length must be $|h(u) - h(u)| = 0$.

Second, suppose that every oriented cycle has net length equal to 0. Then we will show that there exists a homomorphism $h : \mathbf{G} \to \mathbf{L}_\omega$. Without loss of generality suppose $\mathbf{G}$ is connected. Take a spanning tree $\mathbf{T}$ of $\mathbf{G}$, and fix some vertex v of $\mathbf{G}$. For every vertex u, let $h(u)$ be the number of edges towards u on the oriented path from v to u, minus the number of edges away from u on that same path. Consider any edge (x, y) of $\mathbf{G}$. Consider the oriented cycle containing the edge (x, y) and the oriented path from y to x in $\mathbf{T}$. By the definition of h, we can see that it has net length equal to $|h(y) - h(x) - 1|$, and hence $h(y) - h(x) - 1 = 0$ and thus $h(y) = h(x) + 1$ — so h is indeed a homomorphism, as required. □

The next step is to reduce to PCSP($\widehat{\mathbf{G}}$, **NAE**) from PCSP($\widehat{\mathbf{D}_{2k}}$, **NAE**) for some appropriate value of $2k$.

Proposition 5.27. Suppose $\mathbf{G} \to \mathbf{D}_2$ contains an oriented cycle with net length $2k \neq 0$. Then PCSP($\widehat{\mathbf{D}_{2k}}$, **NAE**) reduces to PCSP($\widehat{\mathbf{G}}$, **NAE**).

We prove this fact in Section 5.4.2.

From these two propositions, the hardness of Theorem 5.7 follows.

Proof of hardness in Theorem 5.7. Suppose $\mathbf{G} \not\to \mathbf{L}_\omega$. Thus $\mathbf{G}$ contains an oriented cycle with nonzero net length by Lemma 5.26. Since $\mathbf{G} \to \mathbf{D}_2$, and hence $\mathbf{G}$ is bipartite if we ignore edge orientations, this net length must be even — suppose it is $2k > 0$. Then, by Proposition 5.27, PCSP($\widehat{\mathbf{D}_{2k}}$, **NAE**) reduces to PCSP($\widehat{\mathbf{G}}$, **NAE**) — and PCSP($\widehat{\mathbf{D}_{2k}}$, **NAE**) is NP-hard by Proposition 5.24. So PCSP($\widehat{\mathbf{G}}$, **NAE**) is NP-hard. □

5.4.1. Proof for directed cycles

In this section we prove Proposition 5.24, which we restate now.

Proposition 5.24. For any $k \geq 1$, the problem PCSP($\widehat{\mathbf{D}_{2k}}$, **NAE**) is NP-hard.

In general we will focus on the case $k > 1$, since for $k = 1$ we have $\widehat{\mathbf{D}_2} = \mathbf{NAE}$ — so the PCSP in question is the same as CSP(**NAE**), which is well-known to be NP-hard [Sch78]. To prove this fact, we will use the following hardness criterion, from the preliminaries.

Theorem 2.16 (Easy corollary of [BBKO21, Corollary 5.2]). Suppose $\mathrm{Pol}^{(3)}(\mathbf{A}, \mathbf{B})$ contains only essentially unary, non-constant functions (i.e. functions that depend on exactly one input). Then $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is NP-hard.

As one could guess, we will show that $\mathrm{Pol}^{(3)}(\widehat{\mathbf{D}_{2k}}, \widehat{\mathbf{D}_2})$ contains only essentially unary functions.

Lemma 5.28. For $k \geq 2$, if $f \in \mathrm{Pol}^{(3)}(\widehat{\mathbf{D}_{2k}}, \widehat{\mathbf{D}_2})$ then f is essentially unary and non-constant.

Proof. We will prove various facts about f until our conclusion follows easily. Recall that the domain of $\widehat{\mathbf{D}_n}$ is $\mathbb{Z}_n$, so addition modulo n is assumed for this structure — whether $n = 2k$ or $n = 2$. Observe that we can shift the domain of $\widehat{\mathbf{D}_n}$ by a constant without changing its structure — we will often use this fact to conveniently shift the domain or codomain of f to simplify our proofs without loss of generality.

f is a polymorphism. Let us restate succinctly what this means. Consider any matrix

$$\begin{pmatrix} x & y & z \\ x' & y' & z' \\ x'' & y'' & z'' \end{pmatrix},$$

where (x, x', x''), (y, y', y''), (z, z', z'') — i.e. the columns of the matrix — are in the relation of $\widehat{\mathbf{D}_{2k}}$. Then $(f(x, y, z), f(x', y', z'), f(x'', y'', z''))$ — i.e. f applied to the rows of the matrix — is in the relation of $\widehat{\mathbf{D}_2}$: not all 3 of these values are equal. In particular, we can take $x = x' = x'' - 1, y = y' = y'' - 1, z = z' = z'' - 1$, and then we find that $f(x, y, z) \neq f(x + 1, y + 1, z + 1)$, or equivalently

$$1 + f(x, y, z) = f(x + 1, y + 1, z + 1) \tag{5.1}$$

2-dimensional reduction. Apply (5.1) inductively to find that $f(x, y, z) = f(x - z, y - z, 0) + z$. In other words, f is completely determined by the values in the matrix $M \in \mathbb{Z}_2^{2k \times 2k}$, with $M_{xy} = f(x, y, 0)$.

Adjacent equality implies constant. Consider any two adjacent positions in M, and suppose they are equal. Without loss of generality, suppose $M_{01} = M_{02} = 0$; equivalently $f(0, 1, 0) = f(0, 2, 0) = 0$. Apply the fact that f is a polymorphism to the matrix

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 2 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

to find that $f(1, 1, 1) = 1$. By (5.1), $M_{00} = f(0, 0, 0) = 1 + f(1, 1, 1) = 1 + 1 = 0$. Hence $M_{00} = M_{01}$. In general, if $M_{x,y} = M_{x+1,y}$ then $M_{x-1,y} = M_{x,y}$; and likewise if $M_{x,y} = M_{x,y+1}$ then $M_{x,y-1} = M_{x,y}$. Applying this fact inductively, we find that if any two adjacent elements in M are equal then the entire row or column they are on must be constant.

Adjacent disequality implies alternating. By contrapositive, if we have any two adjacent distinct elements in M, then the entire row or column they are on must alternate between 0 and 1.

Matrix is periodic. The previous two facts imply that M is periodic on both rows and columns with period 2: $M_{x+2,y} = M_{xy} = M_{x,y+2}$. We will prove the typical case, $M_{00} = M_{02}$. There are two subcases. If $M_{00} = M_{01}$ then the entire row is constant and in particular $M_{02} = M_{00}$. If $M_{00} \neq M_{01}$, then the entire row is alternating and $M_{02} = 1 + M_{01} = 2 + M_{00} = M_{00}$. Thus, in particular, the submatrix

$$A = \begin{pmatrix} M_{00} & M_{01} \\ M_{10} & M_{11} \end{pmatrix}$$

encodes the entire matrix.

Forbidden patterns. The following two configurations can never appear in M:

$$\begin{pmatrix} 0 & \star & \star \\ \star & \star & 1 \\ \star & 1 & \star \end{pmatrix}, \qquad \begin{pmatrix} 1 & \star & \star \\ \star & \star & 0 \\ \star & 0 & \star \end{pmatrix}$$

We show why this is impossible by contradiction. The two configurations are the same just swapping 0 and 1. Thus, and by shifting M (i.e. the domain of f) conveniently, assume without loss of generality that $M_{00} = 0$, $M_{12} = 1$, and $M_{21} = 1$. Equivalently $f(0,0,0) = 0$, $f(1,2,0) = 1$, $f(2,1,0) = 1$. Apply the fact that f is a polymorphism to the matrix

$$\begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

to find that $f(1,1,1) = 0$. Hence $f(0,0,0) = 1 + f(1,1,1) = 1$ — a contradiction.

We have now found enough properties of f, M and A to finish the proof. Let us consider the values that A can take. There are 16 possible values; of these, 6 lead to non-constant, essentially unary functions, namely:

$$\begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

We list which essentially unary functions f correspond to these values of A in Table 5.1.

The remaining values of A are the following:

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix},$$

When extending these values of A to M, we will always find a forbidden configuration:

A	f
$\begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}$	$f(x,y,z) = x \mod 2$
$\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$	$f(x,y,z) = 1 + x \mod 2$
$\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$	$f(x,y,z) = y \mod 2$
$\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$	$f(x,y,z) = 1 + y \mod 2$
$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$	$f(x,y,z) = z \mod 2$
$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$	$f(x,y,z) = 1 + z \mod 2$

Table 5.1.: Table of good cases from Lemma 5.28.

$$\left(\begin{smallmatrix} \boxed{1} & 0 & 1 & 0 \\ 0 & 0 & \boxed{0} & 0 \\ 1 & \boxed{0} & 1 & 0 \\ 0 & 0 & 0 & 0 \end{smallmatrix}\right), \left(\begin{smallmatrix} 0 & \boxed{1} & 0 & 1 \\ 0 & 0 & 0 & \boxed{0} \\ 0 & 1 & \boxed{0} & 1 \\ 0 & 0 & 0 & 0 \end{smallmatrix}\right), \left(\begin{smallmatrix} 0 & 0 & 0 & 0 \\ \boxed{1} & 0 & 1 & 0 \\ 0 & 0 & \boxed{0} & 0 \\ 1 & \boxed{0} & 1 & 0 \end{smallmatrix}\right), \left(\begin{smallmatrix} 0 & 0 & 0 & 0 \\ 0 & \boxed{1} & 0 & 1 \\ 0 & 0 & 0 & \boxed{0} \\ 0 & 1 & \boxed{0} & 1 \end{smallmatrix}\right),$$
$$\left(\begin{smallmatrix} \boxed{0} & 1 & 0 & 1 \\ 1 & 1 & \boxed{1} & 1 \\ 0 & \boxed{1} & 0 & 1 \\ 1 & 1 & 1 & 1 \end{smallmatrix}\right), \left(\begin{smallmatrix} 1 & \boxed{0} & 1 & 0 \\ 1 & 1 & 1 & \boxed{1} \\ 1 & 0 & \boxed{1} & 0 \\ 1 & 1 & 1 & 1 \end{smallmatrix}\right), \left(\begin{smallmatrix} 1 & 1 & 1 & 1 \\ \boxed{0} & 1 & 0 & 1 \\ 1 & 1 & \boxed{1} & 1 \\ 0 & \boxed{1} & 0 & 1 \end{smallmatrix}\right), \left(\begin{smallmatrix} 1 & 1 & 1 & 1 \\ 1 & \boxed{0} & 1 & 0 \\ 1 & 1 & 1 & \boxed{1} \\ 1 & 0 & \boxed{1} & 0 \end{smallmatrix}\right),$$
$$\left(\begin{smallmatrix} \boxed{0} & 1 & 0 & 1 \\ 1 & 0 & \boxed{1} & 0 \\ 0 & \boxed{1} & 0 & 1 \\ 1 & 0 & 1 & 0 \end{smallmatrix}\right), \left(\begin{smallmatrix} \boxed{1} & 0 & 1 & 0 \\ 0 & 1 & \boxed{0} & 1 \\ 1 & \boxed{0} & 1 & 0 \\ 0 & 1 & 0 & 1 \end{smallmatrix}\right).$$

Thus only values of A that lead to non-constant, essentially unary values of f are possible, which implies the conclusion □

Proof of Proposition 5.24. For $k = 1$ this follows immediately from the NP-hardness of CSP(**NAE**). For $k \geq 2$, combine Lemma 5.28 with Theorem 2.16. □

5.4.2. Proof of reduction

In this section we prove Proposition 5.27, which we now restate.

Proposition 5.27. Suppose $\mathbf{G} \to \mathbf{D}_2$ contains an oriented cycle with net length $2k \neq 0$. Then PCSP($\widetilde{\mathbf{D}_{2k}}$, **NAE**) reduces to PCSP($\widehat{\mathbf{G}}$, **NAE**).

In order to prove this fact, we will need a powerful result from [HHMN88], together with some notions from there. Firstly, note that every oriented path, say on vertices $s_1, \dots, s_n$, has (at most) two *endpoints* s_1, s_n, and this path can be traversed in two ways: from s_1 to s_n, and the other way around. We say that an oriented path is *minimal* if no subgraph that is also an oriented path has a strictly larger net length.

Theorem 5.29 ([HHMN88, Claim 1]). Let $\mathbf{P}, \mathbf{P}'$ be minimal oriented paths with equal net length. There exists an oriented path $\mathbf{Q}$ so that $\mathbf{Q} \to \mathbf{P}$ and $\mathbf{Q} \to \mathbf{P}'$, where these homomorphisms map the endpoints of $\mathbf{Q}$ to the endpoints of $\mathbf{P}$ and $\mathbf{P}'$ respectively.

Corollary 5.30. Let $\mathbf{P}_1, \dots, \mathbf{P}_n$ be minimal oriented paths with equal net length. There exists an oriented path $\mathbf{Q}$ so that $\mathbf{Q} \to \mathbf{P}_i$ for all $i \in [n]$, where these homomorphisms map the endpoints of $\mathbf{Q}$ to the endpoints of $\mathbf{P}_i$.

Proof. Note in general that for oriented paths $\mathbf{Q}, \mathbf{P}$, if $\mathbf{Q} \to \mathbf{P}$ by a homomorphism that maps endpoints to endpoints, then the net length of $\mathbf{Q}$ is equal to the net length of $\mathbf{P}$. To see why suppose that $\mathbf{Q}$ has vertices $q_1, \dots, q_k$ and $\mathbf{P}$ has vertices $p_1, \dots, p_\ell$, so that if we ignore the orientations of the edges the graphs are $q_1 - \cdots - q_k$ and $p_1 - \cdots - p_\ell$ respectively, and that we have a homomorphism $h : \mathbf{Q} \to \mathbf{P}$ where $h(q_1) = p_1$ and $h(q_k) = p_\ell$. By induction on i, we can prove that the oriented path induced by $q_1, \dots, q_i$ has the same net length as the oriented path induced by $h(q_1), h(q_2), \dots, h(q_i)$. This then implies that $\mathbf{Q}$ and $\mathbf{P}$ have the same net length. Furthermore, if $\mathbf{P}$ is minimal, then $\mathbf{Q}$ must be as well: any subgraph witnessing non-minimality in $\mathbf{Q}$ is mapped by h to a subgraph witnessing non-minimality in $\mathbf{P}$.

This means that we can apply Theorem 5.29 $n - 1$ times to get the desired path. □

We will need some more notations in what follows. For every oriented path we will implicitly choose a traversal direction. The endpoint of an oriented path from which the traversal begins will be called the *near endpoint*, whereas the other endpoint will be called the *far endpoint*.

This being given, for two oriented paths $\mathbf{P}, \mathbf{Q}$, we denote by $\mathbf{P} + \mathbf{Q}$ the oriented path created by taking the disjoint union of $\mathbf{P}$ and $\mathbf{Q}$, then identifying the far endpoint of $\mathbf{P}$ with the near endpoint of $\mathbf{Q}$. The oriented path $\mathbf{P} - \mathbf{Q}$ is similar, but we identify the far endpoint of $\mathbf{P}$ with the far endpoint of $\mathbf{Q}$. In both of these paths the traversal direction starts from the near endpoint of $\mathbf{P}$. Also, for the oriented path $\circ\!\!\to\!\!\circ$, the traversal direction is taken to be from the source of the edge to the target.

We now use Corollary 5.30 to prove the following important technical lemma.

Lemma 5.31. Fix an oriented cycle $\mathbf{C}$ with net length $k > 0$. There exists an oriented path $\mathbf{P}$ and vertices $a_0, \dots, a_{k-1}, a_k = a_0$ in $\mathbf{C}$ so that the following holds.

1. For each $i \in \{0, \dots, k-1\}$, $\mathbf{P} - \mathbf{P}$ maps homomorphically to $\mathbf{C}$ in such a way that the endpoints of $\mathbf{P} - \mathbf{P}$ map to a_i.
2. For each $i \in \{0, \dots, k-1\}$, $\circ\!\!\to\!\!\circ + \mathbf{P} - \mathbf{P}$ maps homomorphically to $\mathbf{C}$ in such a way that the near and far endpoints of $\circ\!\!\to\!\!\circ + \mathbf{P} - \mathbf{P}$ map to a_i and a_{i+1} respectively.

Proof. Suppose $\mathbf{C}$ has n vertices. We will create minimal oriented paths $\mathbf{Q}_i, \mathbf{R}_i, \mathbf{S}_i$, all of which will have net length n, so that $\mathbf{Q}_i - \mathbf{Q}_i$ maps to $\mathbf{C}$ so that the endpoints are taken to a_i, and $\multimap + \mathbf{R}_i - \mathbf{S}_i$ maps to $\mathbf{C}$ so that the near and far endpoints are taken to a_i and a_{i+1} respectively. Applying Corollary 5.30 to all these paths yields the required $\mathbf{P}$.

We first construct an *infinite* oriented path $\mathbf{C}^\star$ in the following way. Start from some vertex in $\mathbf{C}$, then go around the cycle in a consistent direction forever, taking note of the orientation of the edges encountered along the way. The graph $\mathbf{C}^\star$ will be an oriented version of $\mathbf{L}_\omega$ (hence the vertex set is $\mathbb{N}$), with the orientations being chosen to match the orientations found in $\mathbf{C}$ in the traversal mentioned above. Note that $\mathbf{C}^\star$ maps into $\mathbf{C}$ in a natural way, with the preimage of each vertex in $\mathbf{C}$ being an infinite set of vertices in $\mathbf{C}^\star$, spaced regularly every n vertices. Thus, it is sufficient to choose $a_0, \ldots, a_k$ in $\mathbf{C}^\star$ and $\mathbf{Q}_i, \mathbf{P}_i, \mathbf{R}_i$ so that they map appropriately into $\mathbf{C}^\star$ — provided that a_0 and a_k are n vertices apart. Then, when mapping $\mathbf{C}^\star$ into $\mathbf{C}$ the images of a_0 and a_k coincide.

For every vertex i of $\mathbf{C}^\star$, we define its *level*, $\mathrm{level}(i)$, in the following way: $\mathrm{level}(0) = 0$, and $\mathrm{level}(i+1) = \mathrm{level}(i) + 1$ if $i \to i+1$, whereas $\mathrm{level}(i+1) = \mathrm{level}(i) - 1$ if $i + 1 \to i$. We first claim that we can choose a starting point in $\mathbf{C}$ and a traversal direction so that $\mathrm{level}(i) \geq 0$ for all $i \in \mathbb{N}$, with equality only for $i = 0$. First observe that, by construction $\mathrm{level}(i+1) - \mathrm{level}(i) = \mathrm{level}(i+n+1) - \mathrm{level}(i+n)$ — since the difference is $+1$ or -1 depending on the orientation of the same edge in $\mathbf{C}$. Next observe that $|\mathrm{level}(n) - \mathrm{level}(0)| = k$ — and the sign depends on the traversal direction that we choose for $\mathbf{C}$. So choose the direction that makes $\mathrm{level}(n) - \mathrm{level}(0) = k$; for this choice of direction, in general we have that $\mathrm{level}(i+n) = \mathrm{level}(i) + k$. In particular this means we need only prove that $\mathrm{level}(i) \geq 0$ for $0 \leq i < n$. Next, suppose that $i_0 = \arg\min_{0 \leq i < n} \mathrm{level}(i)$, where we choose the largest possible i_0 in case there are multiple. If we start the traversal of $\mathbf{C}$ from (the preimage of) i_0, then it is easy to see that the sequence of levels is truncated so that it starts with i_0, and decreased by $\mathrm{level}(i_0)$. This means that all the levels become non-negative, with only 0 having level 0.

Thus we will assume from now on that $\mathrm{level}(i) \geq 0$ for all $i \in \mathbb{N}$, with equality only for $i = 0$. Observe also that, since $\mathrm{level}(i+n) = \mathrm{level}(i) + k$ and $k > 0$, the levels of vertices get arbitrarily large.

All of the paths we will chose will be segments of $\mathbf{C}^\star$ i.e. paths induced by $x, x+1, \ldots, y$ for some $x \leq y$. The net length of such a path is $|\mathrm{level}(x) - \mathrm{level}(y)|$. Thus, a path of this kind is minimal if and only if $\mathrm{level}(x) \leq \mathrm{level}(i) \leq \mathrm{level}(y)$ for all $x \leq i \leq y$. We will specify our paths by their endpoints, so the path $[x, y]$ is the path induced by $x, x+1, \ldots, y$, for $x \leq y$. The near endpoint is taken to be x, and the far to be y. Of course the path $[x, y]$ embeds homomorphically into $\mathbf{C}^\star$ such that x and y are mapped to themselves. Note that $[x, z] - [y, z]$ similarly maps homomorphically to $\mathbf{C}^\star$ so that x and y map to themselves — our paths will use this fact.

Choosing a_i. For every $i \in \{0, \ldots, k\}$, choose a_i to be the last vertex with level equal to i. The only condition we need to check is that $a_0 + n = a_k$. Since $\mathrm{level}(i) \geq 0$ with equality only for $i = 0$, we have $a_0 = 0$. On the other hand in general $\mathrm{level}(i+n) = \mathrm{level}(i) + k$. Thus $\mathrm{level}(n) = k$, and for all $i > n$, $\mathrm{level}(i) = \mathrm{level}(i-n) + k > 0 + k$. So $a_k = n$, as required.

While this is all we need now, we use some more properties of a_i later. For convenience, extend the definition of a_i to $i > k$. Also, let b_i be the *first* vertex with level equal to i.

The first fact we will use is that $a_i < b_{i+n}$. To see why this is the case, assume $a_i \geq b_{i+n}$, and let i be the minimal value for which this holds. Since $\text{level}(x+n) = \text{level}(x) + k$ in general, we have that $i < k$ by minimality. Since $\text{level}(x) \geq k$ for all $x \geq n$, we have that $a_i < n$, and hence $b_{i+n} < n$. But this is impossible: $\text{level}(b_{i+n}) = i + n$ by definition, $\text{level}(0) = 0$ and the levels change by at most 1 with every successive vertex.

Thus, for any x where $a_{i-1} < x \leq a_i$ and with $\text{level}(x) = i$, the path $[x, b_{i+n}]$ is well-defined. Furthermore, it must be minimal. To see why suppose first there exists some y with $x \leq y \leq b_{i+n}$ but $\text{level}(y) < \text{level}(x) = i$. Since the levels change by at most 1 and $\text{level}(x) = i$, we can assume without loss of generality that $\text{level}(y) = i - 1$. But this contradicts the choice of a_{i-1}. The fact that no y between x and b_{i+n} exists with $\text{level}(y) > \text{level}(b_{i+n}) = i + n$ is proved analogously. All the paths we choose will be of this form, so we can assume that they are well-defined and minimal. Furthermore, such a path always has net length equal to $|\text{level}(x) - \text{level}(b_{i+n})| = |i - (i+n)| = n$, as required.

Choosing $\mathbf{Q}_i$. We want to choose a path $\mathbf{Q}_i$ which is minimal, has net length n, and for which $\mathbf{Q}_i - \mathbf{Q}_i$ maps homomorphically to $\mathbf{C}^\star$ with endpoints mapped to a_i. We will choose the path $[a_i, a_{i+n}]$. By the facts we have already listed it is easy to see that this satisfies all our criteria.

Choosing $\mathbf{R}_i, \mathbf{S}_i$. We want to choose paths $\mathbf{R}_i, \mathbf{S}_i$ that are minimal, with net length n, and for which $\multimap + \mathbf{R}_i - \mathbf{S}_i$ maps homomorphically to $\mathbf{C}^\star$ so that the near and far endpoints are taken to a_i and a_{i+1} respectively. Take $\mathbf{R}_i = [a_i + 1, b_{i+1+n}]$ and $\mathbf{Q}_i = [a_{i+1}, b_{i+1+n}]$. By all the properties established so far, these paths are minimal with appropriate net length, and $\mathbf{R}_i - \mathbf{S}_i$ maps to $\mathbf{C}^\star$ so that the near and far endpoints map to $a_i + 1$ and a_{i+1} respectively. Since $a_i \to (a_i + 1)$ — otherwise $\text{level}(a_i + 1) = i - 1$, and since the levels become arbitrarily large this would mean that a_i is not the last vertex with level i — it follows that $\multimap + \mathbf{R}_i - \mathbf{S}_i$ maps homomorphically to $\mathbf{C}^\star$ with endpoints mapping appropriately □

We are almost ready to describe the reduction we want. We first define a logical formula that encodes a path mapping homomorphically to $\mathbf{G}$, in structures of form $\widehat{\mathbf{G}}$.

Definition 5.32. Consider an oriented path $\mathbf{P}$; without loss of generality suppose its vertex set is $[n]$, its edge set is E, its near endpoint is 1 and its far endpoint is n. Define the primitive-positive formula $v_1 \xrightarrow{\mathbf{P}} v_n$ in the language containing a single ternary relation R by

$$\exists v_2 \cdots \exists v_{n-1} \cdot \bigwedge_{(i,j) \in E} R(v_i, v_i, v_j).$$

Lemma 5.33. $\xrightarrow{\mathbf{P}}$ interpreted in $\widehat{\mathbf{G}}$ yields a relation that admits (u, v) if and only if $\mathbf{P}$ maps to $\mathbf{G}$ such that its near endpoint is mapped to u, and its far endpoint is mapped to v.

Proof. Indeed, the formula can be seen as a first-order definition of the existence of such a homomorphism, with the value of the homomorphism on vertex i is given by the value of v_i in $\xrightarrow{\mathbf{P}}$. □

Using this formula we are ready to create the primitive-positive formula that will give rise to our reduction.

Lemma 5.34. Fix a digraph $\mathbf{G}$ that admits an oriented cycle $\mathbf{C}$ with net length $2k > 0$. There exists a primitive-positive formula $\phi(x, y, z)$ such that the following holds.

1. ϕ interpreted in $\mathbf{NAE}$ yields $\mathbf{NAE}$.
2. ϕ interpreted in $\widehat{\mathbf{G}}$ yields a relational structure $\mathbf{T}$ for which $\widehat{\mathbf{D}_{2k}} \to \mathbf{T}$.

Proof. Let $\mathbf{P}$ be the path and $a_0, \ldots, a_{2k} = a_0$ be the vertices given by Lemma 5.31 applied to $\mathbf{C}$. Take $\phi(x, y, z)$ to be

$$\exists x', y', z' \cdot R(x', y', z') \wedge x \xrightarrow{\mathbf{P}-\mathbf{P}} x' \wedge y \xrightarrow{\mathbf{P}-\mathbf{P}} y' \wedge z \xrightarrow{\mathbf{P}-\mathbf{P}} z'.$$

This is not a primitive-positive formula, but it is equisatisfiable with one — just "pull out" the quantifiers hidden in $\star \xrightarrow{\mathbf{P}-\mathbf{P}} \star$.

First we show that ϕ interpreted in $\mathbf{NAE}$ yields $\mathbf{NAE}$. Observe that $x \xrightarrow{\mathbf{P}-\mathbf{P}} x'$ is true in $\mathbf{NAE} = \widehat{\fish}$ if and only if $\mathbf{P} - \mathbf{P}$ maps homomorphically to $\fish$ with endpoints mapping to x and x'. But due to the structure of $\fish$, and as $\mathbf{P} - \mathbf{P}$ has net length equal to 0, this is only true when $x = x'$! Hence $\phi(x, y, z)$ is true in $\widehat{\fish}$ if and only if (x, y, z) belongs to the relation of $\widehat{\fish} = \mathbf{NAE}$ — which is exactly what we wanted to show.

Second, we show that ϕ interpreted in $\widehat{\mathbf{G}}$ yields a relational structure $\mathbf{T}$ for which $\widehat{\mathbf{D}_{2k}} \to \mathbf{T}$. Indeed, since it is easy to see that the relation defined by ϕ is symmetric (provided that R is symmetric), it is sufficient to show that $\phi(a_i, a_i, a_{i+1})$ is true interpreted in $\widehat{\mathbf{G}}$ — then the homomorphic image of $\widehat{\mathbf{D}_{2k}}$ is given by the vertices $a_0, \ldots, a_{2k-1}$. Let us see why this is the case. By Lemma 5.31, there exists a vertex v such that the edge $a_i \to v$ is in $\mathbf{G}$, and $\mathbf{P} - \mathbf{P}$ maps homomorphically to $\mathbf{G}$, with near and far endpoints mapping to v and a_{i+1} respectively. Hence $v \xrightarrow{\mathbf{P}-\mathbf{P}} a_{i+1}$. Thus we see that $\phi(a_i, a_i, a_{i+1})$ is true when interpreted in $\widehat{\mathbf{G}}$ — take $x' = y' = a_i$ and $z' = v$. □

With all of this in hand we are finally ready to complete the proof of Proposition 5.27.

Proof of Proposition 5.27. As discussed in the preliminaries, if there exists a primitive positive formula ϕ such that ϕ interpreted in $\mathbf{A}$ yields $\mathbf{A}$', and ϕ interpreted in $\mathbf{B}$ yields $\mathbf{B}'$, then $\mathrm{PCSP}(\mathbf{A}', \mathbf{B}')$ reduces to $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ (provided all of these problems are well-defined). Thus Lemma 5.34 implies that $\mathrm{PCSP}(\mathbf{T}, \mathbf{NAE})$ reduces to $\mathrm{PCSP}(\widehat{\mathbf{G}}, \mathbf{NAE})$, for some $\mathbf{T}$ to which $\widehat{\mathbf{D}_{2k}}$ maps homomorphically. By homomorphic relaxation, we find that $\mathrm{PCSP}(\widehat{\mathbf{D}_{2k}}, \mathbf{NAE})$ reduces to $\mathrm{PCSP}(\widehat{\mathbf{G}}, \mathbf{NAE})$, as required. □

6. Functional Structures

In this chapter, we will give complexity classifications for PCSPs where the template is *functional*. These kinds of templates can be seen as generalisations of linear equations — a symmetric relation is functional if and only if no two tuples within it have Hamming distance exactly 1; and a relational structure is functional if all its relations are. We will give classifications for PCSP($\mathbf{A}, \mathbf{B}$) where $\mathbf{A}$ is symmetric, $\mathbf{B}$ is functional, and $(\mathbf{A}, \mathbf{B})$ satisfies two technical conditions we introduce called *additivity* and *dependency*. In particular, this holds if $\mathbf{A}$ is Boolean, or if $\mathbf{A}, \mathbf{B}$ have exactly one relation of arity at most 4, among other cases. This chapter will present results from the following previously published works.

[NŽ23a] Tamio-Vesa Nakajima and Stanislav Živný. 'Boolean symmetric vs. functional PCSP dichotomy'. In: *LICS*. IEEE, 2023, pp. 1–12. DOI: 10.1109/lics56636.2023.10175746.

[NŽ24c] Tamio-Vesa Nakajima and Stanislav Živný. 'On the Complexity of Symmetric vs. Functional PCSPs'. In: *ACM Trans. Algorithms* 20.4 (2024), 33:1–33:29. DOI: 10.1145/3673655.

The works [NŽ23a; NŽ24c] were produced in collaboration with my supervisor. I thank Marcin Kozik and Demian Banakh for useful discussions relating to these results.

6.1. Introduction

One of the most important Constraint Satisfaction Problems is solving *linear equations*. Recall that the problem of solving r-uniform equations over A, where A is some Abelian group (perhaps finite, perhaps infinite) is as follows: given a set of constraints of form $x_1 + \cdots + x_r = a$, where $x_1, \ldots, x_r$ are variables and $a \in A$ is a constant, find an assignment of values to variables that ensures the truth of all equations.

Example 6.1. Define, for $r \in \mathbb{N}_{\geq 1}$ and a group A, the relational structure $\mathbf{Eqn}_r(A)$, on domain A, and with a relation R^r_a for $a \in A$ given by

$$R^r_a = \{(x_1, \ldots, x_r) \mid x_1 + \cdots + x_r = a\}.$$

Then solving r-uniform equations over A is CSP($\mathbf{Eqn}_r(A)$).[1]

For any finite Abelian group A, one can solve CSP($\mathbf{Eqn}_r(A)$) is polynomial time via Gaussian elimination. One can also solve CSP($\mathbf{Eqn}_r(\mathbb{Z})$), by computing the *Smith Normal Form* of the matrix which encodes the linear equations [KB79]. Linear equations have many useful

[1] In the case where A is not finite, this structure is not finite.

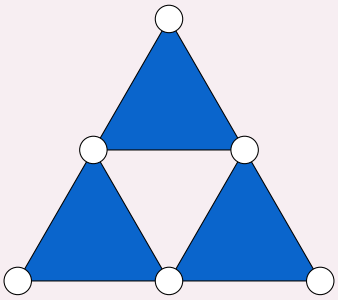
properties — however one that we will try to generalise in particular is the fact that *the final coordinate is uniquely determined by the other coordinates*. Furthermore, the relations are all symmetric: we may arbitrarily permute the coordinates in a relation without changing it.

This leads us to create the following definition.

Definition 6.2. We say that a relation R of arity r is *functional* if, for every tuple of values $x_1, \dots, x_{r-1}, y, z$, if $(x_1, \dots, x_{r-1}, y) \in R$ and $(x_1, \dots, x_{r-1}, z) \in R$, then $y = z$. In other words, $R \in A^r$ is functional if it is the graph of a, perhaps partial,[2] function $f : A^{r-1} \to A$, where f is defined on $x_1, \dots, x_{r-1}$ if there exists (a unique) y such that $(x_1, \dots, x_{r-1}, y) \in R$ — and indeed $f(x_1, \dots, x_{r-1}) = y$. A relational structure $\mathbf{A}$ is called functional if all its relations are functional.

Remark 6.3. For symmetric R, R is functional if and only if no two tuples in R differ in exactly one coordinate. In other words, the Hamming distance between two tuples $t, u \in R$ is either 0 or > 1. We will mostly consider the notion of functionality for symmetric relations.

Example 6.4. Consider the 3-uniform hypergraph.



Seen as a relational structure, the hypergraph is functional. (Indeed, a 3-uniform hypergraph is functional if and only if it is linear, cf. Definition 3.15.)
Fix a set S where, if $i, j \in S$ and $i \neq j$, then $|i - j| > 1$. Then the following is a functional relational structure:

$$(\{0, 1\}; \{(x_1, \dots, x_r) \mid x_1 + \dots + x_r \in S\}).$$

On the other hand, many relational structures are not functional. A simple example is the k-clique, $\mathbf{K}_k$, for $k \geq 3$. Indeed, a graph is functional if and only if it is a disjoint union of edges, loops and isolated vertices.

Our goal is to classify PCSP($\mathbf{A}, \mathbf{B}$) when $\mathbf{B}$ is functional and $\mathbf{A}$ is symmetric. (It is well-known, cf. [BBB21], that if $\mathbf{A}$ is symmetric, one can take the maximal symmetric substructure of $\mathbf{B}$ to get an equivalent problem. Since removing tuples from $\mathbf{B}$ keeps $\mathbf{B}$ functional, we will assume throughout that $\mathbf{B}$ is also symmetric.) How can we use functionality? Consider

[2] Indeed, perhaps "partial-functional" would be a better name!

some polymorphism $f \in \mathrm{Pol}^{(n)}(\mathbf{A}, \mathbf{B})$; we want to use functionality to constrain the values of f. Suppose the template $(\mathbf{A}, \mathbf{B})$ admits a relation R of arity r, and suppose that $M \in A^{r \times n}$ is a matrix whose columns are tuples of $R^{\mathbf{A}}$. Applying f to the rows of M gives us a tuple of $R^{\mathbf{B}}$ by definition — however *the evaluation of the last row is uniquely determined by the evaluations of the first* $n - 1$ *rows.*

Thus we will reason step-by-step in this fashion, determining more and more of the values of some polymorphism from just a few of its values. We will codify a formal logic for this kind of reasoning in Section 6.2. (This is not a mere whim, we will do induction on "proofs" in this system to bootstrap more and more information about our polymorphism!) The ultimate goal will be to prove either that all polymorphisms are "inexpressive" in a formal sense (in which case $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is NP-hard), or to use a sufficiently "expressive" polymorphism f to impose a group structure on $\mathbf{B}$. In this latter case, the problem will be solvable by reducing to solving modular linear equations — thus showing that linear equations are, in some sense, the "essential" functional problem.

