Complexity of Constraint Satisfaction Problems

by

Michal Rolínek

May, 2017

A thesis presented to the
Graduate School
of the
Institute of Science and Technology Austria, Klosterneuburg, Austria
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

© by Michal Rolínek, May, 2017
All Rights Reserved

I hereby declare that this thesis is my own work and that it does not contain other people's work without this being so stated; this thesis does not contain my previous work without this being stated, and the bibliography contains all the literature that I used in writing the dissertation.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee, and that this thesis has not been submitted for a higher degree to any other university or institution.

I certify that any republication of materials presented in this thesis has been approved by the relevant publishers and co-authors.

Signature: _____

Michal Rolínek
May, 2017

Acknowledgments

There are many individuals who made my stay at IST memorable, helped me develop as a scientist, and offered me support. These would include my terrific office mates, my dancing partners, my collaborators, my flatmates, my parents, my football mates and others. I am grateful and I will, of course, let them know. However, this acknowledgement I dedicate to just one person[1].

Dear Vladimir,

You are not very sentimental, at least not on the outside, so I decided to minimize the sugary parts. Instead, I will tell three short stories from our time together and let them speak for themselves.

Let me go in order and start with our rotation project. Having had one free slot for an experimental rotation, I felt like trying something more practical. After being rejected by Gašper (Well, that had consequences!) I turned to you. Working with a tough-looking Russian who does crazy things with images and has his own algorithm running in MS Office sounded exciting. As the rotation approached, I was becoming aware of your reputation and it was not very encouraging. Apparently, having an online math discussion with you was almost impossible since you were communicating your ideas by what was described as “incomprehensible, but probably very smart drawings”. Of course, nothing could be further from the truth. Over the three months, we spent countless hours in your office and became comfortable enough to call each other’s ideas bullsh*t, in a friendly tone, if needed. At that point I was hooked, regardless of the topic.

The second story starts at a VCSP workshop in Krakow in January 2015. The news about Theorem 2.1.19 broke out and clearly the community thought the time was ripe

[1] . . . and to the European Research Council under the European Community’s Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement no 616160.

to prove (some form of) Theorem 2.2.3. One could really feel the tension as groups of distinguished community members sat in adjacent rooms and all were trying to prove the same thing. In the end, we left Krakow without anybody announcing the proof, but we all knew the race was on. The next four weeks were probably the most intense period of my PhD.; I even caught myself composing cyclic polymorphisms on a date (sorry Claudia!). We felt we were so close, but we couldn't quite get it all to work...

It culminated when I was spending a week in Texas. The problem had no rest, we were taking shifts based on our timezones, and kept exchanging progress. The very final stretch is best captured by the following e-mail excerpts:

```
VNK: If you are not busy, could you take a look at the proof of Theorem 2.5.3
(and finish it, if you can)?  I hope things work, but I don't really know.
I need to double check, but I think I can prove everything assuming that
Theorem 2.5.3 holds.

M (DFW): I am spending my next twenty hours on planes and at airports so I
might actually have time for it.

VNK: Actually, I was wrong - obtaining the result from Theorem 2.5.3 is still
an open question...

M (LHR): So on my first flight I convinced myself that Theorem 2.5.3 is true.
I hope nothing pops up when I try to write it up on my next flight.

VNK: Great!  I think I can indeed prove the rest using Theorem 2.5.3.

M (VIE): *file attached*
```

These 24 hours were the highlight of my life as a mathematician.

The last story is short, but it speaks volumes. It happened only once during my PhD. that you felt the need to mentor me. Not that you would otherwise be ecstatic about every single thing I did, however your remarks were always specific. This one time it was different; I cannot recall the full context, but I do remember your closing line from this exchange. It was a fantastic message and it will stay with me:

"I also sometimes get stuck on a problem and it gets frustrating. Do you know what I do? I just keep thinking about it until I solve it!"

Thank you, Vladimir, for being exactly the advisor I needed; we've had great times together.