Unfortunately, we can immediately see some issues with this approach. If $\mathbf{A}$ is very sparse (in some suitably defined way), intuitively it gives us very little information about $\mathbf{B}$ at all. Thus we will need to impose some technical constraints on $\mathbf{A}$ and $\mathbf{B}$, which we call *additivity* and *dependency*, whose definition we postpone until Section 6.2. We will then show the following theorem.

Theorem 6.5. Suppose $\mathbf{A}$ is symmetric, $\mathbf{B}$ is functional and $\mathbf{A} \to \mathbf{B}$. Suppose also that $(\mathbf{A}, \mathbf{B})$ is additive and dependent. Then $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is either NP-hard, or solvable in polynomial time by reducing (by homomorphic sandwich, cf. Corollary 2.5) to modular linear equations.

We will prove this theorem in Section 6.3.

We will generally prove that $(\mathbf{A}, \mathbf{B})$ is additive and dependent using a stronger condition, which we call *super-connectedness*, whose definition we likewise postpone to Section 6.2. The key facts we will use are the following.

Proposition 6.6. Suppose $\mathbf{A}$ is symmetric, $\mathbf{B}$ is functional, with $\mathbf{A} \to \mathbf{B}$. If $\mathbf{A}$ is super-connected then $(\mathbf{A}, \mathbf{B})$ is additive and dependent. Furthermore, the following classes of $\mathbf{A}$ are super-connected:

1. Connected 3- or 4-uniform hypergraphs.
2. Structures where there exists a relation $R^{\mathbf{A}}$ of arity 3 or higher, such that, for any distinct $a, b \in A$, there exists a tuple of $R^{\mathbf{A}}$ containing both a and b.

These facts are enough to prove the following two corollaries.

Corollary 6.7. If $\mathbf{A}$ is an r-uniform hypergraph with $r \leq 4$, and $\mathbf{B}$ is functional such that $\mathbf{A} \to \mathbf{B}$, then $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is either NP-hard or solvable in polynomial time.

Proof. If $r = 1$ then the problem PCSP($\mathbf{A}, \mathbf{B}$) is always trivially solvable in polynomial time (each constraint relation is unary). If $r = 2$, then since $\mathbf{A}$ is a functional graph we have that $\mathbf{A}$ is the disjoint union of loops and undirected edges. Thus CSP($\mathbf{A}$) is easy to solve (if $\mathbf{A}$ has any loops every instance is a YES instance; if $\mathbf{A}$ has any edges but no loops then CSP($\mathbf{A}$) is equivalent to 2-colouring; if $\mathbf{A}$ is empty then every nonempty instance is a NO instance). Hence PCSP($\mathbf{A}, \mathbf{B}$) is also solvable in polynomial time.

Thus consider $r = 3, 4$. Suppose $\mathbf{A} = \mathbf{A}_1 + \cdots + \mathbf{A}_k$, where each of $\mathbf{A}_1, \ldots, \mathbf{A}_k$ is connected. If PCSP($\mathbf{A}_i, \mathbf{B}$) is NP-hard for any $i \in [k]$, as $\mathbf{A}_1 \to \mathbf{A}, \ldots, \mathbf{A}_k \to \mathbf{A}$, we have that PCSP($\mathbf{A}, \mathbf{B}$) is NP-hard by homomorphic relaxation. Otherwise, by Proposition 6.6 and Theorem 6.5, all of PCSP($\mathbf{A}_1, \mathbf{B}$), ..., PCSP($\mathbf{A}_k, \mathbf{B}$) are solvable in polynomial time. Then we claim that the following is an algorithm for PCSP($\mathbf{A}, \mathbf{B}$): On input $\mathbf{H}$, split $\mathbf{H}$ into connected components $\mathbf{H} = \mathbf{H}_1 + \cdots + \mathbf{H}_t$. For each component $\mathbf{H}_i$, try to apply each of the polynomial time algorithms for PCSP($\mathbf{A}_j, \mathbf{B}$) until one successfully outputs a homomorphism from $\mathbf{H}_i$ to $\mathbf{B}$. Combine these homomorphisms to find one from $\mathbf{H}$ to $\mathbf{B}$.

To see why this algorithm succeeds, we must prove that, under the promise that $\mathbf{H} \to \mathbf{A}$, for every $\mathbf{H}_i$ there exists $\mathbf{A}_j$ such that $\mathbf{H}_i \to \mathbf{A}_j$. But this is true: since $\mathbf{H}_i$ is connected, its image through the promised homomorphism from $\mathbf{H}$ to $\mathbf{A}$ must also be connected. It must therefore be contained within one of the connected components $\mathbf{A}_j$. Hence $\mathbf{H}_i \to \mathbf{A}_j$ for some $j \in [k]$. Thus, our algorithm will find an $\mathbf{A}_j$ for every $\mathbf{H}_i$ such that $\mathbf{H}_i \to \mathbf{A}_j$, for which the algorithm for PCSP($\mathbf{A}_j, \mathbf{B}$) will produce a valid homomorphism to $\mathbf{B}$. □

Corollary 6.8. If $\mathbf{A}$ is Boolean and symmetric, and $\mathbf{B}$ is functional such that $\mathbf{A} \to \mathbf{B}$, then PCSP($\mathbf{A}, \mathbf{B}$) is either NP-hard or solvable in polynomial time.

Proof. If $\mathbf{A}$ has a relation of arity 3 or higher that contains the two distinct values in the domain of $\mathbf{A}$, then by combining Theorem 6.5 with Proposition 6.6 the conclusion follows.

Suppose therefore that every relation of arity 3 or higher contains only one value. The higher-arity constraint relations are therefore equivalent to unary constant constraint relations. The binary relations in $\mathbf{A}$ are the edge relations of functional graphs, which are disjoint unions of edges, loops or isolated vertices. The only possible relations are therefore $\{(0,0)\}, \{(1,1)\}, =, \neq$. Thus we see that CSP($\mathbf{A}$) can be solved in polynomial time (indeed it can be solved by solving linear equations modulo 2) — whence the conclusion. □

6.2. Deductions

In this chapter, we will prove Proposition 6.6. Before restating this theorem, let us first define all of the notions that are necessary for properly understanding it: additivity, dependency and super-connectedness.

6.2.1. Additivity

Often when dealing with polymorphisms of a Boolean relational structure, one tries to relate $f(S \cup T)$ with $f(S)$ and $f(T)$ (where we see such a polymorphism as an assignment of values to subsets of $[n]$, as in Chapter 5). Indeed, this is the spirit of what we did for the case of Rainbow-free structures, as in Chapter 5. Notwithstanding, we want to deal with non-Boolean relational structures — we will therefore *flatten* the domain of the source relational structure to its 2-element slices.

Definition 6.9. Fix $f \in \mathrm{Pol}^{(n)}(\mathbf{A}, \mathbf{B})$. Take $(i, j) \in A^2$, and define $f_{ij} : 2^{[n]} \to B$ in the following way. For $S \subseteq [n]$, define $x_1, \ldots, x_n$ by $x_k = j$ if $k \in S$ and $x_k = i$ otherwise. Then

$$f_{ij}(S) = f(x_1, \ldots, x_n).$$

Next, define $f^p : 2^{[n]} \to B^{A^2}$ as follows:[3]

$$f^p(S)(i, j) = f_{ij}(S).$$

(Note that $f^p(S)$ can be seen as a matrix; we denote by $f^p(S)(i, j)$ the element on row i and column j.)

With this flattening in hand, we will investigate those relational structures where $f^p(S \cup T)$ is uniquely determined by $f^p(S)$ and $f^p(T)$.

Definition 6.10. Fix $\mathbf{A} \to \mathbf{B}$ with $\mathbf{A}$ symmetric and $\mathbf{B}$ functional. We say that $(\mathbf{A}, \mathbf{B})$ is *additive* if there exists some operator $+ : B^{A^2} \times B^{A^2} \to B^{A^2}$ such that, for any $f \in \mathrm{Pol}^{(n)}(\mathbf{A}, \mathbf{B})$ and *disjoint* $S, T \subseteq [n]$, we have

$$f^p(S) + f^p(T) = f^p(S \cup T).$$

Perhaps unexpectedly, in this setting addition immediately implies subtraction. We will later use this fact to get inverse elements in a group.

Lemma 6.11. Suppose $(\mathbf{A}, \mathbf{B})$ is additive. Then there exists an operator $- : B^{A^2} \times B^{A^2} \to B^{A^2}$ such that, for any $f \in \mathrm{Pol}^{(n)}(\mathbf{A}, \mathbf{B})$ and $S \subseteq T \subseteq [n]$, we have

$$f^p(T) - f^p(S) = f^p(T \setminus S).$$

Proof. Let $\overline{S}$ denote $[n] \setminus S$, and (nonstandardly) let $\overline{M}$ denote the transpose of the matrix M. Note that

$$f^p(\overline{S}) = \overline{f^p(S)}. \tag{6.1}$$

Indeed, $f_{ij}(S) = f_{ji}(\overline{S})$ by definition, which implies (6.1). Thus set

$$x - y = \overline{\overline{x} + y}.$$

[3] f^p stands for "f evaluated on pairs of values".

Then, for $S \subseteq T$,

$$f^p(T) - f^p(S) = \overline{\overline{f^p(T)} + f^p(S)} = f^p(\overline{\overline{T} \cup S}) = f^p(T \setminus S). \qquad \square$$

Now, we will use this fact to show some interesting structural results. Recall that in Chapter 5, a key part of the proof was showing the existence of sets of bounded size with certain images. We will prove two results of this kind.

Lemma 6.12. Suppose $(\mathbf{A}, \mathbf{B})$ is additive, and take $f \in \mathrm{Pol}^{(n)}(\mathbf{A}, \mathbf{B})$. Suppose $\mathcal{A} \subseteq 2^{[n]}$ is a family of at least $|B|^{|A|^2}$ disjoint sets. Then there exists some nonempty $\mathcal{B} \subseteq \mathcal{A}$ with $f^p(\bigcup_{B \in \mathcal{B}} B) = f^p(\varnothing)$.

The proof of this fact is very similar to the famous exercise, "show that any sequence of n integers has a contiguous nonempty subsequence with sum divisible by n". The source of this problem is Vázsonyi and Sved, according to Erdös (according to [AZ04]).

Proof. Let $\mathcal{A} = \{A_1, \ldots\}$ and define $S_t = \bigcup_{1 \le k < t} A_k$ for $t \ge 0$. Since $\mathcal{A}$ has no fewer elements than the codomain of f^p, there exists $0 \le i < j$ such that $f^p(S_i) = f^p(S_j)$. But then

$$f^p(S_j \setminus S_i) = f^p(S_j) - f^p(S_i) = f^p(S_i) - f^p(S_i) = f^p(S_i \setminus S_i) = f^p(\varnothing),$$

and furthermore $S_j \setminus S_i = \bigcup_{i < k \le j} A_k$, and hence is a nonempty union of elements of $\mathcal{A}$, which completes the proof. $\square$

Lemma 6.13 (Subset Principle). Suppose $(\mathbf{A}, \mathbf{B})$ is additive, and take $f \in \mathrm{Pol}^{(n)}(\mathbf{A}, \mathbf{B})$ and $S \subseteq [n]$. There exists some set $T \subseteq S$ with $f^p(S) = f^p(T)$ and $|S| \le |B|^{|A|^2}$.

Proof. Suppose S is a minimal counterexample. If $|S| \le |B|^{|A|^2}$ then we could take $T = S$, and S would be no counterexample at all — so $|S| > |B|^{|A|^2}$. Thus, let $S = \{a_1, \ldots\}$ and apply Lemma 6.12 to $\{\{a_1\}, \ldots\}$ to find a nonempty subset $Z \subseteq S$ with $f^p(Z) = f^p(\varnothing)$. Now take $T = S \setminus Z$; indeed

$$f^p(S \setminus Z) = f^p(S) - f^p(Z) = f^p(S) - f^p(\varnothing) = f^p(S \setminus \varnothing) = f^p(S). \qquad \square$$

6.2.2. Dependency

While additivity tells us how the values of f^p are related to each other, this is of little use if these values do not tell us anything about the value of f. The notion of *dependency* bridges that gap. Note that in the following we use the notation $(f(x_1, \ldots, x_n) \mid x_1 \in A_1, \ldots, x_n \in A_n)$ to denote a tuple indexed by $A_1 \times \cdots \times A_n$, with value $f(x_1, \ldots, x_n)$ at $(x_1, \ldots, x_n)$.

Definition 6.14. For $\mathbf{A} \to \mathbf{B}$ with $\mathbf{A}$ symmetric and $\mathbf{B}$ functional, we say that $(\mathbf{A}, \mathbf{B})$ is *dependent* if there exists some map h such that, for every $f \in \mathrm{Pol}^{(n)}(\mathbf{A}, \mathbf{B})$, we have that

$$f(x_1, \dots, x_n) = h\left(f(\alpha(x_1), \dots, \alpha(x_n)) \mid \alpha : A \to S, S \subseteq \binom{A}{\leq 2}\right),$$

where $\binom{A}{\leq 2}$ is the set of nonempty subsets of A of size at most 2.

(Note that we have a unique h which has this property for every f.)

Example 6.15. If $\mathbf{A}$ is Boolean then $(\mathbf{A}, \mathbf{B})$ is always dependent.
Suppose $\mathbf{A} = \mathbf{B} = \mathbf{Eqn}_3(\mathbb{Z}_5)$; hence $f \in \mathrm{Pol}^{(n)}(\mathbf{A}, \mathbf{B})$ if and only if $f(x_1, \dots, x_n) = a_1 x_1 + \dots + a_n x_n \bmod 5$ for some $a_1, \dots, a_n \in \mathbb{Z}_5$ with $a_1 + \dots + a_5 \equiv 1 \bmod 5$. Note that $(\mathbf{A}, \mathbf{B})$ is dependent. Indeed, consider the family of functions $\alpha_x : \mathbb{Z}_5 \to \{0, x\}$ given by

$$\alpha_x(y) = \begin{cases} x, & x = y, \\ 0, & \text{otherwise.} \end{cases}$$

Then we observe that

$$f(x_1, \dots, x_n) = \sum_{i=0}^{n} a_i x_i = \sum_{x=0}^{4} \sum_{x_i = x} a_i x_i = \sum_{x=0}^{4} \sum_{i=0}^{n} a_i \alpha_x(x_i) = \sum_{x=0}^{4} f(\alpha_x(x_1), \dots, \alpha_x(x_n)).$$

Hence we see that f is dependent.

Notationally, it will be convenient to see functions of form $A^n \to B$ in a different way. We can see a tuple $(a_1, \dots, a_n) \in A^n$ as a partition of $[n]$ into parts $S_a = \{i \mid a_i = a\}$ for every $a \in A$ — in other words as a function from A to $2^{[n]}$ where the images through A form a partition; or respectively a generalised tuple $(S_a \mid a \in A)$ where S_a forms a partition. Thus we will have two equivalent views of functions $f : A^n \to B$: we will evaluate them either at inputs $x_1, \dots, x_n \in A$, or on a partition $(S_a \mid a \in A)$. In particular, defining $\pi^{-1}(S) = \{x \mid x \in S\}$ and $\pi^{-1}(S_a \mid a \in A) = (\pi^{-1}(S_a) \mid a \in A)$, one can easily show that

$$f^\pi(S_a \mid a \in A) = f(\pi^{-1}(S_a) \mid a \in A) = f(\pi^{-1}(S_a \mid a \in A)),$$

and hence $f^\pi = f \circ \pi^{-1}$.

The notation from Definition 6.14 for "the information gained from evaluating a polymorphism on all tuples that result from mapping the input into an (at most) two element set" is rather unwieldy. We will introduce the following notation for more convenience.

Definition 6.16. Fix $f \in \mathrm{Pol}^{(n)}(\mathbf{A}, \mathbf{B})$. We define $f^\star : A^n \to (B^{A^2})^A$ as follows: for any partition $(S_a \mid a \in A)$ of $[n]$, we set

$$f^\star(S_a \mid a \in A) = (f^p(S_a) \mid a \in A).$$

Lemma 6.17. Suppose that $(\mathbf{A}, \mathbf{B})$ is dependent and additive. Then, there exists a map $h : (B^{A^2})^A \to B$ such that, for any $f \in \mathrm{Pol}^{(n)}(\mathbf{A}, \mathbf{B})$, we have $f = h \circ f^\star$.

Proof. Consider any $\alpha : A \to S$ with $S \in \binom{A}{\leq 2}$. We claim that there exists some map h_α such that, for any $f \in \mathrm{Pol}^{(n)}(\mathbf{A}, \mathbf{B})$ and $x_1, \ldots, x_n \in A$, we have

$$f(\alpha(x_1), \ldots, \alpha(x_n)) = h_\alpha(f^\star(x_1, \ldots, x_n)).$$

Indeed, if such a function exists for every α, then by composing h_α with the map guaranteed by the dependency of f, we will get the map h that we are after. Thus we focus on constructing h_α. Suppose throughout that $(S_a \mid a \in A)$ is the partition of $[n]$ induced by $x_1, \ldots, x_n$.

First consider the case where $S = \{s\}$. In this case, $f(\alpha(x_1), \ldots, \alpha(x_n)) = f(s, \ldots, s)$ — and this element appears as an entry of $f^p(S_a)$ for any $a \in A$, namely $f_{ss}(S_a)$. Hence in this case h_α exists.

Next, suppose S is a 2-element set. Without loss of generality let $S = \{x, y\}$, and suppose that $A_i = \alpha^{-1}(i)$ for $i \in \{x, y\}$ — note that the sets A_x, A_y depend *only* on α, not on f. Note that we want to find $f(\alpha(x_1), \ldots, \alpha(x_n))$ in terms of $(f^p(S_a) \mid a \in A)$. Since the tuple $\alpha(x_1), \ldots, \alpha(x_n)$ contains an i at the indices in A_i for $i \in \{x, y\}$, this value is just $f_{xy}(\bigcup_{a \in A_y} S_a)$. This value is contained within $f^p(\bigcup_{a \in A_y} S_a)$, which by additivity is just

$$f^p\left(\bigcup_{a \in A_y} S_a\right) = \sum_{a \in A_y} f^p(S_a).$$

But note that this value is a function of $(f^p(S_a) \mid a \in A)$, uniquely determined by α and the operation $+$. Hence in this case h_α also exists.

Thus in all cases h_α exists, and the map we want can be created as a composite as described above. □

6.2.3. Formal system

We have now described the notions of additivity and dependency that we will use. Next we will describe a formal system that helps us reason about these notions. This system has both a syntax and a semantics — we will begin by describing its semantics.

Definition 6.18. Let $S \subseteq A^n$ be a set of tuples and $t \in A^n$ be a tuple, all of arity n. We write $S \vDash_{\mathbf{A},\mathbf{B}} t$ if there exists a function $h : B^S \to B$ (where B^S is the set of tuples of elements of B indexed by S), such that, for all polymorphisms $f \in \mathrm{Pol}^{(n)}(\mathbf{A}, \mathbf{B})$, we have that

$$f(t) = h(f(u) \mid u \in S).$$

We omit $\mathbf{A}, \mathbf{B}$ if they are obvious from context.

Definition 6.19. Fixing $\mathbf{A}$, we write

$$\Gamma_{\mathbf{A}} = \{(b, a, a), (a, b, a) \mid a, b \in A\},$$
$$\Delta^n_{\mathbf{A}} = \{(a_1, \ldots, a_n) \mid a_1, \ldots, a_n \in A, |\{a_1, \ldots, a_n\}| \leq 2\}.$$

We omit $\mathbf{A}$ if it is obvious from context.

We now observe that additivity and dependence are captured by the semantics of the formal system.

Proposition 6.20. If $\Gamma \vDash (a, a, b)$ for all $a, b \in A$, then $(\mathbf{A}, \mathbf{B})$ is additive for every $\mathbf{A} \to \mathbf{B}$ with $\mathbf{A}$ symmetric and $\mathbf{B}$ functional.

Proof. Suppose $h_{ab} : B^\Gamma \to B$ is the function that witnesses $\Gamma \vDash (a, a, b)$. We must define an operator $+ : B^{A^2} \times B^{A^2} \to B^{A^2}$ such that for every polymorphism $f \in \mathrm{Pol}^{(n)}(\mathbf{A}, \mathbf{B})$, and for every two disjoint sets $S, T \subseteq [n]$, we have $f^p(S) + f^p(T) = f^p(S \cup T)$.

First, note that it is sufficient to define an operator $\tilde{+} : B^{A^2} \times B^{A^2} \to B^{A^2}$ such that, for every disjoint $S, T \subseteq [n]$, we have $f^P(S) \mathbin{\tilde{+}} f^P(T) = f^p(\overline{S \cup T})$ — this is because we can then define $x + y = \overline{x \mathbin{\tilde{+}} y}$, where again $\bar{\cdot}$ has been used to denote transposition of a matrix. This is because $f^p(\overline{S}) = \overline{f^p(S)}$. Next, by applying the minor $\pi : [n] \to [3]$ where $\pi^{-1}(1) = S, \pi^{-1}(2) = T, \pi^{-1}(3) = [n] \setminus (S \cup T)$ to f, we can reduce to the case where $n = 3$, $S = \{1\}, T = \{2\}$. In other words, we must create an operator $\tilde{+}$ which tells us the value of $f(x, x, y)$ given the values of $f(x, y, x)$ and $f(y, x, x)$ for $x, y \in A$, independent of f. This operator is just the combination of all the functions h_{ab} that exist by assumption. □

Proposition 6.21. Suppose without loss of generality that $A = [a]$. If $\Delta^a \vDash (1, \ldots, a)$ then $(\mathbf{A}, \mathbf{B})$ is dependent for every $\mathbf{A} \to \mathbf{B}$ with $\mathbf{A}$ symmetric and $\mathbf{B}$ functional.

Proof. Suppose $h : B^{\Delta^a} \to B$ is the function that witnesses $\Delta^a \vDash (1, \ldots, a)$. We must create a function h' so that, for any $f \in \mathrm{Pol}^{(n)}(\mathbf{A}, \mathbf{B})$ and $x_1, \ldots, x_n \in A$, we have that

$$f(x_1, \ldots, x_n) = h'\left(f(\alpha(x_1), \ldots, \alpha(x_n)) \mid \alpha : A \to S, S \in \binom{A}{\leq 2}\right).$$

Observe that by applying the minor $\pi : [n] \to [a]$ to f where $\pi(i) = x_i$, we can reduce to the case $n = a, x_1 = 1, \ldots, x_n = n$. In other words, we must create a function h' for which

$$f(1, \ldots, a) = h'\left(f(\alpha(1), \ldots, \alpha(n)) \mid \alpha : A \to S, S \in \binom{A}{\leq 2}\right).$$

But note that the tuple $(\alpha(1), \ldots, \alpha(n))$ ranges over Δ^n for $\alpha : A \to S$ with $S \in \binom{A}{\leq 2}$. Hence the tuple

$$\left(f(\alpha(1), \ldots, \alpha(n)) \mid \alpha : A \to S, S \in \binom{A}{\leq 2}\right)$$

contains the exact same information as the tuple $(f(t) \mid t \in \Delta^n)$. This means that we can create h' from h by simply reordering the elements in the input tuple appropriately. □

While the semantics of the system is useful, we will only ever deduce things about it in certain prescribed ways. This will help us apply meta-reasoning on the proofs of statements in this logic later on. We codify this methodology in the following syntactic description of the formal system.

Definition 6.22. Fix some relation $R^{\mathbf{A}}$ of $\mathbf{A}$, of arity r. For a set of tuples $S \subseteq A^n$ and $t \in A^n$, we define the relation $S \vdash_{\mathbf{A},R^{\mathbf{A}}} t$ (omitting $\mathbf{A}, R^{\mathbf{A}}$ where they are obvious from context) to be the minimal relation that satisfies the following.

(i) If $S \subseteq T$ and $S \vdash t$ then $T \vdash t$.

(ii) $t \vdash t$.

(iii) If $S \vdash t$ and $t, T \vdash u$ then $S, T \vdash u$.

(iv) If there exists a matrix with n columns and r rows, whose rows are $t_1, \dots, t_r$ and whose columns are elements in $R^{\mathbf{A}}$, then $t_1, \dots, t_{r-1} \vdash t_r$.

Remark 6.23. Since $\vdash$ is minimal, every fact $S \vdash t$ can be proved true by using a finite proof constructed from the rules defining $\vdash$. Every such proof can be laid out as a *proof tree*, where every vertex is labelled by an n-ary tuple of elements in A. The root is labelled by t, the leaves by tuples of S, and a vertex labelled by t_r can have children $t_1, \dots, t_{r-1}$ if $(t_1, \dots, t_r)$ are as in Definition 6.22 (iv)

Example 6.24. Consider the following proof of the fact that $(0,0,0), (1,0,0), (0,1,0) \vdash (1,1,0)$ when $\mathbf{A} = (\{0,1\}, \{(1,0,0), (0,1,0), (0,0,1)\}) =$ **1-in-3**.

$$\dfrac{\dfrac{(1,0,0) \qquad (0,1,0)}{(0,0,1)} \qquad (0,0,0)}{(1,1,0)}$$

Note that, when dealing with symmetric relational structures, the order of the children of a vertex does not matter.

We will now prove three key facts: first the link between the syntax and semantics of our system; next two meta-reasoning lemmas that allow us to bootstrap simple proofs into more complicated ones.

Proposition 6.25. Fix $\mathbf{A} \longrightarrow \mathbf{B}$ with $\mathbf{A}$ symmetric and $\mathbf{B}$ functional. Then $S \vdash t$ implies $S \vDash t$ (or, more succinctly, $\vdash \subseteq \vDash$).

In the proof of this fact, we will extensively use the fact that a generalised tuple with values in A indexed by I is a function from I to A.

Proof. We have defined $\vdash$ to be the minimal relation that satisfies certain closure properties. If we prove that $\vDash$ also satisfies these properties, then $\vdash \subseteq \vDash$ as required; we thus prove these facts one by one.

(i) Suppose $S \subseteq T \subseteq A^n$ and $S \vDash t$. Hence there exists a function h such that, for any polymorphism $f \in \mathrm{Pol}^{(n)}(\mathbf{A}, \mathbf{B})$, we have

$$f(t) = h(f(s) \mid s \in S).$$

Now, let h' be a function that, given as input a tuple indexed by T, ignores the values corresponding to indices in $T \setminus S$, and returns the value of h on the values indexed by S. It is not difficult to see then that

$$f(t) = h'(f(s) \mid s \in T).$$

Hence $T \vDash t$ as required.

(ii) We must show that $t \vDash t$ for $t \in A^n$. Indeed, consider the function which, if given the unary tuple (x_t), indexed by the singleton set $\{t\}$, returns the values x_t. This function then witnesses $t \vDash t$.

(iii) Suppose $S \vDash t$ and $t, T \vdash u$. If $t \in T$ then what we need to prove follows by the proof for the first rule (as $t, T \subseteq S, T$) — so suppose $t \notin T$. Suppose that these facts are witnessed by maps h, h' such that, for all $f \in \mathrm{Pol}^{(n)}(\mathbf{A}, \mathbf{B})$,

$$\begin{aligned} f(t) &= h(f(s) \mid s \in S) \\ f(u) &= h'(f(t); f(t') \mid t' \in T) \end{aligned} \tag{6.2}$$

(where the tuple $(f(t); f(t') \mid t' \in T)$ is indexed by $t \cup T$, with value $f(t)$ at t and $f(t')$ at $t' \in T$). We must now create a function which, given a tuple indexed by $S \cup T$ containing the value $f(x)$ at $x \in S \cup T$, outputs $f(u)$. We claim that h'', defined by

$$h''(\alpha_x \mid x \in S \cup T) = h'(h(\alpha_s \mid s \in S); \alpha_{t'} \mid t' \in T).$$

That this value for h'' works follows as an easy consequence of (6.2).

(iv) Suppose there exists a matrix with n columns and r rows, whose rows are $t_1, \ldots, t_r$, and where every column is a tuple of $R^{\mathbf{A}}$. Then, since f is a polymorphism, it follows that $(f(t_1), \ldots, f(t_r)) \in R^{\mathbf{B}}$. Since $R^{\mathbf{B}}$ is functional, it is the graph of some partial function h, which takes $(f(t_1), \ldots, f(t_{r-1}))$ to $f(t_r)$. It is easy to see that this h witnesses that $t_1, \ldots, t_{r-1} \vDash t_r$. □

Next, we go through the two meta-reasoning lemmas. For the following, if R is some arbitrary relation, set $\mathrm{supp}(R)$ to be the set of values that are mentioned in R. Formally, if R has arity r,

$$\mathrm{supp}(R) = \bigcup_{i=1}^{r} \{a_i \mid (a_1, \ldots, a_r) \in R\}.$$

Also, for $S \subseteq A^r, t \in A^r, X \subseteq A, x \in A$, write $(t, x) \in A^{r+1}$ for the tuple $(t_1, \ldots, t_r, x)$ and $S \times X$ for the set of tuples $\{(s, x) \mid s \in S, x \in X\}$. Furthermore, observe that a tuple $(x_1, \ldots, x_n) \in A^n$ can be seen as a function $x : [n] \to A$, given by $i \mapsto x_i$. Thus we can precompose by any function $\pi : [m] \to [n]$, and write $x \circ \pi$. Observe that in this case $x \circ \pi \in A^m$, and

$$x \circ \pi = (x_{\pi(1)}, \ldots, x_{\pi(m)}).$$

We also will write $S \circ \pi$ for $\{s \circ \pi \mid s \in S\}$.

Lemma 6.26. Fix $S \subseteq A^n, t \in A^n$ and $\pi : [m] \to [n]$. Then, if $S \vdash t$, then $S \circ \pi \vdash t \circ \pi$.

Proof. Consider the proof tree of $S \vdash t$. Observe that if we transform the label u of every vertex into $u \circ \pi$, we get a valid proof tree of $S \circ \pi \vdash t \circ \pi$. Indeed, the leaves are elements of $S \circ \pi$, the root is $t \circ \pi$ — the only nontrivial part is that the labels of the children of a vertex, together with its own label, form a matrix as in Definition 6.22 (iv). Thus consider any vertex with label t_r, whose children are vertices with label $t_1, \ldots, t_{r-1}$ (where r is the arity of the relation $R^{\mathbf{A}}$, suppressed from the notation). By assumption, every column in the matrix M whose rows are $t_1, \ldots, t_r$ belongs to $R^{\mathbf{A}}$. But, this is also true for the matrix $M \circ \pi$ whose rows are $t_1 \circ \pi, \ldots, t_r \circ \pi$ — indeed column i of this matrix is just column $\pi(i)$ of M. This completes the proof. □

Lemma 6.27. Let $\mathbf{A}$ be symmetric. Fix $S \subseteq A^n$, $t \in A^n$ and $x \in A$. Choose also $R^{\mathbf{A}} = R$ from $\mathbf{A}$ such that $x \in \mathrm{supp}(R)$. Then, if $S \vdash t$, then $S \times \mathrm{supp}(R) \vdash (t, x)$.

Proof. By induction on the proof of the fact that $S \vdash t$. Consider the final step in that proof — it follows one of the rules from Definition 6.22, so we consider each rule in turn.

(i) Suppose $S \subseteq T, S \vdash t$. By induction we know that $S \times \mathrm{supp}(R) \vdash (t, x)$ — we wish to prove $T \times \mathrm{supp}(R) \vdash (t, x)$. But it is easy to see that $S \times \mathrm{supp}(R) \subseteq T \times \mathrm{supp}(R)$, hence this fact follows from Definition 6.22 (i) applied to $S \times \mathrm{supp}(R) \vdash (t, x)$.

(ii) Suppose $\{t\} \vdash t$. We wish then to prove that $\{t\} \times \mathrm{supp}(R) \vdash (t, x)$. But this is an easy combination of Definition 6.22 (ii) and then Definition 6.22 (i), since $(t, x) \in \{t\} \times \mathrm{supp}(R)$, as $x \in R$.

(iii) Suppose $S \vdash s$ and $s, T \vdash t$. By induction $(\{s\} \cup T) \times \mathrm{supp}(R) \vdash (t, x)$ and, *for every* $y \in \mathrm{supp}(R)$, we have that $S \times \mathrm{supp}(R) \vdash (s, y)$. Hence, applying Definition 6.22 (i) $|\mathrm{supp}(R)|$ times, we can transform $(\{s\} \cup T) \times \mathrm{supp}(R) \vdash (t, x)$ into $(S \cup T) \times \mathrm{supp}(R) \vdash (t, x)$ — replacing each (s, y) by $S \times \mathrm{supp}(R)$.

(iv) Suppose that $t_1, \ldots, t_r$ are as in Definition 6.22 (iv), and that we know $t_1, \ldots, t_{r-1} \vdash t_r$. We wish to show that $\{t_1, \ldots, t_{r-1}\} \times \mathrm{supp}(R) \vdash (t_r, x)$ i.e. $\{(t_i, y) \mid i \in [r-1], y \in \mathrm{supp}(R)\} \vdash (t_r, x)$. Since $x \in \mathrm{supp}(R)$ and R is symmetric by assumption, there exists a tuple $(x_1, \ldots, x_r) \in R$ with $x = x_r$. Hence we claim $(t_1, x_1), \ldots, (t_{r-1}, x_{r-1}) \vdash (t_r, x_r) = (t_r, x)$ — from where our conclusion follows from Definition 6.22 (i). □

(We note in passing that Lemma 6.27 may make the length of the proof explode — this is of no consequence for complexity, but may be of independent interest.)

6.2.4. Super-connectedness

We now introduce a purely syntactic notion, called *super-connectedness*, which implies both additivity and dependency. Our goal will then be to prove that various classes of relational structures have this property.

Definition 6.28. We say that a symmetric relational structure $\mathbf{A}$ is *super-connected* if it has a relation $R^{\mathbf{A}}$ such that, for every $x, y, z \in A$ (not necessarily all distinct), we have

$$\Gamma_{\mathbf{A}} \vdash_{\mathbf{A}, R^{\mathbf{A}}} (x, y, z).$$

Proposition 6.29. If $\mathbf{A}$ is super-connected, then for any functional $\mathbf{B}$ with $\mathbf{A} \to \mathbf{B}$ we have that $(\mathbf{A}, \mathbf{B})$ is additive.

Proof. If $\mathbf{A}$ is super-connected, in particular $\Gamma_{\mathbf{A}} \vdash (a, a, b)$ for every $a, b \in A$, and hence by Proposition 6.25 $\Gamma_{\mathbf{A}} \models (a, a, b)$ for every $a, b \in \mathbf{A}$. But then $(\mathbf{A}, \mathbf{B})$ is additive, by Proposition 6.20. □

Proposition 6.30. If $\mathbf{A}$ is super-connected, then for any functional $\mathbf{B}$ with $\mathbf{A} \to \mathbf{B}$ we have that $(\mathbf{A}, \mathbf{B})$ is dependent.

We first prove the following lemmas on the way to proving this fact. Suppose $\mathbf{A}$ is super-connected, with this being witnessed by $R^{\mathbf{A}}$ of arity r, and suppose $|A| > 1$.

Lemma 6.31. $\mathrm{supp}(R^{\mathbf{A}}) = A$.

Proof. Suppose not. Hence take $x \in A \setminus \mathrm{supp}(R^{\mathbf{A}})$ and $y \in A$ with $x \neq y$. Suppose for contradiction that

$$\Gamma_{\mathbf{A}} \vdash (x, x, y).$$

Consider any proof tree concluding with (x, x, y). Since $(x, x, y) \notin \Gamma_{\mathbf{A}}$ (as $x \neq y$), the root vertex must have children. Therefore there must exist a matrix whose columns are tuples of $R^{\mathbf{A}}$, and whose last row is (x, x, y). But $y \notin \mathrm{supp}(R^{\mathbf{A}})$, so this is impossible! Hence our conclusion. □

Lemma 6.32. Fix n and take $a_1, \ldots, a_n \in A$. Then $\Delta^n_{\mathbf{A}} \vdash (a_1, \ldots, a_n)$.

Proof. By double induction, first on n and then on $|\{a_1, \ldots, a_n\}|$. The base case is $n \leq 2$ — in this case $(a_1, \ldots, a_n) \in \Delta^n_{\mathbf{A}}$, and thus the conclusion follows by Definition 6.22 (i). Thus suppose $n > 2$. Consider the case where $a_1, \ldots, a_n$ contain a duplicated value, say $a_{n-1} = a_n$. Consider the map $\pi : [n] \to [n-1]$ given by $\pi(i) = \min(n-1, i)$. By the inductive hypothesis we have that $\Delta^{n-1}_{\mathbf{A}} \vdash (a_1, \ldots, a_{n-1})$. Apply Lemma 6.26 for π to get

$$\Delta^{n-1}_{\mathbf{A}} \circ \pi \vdash (a_1, \ldots, a_{n-1}) \circ \pi = (a_1, \ldots, a_n).$$

Note that if $(b_1, \ldots, b_{n-1}, b_n) \in \Delta^{n-1}_{\mathbf{A}} \circ \pi$, we have $b_{n-1} = b_n$ and $|\{b_1, \ldots, b_n\}| \leq 2$. But then $(b_1, \ldots, b_{n-1}, b_n) \in \Delta^n_{\mathbf{A}}$. Hence by Definition 6.22 (i), $\Delta^n_{\mathbf{A}} \vdash (a_1, \ldots, a_n)$, as required.

Thus consider the case where $a_1, \ldots, a_n$ are all distinct. By assumption, $\Gamma_{\mathbf{A}} \vdash (a_1, a_2, a_3)$. By applying Lemma 6.27 $n-3$ times, and noting that $\mathrm{supp}(R^{\mathbf{A}}) = A$ by Lemma 6.31, we get $\Gamma_{\mathbf{A}} \times A^{n-3} \vdash (a_1, \ldots, a_n)$. But every tuple $t \in \Gamma_{\mathbf{A}} \times A^{n-3}$ includes *at most* $n-1$ *distinct values*. Hence, by the inductive hypothesis, we have $\Delta^n_{\mathbf{A}} \vdash t$ for $t \in \Gamma_{\mathbf{A}} \times A^{n-3}$. Applying Definition 6.22 (iii) once for each such t, we find that $\Delta^n_{\mathbf{A}} \vdash (a_1, \ldots, a_n)$. This completes the proof. □

Proof of Proposition 6.30. If $\mathbf{A}$ has a unary domain, then it is obviously dependent. Thus take $|A| > 1$ — suppose furthermore $A = [a]$ without loss of generality. Then $\Delta^a_{\mathbf{A}} \vdash (1, \ldots, a)$ by Lemma 6.32, and thus by Proposition 6.25 $\Delta^a_{\mathbf{A}} \models (1, \ldots, a)$. Finally, this implies dependency by Proposition 6.21. □

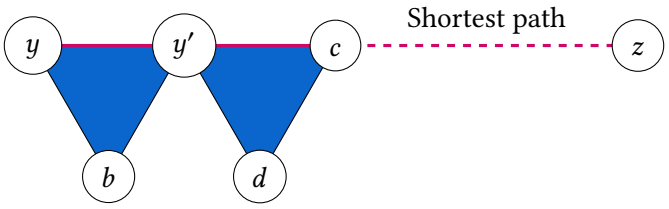


Figure 6.1.

6.2.5. Proofs of super-connectedness

In this section we are finally able to prove Proposition 6.6, which we now restate.

Proposition 6.6. Suppose $\mathbf{A}$ is symmetric, $\mathbf{B}$ is functional, with $\mathbf{A} \to \mathbf{B}$. If $\mathbf{A}$ is super-connected then $(\mathbf{A}, \mathbf{B})$ is additive and dependent. Furthermore, the following classes of $\mathbf{A}$ are super-connected:

1. Connected 3- or 4-uniform hypergraphs.
2. Structures where there exists a relation $R^{\mathbf{A}}$ of arity 3 or higher, such that, for any distinct $a, b \in A$, there exists a tuple of $R^{\mathbf{A}}$ containing both a and b.

For all that follows, if $\mathbf{H}$ is a hypergraph and u, v are vertices, then $\mathrm{dist}(u, v)$ is the number of hyperedges that need to be traversed to reach u from v. In other words, it is the distance between u and v in the graph formed by replacing each hyperedge of $\mathbf{H}$ by a clique.

Lemma 6.33. If $\mathbf{H} = (V, E)$ is a connected 3-uniform hypergraph, then it is superconnected.

Proof. If $\mathbf{H}$ has only one vertex nothing needs to be proved, so suppose it has at least two.

We must prove $\Gamma \vdash (x, y, z)$ for all vertices $x, y, z \in V$. We do induction first on the minimum $\min(\mathrm{dist}(x, z), \mathrm{dist}(y, z))$, then on the maximum $\max(\mathrm{dist}(x, z), \mathrm{dist}(y, z))$. Assume without loss of generality that $\mathrm{dist}(x, z) \leq \mathrm{dist}(y, z)$ — we have a few cases.

$\mathrm{dist}(x, z) = 0$. In this case $(x, y, z) = (x, y, x) \in \Gamma$, so the result follows by Definition 6.22 (ii) and Definition 6.22 (i).

$\mathrm{dist}(x, z) = \mathrm{dist}(y, z) = 1$. In this case there exist edges (x, a, z) and (y, b, z), and we can prove our result as follows:

$$\dfrac{\dfrac{\dfrac{(x,z,x) \qquad (z,a,a)}{(a,x,z)} \qquad (x,a,a)}{(z,z,x)} \qquad (a,b,a)}{(x,y,z)}$$

$\mathrm{dist}(y, z) > \mathrm{dist}(x, z) = 1$. In this case there exist edges (x, a, z) and (y, b, y') with $\mathrm{dist}(y', z) = \mathrm{dist}(y, z) - 1 > 0$. Since $\mathrm{dist}(y', z) > 0$, there exists an edge (y', c, d) with $\mathrm{dist}(c, z) < \mathrm{dist}(y, z)$ and $\mathrm{dist}(d, z) < \mathrm{dist}(y, z)$ (cf. Figure 6.1, where the shortest path in the graph where hyperedges are replaced by cliques from x to z is highlighted). Now we prove

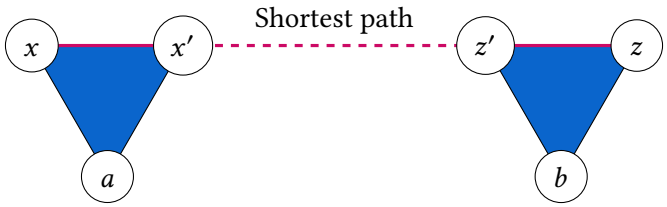


Figure 6.2.

$$\dfrac{\dfrac{(x,d,z) \qquad (a,c,a)}{(z,y',x)} \qquad (a,b,a)}{(x,y,z)}$$

Hence we have shown $(x,d,z),\Gamma \vdash (x,y,z)$. Since $\mathrm{dist}(d,z) < \mathrm{dist}(y,z)$ we can apply the inductive hypothesis to deduce $\Gamma \vdash (x,d,z)$ — applying Definition 6.22 (iii) then completes the proof.

$\mathrm{dist}(y,z) \geq \mathrm{dist}(x,z) > 1$. There exist edges $(x,a,x'),(z,b,z')$ for which $\mathrm{dist}(a,z') < \mathrm{dist}(x,z)$ and $\mathrm{dist}(x',b) < \mathrm{dist}(x,z)$ (cf. Figure 6.2). Also, since the hypergraph is connected and has more than one vertex, an edge (y,c,d) must exist. Then consider the proof

$$\dfrac{(a,c,z') \qquad (x',d,b)}{(x,y,z)}$$

Noting that $\Gamma \vdash (a,c,z')$ and $\Gamma \vdash (x',d,b)$ by the inductive hypothesis and applying Definition 6.22 (iii) twice gives us that $\Gamma \vdash (x,y,z)$ as required. □

Lemma 6.34. If $\mathbf{H} = (V,E)$ is a connected 4-uniform hypergraph, then it is superconnected.

Proof. As in Lemma 6.33, we can assume there are at least two vertices. Similarly to above, we prove $\Gamma \vdash (x,y,z)$ by induction on the minimum $\min(\mathrm{dist}(x,z),\mathrm{dist}(y,z))$ and then on the maximum $\max(\mathrm{dist}(x,z),\mathrm{dist}(y,z))$, assuming without loss of generality that $\mathrm{dist}(x,z) \leq \mathrm{dist}(y,z)$. Consider the following cases.

$\mathrm{dist}(x,z) = 0$. In this case $(x,y,z) = (x,y,x) \in \Gamma$, so the result follows by Definition 6.22 (ii) and Definition 6.22 (i).

$\mathrm{dist}(x,z) = \mathrm{dist}(y,z) = 1$. In this case there exist edges (x,a,a',z) and (y,b,b',z) — the proof of our result can be found in Figure 6.3.

$\mathrm{dist}(y,z) > \mathrm{dist}(x,z) = 1$. In this case there exist edges (x,a,a',z) and (y,b,b',y') with $\mathrm{dist}(y',z) = \mathrm{dist}(y,z) - 1 > 0$. Since $\mathrm{dist}(y',z) > 0$, there exists an edge (y',c,d,e) with $\mathrm{dist}(c,z) < \mathrm{dist}(y,z)$, $\mathrm{dist}(d,z) < \mathrm{dist}(y,z)$ and $\mathrm{dist}(e,z) < \mathrm{dist}(y,z)$ (cf. Figure 6.5). Now we prove:

$$\dfrac{\dfrac{(a',c,a') \qquad \dfrac{(x,d,z) \qquad (a,e,a)}{(z,y',x)} \qquad (a,b,a)}{(x,y,z)} \qquad (a',b',a')}{(x,y,z)}$$

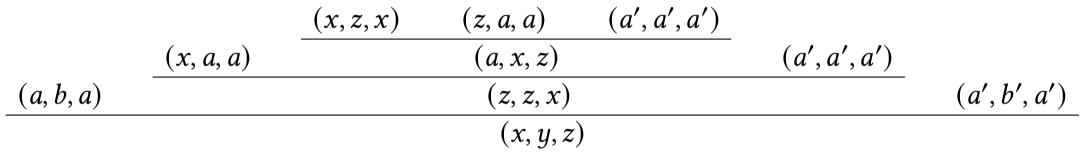


Figure 6.3.: Proof from Lemma 6.34.

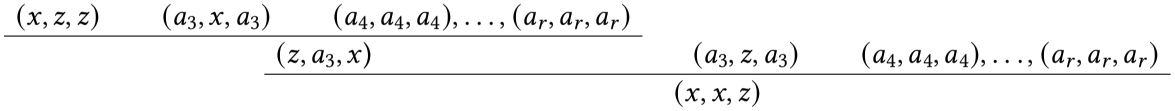


Figure 6.4.: Proof from Lemma 6.35.

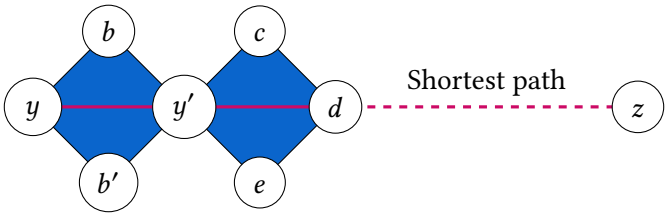


Figure 6.5.

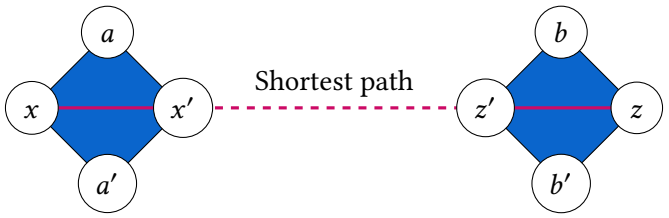


Figure 6.6.

Hence we have shown $(x, d, z), \Gamma \vdash (x, y, z)$. Since $\mathrm{dist}(d, z) < \mathrm{dist}(y, z)$ we can apply the inductive hypothesis to deduce $\Gamma \vdash (x, d, z)$ — applying Definition 6.22 (iii) then completes the proof.

$\mathrm{dist}(y, z) \geq \mathrm{dist}(x, z) > 1$. There exist edges (x, a, a', x'), (z, b, b', z') with $\mathrm{dist}(a, z') < \mathrm{dist}(x, z)$, $\mathrm{dist}(a', z') < \mathrm{dist}(x, z)$, $\mathrm{dist}(x', b) < \mathrm{dist}(x, z)$ and $\mathrm{dist}(x', b') < \mathrm{dist}(x, z)$ (cf. Figure 6.6). Since $\mathrm{dist}(y, z) > 1$, we can find edges (y, c, c', y') and (y', d, d', y'') with $\mathrm{dist}(y'', z) < \mathrm{dist}(y, z)$. Thus:

$$\dfrac{\dfrac{(a', d, z') \qquad (x', d', b') \qquad (x, y'', z)}{(a, y', b)} \qquad (a', c, z') \qquad (x', c', b')}{(x, y, z)}$$

Noting that Γ can prove all the preconditions in this proof tree (i.e. all the leaves in the tree) by the inductive hypothesis, and applying Definition 6.22 (iii) several times completes the proof. □

Lemma 6.35. Suppose $\mathbf{A}$ is a symmetric relational structure with a relation $R = R^{\mathbf{A}}$, of arity 3 or higher, such that any two distinct elements $x, y \in A$ can be extended to a tuple of R. Then $\mathbf{A}$ is super-connected.

Proof. Suppose R has arity r. Take $x, y, z \in A$ — we prove $\Gamma \vdash (x, y, z)$, splitting into cases.

$x = z$ or $y = z$. Here $(x, y, z) \in \Gamma$ and there is nothing to prove.

$x = y \neq z$. By assumption there exists an edge $(x, z, a_3, \ldots, a_r) \in R$. The proof can be found in Figure 6.4. Noting that $(x, x, z) = (x, y, z)$ completes the proof.

$x \neq y \neq z \neq x$. By assumption there exist edges $(x, z, a_3, \ldots, a_r), (x, y, b_3, \ldots, b_r) \in R$. Then we prove:

$$\dfrac{(z, x, x) \qquad (a_3, b_3, a_3), \ldots, (a_r, b_r, a_r)}{(x, y, z)}$$

This concludes the proof. □

Will all of this, we are now ready to prove the main fact of this section.

Proof of Proposition 6.6. That the super-connectedness of $\mathbf{A}$ implies the additivity and dependency of $(\mathbf{A}, \mathbf{B})$ follows from Propositions 6.29 and 6.30. The fact that 3-uniform hypergraphs, 4-uniform hypergraphs, and relational structure with a relation as in the statement of the proposition are all super-connected follows from Lemmas 6.33 to 6.35. □

6.3. Proof of dichotomy

In this section, we prove Theorem 6.5, which we now restate.

Theorem 6.5. Suppose $\mathbf{A}$ is symmetric, $\mathbf{B}$ is functional and $\mathbf{A} \to \mathbf{B}$. Suppose also that $(\mathbf{A}, \mathbf{B})$ is additive and dependent. Then $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is either NP-hard, or solvable in polynomial time by reducing (by homomorphic sandwich, cf. Corollary 2.5) to modular linear equations.

The proof of this fact will be split into two parts, a proof of tractability and a proof of hardness. Throughout this section we will assume $\mathbf{A} \to \mathbf{B}$, $\mathbf{A}$ symmetric, $\mathbf{B}$ functional, and $(\mathbf{A}, \mathbf{B})$ dependent and additive. Our proofs revolve around two further notions: *degeneracy* and *hard sets*.

Definition 6.36. Consider $f \in \mathrm{Pol}^{(n)}(\mathbf{A}, \mathbf{B})$. We say that f is *k-degenerate* if there exist $x_1, \dots, x_k \in \mathrm{range}(f^p)$ such that, whenever $S_1, \dots, S_k \subseteq [n]$ such that $x_1 = f^p(S_1), \dots, x_k = f^p(S_k)$, it follows that $S_1, \dots, S_k$ are not all disjoint. We say that $x_1, \dots, x_k$ *witness* the k-degeneracy of f.

Definition 6.37. Consider $f \in \mathrm{Pol}^{(n)}(\mathbf{A}, \mathbf{B})$. A *hard set* is a set $S \subseteq [n]$ such that, for any $S \subseteq T \subseteq [n]$, we have $f^p(T) \neq f^p(\varnothing)$.

These notions are similar to *unbounded antichains* and *fixing sets* [FKOS19]. The notion of a hard set is similar to the notion of an $f^p(\varnothing)$-avoiding set from [DRS05; Wro22].

Define two constants $N_d = \max(1 + |B|^{|A|^2}|A|^{2r_M}, 3)$, and $N_h = |B|^{|A|^2}$, where r_M is the maximum arity of any relation in $\mathbf{A}$. These constants could be picked to be any numbers that are at least this large.

Proposition 6.38 (Tractability). Suppose $\mathbf{A} \to \mathbf{B}$, $\mathbf{A}$ symmetric, $\mathbf{B}$ functional, $(\mathbf{A}, \mathbf{B})$ additive and dependent. If $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ contains a polymorphism that is not k-degenerate for any $k \leq N_d$, and has no hard sets of size at most N_h, then $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is solvable in polynomial time, by reducing to linear modular equations by homomorphic sandwich.

Proposition 6.39 (Hardness). Suppose $\mathbf{A} \to \mathbf{B}$, $\mathbf{A}$ symmetric, $\mathbf{B}$ functional, $(\mathbf{A}, \mathbf{B})$ additive and dependent. If every polymorphism $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ is k-degenerate for $k \leq N_d$, or has a hard set of size at most N_h, or both, then $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is NP-hard.

We prove these facts in Sections 6.3.1 and 6.3.2 respectively.

6.3.1. Tractability

In this section we will prove our tractability result. In particular, we will assume that there exists some polymorphism $f \in \mathrm{Pol}^{(n)}(\mathbf{A}, \mathbf{B})$ that is not k-degenerate for $k \leq N_d$, and has no hard set of size at most N_h, in addition to our assumptions on $\mathbf{A}, \mathbf{B}$. Our proof will show that we can find a suitably defined group in $\mathbf{B}$.

Definition 6.40. Let $0 = f^p(\varnothing)$ and $1 = f^p([n])$.

Lemma 6.41. $(\mathrm{range}(f^p), +, 0)$ forms a group.

(Indeed, this group is Abelian, but this is not needed and not proved.)

Proof. We prove the group axioms one by one.

Well-definedness, closure. Consider $x, y \in \mathrm{range}(f^p)$. Since f is not 2-degenerate, there exist disjoint $S, T \subseteq [n]$ such that $x = f^p(S), y = f^p(T)$. Now,

$$x + y = f^p(S) + f^p(T) = f^p(S \cup T) \in \mathrm{range}(f^p),$$

so closure and well-definedness follows.

Associativity. Consider $x, y, z \in \mathrm{range}(f^p)$. Since f is not 3-degenerate, there exist disjoint $S, T, U \subseteq [n]$ such that $x = f^p(S), y = f^p(T), z = f^p(U)$. Now

$$x + (y + z) = f^p(S) + (f^p(T) + f^p(U)) = f^p(S \cup (T \cup U)) = \\ f^p((S \cup T) \cup U) = (f^p(S) + f^p(T)) + f^p(U) = (x + y) + z$$

Hence associativity follows.

Identity. Consider $x \in \mathrm{range}(f^p)$. Take $S \subseteq [n]$ such that $x = f^p(S)$. Observe

$$x + 0 = f^p(S) + f^p(\varnothing) = f^p(S \cup \varnothing) = f^p(S) = x.$$

So 0 is an identity element.

Inverses. Consider any $x \in \mathrm{range}(f^p)$. Take $S \subseteq [n]$ such that $x = f^p(S)$. By Lemma 6.13, there exists $T \subseteq S$ with $|T| \leq |B|^{|A|^2} = N_h$ with $f^p(T) = f^p(S) = x$. By assumption T is not a hard set, so there exists $T \subseteq U \subseteq [n]$ with $f^p(U) = f^p(\varnothing) = 0$. But then,

$$x + f^p(U \setminus T) = f^p(T) + f^p(U \setminus T) = f^p(T \cup (U \setminus T)) = f^p(U) = 0.$$

So every element has an inverse. □

Recall that the image of $f^\star$ is a vector of images of f^p. It will pay, therefore, to extend this group notion to vectors of elements.

Definition 6.42. Let G be the subgroup of $(\operatorname{range}(f^p), +, 0)$ generated by 1. Denote the element $1 + 1$ by 2, $2 + 1$ by 3 and so on — with this notation, we see that $G = \mathbb{Z}_m$, with $m \leq |\operatorname{range}(f^p)| \leq |B|^{|A|^2}$. Define the Abelian group $(H, +) = \mathbb{Z}_m^A$ to be the free $\mathbb{Z}_m$-module generated by A — in other words, elements of $(H, +)$ are formal linear combinations of elements in A, with coefficients in $\mathbb{Z}_m$. In particular, we take the convention that $A \subseteq H$. Furthermore, we consider $A^r \subseteq H^r$.

Definition 6.43. For any relation $R \subseteq A^r$, we define $M(R) = \langle p - q \mid p, q \in R \rangle \subseteq H^r$ to be the subgroup generated by $p - q$ for $p, q \in R \subseteq A^r$. In particular, as H^r is Abelian, $M(R)$ is a normal subgroup.

Lemma 6.44. Consider a relation $R^{\mathbf{A}}$ of $\mathbf{A}$, of arity r. Choose a tuple $t \in R^{\mathbf{A}}$. Let $M = M(R)$. Consider $(a_1, \ldots, a_r) \equiv t \bmod M$ i.e. $(a_1, \ldots, a_r) \in t + M$. Then, there exists a matrix (x_{ij}) with $N \leq N_d$ columns and r rows, where $N \equiv 1 \bmod m$, with elements in A, such that each column is a tuple of $R^{\mathbf{A}}$, and for each row i we have

$$\sum_{j=1}^{N} x_{ij} = a_i.$$

(The sum is taken in H.)

Proof. Observe $H^r = (\mathbb{Z}_m^A)^r$ is a $\mathbb{Z}_m$-module, so every element in H^r has order which divides m. Since $(a_1, \ldots, a_r) \equiv t \bmod M$, with M being a subgroup of H^r generated by $p - q = p + (m-1)q$ for $p, q \in R^{\mathbf{A}}$, expanding definitions we find that there exist coefficients $c_{pq} \in \{0, \ldots, m-1\}$ such that

$$(a_1, \ldots, a_r) = t + \sum_{p,q \in R^{\mathbf{A}}} c_{pq}(p - q) = t + \sum_{p,q \in R^{\mathbf{A}}} c_{pq}p + (m - c_{pq})q.$$

If one decodes this statement, it is equivalent to the following: given a matrix whose first column is $t \in R^{\mathbf{A}}$ and, subsequently, contains c_{pq} copies of p and $m - c_{pq}$ copies of q for each $p, q \in R^{\mathbf{A}}$ as columns, summing the rows as elements of H gives us $(a_1, \ldots, a_r)$. Noting that this matrix has $1 + m|R^{\mathbf{A}}|^2 \leq 1 + |B|^{|A|^2}|A|^{2r_M} \leq N_d$ columns with $1 + m|R^{\mathbf{A}}|^2 \equiv 1 \bmod m$ completes the proof. □

For the following, note that a formal linear combination of elements in A can be seen as a generalised tuple indexed by A, under the correspondence

$$\sum_{c_a \in A} c_a a = (c_a \mid a \in A).$$

For convenience we will often mix these two notations for fundamentally the same object.

Lemma 6.45. Suppose $f = h \circ f^\star$. For every $N \leq N_d$ with $N \equiv 1 \bmod m$, the function $h_N : A^N \to B$ given by

$$h_N(x_1, \ldots, x_N) = h\left(\sum_{i=1}^{N} x_i\right)$$

is a polymorphism of $(\mathbf{A}, \mathbf{B})$. (The sum is again taken in H.)

Proof. We have that f is a polymorphism m that is not $N \leq N_d$-degenerate. Thus, there exist disjoint sets $S_1, \ldots, S_N$ such that $f^p(S_1) = \cdots = f^p(S_N) = 1 \in \mathrm{range}(f^p)$. Let $T = [n] \setminus (S_1 \cup \ldots \cup S_N)$. Note that $S_1, \ldots, S_N, T$ form a partition of $[n]$, with

$$1 = f^p([n]) = f^p\left(T \cup \bigcup_{i=1}^{N} S_i\right) = f^p(T) + \sum_{i=1}^{N} f^p(S_i) = f^p(T) + N.$$

But since $N \equiv 1 \bmod m$, this implies $1 = f^p(T) + 1$, so $f^p(T) = 0$.

We will now show that h_N can be created by taking a minor of f (and ignoring a coordinate that does not affect the output). Take $\pi : [n] \to [N+1]$ according to the partition $S_1, \ldots, S_N, T$ — i.e.

$$\pi(i) = \begin{cases} j, & i \in S_j \\ N+1, & i \in T. \end{cases}$$

We will prove that $f^\pi(x_1, \ldots, x_N, y) = h_N(x_1, \ldots, x_N)$ — this immediately implies that h is a polymorphism as required. To see that this is the case recall that $f^\pi = f \circ \pi^{-1} = h \circ f^\star \circ \pi^{-1}$, and since $h_N = h\left(\sum_{i=1}^{N} x_i\right)$, we merely show that

$$(f^\star \circ \pi^{-1})(x_1, \ldots, x_N, y) = \sum_{i=1}^{N} x_i.$$

(Applying h to both sides implies the desired result.) Now, suppose $(T_a \mid a \in A)$ is the partition of $[N+1]$ given by $x_1, \ldots, x_N, y$ — expanding definitions and grouping terms, it is sufficient to show that

$$f^\star(\pi^{-1}(T_a) \mid a \in A) = \sum_{a \in A} |T_a| a.$$

Recall at this point that both the left and right side are essentially tuples of elements of $G = \mathbb{Z}_m$ indexed by A. It is sufficient to compare the left and right sides in every component (i.e. the coefficients of each element in A must be the same). Hence, recalling the definition of $f^\star$, it is sufficient to show that

$$f^p(\pi^{-1}(T_a)) \equiv |T_a| \bmod m,$$

for every $a \in A$. But this is true — let the set $X_a = \varnothing$ if $y \neq a$, and $X_a = T$ if $y = a$. In any case, $f^p(X_a) = f^p(\varnothing) = 0$. Observe that $\pi^{-1}(T_a) = X_a \cup \bigcup_{i \in T_a \cap [N]} S_i$. Hence

$$f^p(\pi^{-1}(T_a)) = f^p\left(X_a \cup \bigcup_{i \in T_a \cap [N]} S_i\right) = f^p(X_a) + \sum_{i \in T_a} f^p(S_i) = 0 + \sum_{i \in T_a} 1 \equiv |T_a| \mod m.$$

This completes the proof. □

We can now prove our tractability result.

Proof of Proposition 6.38. We will construct a relational structure $\mathbf{E}$ such that $\mathbf{A} \to \mathbf{E} \to \mathbf{B}$, and E has domain H, with relations that are either empty, or of form $c + M$ for some $c \in H^r$ and $M \subseteq H^r$, a normal subgroup. Since H is a free $\mathbb{Z}_m$-module, CSP($\mathbf{E}$) can be solved by solving a large system of modulo m linear equations. Indeed, a constraint $x \in c + M$ of arity r equivalent to: There exists $p_v \in \mathbb{Z}_m$ for $v \in M$ such that, for each $i \in [r]$, we have

$$x_i = c_i + \sum_{(v_1,\dots,v_r)=v\in M} p_v v_i.$$

This equation is over H — thus, each such equation is further equivalent to a system of $|A|$ modular equations, one for each component of H.

The homomorphism $\mathbf{A} \to \mathbf{E}$ will be given by the inclusion $A \subseteq \mathbb{Z}_m^A \subseteq H$. The homomorphism $\mathbf{E} \to \mathbf{B}$ will be given by any function h such that $f = h \circ f^\star$ — of course, such a function exists by Lemma 6.17.

We will now show how to construct $\mathbf{E}$ relation by relation. Consider some relation $R^{\mathbf{A}}$ in $\mathbf{A}$ of arity r, corresponding to $R^{\mathbf{B}}$ in $\mathbf{B}$. If $R^{\mathbf{A}}$ is empty, then we can take $R^{\mathbf{E}}$ to be empty — $R^{\mathbf{A}}$ then trivially maps to $R^{\mathbf{E}}$, which then trivially maps to $R^{\mathbf{B}}$. So assume $R^{\mathbf{A}}$ is not empty, and in particular suppose $t \in R^{\mathbf{A}}$. Take $M = M(R^{\mathbf{A}})$. We will take $R^{\mathbf{E}} = t + M$.

First, let us verify that the natural inclusion $A \subseteq H$ satisfies the conditions for a homomorphism from $\mathbf{A}$ to $\mathbf{E}$ for the relation R. It is sufficient to show that, for any $s \in R^{\mathbf{A}}$, we have $s \in t + M$. But, $s = t + (s - t) \in t + M$, as required.

Next, let us verify that h satisfies the conditions for a homomorphism from $\mathbf{E}$ to $\mathbf{B}$ for the relation R. This will be a straight forward combination of Lemmas 6.44 and 6.45. Consider any tuple $(a_1, \dots, a_r) \in t + M$. By Lemma 6.44 there exists a matrix (x_{ij}) with r rows and N columns, with $N \le N_d$ and $N \equiv 1 \bmod m$. Every column of (x_{ij}) is a tuple of $R^{\mathbf{A}}$, and for each row i we have $\sum_{j=1}^{N} x_{ij} = a_i$, where the sum is taken in H. Since every column is a tuple of $R^{\mathbf{A}}$, and since $h_N(x_1, \dots, x_N) = h\left(\sum_{i=1}^{N} x_i\right)$ is a polymorphism by Lemma 6.45, we can apply h_N row-wise to get a tuple of $R^{\mathbf{B}}$. Now, h_N applied to row i of (x_{ij}) yields:

$$h_N(x_{i1}, \dots, x_{iN}) = h\left(\sum_{j=1}^{N} x_{ij}\right) = h(a_i).$$

Hence $(h(a_1), \dots, h(a_N))$ is a tuple of $R^{\mathbf{B}}$ which completes the proof. □

6.3.2. Hardness

In this section we prove our hardness result. In particular, we will assume that every polymorphism in Pol($\mathbf{A}, \mathbf{B}$) is either k-degenerate for some $k \le N_d$, or has a hard set of size at most N_h. We will prove then that PCSP($\mathbf{A}, \mathbf{B}$) is NP-hard.

Let us first define some convenient notations for subsets of Pol($\mathbf{A}, \mathbf{B}$).

Definition 6.46. Let $\mathcal{M}_h \subseteq \mathrm{Pol}(\mathbf{A}, \mathbf{B})$ consist of those polymorphisms $f \in \mathrm{Pol}(\mathbf{A}, \mathbf{B})$ that have a hard set of size at most N_h. For any sequence $x_1, \ldots, x_k \in B^{A^2}$, let $\mathcal{M}_{x_1,\ldots,x_k}$ be the set of polymorphisms $f \in \mathrm{Pol}(\mathbf{A}, \mathbf{B})$ that are *not* $(k-1)$-degenerate, and whose k-degeneracy is witnessed by $x_1, \ldots, x_k$.

Let us quickly recall the hardness criteria that we will use for this proof.

Theorem 2.23. Fix C, k. There exists some N such that the following holds. Consider some $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$, and let $\mathcal{M} = \mathrm{Pol}(\mathbf{A}, \mathbf{B})$. Cover $\mathcal{M}$ by k sets $\mathcal{M}_1, \ldots, \mathcal{M}_k$. For every $i \in [k]$, suppose there exists a mapping I_i that takes $f \in \mathcal{M}_i$ of *arity* $n \leq N$ to a set $I_i(f) \subseteq [n]$ of size at most C. Suppose that for every $f, g \in \mathcal{M}_i$ *where* $\mathrm{ar}(f) \leq N, \mathrm{ar}(g) \leq N$ and π such that $f \xrightarrow{\pi} g$ we have that $\pi(I_i(f)) \cap I_i(g) \neq \varnothing$. Then $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is NP-hard.

We will split $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ into $\mathcal{M}_h$ and $\mathcal{M}_{x_1,\ldots,x_k}$ for all appropriate $x_1, \ldots, x_k$, then apply this theorem. We now show that we can create the mapping I for each of these sets.

Lemma 6.47. There exists a mapping I which takes $f \in \mathcal{M}_h$, say of arity n, to a subset of $[n]$ of size at most N_h^2 such that, for any $f, g \in \mathcal{M}_h$, say of arity n and m, and $\pi : [n] \to [m]$ such that $f \xrightarrow{\pi} g$, we have $\pi(I(f)) \cap I(g) \neq \varnothing$.

Proof. For any fixed $f \in \mathrm{Pol}^{(n)}(\mathbf{A}, \mathbf{B})$, observe that there cannot be more than $N_h = |B|^{|A|^2}$ disjoint hard sets, by Lemma 6.12. Indeed, this lemma indicates that, in any family of at least N_h disjoint sets, some nonempty union of these sets must have image through f^p equal to $f^p(\varnothing)$. Hence at least one set in the family is not a hard set.

This indicates to us what set we will take. Let $I(f)$ be the union of any maximal family of disjoint hard sets of size at most N_h. Note that $I(f)$ will be nonempty — at least one such hard set exists since $f \in \mathcal{M}_h$. Furthermore, since each such set has size at most N_h, and there are at most N_h of them, $|I(f)| \leq N_h^2$.

We now check that I has the required property. Let f, g, π be as in the theorem statement — we must prove that $\pi(I(f)) \cap I(g) \neq \varnothing$. We claim that if S is a hard set of f, then $\pi(S)$ is a hard set of g. Indeed, suppose that $\pi(S)$ is not a hard set of g — then there exists some $T \supseteq \pi(S)$ such that $g^p(T) = g^p(\varnothing)$. But then we observe $\pi^{-1}(T) \supseteq S$, and

$$f^p(\pi^{-1}(T)) = (f^\pi)^p(T) = g^p(T) = g^p(\varnothing) = (f^\pi)^p(\varnothing) = f^p(\pi^{-1}(\varnothing)) = f^p(\varnothing).$$

Hence S is not a hard set of f and our claim follows by contrapositive.