Michal

Abstract

An instance of the Constraint Satisfaction Problem (CSP) is given by a finite set of variables, a finite domain of labels, and a set of constraints, each constraint acting on a subset of the variables. The goal is to find an assignment of labels to its variables that satisfies all constraints (or decide whether one exists). If we allow more general "soft" constraints, which come with (possibly infinite) costs of particular assignments, we obtain instances from a richer class called Valued Constraint Satisfaction Problem (VCSP). There the goal is to find an assignment with minimum total cost.

In this thesis, we focus (assuming that P $\neq$ NP) on classifying computational complexity of CSPs and VCSPs under certain restricting conditions. Two results are the core content of the work. In one of them, we consider VCSPs parametrized by a constraint language, that is the set of "soft" constraints allowed to form the instances, and finish the complexity classification modulo (missing pieces of) complexity classification for analogously parametrized CSP. The other result is a generalization of Edmonds' perfect matching algorithm. This generalization contributes to complexity classfications in two ways. First, it gives a new (largest known) polynomial-time solvable class of Boolean CSPs in which every variable may appear in at most two constraints and second, it settles full classification of Boolean CSPs with planar drawing (again parametrized by a constraint language).

Table of Contents

1 Introduction

Computational problems from many different areas involve finding an assignment of labels to a set of variables, where that assignment must satisfy some specified feasibility conditions and/or optimize some specified objective function. In many such problems, the feasibility conditions are local and also the objective function can be represented as a sum of functions, each of which depends on some subset of the variables. Examples include: Gibbs energy minimization, Markov Random Fields (MRF), Conditional Random Fields (CRF), Min-Sum Problems, Minimum Cost Homomorphism, Constraint Optimization Problems (COP) and Valued Constraint Satisfaction Problems (VCSP) [8, 20, 56, 62, 72],.

The constraint satisfaction problem provides a common framework for many theoretical and practical problems in computer science [21, 62]. An instance of the *constraint satisfaction problem* (CSP) consists of a collection of variables that must be assigned labels from a given domain subject to specified constraints [59]. The CSP is equivalent to the problem of evaluating conjunctive queries on databases [48], and to the homomorphism problem for relational structures [31]. The CSP deals only with the feasibility issue: can all constraints be satisfied simultaneously?

There are several natural optimization versions of the CSP: MAX CSP (or MIN CSP) where the goal is to find the assignment maximizing the number of satisfied constraints (or minimizing the number of unsatisfied constraints) [17, 21, 44, 45], problems like MAX-ONES and MIN-HOM where the constraints must be satisfied and some additional function of the assignment is to be optimized [21, 46, 65], and, the most general version, *valued CSP* or VCSP (also known as soft CSP), where each combination of values for variables in a constraint has a cost and the goal is to minimize the aggregate

cost [15, 19, 51, 67]. Thus, an instance of the VCSP amounts to minimizing a sum of functions, each depending on a subset of variables. By using infinite costs to indicate infeasible combinations, VCSP can model both feasibility and optimization aspects and so considerably generalises all the problems mentioned above [15, 19, 55]. There is much activity and very strong results concerning various aspects of approximability of (V)CSPs (see e.g. [5, 9, 14, 21, 25, 28, 37, 61] for a small sample), but in this thesis we focus on solving VCSPs to optimality.

We assume throughout the thesis that P $\neq$ NP. Since all the above problems are NP-hard in full generality, a major line of research in CSP tries to identify the tractable cases of such problems (see books/surveys [18, 21, 23, 55]), the primary motivation being the general picture rather than specific applications. This is also the focus of the thesis.

The two main ingredients of a constraint are (a) variables to which it is applied and (b) relations/functions specifying the allowed combinations of values or the costs for all combinations. Therefore, the main types of restrictions on CSP are (a) *structural* where the hypergraph formed by sets of variables appearing in individual constraints is restricted [36, 58], and (b) *language-based* where the constraint language, i.e. the set of relations/functions that can appear in constraints, is fixed (see, e.g. [12, 18, 21, 31, 67]). The ultimate sort of results in these directions are *dichotomy* results, pioneered by [63], which characterise the tractable restrictions and show that the rest are as hard as the corresponding general problem (which cannot generally be taken for granted). The language-based direction is considerably more active than the structural one, there are many partial language-based dichotomy results, e.g. [10, 11, 19, 21, 44, 45, 52, 65], but many central questions are still open. In this work, we address a language-based restriction on VCSPs and two combined (both structural and language-based) restrictions on CSPs with Boolean variables.