With this fact established, observe that there exists some hard set $S \subseteq I(f)$ by construction, of size at most N_h. Thus $\pi(S)$ is a hard set of g, also of size at most N_h. By the maximality of $I(g)$, it must be the case that $\pi(S)$ is not disjoint from $I(g)$. Hence $\pi(S) \cap I(G) \neq \varnothing$ and, a fortiori, $\pi(I(f)) \cap I(G) \neq \varnothing$. □

Lemma 6.48. Fix $x_1, \ldots, x_k \in B^{A^2}$, with $2 \leq k \leq N_d$. There exists a mapping I which takes any $f \in \mathcal{M}_{x_1,\ldots,x_k}$, say of arity n, to a subset of $[n]$ of size at most kN_h such that, for any $f, g \in \mathcal{M}_{x_1,\ldots,x_k}$, say of arity n and m, and $\pi : [n] \to [m]$ such that $f \xrightarrow{\pi} g$, we have $\pi(I(f)) \cap I(g) \neq \varnothing$.

Proof. For any $f \in \mathrm{Pol}^{(n)}(\mathbf{A}, \mathbf{B})$, we define $I(f)$ in the following way. Note that f is *not* $(k-1)$-degenerate. Hence, since $x_1, \dots, x_{k-1} \in \mathrm{range}(f)$, there exist *disjoint* $S_1, \dots, S_{k-1}$ with $f^p(S_i) = x_i$. Furthermore, by applying Lemma 6.13, we can take $|S_i| \leq N_h$. Since $x_k \in \mathrm{range}(f)$, there exists also some T with $f^p(T) = x_k$ — again by applying Lemma 6.13 we take $|T| \leq N_h$. Set $I(f) = T \cup \bigcup_{i=1}^{k-1} S_i$. Clearly $|I(f)| \leq kN_h$ — all that remains is the minoring property from the statement.

Let f, g, π be as in the theorem statement. Let $S_1, \dots, S_{k-1}, T \subseteq [n], S'_1, \dots, S'_{k-1}, T' \subseteq [m]$ be the sets used to construct $I(f), I(g)$. Note that

$$f^p(\pi^{-1}(T')) = g^p(T') = x_k.$$

Hence, since f is k-degenerate, with k-degeneracy witnessed by $x_1, \dots, x_k$, it follows that the sets $S_1, \dots, S_{k-1}, \pi^{-1}(T')$ cannot be all disjoint. But since $S_1, \dots, S_{k-1}$ are all disjoint by construction, we have that $\pi^{-1}(T')$ must intersect them. This implies that $I(f) \cap \pi^{-1}(I(g)) \neq \emptyset$, and hence $\pi(I(f)) \cap I(g) \neq \emptyset$, as required. □

We are now ready to put all of this together, and prove the final proposition in this chapter.

Proof of Proposition 6.39. First, note that no polymorphism can be 1-degenerate. This is immediate by unpacking the definition in the case $k = 1$. Thus, since we have assumed that each polymorphism is either k-degenerate for some $k \leq N_d$, or has a hard set of size at most N_h, we have

$$\mathrm{Pol}(\mathbf{A}, \mathbf{B}) = \mathcal{M}_h \cup \bigcup_{k=2}^{N_d} \bigcup_{x_1, \dots, x_k \in B^{A^2}} \mathcal{M}_{x_1, \dots, x_k}.$$

In other words we have split $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ into a *bounded* number of subsets. Applying Lemmas 6.47 and 6.48 we see that each of these subsets has a function I which assigns each polymorphism a bounded subset of its domain so that the condition from Theorem 2.23 is satisfied. Hence, by Theorem 2.23, $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is NP-hard. □

7. Equivalencies Between Classical Algorithms

In this chapter, we will show that for certain classes of templates $(\mathbf{A}, \mathbf{B})$, we have that the *Affine Integer Programming relaxation (AIP)* is equivalent to the *combined Basic Linear Programming and Affine Integer Programming relaxation (BLP+AIP)*. We prove that this is the case when $\mathbf{A}$ has one relation which is *balanced* — in particular, all symmetric relations are balanced, as well as digraphs that are disjoint unions of strongly connected digraphs. This serves as evidence for a conjecture first formally stated in [NŽ23a] (but informally circulated in the community prior to that) — that AIP solves all PCSPs of form PCSP(**1-in-3**, $\mathbf{B}$) that can be solved in polynomial time. This chapter will present results from the following previously published works.

[NŽ23a] Tamio-Vesa Nakajima and Stanislav Živný. 'Boolean symmetric vs. functional PCSP dichotomy'. In: *LICS*. IEEE, 2023, pp. 1–12. DOI: 10.1109/lics56636.2023.10175746.

[NŽ24c] Tamio-Vesa Nakajima and Stanislav Živný. 'On the Complexity of Symmetric vs. Functional PCSPs'. In: *ACM Trans. Algorithms* 20.4 (2024), 33:1–33:29. DOI: 10.1145/3673655.

The works [NŽ23a; NŽ24c] were produced in collaboration with my supervisor.

7.1. Introduction

In the PCSP landscape, the most used paradigm for proving tractability is by *relaxing* then *rounding*. Let us consider a prototypical example: PCSP(**1-in-3**, **NAE**). This was first proved solvable in polynomial time in [BG21], and the algorithm is simple and elegant. Note that **1-in-3** is equivalent to $x + y + z = 1$ for $x, y, z \in \{0, 1\}$. Relax this constraint to $x, y, z \in \mathbb{Z}$, then solve this simplified system. Next, round the solution according to its sign: a number ≥ 1 is rounded to 1, and one ≤ 0 is rounded to 0. It is then not difficult to see that, in any constraint, the three rounded values cannot all be equal.

This example is a particular case of the *Affine Integer Programming relaxation* [BG21]. The precise definition of this relaxation is not essential to us — it suffices to say that, for all PCSPs of form PCSP(**1-in-3**, $\mathbf{B}$) that are known to be solvable in polynomial time, this algorithm also solves them in polynomial time. This is by far not the only relaxation that is commonly used — many others exist, including the *Basic Linear Programming relaxation (BLP)* [BG21], the *Combined Basic Linear Programming and Affine Integer Programming relaxation (BLP+AIP)* [BGWŽ20], the *Combined Constraint Basic Linear Programming and Affine Integer Programming Relaxation (CLAP)* [CŽ23a], as well as various other hierarchies

of relaxations [CŽ23b]. One other striking result is that BLP+AIP solves all *Boolean, symmetric* PCSPs [FKOS19] that can be solved in polynomial time.

In principle, one could now proceed with an in-depth discussion on the exact description of these algorithms, together with an extended explanation of when they are applicable. The beauty of BLP, AIP and BLP+AIP is that this is not necessary: there exists a simple, polymorphism based description of the templates for which they work. Furthermore, the algorithms can be restated as "For an appropriately defined minion $\mathcal{M}$, solve $\mathrm{CSP}(\mathbf{F}_{\mathcal{M}}(\mathbf{A}))$ i.e. the CSP given by the free structure of $\mathcal{M}$ generated by $\mathbf{A}$". This description first appeared in [BBKO21], and we refer the reader to that work for the precise definitions of the minions for each of the three algorithms. We will be interested not in the inner workings of the algorithms, but rather the conditions for their applicability — we give them now.

Definition 7.1 ([BBKO21; BGWŽ20]). (i) An operation $f : A^n \to B$ is called *symmetric* if $f^\pi = f$ for any permutation $\pi : [n] \to [n]$.

(ii) An operation $f : A^{2n+1} \to B$ is called *2-block symmetric* if $f^\pi = f$ for any permutation $\pi : [2n+1] \to [2n+1]$ that sends odd numbers to odd numbers and even numbers to even numbers.

(iii) An operation $f : A^{2n+1} \to B$ is called *alternating* if it is 2-block symmetric, and furthermore, for any $a, b, x_3, \ldots, x_{2n+1} \in A$, we have

$$f(a, a, x_3, \ldots, x_{2n+1}) = f(b, b, x_3, \ldots, x_{2n+1}).$$

Example 7.2. Define

$$\begin{aligned} f(x_1, \ldots, x_n) &= \sum_{i=1}^{n} x_i \\ g(x_1, \ldots, x_{2n+1}) &= \left(\sum_{i=0}^{n} x_{2i+1}\right)\left(\sum_{i=1}^{n} x_{2i}\right) \\ h(x_1, \ldots, x_{2n+1}) &= \left(\sum_{i=0}^{n} x_{2i+1}\right) - \left(\sum_{i=1}^{n} x_{2i}\right). \end{aligned}$$

Then f is symmetric, g is 2-block symmetric, and h is alternating.

Theorem 7.3 ([BBKO21; BGWŽ20]). $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is solved by:

1. BLP if and only if $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ contains symmetric operations of all arities.
2. AIP if and only if $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ contains alternating operations of all odd arities.
3. BLP+AIP if and only if $\mathrm{Pol}(\mathbf{A}, \mathbf{B})$ contains 2-block symmetric operations of all odd arities.

Plainly every symmetric (resp. alternating) operation is also 2-block symmetric. Hence, in general BLP+AIP is at least as strong as either BLP or AIP. In this chapter, we will prove that for a large class of relational structure $\mathbf{A}$, namely *balanced relational structures*, AIP and BLP+AIP have equivalent power for $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$. Thus AIP is unexpectedly powerful.

In particular, all symmetric relational structures (and hence **1-in-3** also) are balanced. This gives further weight to the following conjecture, first formally stated in [NŽ24c], but informally circulated in the community before that.

Conjecture 1.3 ([NŽ24c]). For every structure $\mathbf{B}$ with **1-in-3** $\to \mathbf{B}$, PCSP(**1-in-3**, $\mathbf{B}$) is either NP-hard or solvable in polynomial time by AIP.

In the intervening time, I have come to believe the following stronger conjecture — although proof still remains elusive.

Conjecture 7.4. Suppose $\mathbf{A}$ is symmetric and connected (i.e. if $\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2$, then one of $\mathbf{A}_1$ and $\mathbf{A}_2$ must have empty domain) with exactly one relation, and $\mathbf{B}$ is a relational structure such that $\mathbf{A} \to \mathbf{B}$. Then PCSP($\mathbf{A}, \mathbf{B}$) is either NP-hard or solvable in polynomial time by AIP.

The necessity of the stipulations that $\mathbf{A}$ be symmetric, connected, with one relation can be seen from the following examples, taken from [NŽ24c].

Example 7.5 (Connectedness is necessary). Consider $\mathbf{A} = \mathbf{B}$ on domain $A = B = \{(0,2),(1,2),(0,3),(1,3),(2,3)\}$, and with one 6-ary relation $R^{\mathbf{A}} = R^{\mathbf{B}}$ defined by

$$\{((x_1,m),\dots,(x_6,m)) \in A^6 \mid \sum_{i=1}^{6} x_i \equiv 1 \pmod{m}\}.$$

In particular, $\mathbf{A} = \mathbf{B}$ can be seen as a disjoint sum of its substructures induced by $\{(0,2),(1,2)\}$ and $\{(0,3),(1,3),(2,3)\}$ respectively.
The problem CSP($\mathbf{A}$) = PCSP($\mathbf{A}, \mathbf{B}$) can be equivalently described as follows: we are given a set V and a 6-ary relation $R \subseteq V^6$. We must choose, for each $x \in V$, a modulus $m(x)$, and a value $c(x) \in \{0,\dots,m(x)-1\}$, so that the following hold: for every $(x_1,\dots,x_6) \in R$, $m(x_1) = \cdots = m(x_6)$, and furthermore $\sum_{i=1}^{6} x_i \equiv 1 \pmod{m(x_1)}$. Hence, any "connected" subset of variables must have the same modulus in any valid solution.
This alternate view helps us see how to solve CSP($\mathbf{A}$): for every connected subset of variables, exhaustively try every modulus $m \in \{2,3\}$, and try to solve the system of modulo m equations induced by the variables. If this is possible, it gives us a solution to the CSP; otherwise none exists.
On the other hand, this problem is not solved by AIP — or even by BLP+AIP! Indeed, it cannot have a 2-block symmetric polymorphism of arity 5. Consider the following matrix of inputs:

$$\begin{pmatrix} (0,3) & (1,2) & (1,3) & (0,2) & (1,3) \\ (1,3) & (0,2) & (0,3) & (1,2) & (1,3) \\ (1,3) & (1,2) & (1,3) & (0,2) & (0,3) \\ (0,3) & (0,2) & (1,3) & (1,2) & (1,3) \\ (1,3) & (1,2) & (0,3) & (0,2) & (1,3) \\ (1,3) & (0,2) & (1,3) & (1,2) & (0,3) \end{pmatrix}$$

Assume f were a 2-block symmetric polymorphism of arity 5. By 2-block symmetry, the image of f applied to each of the rows is the same. This is impossible, since the relation of $\mathbf{B}$ has no constant tuples.

Example 7.6 (One relation is necessary). Consider $\mathbf{A} = (\{0,1\}; R, Q)$, with $R = \{(0)\}$ being the unary constant 0 constraint, and $Q = \{(0,1),(1,0),(1,1)\}$ being the constraint $x \vee y$. Then $\mathrm{CSP}(\mathbf{A}) = \mathrm{PCSP}(\mathbf{A}, \mathbf{A})$ is solvable in polynomial time, being solved by BLP+AIP (and indeed by BLP alone): $(x_1, \ldots, x_k) \mapsto \max(x_1, \ldots, x_k)$ is easily seen to be a symmetric polymorphism for each arity k. On the other hand, this template is not solved by AIP. Indeed it cannot have a ternary alternating polymorphism: suppose f were such a polymorphism. Then $f(1,1,0) = f(0,0,0) = f(0,1,1)$. Furthermore, due to R we have $f(0,0,0) = 0$. But, consider the matrix

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}.$$

Each column is a tuple of Q, but f applied row-wise yields $(0,0)$, which is not.

Example 7.7 (Symmetry is necessary). Here we exploit the fact that asymmetric relations can simulate arbitrary relations. In particular, take $\mathbf{B} = (\{0,1\}; S)$, with $S = \{(0,1,0),(0,0,1),(0,1,1)\}$. Note that S can pp-define R and Q from the previous example, and R and Q together can pp-define S, viz.

$$R(x) = \exists y \cdot \exists z \cdot S(x,y,z) \qquad Q(y,z) = \exists x \cdot S(x,y,z)$$
$$S(x,y,z) = R(x) \wedge Q(y,z).$$

Hence the reasoning in the previous example can be carried out analogously here — and thus this example is also solved by BLP but not by AIP.

We will now define what a balanced relation is.

Definition 7.8. Consider a relation $R \subseteq A^r$. We say that it is *balanced* if there exists a matrix M with r rows and n columns, such that the following holds.

1. Every column of M is a tuple of R.
2. Every tuple of R appears at least once as a column in M.
3. Every row of M contains the same elements just permuted.

We say that a relational structure $\mathbf{A}$ is *balanced* if it has *exactly* one relation, and that relation is balanced.

We will prove the following two theorems. The first shows that for balanced $\mathbf{A}$, $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is solved by AIP if and only if it is solved by BLP+AIP. Next, we will show that certain more natural classes of relational structures are balanced.

Theorem 7.9. Suppose $\mathbf{A}$ is balanced and $\mathbf{B}$ is such that $\mathbf{A} \to \mathbf{B}$. Then $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ is solved by AIP if and only if it is solved by BLP+AIP.

Proposition 7.10. If $\mathbf{A}$ has one relation R, preserved by a transitive group G of permutations of the coordinate positions, then $\mathbf{A}$ is balanced.

Proposition 7.11. If G is a digraph where every weakly connected component is strongly connected, then G is balanced.

We will prove Theorem 7.9 in Section 7.2, and Propositions 7.10 and 7.11 in Section 7.3

7.2. Proof of main theorem

In this section we will prove Theorem 7.9. The approach will be similar to some parts of Chapter 6 — except that rather than taking freely generated $\mathbb{Z}_m$ modules, we will take freely generated $\mathbb{Z}$ modules, i.e. freely generated Abelian groups.

Fix $\mathbf{A}$, a relational structure with one balanced r-ary relation $R = R^{\mathbf{A}}$, and $\mathbf{B}$, a relational structure with one relation $R^{\mathbf{B}}$, such that $\mathbf{A} \to \mathbf{B}$. Suppose that $t_1, \ldots, t_N \in R$ are the columns of the matrix that witness the balancedness of $\mathbf{A}$ — hence each tuple of R appears at least once among these tuples, and the rows of the matrix whose columns are $t_1, \ldots, t_N$ are permutations one of the other.

We will consider A as being a subset of $\mathbb{Z}^A$, the Abelian group freely generated by A. The elements in this group are formal linear combinations of elements in A with integer coefficients. We will extend the addition on these elements to vectors of elements in $\mathbb{Z}^A$, and thus implicitly to vectors of elements in A. Hence if, for example, $a, b \in A$, we have

$$(a, b, a) + (b, a, a) = (a + b, a + b, 2a).$$

Furthermore, for any sets $S, T \subseteq (\mathbb{Z}^A)^r$ of r-ary tuples of elements in $\mathbb{Z}^A$, and any r-ary tuple $t \in (\mathbb{Z}^A)^r$, we let kS denote the set of sums of k vectors from S; we let $S + T$ denote the set of sums $x + y$ with $x \in S$, $y \in T$, and we let $S + t$ denote the set of sums $x + t$ with $x \in S$. In particular, we take the view that A is a set of unary tuples of elements in $\mathbb{Z}^A$, and hence kA is the set of sums $a_1 + \cdots + a_k$ for $a_1, \ldots, a_k \in A$. In other words, kA is the set of formal linear combinations of elements in A with nonzero coefficients that sum up to k. Observe that, in particular, $R = R^{\mathbf{A}}$ is a set of r-ary tuples from $A \subseteq \mathbb{Z}^A$.

We will want to bridge the gap between 2-block symmetric and alternating polymorphisms. In particular, we can see an alternating polymorphism as a 2-block symmetric polymorphism where the blocks "cancel out". A prototypical way of doing this is by literally subtracting one block from the other — or, more generally by subtracting the frequencies of one block from those of the other. One can interpret part of the machinery of Chapter 6 as handling "subtraction" of tuples by using the fact that addition was being done modulo m (cf. Lemma 6.44). In other words, "subtracting" k copies of a tuple is the same as adding $m-k$ of them. Now however this will be harder to do — we will accomplish this by "padding out" with copies of $t_1, \ldots, t_N$. This logic is captured by the following lemma.

Lemma 7.12. Let R be a balanced relation, and let $t_1, \ldots, t_N$ be the columns of a matrix witnessing that R is balanced. For any $k \in \mathbb{N}$, $(k+1)R - kR + k\sum_{i=1}^{N} t_i \subseteq (kN+1)R$.

Proof. Consider any element $s \in (k+1)R - kR + k\sum_{i=1}^{N} t_i$. This element can be written as a sum

$$s = u_1 + \cdots + u_{k+1} - v_1 - \cdots - v_k + k\sum_{i=1}^{N} t_i,$$

for some $u_1, \ldots, u_{k+1}, v_1, \ldots, v_k \in R$. Note however that the elements $t_1, \ldots, t_N$ contain *each* tuple in R at least once. Hence the sum $k\sum_{i=1}^{N} t_i$ contains each element in R at least k times. Thus, we can cancel out the $-v_1 - \cdots - v_k$ terms with positive terms within $k\sum_{i=1}^{N} t_i$, writing s as a sum of $kN+1$ elements from R. Hence $s \in (kN+1)R$. □

Next we will give two lemmas that express solvability by BLP+AIP and AIP in more convenient formats. There is nothing mysterious about them — they merely capture the intuition that a 2-block symmetric polymorphism only sees the frequencies of the elements in its two blocks (i.e. their sum as elements of $\mathbb{Z}^A$), whereas an alternating polymorphism only sees the difference between these frequencies.

Lemma 7.13. Suppose $(\mathbf{A}, \mathbf{B})$ admits a 2-block symmetric polymorphism f of arity $2k+1$. Then there exists a function $g : kA \times (k+1)A \to B$ such that, for any $(x_1, \dots, x_r) \in kR$, $(y_1, \dots, y_r) \in (k+1)R$, we have $(g(x_1, y_1), \dots, g(x_r, y_r)) \in R^{\mathbf{B}}$.

(Note that $kA \times (k+1)A$ is the set of pairs of elements, one from kA, the other from $(k+1)A$.)

Proof. Define g as follow: on input $(a_1 + \cdots + a_k, b_1 + \cdots + b_{k+1})$, let the image of g be

$$f(b_1, a_1, \dots, a_k, b_{k+1}). \tag{7.1}$$

The order of $a_1, \dots, a_k$ and $b_1, \dots, b_{k+1}$ respectively is inessential since f is 2-block symmetric. We must now prove that if $(x_1, \dots, x_r) \in kR$ and $(y_1, \dots, y_r) \in (k+1)R$, we have $(g(x_1, y_1), \dots, g(x_r, y_r)) \in R^{\mathbf{B}}$.

Observe that $(x_1, \dots, x_r) \in kR$ is equivalent to the existence of a matrix P with r rows and k columns, where each column is a tuple of R, and summing the rows (as elements of $\mathbb{Z}^A$) gives us $(x_1, \dots, x_r)$. Similarly, $(y_1, \dots, y_r) \in (k+1)R$ is equivalent to the existence of a matrix Q with r rows and $k+1$ columns, defined analogously. Now, interleave the columns of P and Q (similarly to the interleaving of a and b in (7.1)) to create a matrix M with r rows and $2k+1$ columns, where each column is a tuple of R, and the odd (respectively even) coordinates in row i sum up to x_i (respectively y_i).

Now, we claim that applying f to row i yields $g(x_i, y_i)$. Indeed, this follows by 2-block symmetry of f. This implies that $(g(x_1, y_1), \dots, g(x_r, y_r)) \in R^{\mathbf{B}}$, since f is a polymorphism. This completes the proof. □

Lemma 7.14. Fix $(\mathbf{A}, \mathbf{B})$ and suppose there exists a function $f : ((k+1)A - kA) \to B$ such that, for any $(x_1, \dots, x_r) \in (k+1)R - kR$, we have $(f(x_1), \dots, f(x_r)) \in R^{\mathbf{B}}$. Then $\mathrm{PCSP}(\mathbf{A}, \mathbf{B})$ admits an alternating polymorphism of arity $2k+1$.

Proof. Indeed, this polymorphism is given by

$$g(x_1, \dots, x_{2k+1}) = f(x_1 + x_3 + \cdots + x_{2k+1} - x_2 - \cdots - x_{2k}).$$

That this function is alternating is immediate (since indeed the sum within its definition is alternating). To see that it is a polymorphism, consider any matrix with r rows and $2k+1$ columns, all of which are within R. Suppose the columns are $c_1, \dots, c_{2k+1}$. Consider the sum

$$c_1 + c_3 + \cdots + c_{2k+1} - c_2 - \cdots - c_{2k} \in (k+1)R - kR.$$

By assumption, applying f componentwise to this tuple results in a tuple from $R^{\mathbf{B}}$. However, applying f componentwise to this sum yields the same result as applying g to the rows of the matrix. Hence g is a polymorphism. □

These lemmas suffice to prove our main result.

Proof of Theorem 7.9. We know, by Theorem 7.3, that AIP solves PCSP($\mathbf{A}, \mathbf{B}$) if and only if ($\mathbf{A}, \mathbf{B}$) admits alternating polymorphisms of all odd arities, and BLP+AIP solves PCSP($\mathbf{A}, \mathbf{B}$) if and only if ($\mathbf{A}, \mathbf{B}$) admits 2-block symmetric polymorphism of all odd arities. Since any alternating polymorphism is, by definition, 2-block symmetric, it follows that if PCSP($\mathbf{A}, \mathbf{B}$) is solved by AIP, it is also solved by BLP+AIP. The more non-trivial part of the result is proving the converse — we will exploit the fact that we have 2-block symmetric polymorphisms of *all* odd arities, and construct an alternating polymorphism of arity $2k+1$ using a 2-block symmetric polymorphism of arity $2kN+1$ (where, recall, N is the number of columns in the matrix that witnesses the balancedness of $\mathbf{A}$).

Thus, we will show how to construct an alternating polymorphism of arity $2k+1$ starting from a 2-block symmetric polymorphism of arity $2kN+1$. First a short digression: consider the sum $t_1 + \cdots + t_N$. By assumption, the matrix with columns $t_1, \ldots, t_N$ has rows that are permutations of each other. Each element in the tuple $t_1 + \cdots + t_N$ is the sum of a row in this matrix (where elements are seen as members of $\mathbb{Z}^A$) — hence each element in this tuple *must be equal*. Hence, suppose $t_1 + \cdots + t_N = (c, \ldots, c)$ for some $c \in NA$.

Apply Lemma 7.13 to generate a function $g : kNA \times (kN+1)A \to B$ with properties as in that lemma. Consider the function $f : ((k+1)A - kA) \to B$ given by

$$f(x) = g(kc, x + kc).$$

We claim that f is a function to which we can apply Lemma 7.14, in order to generate the requisite alternating polymorphism. Hence consider $(x_1, \ldots, x_r) \in (k+1)R - kR$. Observe that

$$(kc, \ldots, kc) = k(c, \ldots, c) = k \sum_{i=1}^{N} t_i \in kNR$$

$$(x_1 + kc, \ldots, x_r + kc) = (x_1, \ldots, x_r) + k(c, \ldots, c) \in (k+1)R - kR + k \sum_{i=1}^{N} t_i \subseteq (kN+1)R,$$

where the last step is by Lemma 7.12. Hence we see that, by applying the properties guaranteed by Lemma 7.13 to g, we have that

$$(f(x_1), \ldots, f(x_r)) = (g(kc, x_1 + kc), \ldots, g(kc, x_r + kc)) \in R^{\mathbf{B}},$$

as required. This completes the proof. □

7.3. Proof of characterisations

In this section we prove Propositions 7.10 and 7.11. These proofs are fairly simple.

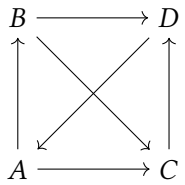
Proof of Proposition 7.10. Suppose that $\mathbf{A} = (A, R)$ and R is preserved by some transitive group G. In other words, if R has arity r, then G is a subgroup of S_r, and for any $\pi \in G$, if $(x_1, \ldots, x_r) \in R$ then $(x_{\pi(1)}, \ldots, x_{\pi(r)}) \in R$. We must create a matrix whose columns are

tuples of R, where each tuple appears at least once, and where the rows are permutations of each other. Indeed, the matrix where each tuple of R appears *exactly once* suffices.

To see why, suppose M is this matrix — we want to show that every two rows are permutations of each other. Without loss of generality, we show that the first is a permutation of the second. Let u, v be the first and the second row respectively, and suppose π is some permutation in G sending 1 to 2 (it exists by transitivity). Apply π to each column in M, yielding the matrix πM. Since $\pi^{-1}\pi M = M$ (i.e. unpermuting the rows undoes the operation), no two columns in πM are equal — else there would be two equal columns in M as well. Furthermore, since R is closed under G, the columns of πM are still tuples of R — hence πM contains every tuple of R exactly once. But this means there exists some σ acting on the *columns* of M such that $M\sigma = \pi M$. Inspect row 2 in this matrix. In πM, this is just u In $M\sigma$, this is v permuted by σ. Hence u and v are permutations of each other. □

Proof of Proposition 7.11. Suppose $G = (V, E)$ is a digraph, where each weakly connected component is strongly connected. Equivalently, for every edge $(x, y) \in E$, there exists a path from y to x in G. By taking the union of all of these paths, together with (x, y), we see that we can cover G by a union of cycles. Now, lay out the edges in all of these cycles as a matrix with two rows, and as many columns as there are edges in all the cycles. Each column in the matrix is an edge of G; furthermore, the first row is a permutation of the second one. Indeed, we can get the first row by cyclically permuting the vertices corresponding to each cycle.

For example, consider the following graph.



For edge $A \to B$ we have the path $B \to D \to A$; for $B \to C$ we have $C \to D \to A \to B$; for $C \to D$ we have $D \to A \to C$; for $D \to A$ we have $A \to B \to D$; for $A \to C$ we have $C \to D \to A$; and for $B \to D$ we have $D \to A \to B$. Hence we construct the matrix

$$\left(\begin{array}{ccc|cccc|ccc|ccc|ccc|ccc} A & B & D & B & C & D & A & C & D & A & D & A & B & A & C & D & B & D & A \\ B & D & A & C & D & A & B & D & A & C & A & B & D & C & D & A & D & A & B \end{array}\right)$$

We can check that, by permuting each block cyclically, the first row can be transformed into the second. □

Part II.

Maximum Promise Constraint Satisfaction Problems

8. A Dichotomy for Boolean, Binary maxPCSPs

In this chapter, we will fully classify the *Boolean, binary maxPCSP*. In addition to this, we will provide algorithms for the following problems: "maximum-and vs. even-SAT", and "maximum-dicut vs. acyclic subgraph". All of these will be solved by essentially the same approach: finding a half-integral solution to a linear programming relaxation of the original problem. This line of work has appeared previously in the following work.

[NŽ25b] Tamio-Vesa Nakajima and Stanislav Živný. 'Maximum And- vs. Even-SAT'. In: *APPROX/RANDOM*. Vol. 353. LIPIcs. LZI, 2025, 3:1–3:8. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2025.3.

The works were produced in collaboration with my supervisor. I am extremely thankful to an anonymous reviewer who suggested the simplification of the SDP-based algorithm from [NŽ24b] to the LP-based algorithm from [NŽ25b].

8.1. Introduction

The most basic problem in the maxCSP literature is the *maximum cut (max-cut)* problem: Given a graph $G = (V, E)$ which admits a cut of value ρ, find a cut with as large a value as possible. (Recall that a *cut* in a graph is a subset of the edges of the graph that induce a bipartite subgraph. Equivalently, it can be seen as a colouring of the vertices with two colours, and the value of the cut is given by the fraction of edges that connect vertices with different colours.) We say that we can α-approximate the problem if we can output a cut with value at least $\alpha\rho$.

There is an easy randomised algorithm that $\frac{1}{2}$-approximates max-cut: assign each vertex a random bit, and output the set of edges that connects two distinct bits. This algorithm is, however, far from optimal.

Theorem 8.1 (Goemans-Williamson algorithm). There exists a polynomial-time randomised algorithm that $(0.878\cdots)$-approximates max-cut [GW95]. This algorithm can be derandomised [MR99], and, assuming the Unique Games Conjecture (UGC) of Khot [Kho02], approximating with a better approximation ratio is NP-hard [KKMO07].

These results can be massively generalised. Note that max-cut can be cast as a maximisation version of 2-colouring: Given a graph G, find an assignment of two colours to the vertices such that, for as many edges (x, y) as possible, we have $x \neq y$. Along similar lines, for every relational structure $\mathbf{A}$, one can define maxCSP($\mathbf{A}$) as being the problem of simultaneously

solving as many constraints of an instance of CSP($\mathbf{A}$) as possible. We call the *value* of an assignment the proportion of constraints which it satisfies. The following theorem is then the culmination of a long line of work started by the Goemans-Williamson algorithm.

Theorem 8.2 ([Rag08; RS09]). Fix any relational structure $\mathbf{A}$. There exists some real number α such that the following holds. Given an instance of CSP($\mathbf{A}$) that admits an assignment with value ρ, one can find an assignment with value at least $(\alpha - \varepsilon)\rho$ in polynomial time for any fixed ε. Furthermore, for all small enough ε, it is NP-hard (assuming UGC) to find an assignment with value at least $(\alpha + \varepsilon)\rho$. Finally, given $\mathbf{A}$ and ε, α can be computed up to error ε in doubly exponential time with respect to a polynomial of $1/\varepsilon$ and the size of $\mathbf{A}$.

The maxPCSP is defined so as to have the same relation to maxCSPs as PCSPs have to CSPs. More concretely: in maxPCSP($\mathbf{A}, \mathbf{B}$) with $\mathbf{A} \to \mathbf{B}$, given an instance of CSP($\mathbf{A}$) with value ρ, find an assignment for it *as an instance of* CSP($\mathbf{B}$) with value at least ρ.[1] This is the search version of the problem — similarly to PCSPs there is also a decision version which is not yet known to be equivalent. In it, we are given an instance of CSP($\mathbf{A}$) and a constant ρ, and we must answer YES if it has value at least ρ, and NO if it has value less than ρ *as an instance of* CSP($\mathbf{B}$). Furthermore, we can α-approximate this problem if (i) in the search version we return an assignment with value at least $\alpha\rho$ as an instance of CSP($\mathbf{B}$), and (ii) in the decision version we can decide between the instance having value at least ρ or not even having value $\alpha\rho$ as an instance of CSP($\mathbf{B}$).

Note that the approximation ratio in Theorem 8.2 is, while computable, in a sense poorly understood. Indeed, we can only compute it up to some error in *doubly exponential time*. Furthermore, the theorem does not, as stated, give any hardness results for $\alpha = 1$. For exact (i.e. $\alpha = 1$) solvability of maxCSP($\mathbf{A}$), a complete classification was given nonetheless by Thapper and Živný [TŽ16]. (The case maxCSP($\mathbf{A}$) corresponds to the special case of "0–1 valued CSPs".)

This indicates to us that the exact solvability of maxPCSPs will lead, perhaps, to a fruitful line of work. The first fragment that one would naturally try to investigate is the case of maxPCSP($\mathbf{A}, \mathbf{B}$) where $\mathbf{A}$ and $\mathbf{B}$ have one Boolean, binary relation. If all of these problems were trivial or NP-hard, this would be an early indication that this problem is uninteresting (it is either too hard or too easy). Fortunately, we get some interesting results even in this very restricted fragment. For the following, note that if $\mathbf{A}$ or $\mathbf{B}$ have a constant tuple or are empty, then maxPCSP($\mathbf{A}, \mathbf{B}$) is trivially solvable.

Definition 8.3. Recall that $\multimap = (\{0,1\}, \{(0,1)\})$ and $\rightleftarrows = (\{0,1\}, \{(0,1),(1,0)\})$. Up to relabelling vertices and transposing the relation, these are the only relational structures with one Boolean, binary relation, whose relation is nonempty and has no constant tuples.

We are ready now to state the classification of Boolean binary maxPCSPs. This follows by combining the results of [TSSW00; Hås01] together with a new algorithmic result.

[1] We explicitly allow *parallel edges*; in other words, a constraint may appear multiple times, and it then counts multiple times to the value

Theorem 8.4. Excluding the cases that have a constant tuple or empty relations, there are essentially three Boolean, binary maxPCSPs: maxCSP(⊸), maxPCSP(⊸, ↫) and maxCSP(↫). Of these, maxPCSP(⊸, ↫) is solvable exactly in polynomial time, whereas it is NP-hard to approximate maxCSP(↫) with approximation ratio $16/17 + \varepsilon$, and maxCSP(⊸) with approximation ratio $12/13 + \varepsilon$.

The two maxCSPs are well-studied: maxCSP(↫) is identical to max-cut, and maxCSP(⊸) is identical to a problem known as *max-dicut*. In the max-dicut problem we are given a directed graph, and are asked to split the vertices into two sets, so that as many edges as possible go from the first set to the second. We call a set of edges of this form a *directed cut* or *dicut*. The hardness of these problems was proved in the early 2000s, as indicated below.

Theorem 8.5 ([TSSW00; Hås01]). It is NP-hard to approximate maxCSP(↫) with approximation ratio $16/17 + \varepsilon$, and to approximate maxCSP(⊸) with approximation ratio $12/13 + \varepsilon$.

The main contribution in this chapter is to complete the classification, by showing that the remaining problem is solvable in polynomial time.

Corollary 8.6. maxPCSP(⊸, ↫) can be solved in polynomial time.

We will prove this by providing an algorithm that solves a more general problem, which we call the *promise max-and* problem.

Definition 8.7. An instance of the *promise max-and* problem is given by a set V of variables, and a multiset C of clauses, where each clause is a multiset $C \in C$ of literals (these clauses may be of different sizes). Each literal is of form either $+x$ or $-x$ for some variable $x \in V$. We denote literals by sx for $s \in \{\pm 1\}, x \in V$.
A *strong assignment* to this instance is a function $c : V \mapsto \{\pm 1\}$. The value of a clause $\{s_1x_1, \ldots, s_kx_k\}$ under such an assignment is $\frac{1}{2} + \frac{1}{2} \min_i s_i c(x_i)$. The value of c is the mean of the values of all clauses under c. The strong value of the instance is the maximum value of any strong assignment.
A *weak assignment* is a function $c : V \mapsto \{-1, 0, 1\}$. The value of a clause $\{s_1x_1, \ldots, s_kx_k\}$ under such an assignment is still $\frac{1}{2} + \frac{1}{2} \min_i s_i c(x_i)$. The value of c is the mean of the values of all clauses under c. Note that every strong assignment is also a weak assignment with the same value.
In the promise max-and problem, we are given an instance with strong value ρ (without being given ρ), and are asked to find a weak assignment with value at least ρ, in polynomial time.

We will prove that this problem is tractable in Section 8.2

Theorem 8.8. The promise max-and problem is solvable in polynomial time.

We will show how to reduce maxPCSP(⊸, ↫) to the promise max-and problem. Indeed, we will show how to solve a more general problem: the *max-and vs even problem*.

Definition 8.9. Consider an instance of promise max-and, with variables V and clauses C. An assignment up to parity is a map $c : V \to \{\pm 1\}$. We say that the value of a clause $C = \{s_1 x_1, \ldots, s_k x_k\}$ *up to parity* is 1 if $s_i c(x_i) = -1$ for an *even* number of indices i. Equivalently, the value of the clause up to parity is given by $\frac{1}{2} + \frac{1}{2} \prod_i s_i c(x_i)$. (This takes advantage of the fact that $\mathbb{Z}_2$ is isomorphic to $\{\pm 1\}$ under multiplication.) The value of c up to parity is the mean of the values of all clauses up to parity.
In the *max-and vs. even* problem, we are given an instance of promise max-and with strong value ρ, and we are asked to create an assignment up to parity c whose value up to parity is at least ρ.

Lemma 8.10. maxPCSP($\multimap, \not\rightarrow$) is a special case of the max-and vs. even problem.

Proof. Indeed, maxPCSP($\multimap, \not\rightarrow$) on instance (V, E) is exactly equivalent to the max-and vs. even problem on the instance (V, C), where C contains the clause $\{-x, y\}$ for every edge $(x, y) \in E$. A strong assignment is identical to a directed cut, and an assignment up to parity is identical to a cut. □

We will prove that this problem is tractable, in Section 8.3

Corollary 8.11. The max-and vs. even problem can be solved in polynomial time.

Proof of Corollary 8.6. Combine Lemma 8.10 with Corollary 8.11. □

Acyclic subgraphs. Having considered the case of maximum promise CSPs on Boolean domains with one binary relation, it is worth thinking about what other natural problems on digraphs we could consider. One of the most natural is the problem of finding a maximum acyclic subgraph in a given graph. Observe that any digraph that contains a dicut with value ρ will have an acyclic subgraph with value ρ as well — just consider the edges included in the dicut. This naturally leads us to ask the following question: Given a digraph with a dicut with value ρ, can we find an acyclic subgraph with value ρ? Let $\omega = (\mathbb{N}, <)$ denote the poset of natural numbers ordered in the natural way. We interpret ω as a digraph whose vertex set is $\mathbb{N}$ and whose edge set is $<$ — with this convention observe that a homomorphism from a digraph G to ω exists if and only if G is acyclic, and the preimages of this homomorphism give rise to a topological ordering of G. Hence, maxCSP(ω) is the problem of finding a maximum acyclic subgraph of an input digraph — and the problem of finding an acyclic subgraph with value at least as large as the maximum value of any dicut is maxPCSP($\multimap, \omega$).

We make progress towards a more general theory of infinite-domain maximum promise CSPs by showing that this problem is tractable.

Theorem 8.12. maxPCSP($\multimap, \omega$) is solvable in polynomial time.

We will prove this fact in Section 8.3

8.2. Tractability of promise max-and

In this section we prove Theorem 8.8. We are given an instance of promise max-and with variable set V and clause set C, such that

$$\rho = \max_{c:V\to\{\pm 1\}} \underbrace{\sum_{\{s_1x_1,\ldots,s_kx_k\}\in C} \min_{1\le i\le k} s_i c(x_i)}_{F(c)}.$$

We are then asked to compute $c^\star : V \to \{-1,0,1\}$ such that $F(c^\star) \ge \rho$.[2]

Let us begin by computing $\tilde{c} : V \to [-1,1]$ that maximises the expression $F(\tilde{c})$. This can be done by solving a linear program. Indeed, for every $C = \{s_1x_1,\ldots,s_kx_k\} \in C$, introduce a variable $t(C)$ and then solve

$$\begin{aligned} \max_{c,\,t} \quad & \sum_{C\in C} t(C) \\ \text{s.t.} \quad & c(x) \in [-1,1], \quad x \in V, \\ & t(C) \le s_i c(x_i), \quad C = \{s_1x_1,\ldots,s_kx_k\} \in C, i \in [k] \end{aligned}$$

In any optimal assignment the value of $t(C)$ will be taken to be exactly $\min_{1\le i\le k} s_i c(x_i)$, and hence solving this linear program maximises $F(c)$.

By negating variables, we can assume that $\tilde{c}(x) \ge 0$ for all $x \in V$. Furthermore, let us suppose $V = \{1,\ldots,n\}$ and $\tilde{c}(1) \le \cdots \le \tilde{c}(n)$. We will now show how to shift the solution $\tilde{c}$, without modifying $F(\tilde{c})$, so that $\tilde{c}(x) \in \{0,1\}$ for all $x \in V$. After undoing the negations we implicitly applied initially, this solution will have values in $\{-1,0,1\}$

Under the assumption that $c(1) \le \cdots \le c(n)$ (i.e. c is monotonic), for any clause $C = \{s_1x_1,\ldots,s_kx_k\}$, we can select an index $i(C)$ such that $s_{i(C)}c(x_{i(C)})$ does not exceed $s_jc(x_j)$ for any $j \in [k]$, regardless of the exact values of $c(1),\ldots,c(n)$. Indeed, if all $s_1,\ldots,s_k$ are equal to 1, then we set $i(C) = \arg\min_{i\in[k]} x_i$, otherwise we set $i(C) = \arg\max_{i\in[k],s_i=-1} x_i$. But then $F(c)$ is a linear function of c — indeed

$$F(c) = \sum_{C=\{s_1x_1,\ldots,s_kx_k\}\in C} s_{i(C)}c(x_{i(C)}).$$

Now, defining $\tilde{c}(0) = 0$, $\tilde{c}(n+1) = 1$, let us suppose that there exist $0 < a \le b < n+1$ such that

$$0 = \tilde{c}(0) = \cdots = \tilde{c}(a-1) < \tilde{c}(a) = \cdots = \tilde{c}(b) < \tilde{c}(b+1) \le \cdots \le \tilde{c}(n+1) = 1.$$

If no such a,b exist, then all the values of $\tilde{c}$ are within $\{0,1\}$ and there is nothing to prove. Suppose we keep all of $\tilde{c}$ fixed save for $\tilde{c}(a) = \cdots = \tilde{c}(b)$; let $G(t)$ be the function taking t to the value of F for $\tilde{c}$ with $\tilde{c}(a) = \cdots = \tilde{c}(b)$ changed to t. For $t \in [0,\tilde{c}(b+1)]$, we see that $G(t)$ is a linear function (since $F(c)$ is linear so long as c is monotonic). Since a global minimum of G is hit for $\tilde{c}(a) \in (0,\tilde{c}(b+1))$, we see that $G(t)$ must be constant for $t \in [0,\tilde{c}(b+1)]$. Hence we can set $t = 0$ without modifying the value of $G(t)$. However by doing this we have reduced the number of distinct values in the image of $\tilde{c}$ by 1. Hence, by repeating this procedure at most n times we can achieve our goal, in polynomial time.

[2] In this formulation of the problem, we have multiplied by $2|C|$ and subtracted a constant factor of $|C|$ from the value. This rescaling does not essentially affect the problem, and is merely cosmetic.

8.3. Applications of promise max-and

Here we prove Corollary 8.11 and theorem 8.12, which we now restate.

Corollary 8.11. The max-and vs. even problem can be solved in polynomial time.

Proof. In this problem, we are given a vertex set V and a clause set C, such that the strong value of (V, C) is at least ρ. We must then find an assignment up to parity with value at least ρ. Our algorithm will be as follows: first do some simple preprocessing to our instance. Then solve (V, C) as an instance of promise max-and, yielding an assignment $\tilde{c}$. Then *randomly round* this assignment into a assignment up to parity. For such a rounding scheme to work, the probability that a clause is randomly rounded into a clause solved up to parity must be at least equal to the weak value of that clause under $\tilde{c}$ — linearity of expectation will then give that the expected fraction of clauses solved up to parity is at least the weak value of $\tilde{c}$, which is at least the strong value of (V, C). Finally we will derandomise this rounding scheme.

First, let us describe our preprocessing. Any clause C such that $+x, -x \in C$ can be safely deleted from C — indeed such a clause can never contribute to the strong value of (V, C), and thus removing it never lowers the strong value. Next, if sx appears twice in a clause C (recall that clauses are *multisets*), then we can remove two appearances of sx from C. This may increase the strong value of (V, C), but it will never modify the value of any assignment up to parity (since the two appearances of sx cancel out). After this preprocessing, each variable can appear in each clause at most once.

Our rounding scheme is very simple. Suppose the output of our rounding scheme is $c^\star : V \to \{\pm 1\}$. If $|\tilde{c}(x)| = 1$ then set $c^\star(x) = \tilde{c}(x)$ — otherwise set $c^\star(x)$ to ± 1 uniformly and independently at random. To see that this rounding scheme is correct in expectation, consider some clause $C = \{s_1 x_1, \ldots, s_k x_k\}$. We must show that the probability that this clause is rounded correctly up to parity is at least the weak value of the clause. There are three cases.

- **Weak value is 1**. In this case $s_i \tilde{c}(x_i) = 1$, hence in particular $|\tilde{c}(x_i)| = 1$ and thus $c^\star(x_i) = \tilde{c}(x_i)$ with probability 1 for all $i \in [k]$. Since $\tilde{c}$ takes only the value ± 1 for the variables in C, and it weakly satisfies C, we deduce that it also *strongly* satisfies C. Hence taking $c^\star(x_i) = \tilde{c}(x_i)$ also strongly satisfies C, and, a fortiori, satisfies C up to parity.

- **Weak value is $\frac{1}{2}$**. In this case there exists at least one $i \in [k]$ such that $s_i \tilde{c}(x_i) = 0$, and hence $\tilde{c}(x_i) = 0$. Thus $c^\star(x_i)$ is set uniformly and independently at random — considering the probability of success if $c^\star(x_i)$ is set after all the other variables shows us that the clause is satisfied up to parity with probability $\frac{1}{2}$. (This case is why the preprocessing step was necessary: without it, it might be the case that $\tilde{c}(x_i)$ has no effect upon the value of C up to parity.)

- **Weak value is 0**. Probabilities are non-negative, so there is nothing to prove in this case.

Hence we have shown that the expected value of the rounded assignment (up to parity) is at least the weak value of the original assignment.

The last step is to derandomise our algorithm. Observe that, after removing the variables that are set to +1 or −1 with probability 1, we are left with a set of variables set to ±1 uniformly and independently at random, and our constraints become parity constraints over these variables. We must now deterministically satisfy at least as many constraints as are expected of the random assignment. This is easy to do, and has been done before e.g. by [KSTW00] — we show how to do this for completeness. We use the method of conditional expectation: we fix variables one by one, maximising the expectation conditional on the fixed variable. Since every parity constraint contains each variable at most once (due to our preprocessing), the expectation conditional on any subset of fixed variables is easy to compute: any constraint that has any unfixed variables is satisfied with probability $\frac{1}{2}$, all the others are either certainly satisfied or certainly not satisfied. □

Theorem 8.12. maxPCSP$(\multimap, \omega)$ is solvable in polynomial time.

Proof. In this problem we are given a digraph $G = (V, E)$ which admits a dicut with value ρ, and we must find an acyclic subgraph with value ρ. Similarly to the proof of Corollary 8.11, we will proceed by first preprocessing, then solving an instance of promise max-and, then randomly rounding, then derandomising. The preprocessing step consists of removing loops from G: this does not modify the value of the promised dicut, nor the value of the acyclic subgraph that we return.

Next, consider the max-and instance with variables set V and clauses $\{-x, y\}$ for each edge $(x, y) \in E$. Similarly to Lemma 8.10, strong assignments of this instance correspond to dicuts in G, and hence the strong value of this instance is ρ. Using Theorem 8.8, compute a weak assignment $\tilde{c} : V \to \{-1, 0, 1\}$ with value ρ. We will now show how to randomly round this assignment into an acyclic subgraph of G.

In fact, we will not output an acyclic subgraph — we will output an ordering of V. The acyclic subgraph will consist of those edges that go from a smaller vertex to a larger vertex in this ordering; clearly such a subgraph will be acyclic. The ordering we output is as follows: first order according to the image through $\tilde{c}$ (in increasing order). In case of equal values, order at random.

To show that this rounding scheme outputs an acyclic subgraph with expected value ρ, it is sufficient to show that the probability that an edge (x, y) is ordered correctly by the outputted ordering is at least the weak value of $\{-x, y\}$ under $\tilde{c}$. There are three cases.

- $\boldsymbol{\tilde{c}(x) < \tilde{c}(y)}$. In this case (x, y) are ordered correctly with probability 1, and there is nothing to prove.
- $\boldsymbol{\tilde{c}(x) = \tilde{c}(y)}$. In this case the weak value is $\frac{1}{2} + \frac{1}{2}\min(-\tilde{c}(x)\tilde{c}(x)) \leq \frac{1}{2} + 0 = \frac{1}{2}$, with equality only for $\tilde{c}(x) = \tilde{c}(y) = 0$ — in all other cases the weak value is 0. Furthermore, in this case (x, y) is ordered correctly with probability $\frac{1}{2}$ (since x and y are ordered randomly).
- $\boldsymbol{\tilde{c}(x) > \tilde{c}(y)}$. By considering the possible values of $\tilde{c}(x), \tilde{c}(y)$, we see that in this case the weak value is 0 and there is nothing to prove.

Hence we have shown that our algorithm outputs an ordering that satisfies a ρ-fraction of edges in expectation i.e. an acyclic subgraph with value ρ in expectation. Now let us derandomise our algorithm. We observe that, to satisfy the same fraction of edges as our randomised scheme does in expectation, it suffices to satisfy all the edges (x, y) with $\tilde{c}(x) < \tilde{c}(y)$, and half of the edges with $\tilde{c}(x) = \tilde{c}(y) = 0$. So it suffices to consider two orderings: let π_{-1}, π_0, π_1 be arbitrary orderings on $\tilde{c}^{-1}(-1), \tilde{c}^{-1}(0)$ and $\tilde{c}^{-1}(1)$ respectively. Then, one of the orderings π_{-1}, π_0, π_1 or $\pi_{-1}, \text{reverse}(\pi_0), \pi_1$ will satisfy all the edges (x, y) with $\tilde{c}(x) < \tilde{c}(y)$ and at least half of the edges with $\tilde{c}(x) = \tilde{c}(y) = 0$. Thus it suffices to consider these two orderings and return the better of the two. □

9. A Dichotomy for maxPCSPs on Graphs

In this chapter, we will fully classify the *maxPCSP on graphs*: if G, H are two undirected, nonempty loopless graphs, then we will determine the complexity of $\mathrm{maxPCSP}(G, H)$. Our results will show that this problem is solvable in polynomial time if $G \to K_2 \to K_3 \to H$, and is NP-hard (assuming the Unique Games Conjecture of Khot [Kho02]) otherwise. In other words, either the algorithm for $\mathrm{maxPCSP}(K_2, K_3)$ essentially solves our problem, or it is (likely) unsolvable. Along the way, we will also investigate the *maximum bipartite vs. triangle-free subgraph problem*, denoted by $\mathrm{maxPCSP}(K_2, G_3)$. The hardness of this problem is important to the classification mentioned above; we will also give a nontrivial approximation algorithm. This work has appeared previously in the following works.

[NŽ25a] Tamio-Vesa Nakajima and Stanislav Živný. *A Dichotomy for Maximum PCSPs on Graphs*. 2025. arXiv: 2406.20069 [cs.DS].

[NŽ25c] Tamio-Vesa Nakajima and Stanislav Živný. 'Maximum Bipartite vs. Triangle-Free Subgraph'. In: *ICALP*. Vol. 334. LIPIcs. LZI, 2025, 121:1–121:16. DOI: 10.4230/lipics.icalp.2025.121.

Note that [NŽ25a] is a preprint full version of [NŽ25c]. The work in this chapter was produced in collaboration with my supervisor. I am thankful to Jakub Opršal, who suggested this direction of work, as well as to Pravesh Kothari, for suggesting finding the performance of the approximation algorithm for $\mathrm{maxPCSP}(K_2, G_3)$ in the $1 - \varepsilon$ regime, leading to part of Theorem 9.5.

9.1. Introduction

In this section, we will endeavour to produce a dichotomy for the $\mathrm{maxPCSP}(G, H)$ problem, where $G \to H$ are undirected, loopless, nonempty graphs.[1] This problem is very natural to consider once one decides to study the maxPCSP problem. Indeed, if one considers the history of the CSP, graph CSPs were among the first to be classified.

Theorem 9.1 ([HN90]). If G is an undirected, loopless, nonempty graph, then $\mathrm{CSP}(G)$ is tractable if and only if G is bipartite; otherwise it is NP-hard.

A further motivation towards studying this problem is the corresponding PCSP. There as well, the *approximate graph homomorphism* problem, $\mathrm{PCSP}(G, H)$, was among the first to be studied [BG21]. (Indeed, it is a generalisation of approximate graph colouring, which was

[1] The caveats "loopless" and "nonempty" hide two trivial cases of the problem, which needlessly complicate classifications later on.

studied independently years before [GJ76; Wig82].) Brakensiek and Guruswami conjecture the following.

Conjecture 1.4 ([BG21]). If $G \to H$ are undirected, loopless, nonempty graphs, then $\mathrm{PCSP}(G, H)$ is tractable if and only if G is bipartite; otherwise it is NP-hard.

Despite many years of work, this conjecture has not been resolved. Even the weaker conjecture of Garey and Johnson [GJ76], which covers the case where G, H are cliques, is not close to being solved. In the case of the non-promise CSP, the finite-valued CSP was solved completely before the non-valued CSP was (cf. compare [TŽ16] with [Zhu20; Bul17]). This suggests that proving a finite-valued version of Conjecture 1.4, i.e. classifying $\mathrm{maxPCSP}(G, H)$, may be easier than proving Conjecture 1.4. Additionally, perhaps studying the complexity of $\mathrm{maxPCSP}(G, H)$ will lead to ideas about the conjecture for $\mathrm{PCSP}(G, H)$?

With this motivation in hand, observe that $\mathrm{PCSP}(G, H)$ trivially reduces to $\mathrm{maxPCSP}(G, H)$ (in the sense that every instance to $\mathrm{PCSP}(G, H)$ is also an instance of $\mathrm{maxPCSP}(G, H)$, with the same desired output). This means that, if one believe Conjecture 1.4, then we should only hope for algorithms in the case where G is bipartite; this is equivalent to the case $G = K_2$. Heuristically, the obvious way to make an algorithm for this problem is to solve (some relaxation of) $\mathrm{maxCSP}(K_2)$, then round the assignment in such a way as to create an H-colouring. This motivates looking into optimal algorithms for $\mathrm{maxCSP}(K_2)$, as our algorithm will be a modification of this algorithm. This is indeed the case — our algorithm will be a combination between the Goemans-Williamson algorithm for max-cut [GW95], together with the Frieze-Jerrum rounding scheme for max-3-cut [FJ97]. (This rounding scheme is essentially the same as the rounding scheme of Karger, Motwani and Sudan for approximate graph colouring [KMS98].) The following is proved in Section 9.2.[2]

Proposition 9.2. $\mathrm{maxPCSP}(K_2, K_3)$ can be solved in polynomial time.

Next, we move on to proving a hardness result for $\mathrm{maxPCSP}(K_2, H)$ where H is triangle-free. To this end, we define the (infinite) graph G_3 as being any graph such that, for finite H, $H \to G_3$ if and only if H is triangle-free. One particular way to construct G_3 is to take the disjoint union of all finite triangle-free graphs. In some sense the most elegant model for G_3 is the *Henson graph*, which has the additional property of being *homogeneous* (i.e. every isomorphism between finite induced subgraphs extends to an automorphism of the entire graph). For more on this graph see [Hen71], however all that we care about is the property that any finite H maps homomorphically to G_3 if and only if H is triangle-free.

We state, for clarity, the precise definition of the problem $\mathrm{maxPCSP}(K_2, G_3)$. In this problem, we are given a graph with a cut of value ρ (i.e. a bipartite subgraph containing a ρ-fraction of the edges in the input), and are asked to find a triangle-free subgraph with value ρ. (In both of these sentences, a subgraph is a subset of the edges of the graph.) With this notation in mind, we see that for triangle-free H, the problem $\mathrm{maxPCSP}(K_2, G_3)$ reduces

[2] To be explicit: Recall that a 3-cut is a partition of the vertices of a graph into 3 sets so that as many edges as possible cross from one set into another, and that finding a maximum 3-cut is identical to $\mathrm{maxCSP}(K_3)$. Furthermore, in $\mathrm{maxPCSP}(K_2, K_3)$, we are given a graph G, and we must create a 3-cut with at least as many edges as any cut.

homomorphically[3] to maxPCSP(K_2, H). Hence, for the hardness of maxPCSP(K_2, H) it suffices to prove the hardness of maxPCSP(K_2, G_3).

How should one go about proving such a result? Observe that a graph is triangle-free if and only if it is "locally bipartite", i.e. it is bipartite if one looks only at subgraphs of size at most 4. Hence, heuristically one wants to use a "highly connected" gadget reduction that proves the hardness of max-cut, then prove that one can decode the gadget even if we merely find a triangle-free subgraph in it. This approach indeed works, and we use (and generalise) a gadget of Bellare, Goldreich and Sudan [BGS98] for max-cut to prove our result.

Proposition 9.3. maxPCSP(K_2, G_3) is NP-hard; indeed it is NP-hard even to $(25/26 + \varepsilon)$-approximate, for any $\varepsilon > 0$.

We prove this result in Section 9.3.

The remaining hardness result is maxPCSP(C_{2k+1}, K_ℓ) for $k \geq 1$ and $\ell \geq 3$. Luckily, Dinur, Mossel and Regev have shown [DMR09] a conditional hardness result for *almost colouring* i.e. given a graph that has a large *induced* 3-colourable subgraph, find a large independent set. By generalising this result, combined with a *multilayered Unique Games instance*, in the style of [DRS05; BWŽ21], we establish the following.

Proposition 9.4. maxPCSP(C_{2k+1}, K_ℓ) is NP-hard for $k \geq 1$ and $\ell \geq 3$, assuming the Unique Games Conjecture of Khot.

This result is proved in Section 9.4.

Finally, let us return to the problem maxPCSP(K_2, G_3). We have proved that it is NP-hard to approximate any better than $25/26$. Furthermore, this problem reduces to max-cut, hence it can be approximated within $0.878\cdots$ using the Goemans-Williamson algorithm. The Goemans-Williamson algorithm also returns a cut with value $1 - O(\sqrt{\varepsilon})$ when given a graph whose max-cut has value $1 - \varepsilon$. Can we do any better? Indeed we can — an algorithm better than the Goemans-Williamson algorithm is possible. Our approach uses the key fact that the worst-case instance for the Goemans-Williamson algorithm *is triangle-free itself*, and hence is a trivial instance for maxPCSP(K_2, G_3) (i.e. we can return the entire graph).

Theorem 9.5. maxPCSP(K_2, G_3) can be 0.8823-approximated in polynomial time. Furthermore, on a graph with a cut with value $1 - \delta$ edges, our algorithm outputs a triangle-free graph with value $1 - O(\delta)$.

This is proved in Section 9.5.

With all of these results in hand, we state and prove our classification theorem.

Theorem 1.5. Let $G \to H$ be two loopless, nonempty graphs. Then maxPCSP(G, H) can be solved in polynomial time if G is bipartite and H contains a triangle; otherwise it is NP-hard assuming the Unique Games Conjecture.

[3] Observe that homomorphic relaxations generalise from PCSPs to maxPCSPs, i.e. if $\mathbf{A} \to \mathbf{A}' \to \mathbf{B}' \to \mathbf{B}$, then maxPCSP$(\mathbf{A}, \mathbf{B})$ reduces to maxPCSP$(\mathbf{A}', \mathbf{B}')$.

Proof. If G is bipartite and H contains a triangle, then $G \to K_2 \to K_3 \to H$, and hence $\mathrm{maxPCSP}(G,H)$ reduces by homomorphic relaxation to $\mathrm{maxPCSP}(K_2,K_3)$, which is solvable in polynomial time by Proposition 9.2. If G is bipartite but H is triangle-free, then since $K_2 \to G$ as G is nonempty, we have that $\mathrm{maxPCSP}(K_2,G_3)$ reduces by homomorphic relaxation to $\mathrm{maxPCSP}(G,H)$, hence the latter problem is NP-hard by Proposition 9.3. Finally, suppose G is not bipartite; in this case $C_{2k+1} \to G \to H \to K_\ell$ for some values of $k \geq 1, \ell \geq 3$. Thus $\mathrm{maxPCSP}(C_{2k+1},K_\ell)$ reduces by homomorphic relaxation to $\mathrm{maxPCSP}(G,H)$, hence the latter problem is NP-hard under UGC by Proposition 9.4. □

Let us return to the question of whether $\mathrm{maxPCSP}(G,H)$ gives us insight on $\mathrm{PCSP}(G,H)$. On the one hand, our classification of $\mathrm{maxPCSP}(G,H)$ is as one would expect assuming Conjecture 1.4; we can thus interpret this classification as further evidence for Conjecture 1.4. On the other hand, the technical ingredients of Theorem 1.5 do not seem to immediately lead to a proof of Conjecture 1.4. This situation is to some extent disappointing. The general methodology of using maxPCSPs in order to study PCSPs may be more successful for the case of Boolean PCSPs, for which we do not even have a conjecture of where the line between easy and hard problems should lie. In such a case, a maxPCSP classification may give us a hint to where the boundary is.

9.2. Algorithm for $\mathrm{maxPCSP}(K_2,K_3)$

9.2.1. Goemans-Williamson algorithm

In this section we provide an algorithm for $\mathrm{maxPCSP}(K_2,K_3)$. This algorithm generalises the Goemans-Williamson algorithm [GW95], so let us first explain how that algorithm works. First, consider the following "arithmetic" formulation of the max-cut problem: given a graph $G=(V,E)$, introducing variables $x_u \in \mathbb{R}$ for $u \in V$, our goal is to solve the following optimisation problem:

$$\begin{aligned} \max_{x_u} \quad & \frac{1}{|E|} \sum_{(u,v)\in E} \frac{1-x_u x_v}{2} \\ \text{s.t.} \quad & x_u^2 = 1, \quad u \in V \end{aligned}$$

In other words: assign each variable ± 1, such that the (normalised) sum of $(1-x_u x_v)/2$ — which is 1 when $x_u \neq x_v$ and 0 when $x_u = x_v$ — over all edges (u,v) is maximised. Unfortunately, it is impossible to solve this problem directly. However, we can *relax* our problem using *semi-definite programming (SDP)*. More precisely, considering the variables $\boldsymbol{x}_u \in \mathbb{R}^{|V|}$ for $u \in V$, for the following problem we can find a feasible assignment whose value is at most ε away from the optimal assignment, in polynomial time with respect to $|G|$ and $\log(1/\varepsilon)$:

$$\begin{aligned} \max_{\boldsymbol{x}_u} \quad & \frac{1}{|E|} \sum_{(u,v)\in E} \frac{1-\boldsymbol{x}_u \cdot \boldsymbol{x}_v}{2} \\ \text{s.t.} \quad & \boldsymbol{x}_u \cdot \boldsymbol{x}_u = 1, \quad u \in V \end{aligned} \tag{9.1}$$

This problem now has a natural geometric interpretation vis-a-vis drawing graphs. In particular, one can see this problem in the following light: we want to embed the vertices of our graph into the $(|V|-1)$-dimensional hypersphere, so that the (normalised) sum of $(1-\boldsymbol{x}_u \cdot \boldsymbol{x}_v)/2 \propto ||\boldsymbol{x}_u - \boldsymbol{x}_v||_2^2$ for $(u,v) \in E$ is as large as possible. This is a natural target for nicely drawing graphs in very high-dimensional space.

Now that we have embedded our graph in space, how will we create a cut? An obvious way to do this is by selecting, uniformly at random, a hyperplane through the origin, and outputting the cut induced by that hyperplane (i.e. vertices are partitioned according to which side of the hyperplane they are on). Consider then some edge (u,v), and suppose that $\boldsymbol{x}_u \cdot \boldsymbol{x}_v = \sigma$, and hence the angle between $\boldsymbol{x}_u$ and $\boldsymbol{x}_v$ is $\theta = \arccos \sigma$. Projecting down into just the space spanned by $\boldsymbol{x}_u$ and $\boldsymbol{x}_v$, observe that the random hyperplane now becomes a random line through the origin. The probability that $\boldsymbol{x}_u$ and $\boldsymbol{x}_v$ are separated by such a line is proportional to θ, with it being 0 when $\theta = 0$ and 1 when $\theta = \pi$. Hence the probability is θ/π. In order to analyse what the worst possible approximation ratio is, let us consider the ratio between this probability and the value the edge brings to the SDP objective value i.e. $(1-\sigma)/2$. We are therefore interested in the ratio

$$\min_{0 \leq \theta \leq \pi} \frac{2}{\pi} \frac{\theta}{1-\cos\theta} = 0.878\cdots,$$

where the minimum is hit at $\theta \approx 0.742\pi$. Hence we conclude that the approximation ratio is at least $0.878\cdots$.

9.2.2. Frieze-Jerrum rounding; our algorithm

We now want to round (9.1) so as to produce a 3-cut. Luckily, Frieze and Jerrum [FJ97], when trying to approximate max-3-cut, have already done this (albeit for a slightly different SDP, one in which we also constrain that each edge must have angle at most $2\pi/3$). Their key insight was that a random hyperplane can be sampled in the following way: take two vectors $\boldsymbol{y}_1, \boldsymbol{y}_2$ from a radially symmetric distribution, and consider the hyperplane of points $\boldsymbol{x}$ for which $\boldsymbol{x} \cdot \boldsymbol{y}_1 = \boldsymbol{x} \cdot \boldsymbol{y}_2$. In the Goemans-Williamson algorithm, this amounts to colouring a vertex u with the colour $\arg\max_i \boldsymbol{x}_u \cdot \boldsymbol{y}_i$. Frieze and Jerrum generalise this approach to max-k-cut, by randomly sampling $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ from a multivariate normal distribution. (Let us mention again that this approach is essentially the same as that of Karger, Motwani and Sudan [KMS98].)

We will analyse this approach when applied to rounding (9.1). Our key technical contribution is an explicit form for the performance of Frieze and Jerrum's rounding scheme for max-3-cut, using a formula of Cheng [Che68; Che69]. First, let us go through the technical machinery necessary for the algorithm. Let us first introduce some standard notation for random variables.

Definition 9.6. A *Normalised Binomial random variable with parameter* $n \in \mathbb{N}$, denoted by $\mathrm{NBin}(n)$, is a random variable distributed identically to $X_1 + \cdots + X_n$, where $X_1, \ldots, X_n$ are independent and identically distributed, with X_i being uniformly distributed within $\{\pm\frac{1}{\sqrt{n}}\}$. In particular, $\mathrm{NBin}(n)$ has mean 0 and variance 1, and it has has a domain of size $O(n)$.

Definition 9.7. A *normally distributed variable with mean* $\mu \in \mathbb{R}$ *and variance* $\sigma^2 \in \mathbb{R}$, denoted by $N(\mu, \sigma)$, is a continuous random variable with real domain, whose probability density function at x is given by

$$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

The case $\mu = 0, \sigma = 1$ is called the *standard normal distribution.*

Definition 9.8. A t*-variate normally distributed variable with mean* $\boldsymbol{\mu} \in \mathbb{R}^t$ *and covariance matrix* $\Sigma \in \mathbb{R}^{t\times t}$*, denoted by* $N(\boldsymbol{\mu}, \Sigma)$, where Σ must be invertible, is a continuous random variable with domain $\mathbb{R}^t$, whose probability density function at $\boldsymbol{x}$ is given by

$$(2\pi)^{-t/2}\det(\Sigma)^{-1/2} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right).$$

In particular, if $\boldsymbol{\mu} = (0, \ldots, 0) = \mathbf{0}$ and $\Sigma = I_t$ is the identity matrix, then $N(\boldsymbol{\mu}, \Sigma)$ is distributed as t independent and identically distributed standard normal variables. We call this case the *standard* t*-variate normal distribution.*

One particular fact we will use is that, if $\Sigma = AA^T$, then $N(\boldsymbol{\mu}, AA^T)$ is distributed identically to $A\boldsymbol{x} + \boldsymbol{\mu}$, where $\boldsymbol{x}$ is a standard t-variate normal variable.

Theorem 9.9 ([Che68], [Che69, Equation (2.18)]). Take $a, b \in [-1, 1]$ and let

$$\Sigma = \begin{pmatrix} 1 & a & b & ab \\ a & 1 & ab & b \\ b & ab & 1 & a \\ ab & b & a & 1 \end{pmatrix}.$$

Suppose $(s_1, s_2, s_3, s_4) \sim N(\mathbf{0}, \Sigma)$. Then the *orthant probability* of (s_1, s_2, s_3, s_4) is given by

$$\Pr[s_1 \geq 0, s_2 \geq 0, s_3 \geq 0, s_4 \geq 0] = \frac{1}{16} + \frac{\arcsin a + \arcsin b + \arcsin ab}{4\pi} + \frac{(\arcsin a)^2 + (\arcsin b)^2 - (\arcsin ab)^2}{4\pi^2}.$$

Corollary 9.10. Fix $\sigma \in [-1, 1]$ and suppose $x_1, x_2, x_3, y_1, y_2, y_3 \sim N(0, 1)$ are independent, identically distributed normal variables. The probability that

$$\begin{aligned} x_1 &\geq x_2 \\ x_1 &\geq x_3 \\ \sigma x_1 + \sqrt{1-\sigma^2}y_1 &\geq \sigma x_2 + \sqrt{1-\sigma^2}y_2 \\ \sigma x_1 + \sqrt{1-\sigma^2}y_1 &\geq \sigma x_3 + \sqrt{1-\sigma^2}y_3 \end{aligned} \tag{9.2}$$

is given by $P(\sigma) = \frac{1}{9} + \frac{\arcsin\sigma + \arcsin\frac{\sigma}{2}}{4\pi} + \frac{(\arcsin\sigma)^2 - (\arcsin\frac{\sigma}{2})^2}{4\pi^2}$.