1.1 Classifying VCSPs

In Chapter 2, we study VCSPs with a fixed constraint language on a finite domain and give complete classification of the complexity of VCSPs modulo the complexity of

CSPs. The results are joint work with V. Kolmogorov and A. Krokhin and appeared as a conference paper at FOCS '15 [49].

Clearly, for a VCSP to be tractable, it is necessary that the corresponding feasibility CSP is tractable. We prove that any VCSP satisfying this necessary condition and an algebraic necessary condition formulated by Kozik and Ochremiak [53] is tractable.

The classification result has the following several unexpected features. One is that the algorithm that solves all tractable VCSPs uses feasibility checking only as a black-box. The other is that the algorithm is simply feasibility preprocessing followed by solving a linear programming relaxation - this was unexpected, for example, because higher levels of the Sherali-Adams hierarchy were used in [66] to prove tractability of a wide class of VCSPs. Finally, the proof of our result avoids structural universal algebra present in most CSP classifications and in [53, 54].

Our dichotomy theorem generalizes the dichotomy for finite-valued CSPs from [67], and, with the help of the CSP tractability result from [4], it also implies the tractability of VCSPs shown tractable in [66, 68]. It is also the culmination of research into complexity classification of language-based VCSPs in the sense that its scope cannot be widened, the yet unclassified part of the VCSP landscape is the (non-valued) CSP. One could, of course, extend the classification framework by looking at other forms of algorithmic tractability, say, approximation algorithms or fixed-parameter tractability, and such extensions will have many open questions. It is also interesting to obtain tighter and more explicit characterisations for important special cases of VCSP (as done in [68], for example), by deriving them from our main result or otherwise.

1.2 Classifying Boolean CSPs

The content of Chapter 3 will have more combinatorial flavor. We will discuss an extension of a classical perfect matching algorithm and its connections to certain types of Boolean CSPs. This joint work with A. Kazda and V. Kolmogorov was published in SODA '17 [47].

We address two special structural restrictions for CSPs with Boolean variables. One is limiting to at most two constraints per variable and the other requires the constraint

network to have a planar representation. The first type, introduced by Feder [29], has very natural interpretation as CSPs in which edges play the role of variables and nodes the role of constraints, which is why we choose to refer to it as *edge CSP*. It was Feder who showed the following hardness result: Assume the constraint language Γ contains both unary constant relations. Then unless all relations in Γ are Δ-matroids, the edge CSP with constraint language Γ has the same complexity as the unrestricted CSP with constraint language Γ. Since then, there has also been progress on the algorithmic side. Several tractable classes of Δ-matroids were identified [29, 30, 41, 24, 34, 26]. A recurring theme is the connection between Δ-matroids and matching problems.

A setting for planar CSPs appears already in [22] and was recently brought back into attention by Dvořák and Kupec [26]. In their work, they provide certain hardness results together with a reduction of the remaining cases to Boolean edge CSP. Dvořák and Kupec's results imply that completing the complexity classification of Boolean planar CSPs is equivalent to establishing the complexity of (planar) Boolean edge CSP where all the constraints are *even Δ-matroids*. In their paper, Dvořák and Kupec provided a tractable subclass of even Δ-matroids along with computer-aided evidence that the subclass (matching realizable even Δ-matroids) covers all even Δ-matroids of arity at most 5. However, it turns out that there exist even Δ-matroids of arity 6 that are not matching realizable; we provide an example of such a Δ-matroid.

The main contribution of the work is a generalization of the classical Edmonds' blossom-shrinking algorithm for matchings [27] that we use to efficiently solve edge CSPs with even Δ-matroid constraints. This settles the complexity classification of planar CSP. Moreover, we give an extension of the algorithm to cover a wider class of Δ-matroids. This extension subsumes (to our best knowledge) all previously known tractable classes.