Proof. Define

$$s_1 = \frac{1}{\sqrt{2}}(x_1 - x_2) \qquad s_3 = \frac{\sigma}{\sqrt{2}}(x_1 - x_2) + \sqrt{\frac{1-\sigma^2}{2}}(y_1 - y_2)$$
$$s_2 = \frac{1}{\sqrt{2}}(x_1 - x_3) \qquad s_4 = \frac{\sigma}{\sqrt{2}}(x_1 - x_3) + \sqrt{\frac{1-\sigma^2}{2}}(y_1 - y_3)$$

In other words, if we collect $x_1, x_2, x_3, y_1, y_2, y_3$ into the vector $(x_1, x_2, x_3, y_1, y_2, y_3) = \boldsymbol{x} \sim N(\mathbf{0}, I_6)$ (recalling I_6 is the 6×6 identity matrix), then the vector $\boldsymbol{s} = (s_1, s_2, s_3, s_4)$ is defined by $\boldsymbol{s} = A\boldsymbol{x}$ where

$$A = \left(\begin{array}{c} \boldsymbol{a}_1 \\ \hline \boldsymbol{a}_2 \\ \hline \boldsymbol{a}_3 \\ \hline \boldsymbol{a}_4 \end{array}\right) = \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ \frac{\sigma}{\sqrt{2}} & -\frac{\sigma}{\sqrt{2}} & 0 & \sqrt{\frac{1-\sigma^2}{2}} & -\sqrt{\frac{1-\sigma^2}{2}} & 0 \\ \frac{\sigma}{\sqrt{2}} & 0 & -\frac{\sigma}{\sqrt{2}} & \sqrt{\frac{1-\sigma^2}{2}} & 0 & -\sqrt{\frac{1-\sigma^2}{2}}. \end{pmatrix}$$

We now note that $\boldsymbol{s}$ is a multivariate normal variable, distributed according to $N(\mathbf{0}, AA^T)$. Let us compute AA^T. This matrix contains the inner products $\boldsymbol{a}_i \cdot \boldsymbol{a}_j$ for $i, j \in [4]$; let us compute these values. We save some computation by noting that $\boldsymbol{a}_1 \cdot \boldsymbol{a}_1 = \boldsymbol{a}_2 \cdot \boldsymbol{a}_2$, $\boldsymbol{a}_3 \cdot \boldsymbol{a}_3 = \boldsymbol{a}_4 \cdot \boldsymbol{a}_4$, $\boldsymbol{a}_1 \cdot \boldsymbol{a}_3 = \boldsymbol{a}_2 \cdot \boldsymbol{a}_4$ and $\boldsymbol{a}_1 \cdot \boldsymbol{a}_4 = \boldsymbol{a}_2 \cdot \boldsymbol{a}_3$ by symmetry.

$$\boldsymbol{a}_1 \cdot \boldsymbol{a}_1 = \left(\frac{1}{\sqrt{2}}\right)^2 + \left(-\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2} + \frac{1}{2} = 1,$$

$$\boldsymbol{a}_1 \cdot \boldsymbol{a}_2 = \frac{1}{\sqrt{2}} \times \frac{1}{\sqrt{2}} = \frac{1}{2},$$

$$\boldsymbol{a}_1 \cdot \boldsymbol{a}_3 = \frac{1}{\sqrt{2}} \times \frac{\sigma}{\sqrt{2}} + \left(-\frac{1}{\sqrt{2}}\right) \times \left(-\frac{\sigma}{\sqrt{2}}\right) = \frac{\sigma}{2} + \frac{\sigma}{2} = \sigma,$$

$$\boldsymbol{a}_1 \cdot \boldsymbol{a}_4 = \frac{1}{\sqrt{2}} \times \frac{\sigma}{\sqrt{2}} = \frac{\sigma}{2},$$

$$\boldsymbol{a}_3 \cdot \boldsymbol{a}_3 = \left(\frac{\sigma}{\sqrt{2}}\right)^2 + \left(-\frac{\sigma}{\sqrt{2}}\right)^2 + \left(\sqrt{\frac{1-\sigma^2}{2}}\right)^2 + \left(-\sqrt{\frac{1-\sigma^2}{2}}\right)^2$$
$$= \frac{\sigma^2}{2} + \frac{\sigma^2}{2} + \frac{1-\sigma^2}{2} + \frac{1-\sigma^2}{2} = 1,$$

$$\boldsymbol{a}_3 \cdot \boldsymbol{a}_4 = \frac{\sigma}{\sqrt{2}} \times \frac{\sigma}{\sqrt{2}} + \sqrt{\frac{1-\sigma^2}{2}} \times \sqrt{\frac{1-\sigma^2}{2}} = \frac{\sigma^2}{2} + \frac{1-\sigma^2}{2} = \frac{1}{2}.$$

Thus,

$$AA^T = \begin{pmatrix} 1 & \frac{1}{2} & \sigma & \frac{\sigma}{2} \\ \frac{1}{2} & 1 & \frac{\sigma}{2} & \sigma \\ \sigma & \frac{\sigma}{2} & 1 & \frac{1}{2} \\ \frac{\sigma}{2} & \sigma & \frac{1}{2} & 1 \end{pmatrix}.$$

Furthermore the event (9.2) is equivalent to $s_1 \geq 0, s_2 \geq 0, s_3 \geq 0, s_4 \geq 0$. Thus, applying Theorem 9.9, we see that the probability of (9.2) is

$$P(\sigma) = \frac{1}{16} + \frac{\arcsin\frac{1}{2} + \arcsin\sigma + \arcsin\frac{\sigma}{2}}{4\pi} + \frac{\left(\arcsin\frac{1}{2}\right)^2 + (\arcsin\sigma)^2 - \left(\arcsin\frac{\sigma}{2}\right)^2}{4\pi^2}$$
$$= \frac{1}{9} + \frac{\arcsin\sigma + \arcsin\frac{\sigma}{2}}{4\pi} + \frac{(\arcsin\sigma)^2 - \left(\arcsin\frac{\sigma}{2}\right)^2}{4\pi^2}. \qquad \square$$

Lemma 9.11. For $\sigma \in [-1, 1]$, we have

$$1 - 3P(\sigma) \geq \frac{1-\sigma}{2}.$$

Proof. Equivalently, we want to prove that

$$P(\sigma) \leq \frac{1+\sigma}{6} \tag{9.3}$$

for $\sigma \in [-1, 1]$.

First, consider the third derivative of P. Note that (cf. Figure 9.1)

$$\min_{-1<\sigma<1} P'''(\sigma) \approx 0.00484 > 0.$$

In particular this means that P'' is increasing on $(-1, 1)$ and P' is convex on $(-1, 1)$. Next, defining P'_+ to be the right derivative and P'_- to be the left derivative, we have

$$P'_+(-1) \approx 0.111 < \frac{1}{6}$$
$$P'_-(1) = \infty > \frac{1}{6}.$$

By the convexity of P' it follows that P' can be equal to $\frac{1}{6}$ exactly once on the interval $[-1, 1]$, it is less than $\frac{1}{6}$ to the left of that point, and greater to the right.

Thus, and since (9.3) holds with equality for $\sigma = \pm 1$, we observe that $P(\sigma) - \frac{1+\sigma}{6}$ is zero at -1, decreases up to a point, then increases until it becomes zero again at $+1$. This implies our result. $\square$

We first provide a randomised algorithm for maxPCSP(K_2, K_3), i.e. one that in a graph with a cut with value ρ, outputs a 3-cut with expected value $\rho - \varepsilon$, in polynomial time in the size of the graph and $\log(1/\varepsilon)$. Again, this is just an application of the rounding scheme of [FJ97], but to the SDP of [GW95]. Next we will derandomise this algorithm.

Theorem 9.12. There exists a polynomial-time randomised algorithm for maxPCSP(K_2, K_3), i.e. given a graph G that admits a cut with value ρ, we output a 3-cut with expected value $\rho - \varepsilon$, in polynomial time with respect to $|G|$ and $\log(1/\varepsilon)$.

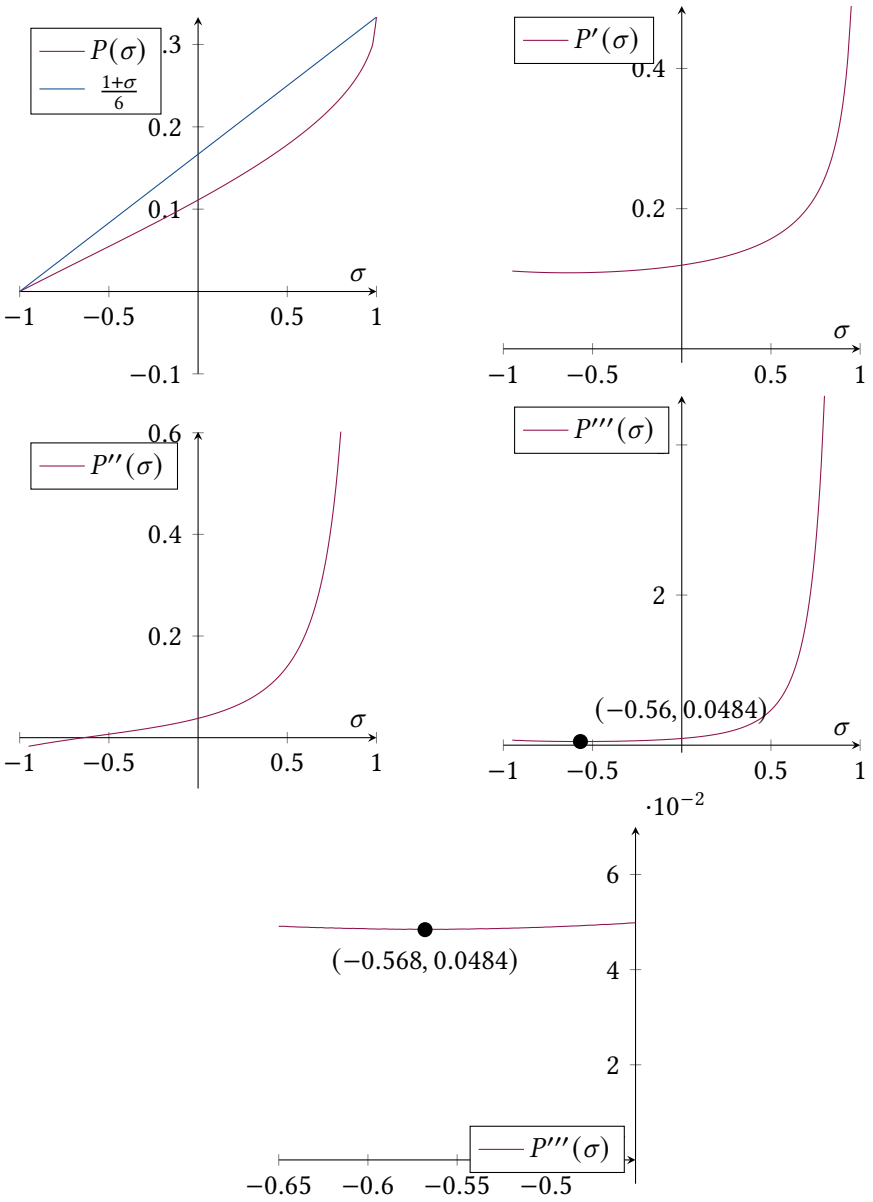


Figure 9.1.: Plots of functions from Lemma 9.11.

Proof. Suppose $G = (V, E)$ is the input graph, and suppose that it admits a cut with value ρ. This implies that the value of (9.1) is at least ρ (indeed, set the vectors to opposite unit vectors depending on the side of the ρ-edge cut their vertices find themselves on). We first solve (9.1) to within ε in polynomial time with respect to $|G|$ and $\log(1/\varepsilon)$. Hence, generate vectors $\{\boldsymbol{x}_u\}_{u \in V}$, with $\boldsymbol{x}_u \in \mathbb{R}^{|V|}$, $\boldsymbol{x}_u \cdot \boldsymbol{x}_u = 1$ and

$$\frac{1}{|E|} \sum_{(u,v) \in E} \frac{1 - \boldsymbol{x}_u \cdot \boldsymbol{x}_v}{2} \geq \rho - \varepsilon.$$

Next, generate three independent, identically distributed vectors $\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3 \in \mathbb{R}^{|V|}$ according to a standard $|V|$-variate normal distribution. We decide the colour of vertex u by

$$\arg\max_i \boldsymbol{y}_i \cdot \boldsymbol{x}_u.$$

By linearity of expectation, it is sufficient that an edge (u, v) with $\boldsymbol{x}_u \cdot \boldsymbol{x}_v = \sigma$ is properly cut by this rounding scheme with probability at least $\frac{1-\sigma}{2}$ — in that case, the expected fraction of properly cut edges is at least $\rho - \varepsilon$.

Let us therefore compute the probability that an edge (u, v) with $\boldsymbol{x}_u \cdot \boldsymbol{x}_v = \sigma$ is properly cut. Since the rounding scheme is rotationally symmetric, we can rotate and project our vectors down to the spanning plane of $\boldsymbol{x}_u, \boldsymbol{x}_v$; in particular we can take $\boldsymbol{x}_u = (1, 0)$ and $\boldsymbol{x}_v = (\sigma, \sqrt{1 - \sigma^2})$. Next, the projections of our three vectors $\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3$ to this plane are three independent, identically distributed vectors $(x_1, y_1), (x_2, y_2), (x_3, y_3) \in \mathbb{R}^2$ distributed according to standard bivariate normal distributions.

What is then the probability that both u and v are assigned, say, colour 1? It is precisely the probability of the event (9.2) — hence, the probability that both u and v are assigned colour 1 is exactly $P(\sigma)$. By symmetry, the edge (u, v) is coloured correctly with probability $1 - 3P(\sigma)$, and this is at least $\frac{1-\sigma}{2}$ by Lemma 9.11. This concludes the proof. □

9.2.3. Derandomisation

In this section, we derandomise our algorithm. Our method uses the method of conditional expectations, similarly to Mahajan and Ramesh [MR99] or Bhargava and Kosaraju [BK05]. This latter algorithm is the most similar to ours: it uses conditional expectations, and uses an approximation of the conditional expectation to find a good-enough way of setting the current variable. In both [MR99; BK05], the techniques are claimed to apply to the rounding scheme of Frieze and Jerrum (which, again, is what we use). We create a new derandomisation theorem, which we believe is conceptually simpler.

In particular, we prove the following.

Theorem 9.13. Fix a constant $d \in \mathbb{N}$. Suppose we have $n, m \in \mathbb{N}$ and $x_{ijk}, y_{ij}, \varepsilon \in \mathbb{R}$ for $i \in [m], j \in [d], k \in [n]$. Suppose $a_1, \ldots, a_n$ are independent, identically distributed standard normal variables, and that

$$\sum_{i=1}^{m} \Pr\left[\bigwedge_{j=1}^{d} \sum_{k=1}^{n} x_{ijk} a_k > y_{ij}\right] \geq \rho.$$

Then, we can compute $a_1^*, \ldots, a_n^* \in \mathbb{R}$ such that

$$\sum_{i=1}^{m} \left[\bigwedge_{j=1}^{d} \sum_{k=1}^{n} x_{ijk} a_k^* > y_{ij}\right] \geq \rho - \varepsilon. \tag{9.4}$$

in polynomial time with respect to $n, m, 1/\varepsilon$.[4]

We will use the method of conditional expectations. It is therefore sufficient to show how to set just one of the random variables, and repeat inductively; we codify this observation in the following lemma.

Lemma 9.14. Consider the same situation as in Theorem 9.13. Then we can compute $a_n^* \in \mathbb{R}$ such that

$$\sum_{i=1}^{m} \Pr\left[\bigwedge_{j=1}^{d} \sum_{k=1}^{n-1} x_{ijk} a_k > y_{ij} - x_{ijn} a_n^*\right] \geq \rho - \varepsilon \tag{9.5}$$

in polynomial time with respect to $n, m, 1/\varepsilon$.

Proof of Theorem 9.13. Apply Lemma 9.14 n times, with error ε/n. □

We now devote the rest of this section to the proof of Lemma 9.14. Our general approach will be to approximate the value on the left side of (9.5) as a function of a_n^* by a step function with few steps, then to find the maximum of this approximation. We will do this by first reducing each probability in the expression to a probabilistic statement about at most d independent normal variables, then to approximate the d-variate standard normal distribution by the normalised binomial distribution.

Let us first give the theorem that allows us to do this approximation. We will use the following version of the Berry-Esseen theorem, with explicit bounds, due to Raič.

Theorem 9.15 ([Rai19]). Fix $d, N \in \mathbb{N}$, and suppose that $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^d$ are independent random variables with mean zero. Let $\boldsymbol{x} = \sum_{i=1}^{N} \boldsymbol{x}_i$, and suppose that the covariance matrix of $\boldsymbol{x}$ is I_d (i.e. the $d \times d$ identity matrix). Suppose $\boldsymbol{y}$ is a standard d-variate normal variable, and let $E \subseteq \mathbb{R}^d$ be a convex, measurable set. Then

$$|\Pr[\boldsymbol{x} \in E] - \Pr[\boldsymbol{y} \in E]| \leq (42\sqrt[4]{d} + 16) \sum_{i=1}^{N} \mathbb{E}[||\boldsymbol{x}_i||^3].$$

[4] For (9.4), recall that $[\phi]$ is 1 if ϕ is true and 0 if ϕ is false.

Applying this allows us to approximate $N(0,1)$ by $\mathrm{NBin}(N)$ for large enough N.

Corollary 9.16. Fix $d \in \mathbb{N}, \varepsilon > 0$, and let

$$N \geq \left(\frac{42d^{7/4} + 16d^{3/2}}{\varepsilon}\right)^2 = \frac{O(d^{7/2})}{\varepsilon^2}.$$

Suppose $x_1, \ldots, x_d \sim \mathrm{NBin}(N)$ and, respectively, $y_1, \ldots, y_d \sim N(0,1)$ are independent and identically distributed, and let $E \subseteq \mathbb{R}^d$ be a convex, measurable set. Then

$$|\Pr[(x_1, \ldots, x_d) \in E] - \Pr[(y_1, \ldots, y_d) \in E]| \leq \varepsilon.$$

Proof. Indeed, let X_{ij} be uniformly distributed on $\{\pm 1/\sqrt{N}\}$, for $i \in [d], j \in [N]$, and observe that $x_1, \ldots, x_d$ are identically distributed to $\sum_{j=1}^N X_{1j}, \ldots, \sum_{j=1}^N X_{dj}$. Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ be the columns of X_{ij}, and note that $(x_1, \ldots, x_d) = \boldsymbol{x}_1 + \cdots + \boldsymbol{x}_N$. Apply Theorem 9.15 to the vector $(y_1, \ldots, y_d)$ and to $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$. The sum of the columns is just the vector $(x_1, \ldots, x_d)$, which has covariance matrix I_d as required; furthermore observe $||\boldsymbol{x}_i||^3 = (d/N)^{3/2}$. By our choice of N, we have that

$$(42\sqrt[4]{d} + 16) \sum_{i=1}^{N} \mathbb{E}[||\boldsymbol{x}_i||^3] = (42\sqrt[4]{d} + 16) N (d/N)^{3/2} \leq \varepsilon.$$

Hence the conclusion follows. □

With this in hand we can prove Lemma 9.14.

Proof of Lemma 9.14. Consider first one particular term of the sum in (9.5) as a function of $t = a_n^*$, i.e. for some particular i the value

$$\Pr\left[\bigwedge_{j=1}^{d} \sum_{k=1}^{n-1} x_{ijk} a_k > y_{ij} - x_{ijn} t\right].$$

We will show how to approximate this function up to error δ by a step function with polynomially many steps in $n, 1/\delta$. Setting $\delta = \varepsilon/2m$ and summing these approximations up allows us to approximate (9.5) as a function of a_n^* with error $\varepsilon/2$. Observe that the maximum value of (9.5) (as a function of a_n^*) is at least ρ by assumption; hence the maximum of the approximation is at least $\rho - \varepsilon/2$. Thus, if we set a_n^* to be the value which maximises the approximation, we return a result with true value at least $\rho - \varepsilon$. Hence finding this approximation is sufficient.

Indeed, let us simplify our task further. We rename all the variables in the term above, as well as divide by $|x_{ijn}|$ if it is nonzero, and collect constants together. It is therefore sufficient to approximate the function

$$f(t) = \Pr\left[\bigwedge_{i=1}^{d} \sum_{j=1}^{n} x_{ij} a_j > s_i t + b_i\right],$$

(where $s_j \in \{-1, 0, 1\}$) to within δ with a step function with polynomially many steps in $n, 1/\delta$. Let $X = (x_{ij}) \in \mathbb{R}^{d \times n}$ and collect $a_1, \ldots, a_n; b_1, \ldots, b_d$ and $s_1, \ldots, s_i$ into vectors $\boldsymbol{a} \in \mathbb{R}^n$ and $\boldsymbol{b}, \boldsymbol{s} \in \mathbb{R}^d$. Observe that the components of $\boldsymbol{a}$ are independent, identically distributed standard normal variables, so $\boldsymbol{a} \sim N(\boldsymbol{0}, I_n)$. Let $\boldsymbol{t} = (t, \ldots, t)^T$ denote the vector containing only t, and let $>$ between vectors denote the pointwise order (i.e. a vector is greater than another if and only if it is greater in each component). Let $\odot$ denote the pointwise multiplication of vectors. Then, by substituting definitions,

$$f(t) = \Pr\left[X\boldsymbol{a} > \boldsymbol{s} \odot \boldsymbol{t} + \boldsymbol{b}\right].$$

Observe that $X\boldsymbol{a} \sim N(\boldsymbol{0}, XX^T)$. Let us compute the Cholesky decomposition of the positive semidefinite $d \times d$ matrix XX^T; i.e. let us compute a matrix $X' \in \mathbb{R}^{d \times d}$ in polynomial time such that $XX^T = X'X'^T$. Then we see that $X\boldsymbol{a} \sim N(\boldsymbol{0}, X'X'^T)$ But, let $\boldsymbol{a}' \sim N(\boldsymbol{0}, I_d)$ i.e. let $\boldsymbol{a}' = (a'_1, \ldots, a'_d)^T$ such that $a'_1, \ldots, a'_d$ are independent, identically distributed standard normal variables. Then we see that $X'\boldsymbol{a}' \sim N(\boldsymbol{0}, X'X'^T)$ as well. In other words, $X\boldsymbol{a}$ and $X'\boldsymbol{a}'$ are identically distributed, so

$$f(t) = \Pr\left[X'\boldsymbol{a}' > \boldsymbol{s} \odot \boldsymbol{t} + \boldsymbol{b}\right].$$

What have we gained by this transformation? We have reduced the dimension of our problem from n to d.

Take $N = O(d^{7/2})/\delta^2$ as in Corollary 9.16, let $a''_1, \ldots, a''_d \sim \mathrm{NBin}(N)$, $\boldsymbol{a}'' = (a''_1, \ldots, a''_d)$, and define the set

$$E_t = \{\boldsymbol{a}' \in \mathbb{R}^d \mid X'\boldsymbol{a}' > \boldsymbol{s} \odot \boldsymbol{t} + \boldsymbol{b}\}.$$

This is an intersection of d half-spaces, so it is convex and measurable. Observe $f(t) = \Pr[\boldsymbol{a}' \in E_t]$, and let $g(t) = \Pr[\boldsymbol{a}'' \in E_t]$. Hence, applying Corollary 9.16, we have

$$|f(t) - g(t)| = |\Pr[\boldsymbol{a}' \in E_t] - \Pr[\boldsymbol{a}'' \in E_t]| \leq \delta.$$

Hence the function g is a viable approximation — this is the function we will output.

Now, we will show that g is a step function with polynomially many steps, which we can compute in polynomial time. Indeed, note that the variable $\boldsymbol{a}''$ is a vector of d variables distributed according to $\mathrm{NBin}(N)$. Note that each such variable can take $O(N)$ different values (which are easy to compute). Hence $\boldsymbol{a}''$ takes at most $O(N^d)$ different values, all of which are easy to compute. Let D be the set of these values, and let $p(\boldsymbol{a}'')$ be the probability of one of them. Then

$$g(t) = \Pr[\boldsymbol{a}'' \in E_t] = \sum_{\boldsymbol{a}'' \in D} p(\boldsymbol{a}'')[\boldsymbol{a}'' \in E_t].$$

We will show that $t \mapsto [\boldsymbol{a}'' \in E_t]$ is a step function with at most 2 steps, which is easy to compute. This is sufficient to prove our result. Indeed, note that the value associated to t is 1 precisely when

$$X'\boldsymbol{a}' > \boldsymbol{s} \odot \boldsymbol{t} + \boldsymbol{b}, \tag{9.6}$$

and is zero otherwise. Note that this expression amounts to at most d linear constraints on t, each of which can be computed in polynomial time. Then condition (9.6) is true precisely when t is contained within some interval which can be computed in polynomial time. Hence, $t \mapsto [\boldsymbol{a}'' \in E_t]$ is a step function with at most two steps, the locations of the steps are easy to compute, and the value of the function between the steps is either 0 or 1. This is sufficient to complete the proof. □

Finally, let us use Theorem 9.13 to show the tractability of our problem.

Proposition 9.2. maxPCSP(K_2, K_3) can be solved in polynomial time.

Proof. Assume that $|E| \geq 1$. Note that Theorem 9.12 amounts to the following: Given an input graph $G = (V, E)$ that admits a cut with value ρ, we can assign each vertex $v \in V$ a vector $\boldsymbol{x}_v \in \mathbb{R}^{|V|}$ such that, if $\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3 \sim N(\mathbf{0}, I_{|V|})$ are independent, identically distributed standard $|V|$-variate normal vectors, then

$$\frac{1}{|E|} \sum_{(u,v)\in E} \Pr[\arg\max_i \boldsymbol{x}_u \cdot \boldsymbol{y}_i \neq \arg\max_i \boldsymbol{x}_v \cdot \boldsymbol{y}_i] \geq \rho - \varepsilon,$$

in polynomial time with respect to $|G|, \log(1/\varepsilon)$. Equivalently,

$$\sum_{(u,v)\in E} \sum_{p\neq q} \Pr[\arg\max_i \boldsymbol{x}_u \cdot \boldsymbol{y}_i = p \wedge \arg\max_i \boldsymbol{x}_v \cdot \boldsymbol{y}_i = q] \geq |E|(\rho - \varepsilon).$$

Now, note that each event is a conjunction of $O(1)$ different bounds on linear combinations of the components within $\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3$. Hence, we can apply Theorem 9.13 to this expression, and find some particular $\boldsymbol{y}_1^*, \boldsymbol{y}_2^*, \boldsymbol{y}_3^*$, such that

$$\sum_{(u,v)\in E} \sum_{p\neq q} [\arg\max_i \boldsymbol{x}_u \cdot \boldsymbol{y}_i^* = p \wedge \arg\max_i \boldsymbol{x}_v \cdot \boldsymbol{y}_i^* = q] \geq |E|(\rho - \varepsilon) - \varepsilon \geq |E|(\rho - 2\varepsilon), \quad (9.7)$$

in polynomial time with respect to $|G|, 1/\varepsilon$. Thus, set $\varepsilon = 1/(4|E|)$, and find these particular vectors. Finally, colour u with $\arg\max_i \boldsymbol{x}_u \cdot \boldsymbol{y}_i^*$; this will correctly colour at least a $(\rho - 1/(2|E|))$-fraction of the edges by (9.7). Since the fraction of correctly coloured vectors is an integer divided by $|E|$, as is ρ, we correctly colour at least a ρ-fraction of the edges. □

9.3. Hardness for maxPCSP(K_2, G_3)

In this section we prove Proposition 9.3, which we restate here.

Proposition 9.3. maxPCSP(K_2, G_3) is NP-hard; indeed it is NP-hard even to $(25/26 + \varepsilon)$-approximate, for any $\varepsilon > 0$.

Our proof follows a standard pattern for proving hardness of approximation, namely gadget reducing from the 3-bit PCP of Håstad [Hås01]. For a system of modulo 2 linear equations in 3 variables, we say that the *value* of the system is the maximum proportion of the equations that can be simultaneously satisfied.

Theorem 9.17 ([Hås01]). For all small enough $\varepsilon > 0$, the following problem is NP-hard: Given a system of modulo 2 linear equations in 3 variables, decide whether it has value at least $1 - \varepsilon$, or not even $\frac{1}{2} + \varepsilon$.

This theorem is not quite strong enough for our purposes, however a very small strengthening is. A system of modulo 2 linear equations in 3 variables contains equations of one of two forms: $x + y + z \equiv 0 \bmod 2$ and $x + y + z \equiv 1 \bmod 2$. We say a system is *balanced* if it includes the same number of equations in these two forms. We say that an assignment is *balanced* if it satisfies the same number of equations of the two forms. Furthermore, the *balanced value* of a system is the maximum proportion of the equations that can be simultaneously satisfied by any balanced assignment.

Corollary 9.18 (Implicit in [Hås01]). For all small enough $\varepsilon > 0$, the following problem is NP-hard: Given a *balanced* system of modulo 2 linear equations in 3 variables, decide whether it has *balanced* value at least $1 - \varepsilon$, or (unbalanced) value not even $\frac{1}{2} + \varepsilon$.

While this may follow from the analysis in [Hås01], it can also be proved in a simple and direct way, which we now do here.

Proof. We will show that, for any system of modulo 2 linear equations in three variables $\mathcal{E} = \{x_{a_i} + x_{b_i} + x_{c_i} \equiv t_i\}_{i\in[m]}$, over variables $x_1, \ldots, x_n$, we can compute a *balanced* system of modulo 2 linear equations in three variables that has the same (balanced) value, in polynomial time. Indeed, take the system

$$\mathcal{E}' = \{x_{a_i} + x_{b_i} + z_{c_i} \equiv t_i\}_{i\in[m]} \cup \{y_{a_i} + y_{b_i} + y_{c_i} \equiv 1 - t_i\}_{i\in[m]},$$

over variables $x_1, \ldots, x_n, y_1, \ldots, y_n$. We show that a (balanced or unbalanced as appropriate) assignment to one of the systems can be transformed into an assignment to the other one without losing value. Indeed, any assignment $(\hat{x}_i)_{i\in[n]}$ for $\mathcal{E}$ can be extended into a balanced assignment of the same value for $\mathcal{E}'$ by setting $\hat{y}_i = 1 - \hat{x}_i$. Conversely, for any assignment $(\hat{x}_i, \hat{y}_i)_{i\in[n]}$, one of the two assignments $x_i \mapsto \hat{x}_i$ or $x_i \mapsto \hat{y}_i$ respectively will have at least the same value as the original one. □

With our source of hardness in order, we need to show how to encode these parity constraints as gadgets within the problem maxPCSP(K_2, G_3). We will now spell out what conditions our gadgets need to have — this is similar to the approach used within [TSSW00].

Definition 9.19. A *gadget with parity p and performance α* is a graph $G = (V, E)$, where $0, x, y, z \in V$, with the following properties.

- **Soundness**. For every function $f : \{x, y, z\} \to \{0, 1\}$ such that $f(x) + f(y) + f(z) \equiv p \bmod 2$, there exists a cut with α edges such that, for $u \in \{x, y, z\}$, if $f(u) = 0$ then u is on the same side of the cut as 0, and if $f(u) = 1$ then u is on the opposite side of the cut from 0.
- **Completeness**. Every triangle-free subgraph of G has at most α edges.
- **Connectedness**. If H is an α-edge triangle-free subgraph of G, then 0 and u are at distance at most 2 in H, for any $u \in \{x, y, z\}$.
- **Decodability**. If H is an α-edge triangle-free subgraph of G, then define $f(u)$ to be the parity of length of the shortest path from 0 to u in H, for $u \in \{x, y, z\}$. Then $f(x) + f(y) + f(z) \equiv p \bmod 2$.

We will first show how to use such a gadget to reduce to the 3-bit PCP. This is quite standard, and follows the pattern of [TSSW00]. The only complication is showing that we can decode the output of maxPCSP(K_2, G_3) to that of our original problem in a consistent fashion.

Proposition 9.20. Suppose that for $p \in \{0, 1\}$ there exists a gadget $G_p = (V_p, E_p)$ with parity p and performance α_p. Then it is NP-hard to approximate maxPCSP(K_2, G_3) to within $1 - 1/(\alpha_0 + \alpha_1) + \varepsilon$, for all small enough $\varepsilon > 0$.

Proof. We reduce from the problem from Corollary 9.18 to maxPCSP(K_2, G_3). We first describe the reduction, then the soundness and completeness of our reduction, ending with an analysis of the parameters needed to make the reduction work.

Reduction. Suppose we are given an instance of the problem from Corollary 9.18, i.e. we are given a balanced system of modulo 2 linear equations in three variables $\mathcal{E}$, and we are asked to determine whether it has balanced value at least $1 - \delta$, or unbalanced value not even $\frac{1}{2} + \delta$. Suppose that this instance has n variables, and m equations *of each type* (hence $2m$ equations in total). We reduce to a graph G constructed as follows. Introduce in G a special vertex v_0, as well as a vertex v_x for every variable x in $\mathcal{E}$. For every equation $x + y + z \equiv p \bmod 2$, introduce a disjoint copy of the gadget G_p, but identifying the vertices $0, x, y, z$ in G_p with v_0, v_x, v_y, v_z respectively. Note that this identification acts on vertices and not on edges — hence the output of our reduction may have parallel edges.

Soundness. Suppose the original instance has a balanced assignment with balanced value $1 - \delta$, i.e. it satisfies a $(1 - \delta)$-fraction of the constraints of each type. Suppose this assignment sets the variable x to $c(x)$. We will create a cut of the graph G with value at least $(1-\delta)(\alpha_0+\alpha_1)$, i.e. at least $(1-\delta)m(\alpha_0+\alpha_1)$ edges. Our cut will be created as follows: for any vertex v_x which corresponds to a variable x in $\mathcal{E}$, we place v_x according to $c(x)$. We place v_0 on the same side as those vertices v_x where $c(x) = 0$ (or on the opposite side as those with $c(x) = 1$, in case only such vertices exist). By the soundness of our gadgets, for each of the $(1 - \delta)m$ copies of G_p that correspond to satisfied equations, there exists a cut that extends the choices we have made so far. Furthermore, this cut has α_p edges. It follows that the graph we create has a cut with $(1 - \delta)m(\alpha_0 + \alpha_1)$ edges.

Completeness. Suppose that our graph G has a triangle free subgraph H with value at least $(\alpha_0 + \alpha_1 - 1 + 2\delta)$, i.e. at least $m(\alpha_0 + \alpha_1 - 1 + 2\delta)$ edges. We say that an equation $x + y + z \equiv p \bmod 2$ from $\mathcal{E}$ is *good* if at least α_p of the edges of its copy of G_p are included within H.

First, we claim that we can decode the triangle-free subgraph H into an assignment for $\mathcal{E}$ that satisfies all the good equations. Indeed, by the connectedness of our gadgets we see that the distance from v_0 to v_x for any vertex v_x that corresponds to a variable x of $\mathcal{E}$ is at most 2. This implies that the length of the shortest path from v_0 to v_x in H has the same parity as the length of the shortest path from v_0 to v_x *in the edges of H from any copy of G_p that comes from a good equation.* This is because both of these shortest paths are of length at most 2 and at least 1; if they had different parities, one would be of length 1, the other of

length 2. This would imply the existence of a triangle in H, which is impossible. Hence, the decodability of our gadgets imply that setting variable x to the parity of the length of the shortest path from v_0 to v_x satisfies all the good constraints.

It remains to argue that at least $m + 2m\delta = 2m(\frac{1}{2} + \delta)$ of the equations are good. Observe that every good equation of parity p contributes exactly α_p of the edges in H, whereas every bad equation of parity p contributes at most $\alpha_p - 1$ of the equations in p. Thus, if there are x bad equations, we see that the number of edges in H is at most $m(\alpha_0 + \alpha_1) - x$. Hence

$$m(\alpha_0 + \alpha_1) - x \geq m(\alpha_0 + \alpha_1 - 1 + 2\delta),$$

or equivalently $x \leq m - 2m\delta$. Since there are $2m - x$ good equations, completeness follows.

Choice of parameters. We have thus far proved that, given a graph with a cut of value $(1 - \delta)(\alpha_0 + \alpha_1)$, it is NP-hard to find a triangle-free subgraph of value $(\alpha_0 + \alpha_1 - 1 + 2\delta)$. Hence, it is NP-hard to approximate maxPCSP(K_2, G_3) to within an approximation ratio of

$$\frac{(\alpha_0 + \alpha_1 - 1 + 2\delta)}{(1 - \delta)(\alpha_0 + \alpha_1)} = 1 - \frac{1}{\alpha_0 + \alpha_1} + O(\delta).$$

By setting δ small enough, the desired inapproximability result follows. □

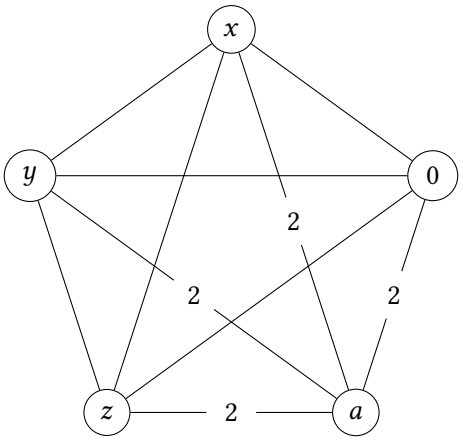
It now remains to provide the required gadgets. The first gadget is identical to one of [BGS98], which they call PC-CUT (although the analysis is slightly more complex but very similar). The second is a slight generalisation.

Lemma 9.21 (PC-CUT [BGS98]). There exists a gadget with parity 1 and performance 9.

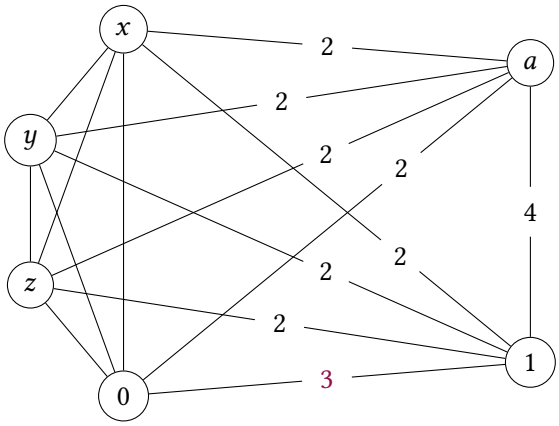
Proof. Consider the gadget from Figure 9.2a, where edges labelled by 2 are in fact parallel edges. If we assign vertices weights, namely $w(a) = 2, w(0) = w(x) = w(y) = w(z) = 1$, then there are $w(i)w(j)$ edges between vertex i and vertex j. Hence the weight of any cut into A and B is $(\sum_{a \in A} w(a))\,(\sum_{b \in B} w(b))$. Hence we see that the optimum cut splits the weight as evenly as possible, i.e. into 3 and 3, giving rise to a cut of size 9. These cuts put a with one of $0, x, y, z$ on one side, and the other three vertices on the other side. As to the non-bipartite triangle-free subgraphs, the maximal ones are copies of C_5 but with parallel edges. Any such maximal non-bipartite triangle-free subgraph has $1 + 1 + 1 + 2 + 2 = 7 < 9$ edges. Hence the only 9-edge triangle-free subgraphs are precisely the bipartite ones mentioned above. It is easy to check that the conditions from Definition 9.19 are satisfied on these subgraphs. □

Lemma 9.22. There exists a gadget with parity 0 and performance 17.

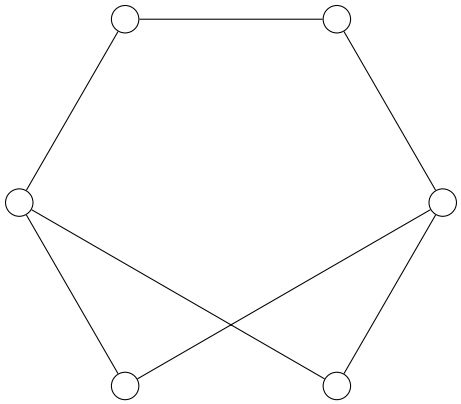
Proof. Consider the gadget from Figure 9.2b, where edges labelled by t are actually t parallel edges. Proceeding similarly to Lemma 9.21, we assign vertices weights: $w(a) = w(1) = 2, w(0) = w(x) = w(y) = w(z) = 1$. We also define a penalty δ given by $\delta(0, 1) = \delta(1, 0) = 1$, and $\delta(\star, \star) = 0$ otherwise. Now there are $w(i)w(j) + \delta(i, j)$ edges between i and j. We see again that the maximum cut splits the total weight of 8 as evenly as possible, putting 0 and



(a) PC-CUT



(b) Modified PC-CUT



(c) Maximal triangle-free non-bipartite 6 vertex graph

Figure 9.2.: Two gadgets based on PC-CUT [BGS98], and a triangle-free non-bipartite subgraph

1 on opposite sides due to the penalty. Thus we want a total weight of 2 on the same side as 1. Hence, in the optimal cuts, either a is on the same side as 1 and x, y, z are on the same side as 0, or a and one of x, y, z are on the same side as 0, and the other two among x, y, z are on the same side as 1. As for the non-bipartite triangle-free subgraphs, the maximal non-bipartite graph on 6 vertices is pictured within Figure 9.2c. Any maximal non-bipartite triangle-free subgraph of our gadget must be isomorphic (up to ignoring parallel edges) to this graph.

We claim that every non-bipartite triangle-free subgraph of our gadget has at most 16 edges. First, remove one edge between 0 and 1 — we claim that any non-bipartite triangle-free subgraph of this graph has at most 15 edges. Suppose this is not the case. Note that the graph from Figure 9.2c has 7 edges; hence, in any non-bipartite triangle-free subgraph of our gadget (ignoring one edge between 0 and 1) with > 15 edges, there must exist a pair of vertices with $> 15/7 > 2$ edges between them. The only pair of vertices which satisfy this are a and 1 (since, again, we removed an edge between 0 and 1). Hence, a and 1 are adjacent in the assumed subgraph. Note that the maximum degree of any 2-vertex subgraph of Figure 9.2c is 3 (this is achieved by taking one vertex with degree 3 and one neighbour). Thus, placing a and 1 on this maximum degree subgraph, we see that the maximum number of edges is $4 + 3 \times 2 + 3 = 13$, which is less than 15. Hence our claim follows.

Thus, the only triangle-free subgraphs with 17 edges are cuts with the structure we mentioned above. It is easy to check that these subgraphs follow the properties required in Definition 9.19. Our result thus follows. □

Proof of Proposition 9.3. Apply Proposition 9.20 using the gadgets from Lemmas 9.21 and 9.22. Since the performance is 9 and 17 respectively, we find that it is NP-hard to $(25/26 + \varepsilon)$-approximate maxPCSP(K_2, G_3). □

9.4. Hardness for maxPCSP(C_{2k+1}, K_ℓ)

In this section, we prove Proposition 9.4, thus completing the proof of Theorem 1.5. We restate the proposition for the reader's convenience.

Proposition 9.4. maxPCSP(C_{2k+1}, K_ℓ) is NP-hard for $k \geq 1$ and $\ell \geq 3$, assuming the Unique Games Conjecture of Khot.

This proof will be very similar to the proof of hardness for *almost 3-colouring* from [DMR09]. Recall that in the almost 3-colouring problem, we are given an n-vertex graph G and we are promised that it has a $(1 - \varepsilon)n$-vertex induced subgraph that is 3-colourable. The task is then to find an independent set with εn vertices. Recall that [DMR09] prove that this problem is hard under UGC, by exploiting a bound on a *noise operator* derived from the random walk on K_3. Our proof will likewise exploit this same bound, but applied to the noise operator derived from the random walk on C_{2k+1}. We will also prove that UGC implies that it is NP-hard to solve *multilayered unique games*, using a construction from [DRS05; BWŽ21] (this has already been proved in the preliminaries).

We will, however, structure our proof in a way that we believe is easier to work with. We will first prove the following purely combinatorial fact.

Proposition 9.23. Fix $k, \ell \in \mathbb{N}$. There exists $s \in \mathbb{N}, \delta > 0$ so that the following holds. For any $N \in \mathbb{N}$, there exists a way to partition the set of functions $[2k+1]^N \to [\ell]$ into ℓ parts $P_1, \dots, P_\ell$ so that the following holds. Consider some part $P = P_i$, consisting of functions $[2k+1]^N \to [\ell]$. There exists an assignment I, which assigns to each $f \in P$ a subset of $[N]$ of size at most s, so that the following two facts hold.

(i) For a permutation $\pi : [N] \to [N]$, we have $f^\pi \in P$, and furthermore $I(f^\pi) = \pi(I(f))$.

(ii) Consider any $f, g \in P$ such that, for a δ-fraction of $(x_1, \dots, x_N), (y_1 \dots, y_N) \in [2k+1]^N$ with $x_i \equiv y_i \pm 1 \bmod 2k+1$, we have $f(x_1, \dots, x_N) \neq g(y_1, \dots, y_N)$. Then $I(f) \cap I(g) \neq \emptyset$.

We will prove this fact in Section 9.4.1. We will then prove that this implies the required conclusion in Section 9.4.2

9.4.1. Proof of Proposition 9.23

Let us begin with the noise operator bound. We start with some simple definitions, then state the bound.

Definition 9.24. Define the matrix $M_n \in \mathbb{R}^{n \times n}$ by setting the element at (i, j) to $\frac{1}{2}$ if $i \equiv j \pm 1 \bmod n$, and to 0 otherwise. In other words M_n is the transition matrix of the random walk on the graph C_n, and we can see M_n as a Markov operator.

Lemma 9.25. The Markov chain M_n is *symmetric* i.e. it is *reversible* with respect to the uniform distribution

Proof. This is immediate from the fact that the uniform distribution is a stationary distribution for any random walk on a regular graph. □

For what follows, if M is a Markov operator then note that all eigenvalues are between -1 and 1. We define the *spectral radius* $\rho(M)$ as the second largest absolute value of any eigenvalue of M. (This is somewhat nonstandard, but it follows [DMR09]. The reason for this notation is that M is always guaranteed to have one eigenvalue equal to 1, and we are interested in what can be said about the rest.)

Lemma 9.26. The eigenvalues of M_n are $\cos(2k\pi/n)$ for $k = 0, \dots, n-1$. In particular, if n is odd then k has one eigenvalue equal to 1 and all other eigenvalues have subunitary absolute value i.e. $\rho(M_n) < 1$ for odd n.

Proof. Observe that M_n is a *circulant matrix*, i.e. every row is the cyclic shift of the row above it. If we let ω be an n-th primitive root of unity, then the eigenvalues of the circulant matrix whose first row is $a_0, \dots, a_{n-1}$ are given by

$$\lambda_k = \sum_{i=0}^{n-1} a_i \omega^{-ki},$$

for $k = 0, \dots, n-1$. In particular, in our case $a_1 = a_{n-1} = \frac{1}{2}$ and all other elements on the first row are 0, so the eigenvalues are given by

$$\lambda_k = \frac{\omega^k + \omega^{(n-1)k}}{2} = \frac{\omega^k + \omega^{-k}}{2} = \cos(2k\pi/n),$$

for $k = 0, \dots, n-1$. □

The statement of the noise inequality from [DMR09] requires some preliminary Fourier-analytic notions. Rather than state all of these definitions in detail, we will simply list the properties of these notions that we will use, and refer the interested reader to [DMR09].

Definition 9.27. Fix $d, N \in \mathbb{N}$ and let $f, g : [d]^N \to [0,1]$ be arbitrary functions. We define $\mathbb{E}[f]$ to be the expected value of $f(x)$ if x is uniformly distributed in $[d]^N$. Furthermore, let M be some Markov chain on $[d]^N$. Then we define $\langle f, Mg \rangle$ as the expectation of $f(x)g(y)$ if x is distributed uniformly, then y is distributed according to 1 step of the Markov chain M starting at x.

Definition 9.28. For fixed $\mu \in (0,1)$, let $t(\mu)$ be a real number so that, if x is a standard normal variable, then the probability that $x \le t(\mu)$ is precisely μ. Suppose $\rho \in \mathbb{R}$, and take (x, y) from a bivariate normal distribution, with the expectation of x, y being 0, the variance of x and y being 1, and the covariance of x and y being ρ. Then, for $\mu, \nu \in (0,1)$, let $\Gamma_\rho(\mu, \nu)$ denote the probability that $x \le t(\mu)$ and $y \ge t(1-\nu)$. Crucially Γ_ρ is non-decreasing in both μ and ν.

Definition 9.29. For any function $f : [d]^N \to [0,1]$, the *low-degree influence of* $i \in [N]$, denoted by $\mathrm{Inf}_i^{\le t}(f)$, is a non-negative real number. The key properties we will use are that $\sum_i \mathrm{Inf}_i^{\le t}(f) \le t$, and that $\mathrm{Inf}_i^{\le t}(f)$ is defined in a way compatible with permuting coordinates (i.e. for a permutation $\pi : [N] \to [N]$, we have $\mathrm{Inf}_{\pi(i)}^{\le t}(f^\pi) = \mathrm{Inf}_i^{\le t}(f)$).

Definition 9.30. If M is a Markov chain on $[d]$, for any N we define $M^{\otimes N}$ to be a Markov chain on $[d]^N$, which behaves as N independent copies of M, one on each coordinate of $[d]^N$.

Theorem 9.31 ([DMR09, Theorem 3.1]). Fix d and a symmetric Markov operator M on $[d]$ with $\rho(M) < 1$. For every $\varepsilon > 0$ there exists $\delta > 0, t \in \mathbb{N}$ so that the following holds. For any $f, g : [d]^N \to [0,1]$ with $\mathbb{E}[f] = \mu, \mathbb{E}[g] = \nu$, with

$$\mathrm{Inf}_i^{\leq t}(f) < \delta,$$
$$\mathrm{Inf}_i^{\leq t}(g) < \delta$$

for all $i \in [N]$, we have

$$\langle f, T^{\otimes N} g \rangle \geq \Gamma_\rho(\mu, \nu) - \varepsilon.$$

We will be using this theorem in the contrapositive, quite similarly to [DMR09, Corollary 4.12].

Corollary 9.32. Fix d and a symmetric Markov operator M on $[d]$ with $\rho(M) < 1$. For every ε there exists $\delta > 0, t \in \mathbb{N}$ so that the following holds. If $f, g : [d]^N \to [0,1]$ are functions with $\mathbb{E}[f] > \varepsilon, \mathbb{E}[g] > \varepsilon$ and $\langle f, T^{\otimes N} g \rangle \leq \delta$, then there exists $i \in [N]$ so that $\mathrm{Inf}_i^{\leq t}(f) \geq \delta$ and $\mathrm{Inf}_i^{\leq t}(f) \geq \delta$.

Proof. Take $\varepsilon' = \min(\varepsilon, \Gamma_\rho(\varepsilon, \varepsilon)/2)$ and apply Theorem 9.31 with ε', d and M; suppose we get δ', t from this application of Theorem 9.31. Then set $\delta < \min(\delta', \Gamma_\rho(\varepsilon, \varepsilon)/2)$.

Suppose that $\mathbb{E}[f] = \mu, \mathbb{E}[g] = \nu$ with $\mu, \nu > \varepsilon$ and $\langle f, T^{\otimes N} g \rangle \leq \delta$. Then observe

$$\langle f, T^{\otimes N} g \rangle \leq \delta \leq \frac{1}{2}\Gamma_\rho(\varepsilon, \varepsilon) \leq \Gamma_\rho(\varepsilon, \varepsilon) - \varepsilon \leq \Gamma_\rho(\mu, \nu) - \varepsilon,$$

where the last inequality follows from the fact that Γ_ρ is non-decreasing. Hence by applying Theorem 9.31, we have that for some i in particular

$$\mathrm{Inf}_i^{\leq t}(f), \mathrm{Inf}_i^{\leq t}(g) \geq \delta' > \delta,$$

as required. □

With this noise inequality in place, we can now prove Proposition 9.23.

Proof of Proposition 9.23. Take $d = 2k+1$, $M = M_{2k+1}$, and apply Corollary 9.32, setting $\varepsilon = 1/\ell$, and receiving δ, t. We take $s = t/\delta$. Split the functions $f : [2k+1]^N \to [\ell]$ into ℓ parts $P_1, \ldots, P_\ell$ according to the most frequent value, taking the minimum most frequent value in case of ties. In other words, P_i contains only those functions f where i appears strictly more frequently than all values $j < i$, and at least as frequently as all values $j \geq i$. Now, consider $f \in P_c$.

First, we explain how we construct the mapping I. This is quite simple — suppose we have some function $f : [2k+1]^N \to [\ell]$, whose most frequent value is c. Define a function $\overline{f} : [2k+1]^N \to [0,1]$ as follows: if $f(x_1, \ldots, x_N) = c$ then $\overline{f}(x_1, \ldots, x_N) = 1$, otherwise $\overline{f}(x_1, \ldots, x_N) = 0$. Then set

$$I(f) = \{i \in [N] \mid \mathrm{Inf}_i^{\leq t}(\overline{f}) \geq \delta\}.$$

Since, as observed earlier, $\sum_i \mathrm{Inf}_i^{\leq t}(\overline{f}) \leq t$ and $\mathrm{Inf}_i^{\leq t}(\overline{f}) \geq 0$, we see that $|I(f)| \leq t/\delta = s$, as required.

Next, observe that $f^\pi \in P_c$, and $I(f^\pi) = \pi(I(f))$. This follows immediately from the fact that (i) f^π and f are equal up to a permutation of their domains, and (ii) Inf_i is defined compatibly with permuting coordinates.

Next, we show that the intersection property is satisfied. Consider any $f, g : [2k+1]^N \to [\ell]$, where the most frequent value is equal, and where $f(x_1, \ldots, x_N) \neq g(y_1, \ldots, y_N)$ for a $(1-\delta)$-fraction of all pairs $(x_1, \ldots, x_N), (y_1, \ldots, y_N) \in [2k+1]^N$ with $x_i \equiv y_i \pm 1 \bmod 2k+1$ for $i \in [N]$.

First observe that $\mathbb{E}[\overline{f}], \mathbb{E}[\overline{g}] \geq 1/\ell = \varepsilon$. This is because, by definition, $\overline{f}$ and $\overline{g}$ are 1 on inputs for which f, g which output their most frequent value. This value must appear a $(1/\ell)$-fraction of the time.

Next, we claim that $\langle \overline{f}, M^{\otimes N} \overline{g} \rangle \leq \delta$. Indeed, this value can be interpreted as the expected value of $\overline{f}(x_1, \ldots, x_N)\overline{g}(y_1, \ldots, y_N)$, where $x_1, \ldots, x_N \in [2k+1]$ are selected uniformly at random, and y_i is selected at random from among $x_i \pm 1 \bmod 2k+1$. This is 1 only when $f(x_1, \ldots, x_N) = g(y_1, \ldots, y_N) = c$, and zero otherwise — hence in particular $1 - \langle \overline{f}, M^{\otimes N} \overline{g} \rangle$ is an upper bound for the probability that $f(x_1, \ldots, x_N) \neq g(y_1, \ldots, y_N)$. This probability is equal to the probability that $f(x_1, \ldots, x_N) \neq g(y_1, \ldots, y_N)$ when $(x_1, \ldots, x_N), (y_1, \ldots, y_N) \in [2k+1]^N$ with $x_i \equiv y_i \pm 1 \bmod 2k+1$ are selected uniformly at random (since each $(x_1, \ldots, x_N)$ corresponds to the same number of $(y_1, \ldots, y_N)$ and vice versa) — which is at least $1 - \delta$ by assumption. Hence $\langle \overline{f}, M^{\otimes N} \overline{g} \rangle \leq \delta$.

Finally, apply Corollary 9.32. This implies that there exists some $i \in [N]$ such that $\mathrm{Inf}_i^{\leq t}(\overline{f}) \geq \delta, \mathrm{Inf}_i^{\leq t}(\overline{g}) \geq \delta$. Thus $i \in I(f) \cap I(g)$, as required. □

9.4.2. Reduction

With Proposition 9.23 in place, we can now give the reduction from the multi-layered UGC to maxPCSP(C_{2k+1}, K_ℓ). Let us define the exact problem we reduce from — this definition has appeared previously in the preliminaries, but we restate it for the reader's convenience.

Definition 9.33. An $(\ell+1)$*-player one-round unique game* is a computational problem consisting of a set of variables X, partitioned into *layers* $X_0, \ldots, X_\ell$, a domain $[N]$, and constraints. Each constraint consists of a sequence of variables $x_0, \ldots, x_\ell$, with $x_i \in X_i$, and a family of permutations $\boldsymbol{\pi} = (\pi_{ij})_{0 \leq i < j \leq \ell}$, with $\pi_{ik} = \pi_{jk} \circ \pi_{ij}$ for $0 \leq i < j < k \leq \ell$.
An *assignment* for this problem is a function $c : X \to [N]$. A constraint given by $x_0, \ldots, x_\ell$ and $\boldsymbol{\pi}$ is *strongly satisfied* by c if $\pi_{ij}(c(x_i)) = c(x_j)$ for all $0 \leq i < j \leq \ell$. It is *weakly satisfied* if $\pi_{ij}(c(x_i)) = c(x_j)$ for *at least one* pair (i, j) with $0 \leq i < j \leq \ell$. We say that the *strong/weak value of* c is the proportion of constraints strongly/weakly satisfied by c. The *strong/weak value* of the game is the maximum strong/weak value of any assignment.

It is easy to see that the weak value of any $(\ell+1)$-player one-round game is at least as large as the strong value. The following theorem (seen before in the preliminaries) is proved simply by applying the reduction of [DRS05; BWŽ21] but starting from the UGC.

Corollary 2.22. Assuming UGC, for any $\ell \in \mathbb{N}$, $\ell > 0$, $\varepsilon \in (0,1)$, there exists some N such that the following problem is NP-hard: Given an $(\ell+1)$-player 1-round unique game with domain $[N]$, decide if the strong value is at least $1-\varepsilon$, or the weak value is not even ε.

With this in hand, we will now describe how we will encode a unique game into an instance of maxPCSP(C_{2k+1}, K_ℓ). Within the definitions that follow, fix k, ℓ — we will define *long-code blocks* which encode variables in a unique game, and *constraint gadgets* that encode the constraints. Since we reduce to maxPCSP(C_{2k+1}, K_ℓ) i.e. a graph problem, all of these objects will be graphs.

Definition 9.34. For fixed N, k, a *long-code block* $\boldsymbol{x}$, is a set of $(2k+1)^N$ independent vertices, one for every vector $(x_1, \dots, x_N) \in [2k+1]^N$.
If we have several long-code blocks $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$, together with an ℓ-colouring f, we let $f_1, \dots, f_n$ denote this colouring restricted to $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ respectively. In particular, note that f_i is a function from $[2k+1]^N$ to $[\ell]$.

Definition 9.35. For fixed N, k, ℓ, a sequence of long-code blocks $\boldsymbol{x}_0, \dots, \boldsymbol{x}_\ell$, and a family of permutations $\boldsymbol{\pi}$ as in Definition 9.33, we define a constraint gadget $E_{\boldsymbol{\pi}}(\boldsymbol{x}_0, \dots, \boldsymbol{x}_\ell)$ as the following set of edges. For every $0 \le i < j \le \ell$, let $\boldsymbol{x} = \boldsymbol{x}_i, \boldsymbol{y} = \boldsymbol{x}_j, \pi = \pi_{ij}$. Then, for every $(x_1, \dots, x_N), (y_1, \dots, y_N) \in [2k+1]^N$ with $x_i \equiv y_i \pm 1 \bmod 2k+1$ for all $i \in [N]$, include the edge $(x_{\pi(1)}, \dots, x_{\pi(N)}) - (y_1, \dots, y_N)$.

We now explain how we will decode this long-code gadget, using Proposition 9.23.

Lemma 9.36. Fix k, ℓ. There exists ε so that the following hold for any N. Consider any ℓ-colouring f of a long code block. There exists a random variable $i(f)$ defined on $[N]$ such that the following holds.
Suppose that $f_0, \dots, f_\ell$ are ℓ-colourings of the long-code blocks $\boldsymbol{x}_0, \dots, \boldsymbol{x}_\ell$. Suppose these blocks are coloured so that a $(1-\varepsilon)$-fraction of the edges in $E_{\boldsymbol{\pi}}(\boldsymbol{x}_0, \dots, \boldsymbol{x}_\ell)$ are coloured properly. Then there exists $0 \le i < j \le \ell$ so that $\pi_{ij}(i(f_i)) = i(f_j)$ with probability at least ε.

Proof. Apply Proposition 9.23 to get I, s, δ. Let $i(f)$ be uniformly distributed among $I(f)$. We claim that if we take

$$\varepsilon = \min\left(\frac{1}{s^2}, \frac{\delta}{(\ell+1)^2}\right),$$

then $i(f)$ satisfies the required conditions.

First, note that in $E_{\boldsymbol{\pi}}(\boldsymbol{x}_0, \dots, \boldsymbol{x}_\ell)$ there are the same number of edges between any two long-code blocks. Hence, since a $(1-\varepsilon)$-fraction of the total edges were correctly coloured, it follows that a $(1-(\ell+1)\varepsilon)$-fraction of the edges between any particular pair of blocks is also coloured correctly. In other words, at least a $(1-\delta)$-fraction of the edges between any two blocks is correctly coloured.

Observe that as there are $\ell+1$ colourings $f_0, \dots, f_\ell$, but we have split the set of functions from $[2k+1]^N$ to $[\ell]$ into ℓ parts in Proposition 9.23, there must exist $0 \le i < j \le \ell$ such

that f_i, f_j are in the same part. We will show that $\pi_{ij}(i(f_i)) = i(f_j)$ with probability at least ε — to this end let $\boldsymbol{x} = \boldsymbol{x}_i, \boldsymbol{y} = \boldsymbol{x}_j, f = f_i, g = f_j, \pi = \pi_{ij}$.

Recall that a $(1-\delta)$-fraction of the edges between $\boldsymbol{x}, \boldsymbol{y}$ are correctly coloured. In other words, a $(1-\delta)$-fraction of vectors $(x_1, \dots, x_N), (y_1, \dots, y_N) \in [2k+1]^N$ with $x_i \equiv y_i \pm 1 \bmod 2k+1$ satisfy

$$f(x_{\pi(1)}, \dots, x_{\pi(N)}) \neq g(y_1, \dots, y_N).$$

Equivalently, the same fraction of vectors satisfy

$$f^\pi(x_1, \dots, x_N) \neq g(y_1, \dots, y_N).$$

By Proposition 9.23, we see that f^π is in the same part as f (and hence as g), and therefore that $I(f^\pi) \cap I(g) \neq \varnothing$. Since $I(f^\pi) = \pi(I(f))$, this means that $\pi(I(f)) \cap I(g) \neq \varnothing$. Since $|I(f)|, |I(g)| \leq s$, by selecting $i(f), i(g)$ uniformly at random from $I(f), I(g)$, we have that $\pi(i(f)) = \pi(i(g))$ with probability at least $1/s^2 \geq \varepsilon$, as required. □

Finally, we simply put together all of the parts we have defined so far to complete the reduction.

Proof of Proposition 9.4. We give a reduction from the problem defined in Definition 9.33 to maxPCSP(C_{2k+1}, K_ℓ). Take ε from Lemma 9.36, and then take N from Corollary 2.22. All our long-code blocks and constraint gadgets will be defined using N, k.

Reduction. Suppose we are given an $(\ell+1)$-player one-round unique game, with variables $X = X_0 \cup \ldots \cup X_\ell$ and domain $[N]$. For every variable $x \in X$, introduce a long-code block $\boldsymbol{x}$. For every constraint, given say by variables $x_0, \dots, x_\ell$ and permutation family $\boldsymbol{\pi}$, introduce the edges of the constraint gadget $E_{\boldsymbol{\pi}}(\boldsymbol{x}_0, \dots, \boldsymbol{x}_\ell)$. If an edge is included by multiple different gadgets, include it multiple times.

Completeness. Suppose that the original unique game had strong value at least $1 - \delta$. Suppose that the assignment with that value is $c : X \to [N]$. We claim that there exists an ℓ-colouring of our graph that satisfies at least a $(1 - \delta)$-fraction of the edges. Indeed, suppose $\phi : [2k+1] \to [\ell]$ is an ℓ-colouring of C_{2k+1}. Then, in every long-code block $\boldsymbol{x}$, corresponding to variable x in the original unique game, colour vertex $(x_1, \dots, x_N)$ by $\phi(x_{c(x)})$. We claim that this satisfies all the edges introduced by every constraint strongly satisfied by c — since each constraint introduces the same number of edges, this implies that a $(1 - \delta)$-fraction of all edges are satisfied.

To see why this is the case, consider some edge introduced by a strongly satisfied constraint. This edge is between two long-code blocks $\boldsymbol{x}$ and $\boldsymbol{y}$, corresponding to variables x, y, where the constraint is that $\pi(c(x)) = c(y)$ for some permutation $\pi : [N] \to [N]$. The edge is between $(x_{\pi(1)}, \dots, x_{\pi(N)})$ and $(y_1, \dots, y_N)$, where $x_i \equiv y_i \pm 1 \bmod 2k+1$ for $i \in [N]$. By construction these vertices are coloured by $x_{\pi(c(x))}$ and $y_{c(y)}$ i.e. $x_{c(y)}$ and $y_{c(y)}$. We know that $x_{c(y)} \equiv y_{c(y)} \pm 1 \bmod 2k+1$, hence $\phi(x_{c(y)}) \neq \phi(y_{c(y)})$ as required (since ϕ is a colouring of C_{2k+1}).

Soundness. Suppose that c is some ℓ-colouring of our graph that satisfies at least a $(1-\delta)$-fraction of the edges. By applying Markov's inequality, we find that in a $(1-\sqrt{\delta})$-fraction of the constraints, a $(1-\sqrt{\delta})$-fraction of the edges are coloured correctly. Assume that $\sqrt{\delta} \leq \varepsilon$. Consider some variable $x \in X$, and suppose that its long-code block $\boldsymbol{x}$ is coloured by the function $f : [2k+1]^N \to [\ell]$. Then assign x the value $i(f)$ — we claim that this assignment weakly satisfies at least a $(1-\sqrt{\delta})\varepsilon$-fraction of the constraints in expectation. Indeed, in all the constraints where a $1-\sqrt{\delta} \geq (1-\varepsilon)$-fraction of edges are properly coloured, by the properties of Lemma 9.36, the constraint is weakly satisfied with probability at least $(1-\sqrt{\delta})\varepsilon$.

Choice of parameters. We are given an $(\ell+1)$-player two-round unique game, and we must decide whether it has strong value $1-\delta$ or weak value less than δ. By the reduction outlined above, we can decide whether it has strong value $1-\delta$ or weak value not even $(1-\sqrt{\delta})\varepsilon$ by reducing to maxPCSP(C_{2k+1}, K_ℓ). Hence, take δ small enough so that $(1-\sqrt{\delta})\varepsilon > \delta$. This completes the proof. □

9.5. Algorithm for maxPCSP(K_2, G_3)

In this section, we provide an algorithm for maxPCSP(K_2, G_3) i.e. given a graph G that admits a large cut, find a large triangle-free subgraph. In particular, our algorithm achieves an approximation ratio of about 0.8823 — greater than the approximation ratio of about 0.878 from the Goemans-Williamson algorithm (which, of course, solves a harder problem). Additionally, when given a graph that admits an almost perfect cut i.e. one containing a $(1-\varepsilon)$-fraction of the edges, our algorithm produces a triangle-free subgraph with a $(1-O(\varepsilon))$-fraction of the edges. This is significantly better than the $1-O(\sqrt{\varepsilon})$ achieved by the Goemans-Williamson algorithm. For the reader's convenience, we restate the result we prove in this section.

Theorem 9.5. maxPCSP(K_2, G_3) can be 0.8823-approximated in polynomial time. Furthermore, on a graph with a cut with value $1-\delta$ edges, our algorithm outputs a triangle-free graph with value $1-O(\delta)$.

In order to prove this, we will need the following technical lemma. Let

$$\theta_{GW} = \arg\min_{0 \leq \theta \leq \pi} \frac{2}{\pi} \frac{\theta}{1-\cos\theta} \approx 0.7\pi.$$

Furthermore, observe that the function

$$\theta \mapsto \frac{2}{\pi} \frac{\theta}{1-\cos\theta} \tag{9.8}$$

decreases from 0 to θ_{GW} and then increases to π.

Lemma 9.37. For some $\alpha \geq 0.88232$, there exist $P, Q \in [0,1]$ with $P + Q = 1$ and $\tau \in [2\pi/3, \theta_{GW}]$ such that the following holds

$$P\frac{\phi}{\pi} + Q \geq \alpha\frac{1-\cos\phi}{2} \qquad \tau \leq \phi \leq \pi \tag{9.9}$$

$$P\frac{\phi}{\pi} + Q\frac{\phi}{\pi} \geq \alpha\frac{1-\cos\phi}{2} \qquad \pi - \tau/2 \leq \phi \leq \tau \tag{9.10}$$

$$P\frac{\phi}{\pi} \geq \alpha\frac{1-\cos\phi}{2} \qquad 0 \leq \phi \leq \pi - \tau/2 \tag{9.11}$$

In particular, take $\tau = 2.18746$, $Q = 1 - P$ and $P \approx 0.987535$.

Proof. Observe that (9.10) does not depend on P, Q at all, since $P + Q = 1$, and the bound becomes just

$$\frac{\phi}{\pi} \geq \alpha\frac{1-\cos\phi}{2}.$$

In particular, by the monotonicity of the function from (9.8), we see that this bound becomes most tight when $\phi = \tau$, and hence this bound becomes simply

$$\alpha \leq \frac{2}{\pi}\frac{\tau}{1-\cos\tau}.$$

Again, due to the monotonicity of (9.8), we see that (9.11) is most tight when $\phi = \pi - \tau/2$. Hence it becomes

$$P \geq \frac{\alpha\pi}{2}\left(\frac{1-\cos(\pi - \frac{\tau}{2})}{\pi - \frac{\tau}{2}}\right).$$

Let

$$X(\tau) = \left(\frac{1-\cos(\pi - \frac{\tau}{2})}{\pi - \frac{\tau}{2}}\right).$$

Then we have $P \geq \frac{\alpha\pi}{2}X(\tau)$.