2 Complexity of Valued CSP

In this chapter, we prove a dichotomy-like theorem for valued CSP. This provides full complexity classification of VCSPs modulo gaps in our understanding of (non-valued) CSP. Before we turn to formal statements, let us shed more light on the previous statement by providing some context.

One major knowledge gap regarding CSPs is captured by the CSP Dichotomy Conjecture. It states that each CSP is either tractable or NP-hard, and was first formulated by Feder and Vardi [31]. A universal-algebraic approach to this problem was discovered in [12, 42, 43], and the precise boundary between the tractable cases and NP-hard cases was conjectured in algebraic terms in [12], in what is now known as the Algebraic CSP Dichotomy Conjecture (see Conjecture 2.1.21). The hardness part was proved in [12], and it is the tractability part that is the essence of the conjecture. This conjecture is still open in full generality and is the object of much investigation, e.g. [2, 3, 4, 1, 6, 12, 11, 18, 40]. It is known to hold for domains with at most 3 elements [10, 63], for smooth digraphs [6], and for the case when all unary relations are available [1, 11]. The main two polynomial-time algorithms used for CSPs are based one on local consistency ("bounded width") and the other on compact representation of solution sets, such as in systems of linear equations ("few subpowers"), and their applicability (in pure form) is fully characterized in [2, 4] and [40], respectively.

At the opposite (to CSP) end of the VCSP spectrum are the finite-valued CSPs, in which functions do not take infinite values. In such VCSPs, the feasibility aspect is trivial, and one has to deal only with the optimization issue. One polynomial-time algorithm that solves tractable finite-valued CSPs is based on the so-called basic linear programming (BLP) relaxation, and its applicability (also for the general-valued case)

was fully characterized in [51] (see Theorem 2.1.22). The complexity of finite-valued CSPs was completely classified in [67], where it is shown that all finite-valued CSPs not solvable by BLP are NP-hard.

For general-valued CSPs, full classifications are known for the Boolean case (i.e., when the domain is two-element) [19] and also for the case when all 0-1-valued unary cost functions are available [52]. The algebraic approach to the CSP was extended to VCSPs in [15, 16, 19, 53], and was also key to much progress. An algebraic necessary condition for a VCSP to be tractable was proved by Kozik and Ochremiak in [53], where this condition was also conjectured to be sufficient (see Theorem 2.1.19 and Conjecture 2.1.20 below). This conjecture can be called the Algebraic VCSP Dichotomy Conjecture, and it is a generalization of the corresponding conjecture for CSP. A large family of VCSPs satisfying the necessary condition from [53] has recently been shown tractable via a low-level Sherali-Adams hierarchy relaxation [66].

Our result says that any dichotomy for CSP (not necessarily the one predicted by the Algebraic CSP Dichotomy Conjecture) will imply a dichotomy for VCSP. However, if the Algebraic CSP Dichotomy Conjecture holds then the necessary algebraic condition of Kozik and Ochremiak guarantees tractability of the feasibility CSP (see [53]), implying that this algebraic condition alone is necessary and sufficient for tractability of a VCSP, and also that all the not tractable VCSPs are NP-hard. In particular, the Algebraic CSP Dichotomy Conjecture implies the Algebraic VCSP Dichotomy Conjecture.

On the technical level, some of our proofs (e.g. those in Section 2.6) use techniques established in [51, 67], while others (e.g. all of Section 2.5) introduce new technical ideas. The technique of “lifting a language” was introduced in [50].

2.1 Preliminaries

Valued Constraint Satisfaction Problems

Throughout the chapter, let D be a fixed finite set and let $\overline{\mathbb{Q}} = \mathbb{Q} \cup \{\infty\}$ denote the set of rational numbers with (positive) infinity.