Thus far we have reduced (9.10) and (9.11) to $\alpha \leq \frac{2}{\pi}\frac{\tau}{1-\cos\tau}$ and $P \geq \frac{\alpha\pi}{2}X(\tau)$. What remains is (9.9). Note that it is advantageous to select P as small as possible, since $\phi/\pi \leq 1$. Hence we choose $P = \frac{\alpha\pi}{2}X(\tau)$, $Q = 1 - \frac{\alpha\pi}{2}X(\tau)$. The equation then becomes

$$\frac{\alpha\phi}{2}X(\tau) + 1 - \frac{\alpha\pi}{2}X(\tau) \geq \alpha\frac{1-\cos\phi}{2}.$$

Rearrange to get

$$\alpha \leq \frac{2}{(\pi-\phi)X(\tau) - \cos\phi + 1}.$$

It is sufficient, then, that

$$\alpha \leq \min_{\tau\leq\phi\leq\pi}\frac{2}{(\pi-\phi)X(\tau) - \cos\phi + 1} = 2\left(1 + \max_{\tau\leq\phi\leq\pi}(\pi-\phi)X(\tau) - \cos\phi\right)^{-1}.$$

Thus, we are interested in the maximum of

$$f(\phi) = (\pi-\phi)X(\tau) - \cos\phi,$$

when $\phi \in [\tau, \pi]$. We will in fact compute the maximum for $\phi \in [2\pi/3, \pi]$ — this is an even stronger bound. Compute

$$f'(\phi) = -X(\tau) + \sin\phi.$$

On $[2\pi/3, \pi]$, it is easy to see that $\sin\phi$ decreases — hence, if $f'(\phi)$ is 0 at any point in this domain, the maximum of f is hit there. One can find numerically that $0.75 > X(\tau) > 0.6$ for $2\pi/3 \leq \tau \leq \pi$. Thus, the derivative does become 0, precisely at $\arcsin X(\tau)$ (as, for the numeric bounds computed above, we get that $\arcsin X(\tau) \in [2\pi/3, \pi]$). Hence we find that (9.9) is equivalent to

$$\alpha \leq 2(1 + (\pi - \arcsin X(\tau))X(\tau) - \cos\arcsin X(\tau))^{-1}$$

Thus, the only conditions on α, P, τ are:

$$\begin{aligned}
\alpha &\leq 2(1 + (\pi - \arcsin X(\tau))X(\tau) - \cos\arcsin X(\tau))^{-1} \\
\alpha &\leq \frac{2}{\pi}\frac{\tau}{1-\cos\tau} \\
P &= \frac{\alpha\pi}{2}X(\tau).
\end{aligned}$$

Take $\tau = 2.18746$ and P, α as dictated by these equations to complete the proof. □

Let us also give some notation for angles: for two vectors $\boldsymbol{x}, \boldsymbol{y}$, we denote by $\angle(\boldsymbol{x}, \boldsymbol{y})$ the angle spanned by $\boldsymbol{x}$ and $\boldsymbol{y}$, i.e.

$$\angle(\boldsymbol{x}, \boldsymbol{y}) = \arccos\left(\frac{\boldsymbol{x}\cdot\boldsymbol{y}}{||\boldsymbol{x}||, ||\boldsymbol{y}||}\right).$$

The following was known even to Euclid.

Theorem 9.38 ([Euc26, Book XI, Proposition 21]). For any three nonzero vectors $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{R}^N$ for any $N \in \mathbb{N}$, we have

$$\angle(\boldsymbol{x}, \boldsymbol{y}) + \angle(\boldsymbol{x}, \boldsymbol{z}) + \angle(\boldsymbol{y}, \boldsymbol{z}) \leq 2\pi.$$

We are now ready to prove Theorem 9.5.

Proof of Theorem 9.5. We will first describe our algorithm, then prove that it has the required performance. In the following, let α, P, Q, τ be as in Lemma 9.37. Assume throughout that we are given an input with at least one edge.

Algorithm. We will first describe a randomised algorithm, then derandomise it. The randomised algorithm is as follows.

1. Receive as input the graph $G = (V, E)$.
2. Solve the Goemans-Williamson SDP (9.1), up to additive error ε. Suppose $(\boldsymbol{x}_v)_{v\in V}$ is the configuration of vectors returned by this.

3. With probability P, we run the Goemans-Williamson rounding scheme on this configuration of vectors i.e. select a uniformly random hyperplane and return all edges separated by this hyperplane.

4. With probability Q, we do the following. Run the Goemans-Williamson rounding scheme on edges (u, v) where $\pi - \tau/2 < \angle(\boldsymbol{x}_u, \boldsymbol{x}_v) \leq \tau$ i.e. select a uniformly random hyperplane and take all edges (u, v) with $\pi - \tau/2 < \angle(\boldsymbol{x}_u, \boldsymbol{x}_v) \leq \tau$ separated by this random hyperplane. Return the edges returned by this procedure, as well as *all* edges (u, v) with $\angle(\boldsymbol{x}_u, \boldsymbol{x}_v) > \tau$.

This algorithm can be easily derandomised: rather than running Items 3 and 4 with probability P or Q respectively, simply run both and return the larger of the two. Furthermore, use a derandomised version of the Goemans-Williamson algorithm (e.g. [MR99]) — this returns at least the expected number of edges minus ε, in polynomial time with respect to $\log(1/\varepsilon)$ and $|G|$.

Correctness. We prove that the graph we output must be triangle-free. In Item 3, we output a bipartite subgraph i.e. a cut, which is necessarily triangle-free. In Item 4, we output a set of edges with angle $> \tau$ (long edges), and a set of edges inducing a bipartite graph with angle $> \pi - \tau/2$ (short edges). Observe that no three short edges form a triangle (since the short edges induce a bipartite graph), whereas any three edges out of which at least one is long cannot induce a triangle. This is because the sum of the angles of the three edges is more than $2(\pi - \tau/2) + \tau = 2\pi$, which is impossible by Theorem 9.38.

Approximation ratio. We first show that the randomised algorithm returns a subgraph with an expected value $\alpha(\rho - \varepsilon)$, if G has a cut with value ρ. Next, we show that the derandomisation error can be made small enough by selecting ε small enough.

Consider some edge (u, v), with $\phi = \angle(\boldsymbol{x}_u, \boldsymbol{x}_v)$. Observe that this edge contributes $(1 - \cos\phi)/2$ to the value of (9.1). Hence, since the value given by $(\boldsymbol{x}_v)_{v \in V}$ is $\rho - \varepsilon$, if we show that the algorithm picks the edge (u, v) with probability at least $\alpha(1 - \cos\phi)/2$, then we output a graph with value $\alpha(\rho - \varepsilon)$ by linearity of expectation.

To see why we select (u, v) with probability at least $\alpha(1 - \cos\phi)/2$, consider what our algorithm does, depending on if $\phi \in [0, \pi - \tau/2]$, $\phi \in (\pi - \tau/2, \tau]$ or $\phi \in (\tau, \pi]$. Observe that in these cases the algorithm picks the edge with probability $P\phi/\pi$, $P\phi/\pi + Q\phi/\pi$ or $P\phi/\pi + Q$ respectively. (This is because a random hyperplane separates (u, v) with probability ϕ/π.) By Lemma 9.37, all three of these expressions are at least $\alpha(1 - \cos\phi)/2$, as required.

Now, let us analyse the derandomised algorithm. Observe that in any case the derandomisation loss is an additive ε — hence, we return a subgraph with value $\alpha(\rho - \varepsilon) - \varepsilon \geq \alpha\rho - 2\varepsilon$. Observe that $\alpha \geq 0.88232$, but we have only claimed that our algorithm has an approximation ratio of 0.8823. Hence, since $\rho \geq 1/|E|$ in any case of interest, it is sufficient to set e.g. $\varepsilon \leq 10^{-9}/|E|$.

Almost perfectly satisfiable regime. We now analyse what happens when the input graph has a cut which contains a $(1-\delta)$-fraction of the edges. If $\delta = 0$ then the input graph is bipartite, and we can simply return the entire graph. Hence assume $\delta > 0$. In fact, since $1-\delta$ is a fraction of form $x/|E|$, we must have that $\delta \geq 1/|E|$.

Consider the SDP solution $(\boldsymbol{x}_v)_{v\in V}$ found by our algorithm. It's value is at least

$$1 - \delta - \frac{10^{-9}}{|E|} \geq 1 - 2\delta.$$

Suppose we have a edges (u,v) where $\angle(\boldsymbol{x}_u, \boldsymbol{x}_v) > \tau$, and $b = |E| - a$ edges where this is not the case. Observe that each edge in the second case contributes at most $(1-\cos(\tau))/2 \leq 0.8$ to the value of $(\boldsymbol{x}_v)_{v\in V}$. Hence, we have

$$\begin{aligned} a + b &= |E| \\ a + 0.8b &\geq |E|(1-2\delta). \end{aligned}$$

Subtract 0.2 times the first equation from the second to find that $0.2a \geq |E|(1-2\delta) - 0.8|E|$, i.e. $a \geq |E|(1-10\delta)$. Thus, at least a $(1-10\delta)$-fraction of edge (u,v) have $\angle(\boldsymbol{x}_u, \boldsymbol{x}_v) > \tau$. But note that in the triangle-free subgraph found in Item 4, all such edges are certainly included. In the deterministic version of the algorithm, this implies that we return a triangle-free subgraph with at least a $(1-10\delta)$-fraction of the edges, as required. □

Bibliography

[AFL86] N. Alon, P. Frankl and L. Lovász. 'The Chromatic Number of Kneser Hypergraphs'. In: *Transactions of the American Mathematical Society* 298.1 (1986), pp. 359–370. ISSN: 00029947. URL: http://www.jstor.org/stable/2000624.

[ALMSS98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan and Mario Szegedy. 'Proof Verification and the Hardness of Approximation Problems'. In: *J. ACM* 45.3 (1998), pp. 501–555. DOI: 10.1145/278298.278306.

[AS98] Sanjeev Arora and Shmuel Safra. 'Probabilistic Checking of Proofs: A New Characterization of NP'. In: *J. ACM* 45.1 (1998), pp. 70–122. DOI: 10.1145/273865.273901.

[AZ04] Martin Aigner and Günter M. Ziegler. *Proofs from THE BOOK (3. ed.)* Springer, 2004.

[BBB21] Libor Barto, Diego Battistelli and Kevin M. Berg. 'Symmetric Promise Constraint Satisfaction Problems: Beyond the Boolean Case'. In: *STACS*. Vol. 187. LIPIcs. LZI, 2021, 10:1–10:16. DOI: 10.4230/lipics.stacs.2021.10.

[BBKO21] Libor Barto, Jakub Bulín, Andrei A. Krokhin and Jakub Opršal. 'Algebraic Approach to Promise Constraint Satisfaction'. In: *J. ACM* 68.4 (2021), 28:1–28:66. DOI: 10.1145/3457606.

[BG08] Manuel Bodirsky and Martin Grohe. 'Non-dichotomies in Constraint Satisfaction Complexity'. In: *ICALP*. Vol. 5126. LNCS. Springer, 2008, pp. 184–196. DOI: 10.1007/978-3-540-70583-3_16.

[BG21] Joshua Brakensiek and Venkatesan Guruswami. 'Promise Constraint Satisfaction: Algebraic Structure and a Symmetric Boolean Dichotomy'. In: *SIAM J. Comput.* 50.6 (2021), pp. 1663–1700. DOI: 10.1137/19m128212x.

[BGS98] Mihir Bellare, Oded Goldreich and Madhu Sudan. 'Free Bits, PCPs, and Nonapproximability-Towards Tight Results'. In: *SIAM J. Comput.* 27.3 (1998), pp. 804–915. DOI: 10.1137/s0097539796302531.

[BGWŽ20] Joshua Brakensiek, Venkatesan Guruswami, Marcin Wrochna and Stanislav Živný. 'The Power of the Combined Basic Linear Programming and Affine Relaxation for Promise Constraint Satisfaction Problems'. In: *SIAM J. Comput.* 49.6 (2020), pp. 1232–1248. DOI: 10.1137/20m1312745.

[BHPZ23] Joshua Brakensiek, Neng Huang, Aaron Potechin and Uri Zwick. 'Separating MAX 2-AND, MAX DI-CUT and MAX CUT'. In: *FOCS*. IEEE, 2023, pp. 234–252. DOI: 10.1109/focs57990.2023.00023.

[BJ01] Andrei Bulatov and Peter Jeavons. *Algebraic structures in combinatorial problems*. Tech. rep. 2001.

[BJK05] Andrei A. Bulatov, Peter Jeavons and Andrei A. Krokhin. 'Classifying the Complexity of Constraints Using Finite Algebras'. In: *SIAM J. Comput.* 34.3 (2005), pp. 720–742. DOI: 10.1137/s0097539700376676.

[BK05] Ankur Bhargava and S. Rao Kosaraju. 'Derandomization of Dimensionality Reduction and SDP Based Algorithms'. In: *WADS*. Vol. 3608. LNCS. Springer, 2005, pp. 396–408. DOI: 10.1007/11534273_35.

[BK10] Manuel Bodirsky and Jan Kára. 'The complexity of temporal constraint satisfaction problems'. In: *J. ACM* 57.2 (2010), 9:1–9:41. DOI: 10.1145/1667053.1667058.

[BKO19] Jakub Bulín, Andrei A. Krokhin and Jakub Opršal. 'Algebraic approach to promise constraint satisfaction'. In: *STOC*. ACM, 2019, pp. 602–613.

[Blu94] Avrim Blum. 'New Approximation Algorithms for Graph Coloring'. In: *J. ACM* 41.3 (1994), pp. 470–516. DOI: 10.1145/176584.176586.

[BNOŽ25] Benjamin Bedert, Tamio-Vesa Nakajima, Karolina Okrasa and Stanislav Živný. 'Strong Sparsification for 1-in-3-SAT via Polynomial Freiman-Ruzsa'. In: *FOCS*. Accepted. 2025. arXiv: 2507.17878 [cs.DS].

[BR90] Bonnie Berger and John Rompel. 'A Better Performance Guarantee for Approximate Graph Coloring'. In: *Algorithmica* 5.3 (1990), pp. 459–466. DOI: 10.1007/bf01840398.

[Bra22] Zarathustra Brady. *Notes on CSPs and Polymorphisms*. 2022. arXiv: 2210.07383.

[Bul17] Andrei A. Bulatov. 'A Dichotomy Theorem for Nonuniform CSPs'. In: *FOCS*. IEEE, 2017, pp. 319–330. DOI: 10.1109/focs.2017.37.

[BWŽ21] Alex Brandts, Marcin Wrochna and Stanislav Živný. 'The Complexity of Promise SAT on Non-Boolean Domains'. In: *ACM Trans. Comput. Theory* 13.4 (2021), 26:1–26:20. DOI: 10.1145/3470867.

[Che68] M. Cheng. 'The clipping loss in correlation detectors for arbitrary input signal-to-noise ratios'. In: *IEEE Trans. Inf. Theory* 14.3 (1968), pp. 382–389. DOI: 10.1109/tit.1968.1054159.

[Che69] M. C. Cheng. 'The Orthant Probabilities of Four Gaussian Variates'. In: *Ann. Math. Stat.* 40.1 (1969), pp. 152–161. ISSN: 00034851. DOI: 10.1214/aoms/1177697812.

[Chu36] Alonzo Church. 'An Unsolvable Problem of Elementary Number Theory'. In: *American Journal of Mathematics* 58.2 (1936), pp. 345–363. ISSN: 00029327, 10806377. URL: http://www.jstor.org/stable/2371045 (visited on 02/08/2025).

[CKKNŽ24] Lorenzo Ciardo, Marcin Kozik, Andrei A. Krokhin, Tamio-Vesa Nakajima and Stanislav Živný. '1-in-3 vs. Not-All-Equal: Dichotomy of a broken promise'. In: *LICS*. ACM, 2024, 24:1–24:12. DOI: 10.1145/3661814.3662069.

[CKKNŽ25] Lorenzo Ciardo, Marcin Kozik, Andrei A. Krokhin, Tamio-Vesa Nakajima and Stanislav Živný. '1-in-3 vs. Not-All-Equal: Dichotomy of a Broken Promise'. In: *ACM Trans. Comput. Log.* 26.2 (2025), 10:1–10:22. DOI: 10.1145/3719007.

[CKP13] Panagiotis Cheilaris, Balázs Keszegh and Dömötör Pálvölgyi. 'Unique-Maximum and Conflict-Free Coloring for Hypergraphs and Tree Graphs'. In: *SIAM J. Discret. Math.* 27.4 (2013), pp. 1775–1787. DOI: 10.1137/120880471.

[CLRS22] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein. *Introduction to Algorithms, fourth edition.* MIT Press, 2022. ISBN: 9780262046305.

[Cob65] Alan Cobham. 'The Intrinsic Computational Difficulty of Functions'. In: *Logic, methodology and philosophy of science.* North-Holland Pub. Co., 1965, pp. 24–30.

[Coh04] David A. Cohen. 'Tractable Decision for a Constraint Language Implies Tractable Search'. In: *Constraints An Int. J.* 9.3 (2004), pp. 219–229. DOI: 10.1023/b:cons.0000036045.82829.94.

[Coo71] Stephen A. Cook. 'The complexity of theorem-proving procedures'. In: *STOC*. Shaker Heights, Ohio, USA: ACM, 1971, pp. 151–158. ISBN: 9781450374644. DOI: 10.1145/800157.805047.

[CŽ23a] Lorenzo Ciardo and Stanislav Živný. 'CLAP: A New Algorithm for Promise CSPs'. In: *SIAM J. Comput.* 52.1 (2023), pp. 1–37. DOI: 10.1137/22m1476435.

[CŽ23b] Lorenzo Ciardo and Stanislav Živný. 'Hierarchies of Minion Tests for PCSPs through Tensors'. In: *Soda*. Siam, 2023, pp. 568–580. DOI: 10.1137/1.9781611977554.ch25.

[DGKR05] Irit Dinur, Venkatesan Guruswami, Subhash Khot and Oded Regev. 'A New Multilayered PCP and the Hardness of Hypergraph Vertex Cover'. In: *SIAM J. Comput.* 34.5 (2005), pp. 1129–1146. DOI: 10.1137/s0097539704443057.

[Din07] Irit Dinur. 'The PCP theorem by gap amplification'. In: *J. ACM* 54.3 (2007), p. 12. DOI: 10.1145/1236457.1236459.

[DMR09] Irit Dinur, Elchanan Mossel and Oded Regev. 'Conditional Hardness for Approximate Coloring'. In: *SIAM J. Comput.* 39.3 (2009), pp. 843–873. DOI: 10.1137/07068062x.

[DRS05] Irit Dinur, Oded Regev and Clifford D. Smyth. 'The Hardness of 3-Uniform Hypergraph Coloring'. In: *Comb.* 25.5 (2005), pp. 519–535. DOI: 10.1007/s00493-005-0032-4.

[Edm65] Jack Edmonds. 'Paths, Trees, and Flowers'. In: *Canadian Journal of Mathematics* 17 (1965), pp. 449–467. DOI: 10.4153/cjm-1965-045-4.

[EKR61] P. Erdös, Chao Ko and R. Rado. 'Intersection Theorems for Systems of Finite Sets'. In: *The Quarterly Journal of Mathematics* 12.1 (1961), pp. 313–320. ISSN: 0033-5606. DOI: 10.1093/qmath/12.1.313.

[ELRS03] Guy Even, Zvi Lotker, Dana Ron and Shakhar Smorodinsky. 'Conflict-Free Colorings of Simple Geometric Regions with Applications to Frequency Assignment in Cellular Networks'. In: *SIAM J. Comput.* 33.1 (2003), pp. 94–136. DOI: 10.1137/s0097539702431840.

[Euc26] Euclid. *The Thirteen Books of Euclid's Elements.* Second. Trans. Sir Thomas Little Heath. Cambridge University Press, 1926. ISBN: 9781107480506.

[FJ97] Alan M. Frieze and Mark Jerrum. 'Improved Approximation Algorithms for MAX k-CUT and MAX BISECTION'. In: *Algorithmica* 18.1 (1997), pp. 67–81. DOI: 10.1007/bf02523688.

[FKOS19] Miron Ficak, Marcin Kozik, Miroslav Olšák and Szymon Stankiewicz. 'Dichotomy for Symmetric Boolean PCSPs'. In: *ICALP*. Vol. 132. LIPIcs. LZI, 2019, 57:1–57:12. DOI: 10.4230/lipics.icalp.2019.57.

[FNOTW24] Marek Filakovský, Tamio-Vesa Nakajima, Jakub Opršal, Gianluca Tasinato and Uli Wagner. 'Hardness of Linearly Ordered 4-Colouring of 3-Colourable 3-Uniform Hypergraphs'. In: *STACS*. Vol. 289. LIPIcs. LZI, 2024, 34:1–34:19. DOI: 10.4230/lipics.stacs.2024.34.

[FV93] Tomás Feder and Moshe Y. Vardi. 'Monotone monadic SNP and constraint satisfaction'. In: *STOC*. ACM, 1993, pp. 612–622. DOI: 10.1145/167088.167245.

[GJ76] M. R. Garey and David S. Johnson. 'The Complexity of Near-Optimal Graph Coloring'. In: *J. ACM* 23.1 (1976), pp. 43–49. DOI: 10.1145/321921.321926.

[GS20] Venkatesan Guruswami and Sai Sandeep. 'd-To-1 Hardness of Coloring 3-Colorable Graphs with O(1) Colors'. In: *ICALP*. Vol. 168. LIPIcs. LZI, 2020, 62:1–62:12. DOI: 10.4230/lipics.icalp.2020.62.

[GT05] Venkatesan Guruswami and Luca Trevisan. 'The Complexity of Making Unique Choices: Approximating 1-in-k SAT'. In: *APPROX/RANDOM*. Vol. 3624. LNCS. Springer, 2005, pp. 99–110. DOI: 10.1007/11538462_9.

[GW95] Michel X. Goemans and David P. Williamson. 'Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming'. In: *J. ACM* 42.6 (1995), pp. 1115–1145. DOI: 10.1145/227683.227684.

[Hås01] Johan Håstad. 'Some optimal inapproximability results'. In: *J. ACM* 48.4 (2001), pp. 798–859. DOI: 10.1145/502090.502098.

[Hen71] C. Ward Henson. 'A family of countable homogeneous graphs.' In: *Pacific Journal of Mathematics* 38.1 (1971), pp. 69–83.

[HHMN88] Roland Häggkvist, Pavol Hell, Donald J. Miller and Victor Neumann-Lara. 'On multiplicative graphs and the product conjecture'. In: *Comb.* 8.1 (1988), pp. 63–74. DOI: 10.1007/bf02122553.

[HM67] A. J. W. Hilton and E. C. Milner. 'Some Intersection Theorems for ystems of Finite Sets'. In: *The Quarterly Journal of Mathematics* 18.1 (1967), pp. 369–384. ISSN: 0033-5606. DOI: 10.1093/qmath/18.1.369.

[HMNŽ24] Johan Håstad, Björn Martinsson, Tamio-Vesa Nakajima and Stanislav Živný. 'A Logarithmic Approximation of Linearly-Ordered Colourings'. In: *APPROX/RANDOM*. Vol. 317. LIPIcs. LZI, 2024, 7:1–7:6. DOI: 10.4230/lipics.approx/random.2024.7.

[HN04] Pavol Hell and Jaroslav Nešetřil. *Graphs and homomorphisms*. Vol. 28. Oxford lecture series in mathematics and its applications. Oxford University Press, 2004.

[HN90] Pavol Hell and Jaroslav Nešetřil. 'On the complexity of H-coloring'. In: *J. Comb. Theory B* 48.1 (1990), pp. 92–110. DOI: 10.1016/0095-8956(90)90132-j.

[JCG97] Peter Jeavons, David A. Cohen and Marc Gyssens. 'Closure properties of constraints'. In: *J. ACM* 44.4 (1997), pp. 527–548. DOI: 10.1145/263867.263489.

[Jea98] Peter Jeavons. 'On the Algebraic Structure of Combinatorial Problems'. In: *Theor. Comput. Sci.* 200.1-2 (1998), pp. 185–204. DOI: 10.1016/s0304-3975(97)00230-2.

[Kar72] Richard M. Karp. 'Reducibility Among Combinatorial Problems'. In: *Complexity of Computer Computations*. The IBM Research Symposia Series. Plenum Press, New York, 1972, pp. 85–103. DOI: 10.1007/978-1-4684-2001-2_9.

[KB79] Ravindran Kannan and Achim Bachem. 'Polynomial Algorithms for Computing the Smith and Hermite Normal Forms of an Integer Matrix'. In: *SIAM Journal on Computing* 8.4 (1979), pp. 499–507. DOI: 10.1137/0208040.

[Kho02] Subhash Khot. 'On the power of unique 2-prover 1-round games'. In: *STOC*. ACM, 2002, pp. 767–775. DOI: 10.1145/509907.510017.

[Kho10] Subhash Khot. 'On the Unique Games Conjecture (Invited Survey)'. In: *CCC*. IEEE, 2010, pp. 99–121. DOI: 10.1109/ccc.2010.19.

[KKMO07] Subhash Khot, Guy Kindler, Elchanan Mossel and Ryan O'Donnell. 'Optimal Inapproximability Results for MAX-CUT and Other 2-Variable CSPs?' In: *SIAM J. Comput.* 37.1 (2007), pp. 319–357. DOI: 10.1137/s0097539705447372.

[KKR17] Vladimir Kolmogorov, Andrei A. Krokhin and Michal Rolínek. 'The Complexity of General-Valued CSPs'. In: *SIAM J. Comput.* 46.3 (2017), pp. 1087–1110. DOI: 10.1137/16m1091836.

[KMS98] David R. Karger, Rajeev Motwani and Madhu Sudan. 'Approximate Graph Coloring by Semidefinite Programming'. In: *J. ACM* 45.2 (1998), pp. 246–265. DOI: 10.1145/274787.274791.

[KO15] Marcin Kozik and Joanna Ochremiak. 'Algebraic Properties of Valued Constraint Satisfaction Problem'. In: *ICALP*. Vol. 9134. LNCS. Springer, 2015, pp. 846–858. DOI: 10.1007/978-3-662-47672-7_69.

[KOWŽ23] Andrei A. Krokhin, Jakub Opršal, Marcin Wrochna and Stanislav Živný. 'Topology and Adjunction in Promise Constraint Satisfaction'. In: *SIAM J. Comput.* 52.1 (2023), pp. 38–79. DOI: 10.1137/20m1378223.

[KSTW00] Sanjeev Khanna, Madhu Sudan, Luca Trevisan and David P. Williamson. 'The Approximability of Constraint Satisfaction Problems'. In: *SIAM J. Comput.* 30.6 (2000), pp. 1863–1920. DOI: 10.1137/s0097539799349948.

[KTY24] Ken-ichi Kawarabayashi, Mikkel Thorup and Hirotaka Yoneda. 'Better Coloring of 3-Colorable Graphs'. In: *STOC*. ACM, 2024, pp. 331–339. DOI: 10.1145/3618260.3649768.

[KV25] Andrei Krokhin and Danny Vagnozzi. *Approximating 1-in-3 SAT by linearly ordered hypergraph 3-colouring is NP-hard*. 2025. arXiv: 2508.14606 [cs.CC]. URL: https://arxiv.org/abs/2508.14606.

[Lad75] Richard E. Ladner. 'On the Structure of Polynomial Time Reducibility'. In: *J. ACM* 22.1 (1975), pp. 155–171. DOI: 10.1145/321864.321877.

[Lar25] Alberto Larrauri. 'Ineffectiveness for Search and Undecidability of PCSP Meta-Problems'. In: *FOCS*. Accepted. 2025. arXiv: 2504.04639 [cs.CC].

[Lev73] Leonid A Levin. 'Universal sequential search problems'. In: *Problems of information transmission* 9.3 (1973), pp. 265–266.

[LNR24] Anand Louis, Alantha Newman and Arka Ray. 'Improved Linearly Ordered Colorings of Hypergraphs via SDP Rounding'. In: *Fsttcs*. Vol. 323. LIPIcs. LZI, 2024, 30:1–30:19. DOI: 10.4230/lipics.fsttcs.2024.30.

[Lov78] László Lovász. 'Kneser's Conjecture, Chromatic Number, and Homotopy'. In: *J. Comb. Theory A* 25.3 (1978), pp. 319–324. DOI: 10.1016/0097-3165(78)90022-5.

[Mot25] Antoine Mottet. *Algebraic and algorithmic synergies between promise and infinite-domain CSPs*. 2025. arXiv: 2501.13740.

[MR99] Sanjeev Mahajan and H. Ramesh. 'Derandomizing Approximation Algorithms Based on Semidefinite Programming'. In: *SIAM J. Comput.* 28.5 (1999), pp. 1641–1663. DOI: 10.1137/s0097539796309326.

[NVWŽ25] Tamio-Vesa Nakajima, Zephyr Verwimp, Marcin Wrochna and Stanislav Živný. 'Complexity of Approximate Conflict-Free, Linearly-Ordered, and Nonmonochromatic Hypergraph Colourings'. In: *ICALP*. Vol. 334. LIPIcs. LZI, 2025, 169:1–169:10. DOI: 10.4230/lipics.icalp.2025.169.

[NŽ22] Tamio-Vesa Nakajima and Stanislav Živný. 'Linearly Ordered Colourings of Hypergraphs'. In: *ICALP*. Vol. 229. LIPIcs. LZI, 2022, 128:1–128:18. DOI: 10.4230/lipics.icalp.2022.128.

[NŽ23a] Tamio-Vesa Nakajima and Stanislav Živný. 'Boolean symmetric vs. functional PCSP dichotomy'. In: *LICS*. IEEE, 2023, pp. 1–12. DOI: 10.1109/lics56636.2023.10175746.

[NŽ23b] Tamio-Vesa Nakajima and Stanislav Živný. 'Linearly Ordered Colourings of Hypergraphs'. In: *ACM Trans. Comput. Theory* 14.3-4 (2023), pp. 1–19. DOI: 10.1145/3570909.

[NŽ24a] Tamio-Vesa Nakajima and Stanislav Živný. *A logarithmic approximation of linearly ordered colourings*. 2024. arXiv: 2404.19556v1 [math.CO].

[NŽ24b] Tamio-Vesa Nakajima and Stanislav Živný. *An approximation algorithm for Maximum DiCut vs. Cut*. 2024. arXiv: 2402.07863.

[NŽ24c] Tamio-Vesa Nakajima and Stanislav Živný. 'On the Complexity of Symmetric vs. Functional PCSPs'. In: *ACM Trans. Algorithms* 20.4 (2024), 33:1–33:29. DOI: 10.1145/3673655.

[NŽ25a] Tamio-Vesa Nakajima and Stanislav Živný. *A Dichotomy for Maximum PCSPs on Graphs*. 2025. arXiv: 2406.20069 [cs.DS].

[NŽ25b] Tamio-Vesa Nakajima and Stanislav Živný. 'Maximum And- vs. Even-SAT'. In: *APPROX/RANDOM*. Vol. 353. LIPIcs. LZI, 2025, 3:1–3:8. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2025.3.

[NŽ25c] Tamio-Vesa Nakajima and Stanislav Živný. 'Maximum Bipartite vs. Triangle-Free Subgraph'. In: *ICALP*. Vol. 334. LIPIcs. LZI, 2025, 121:1–121:16. DOI: 10.4230/lipics.icalp.2025.121.

[Pos41] Emil L. Post. *The Two-Valued Iterative Systems of Mathematical Logic. (AM-5)*. Princeton University Press, 1941. ISBN: 9780691095707. URL: http://www.jstor.org/stable/j.ctt1bgzb1r (visited on 05/08/2025).

[PRSS25] Michael Pinsker, Jakub Rydval, Moritz Schöbi and Christoph Spiess. *Three meta-questions on infinite-domain Constraint Satisfaction Problems*. 2025. arXiv: 2502.06621.

[Rag08] Prasad Raghavendra. 'Optimal algorithms and inapproximability results for every CSP?' In: *STOC*. ACM, 2008, pp. 245–254. DOI: 10.1145/1374376.1374414.

[Rai19] Martin Raič. 'A multivariate Berry–Esseen theorem with explicit constants'. In: *Bernoulli* 25.4a (2019), pp. 2824–2853. ISSN: 13507265, 15739759. DOI: 10.3150/18-bej1072.

[Raz95] Ran Raz. 'A parallel repetition theorem'. In: *STOC*. ACM, 1995, pp. 447–456. DOI: 10.1145/225058.225181.

[Rei08] Omer Reingold. 'Undirected connectivity in log-space'. In: *J. ACM* 55.4 (2008), 17:1–17:24. DOI: 10.1145/1391289.1391291.

[RS09] Prasad Raghavendra and David Steurer. 'How to Round Any CSP'. In: *FOCS*. IEEE, 2009, pp. 586–594. DOI: 10.1109/focs.2009.74.

[Sch78] Thomas J. Schaefer. 'The Complexity of Satisfiability Problems'. In: *STOC*. ACM, 1978, pp. 216–226. DOI: 10.1145/800133.804350.

[Sch98] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley Series in Discrete Mathematics & Optimization. Wiley, 1998. ISBN: 9780471982326.

[Smo10] Shakhar Smorodinsky. *Conflict-Free Coloring and its Applications*. 2010. arXiv: 1005.3616.

[TSSW00] Luca Trevisan, Gregory B. Sorkin, Madhu Sudan and David P. Williamson. 'Gadgets, Approximation, and Linear Programming'. In: *SIAM J. Comput.* 29.6 (2000), pp. 2074–2097. DOI: 10.1137/s0097539797328847.

[Tur04] Alan Turing. 'Intelligent Machinery (1948)'. In: *The Essential Turing*. Oxford University Press, Sept. 2004. ISBN: 9780198250791. DOI: 10.1093/oso/9780198250791.003.0016.

[Tur37] Alan M. Turing. 'On computable numbers, with an application to the Entscheidungsproblem'. In: *Proc. London Math. Soc.* s2-42.1 (1937), pp. 230–265. DOI: 10.1112/plms/s2-42.1.230.

[TŽ16] Johan Thapper and Stanislav Živný. 'The Complexity of Finite-Valued CSPs'. In: *J. ACM* 63.4 (2016), 37:1–37:33. DOI: 10.1145/2974019.

[Wig82] Avi Wigderson. 'A New Approximate Graph Coloring Algorithm'. In: *STOC*. ACM, 1982, pp. 325–329. DOI: 10.1145/800070.802207.

[Wro22] Marcin Wrochna. *A note on hardness of promise hypergraph colouring*. 2022. arXiv: 2205.14719.

[Zhu20] Dmitriy Zhuk. 'A Proof of the CSP Dichotomy Conjecture'. In: *J. ACM* 67.5 (2020), 30:1–30:78. DOI: 10.1145/3402029.

[Zuc07] David Zuckerman. 'Linear Degree Extractors and the Inapproximability of Max Clique and Chromatic Number'. In: *Theory Comput.* 3.1 (2007), pp. 103–128. DOI: 10.4086/toc.2007.v003a006.