Definition 2.1.1. We denote the set of all functions $f : D^n \to \overline{\mathbb{Q}}$ by $\mathcal{F}_D^{(n)}$ and let $\mathcal{F}_D = \bigcup_{n\geq 1} \mathcal{F}_D^{(n)}$. We will often call the functions in $\mathcal{F}_D$ *cost functions* over D. For every

cost function $f \in \mathcal{F}_D^{(n)}$, let $\operatorname{dom} f = \{x \mid f(x) < \infty\}$. Note that $\operatorname{dom} f$ can be considered both as an n-ary relation and as a n-ary function such that $\operatorname{dom} f(x) = 0$ if and only if $f(x)$ is finite.

We will call the set D the *domain*, elements of D *labels* (for variables), and say that the cost functions in $\mathcal{F}_D$ take *values*. Note that in some papers on VCSP, e.g. [15, 66], cost functions are called weighted relations.

Definition 2.1.2. An instance of the *valued constraint satisfaction problem* (VCSP) is a function from D^V to $\overline{\mathbb{Q}}$ given by

$$f_{\mathcal{I}}(x) = \sum_{t \in T} f_t(x_{v(t,1)}, \ldots, x_{v(t,n_t)}), \tag{2.1}$$

where V is a finite set of variables, T is a finite set of constraints, each constraint is specified by a cost function f_t of arity n_t and indices $v(t,k)$, $k = 1, \ldots, n_t$. The goal is to find an *assignment* (or *labeling*) $x \in D^V$ that minimizes $f_{\mathcal{I}}$. The value of an optimal assignment is denoted by $\operatorname{Opt}(\mathcal{I})$.

Definition 2.1.3. Any set $\Gamma \subseteq \mathcal{F}_D$ is called a *valued constraint language* over D, or simply a *language*. We will denote by $\operatorname{VCSP}(\Gamma)$ the class of all VCSP instances in which the constraint functions f_t are all contained in Γ. Instances of $\operatorname{VCSP}(\Gamma)$ will sometimes be called just Γ*-instances*.

Note that if every function in Γ takes values in $\{0, \infty\}$ (such functions are often called *crisp*) then $\operatorname{VCSP}(\Gamma)$ is a pure feasibility problem, commonly known as $\operatorname{CSP}(\Gamma)$.

Example 2.1.4. *Let us fix $k \geq 2$ set $D = \{1, \ldots, k\}$, and $\Gamma_{color} = \{f\}$, where $f \colon D^2 \to \overline{\mathbb{Q}}$ is defined for $(x, y) \in D^2$ as*

$$f(x,y) = \begin{cases} 0 & \text{if } x \neq y, \\ \infty & \text{otherwise.} \end{cases}$$

*Then $\operatorname{VCSP}(\Gamma_{color})$, which coincides with $\operatorname{CSP}(\Gamma_{color})$ in this case, models exactly k-*COLORING *of graphs (edges correspond to pairs of variables constrained by f).*

Example 2.1.5. *Let* $D = \{0, 1, 2\}$ *and* $\Gamma_{eq} = \{f_{eq}\}$*, where* $f_{eq} \colon D^3 \to \overline{\mathbb{Q}}$ *is defined for* $(x, y, z) \in D^3$ *as*

$$f_{eq}(x, y, z) = \begin{cases} 0 & \text{if } x + y + z \equiv 1 \pmod 3, \\ \infty & \text{otherwise.} \end{cases}.$$

Then $\mathrm{VCSP}(\Gamma_{eq})$ *(as well as* $\mathrm{CSP}(\Gamma_{eq})$*) models systems of (specific) linear equations modulo 3.*

Example 2.1.6. *Let* Γ_{mc} *be a constraint language on a Boolean domain* $D = \{0, 1\}$ *containing a single binary function* $f_{mc} \colon D^2 \to \overline{\mathbb{Q}}$ *defined by*

$$f_{mc}(x, y) = \begin{cases} 0 & \text{if } x \neq y, \\ 1 & \text{if } x = y. \end{cases}.$$

Then the problem $\mathrm{VCSP}(\Gamma_{mc})$ *is equivalent to the NP-Hard* MAX-CUT *[33].*

Example 2.1.7. *On a Boolean domain* $D = \{0, 1\}$ *consider* $\Gamma_{mis} = \{f, u\}$ *where* $f \colon D^2 \to \overline{\mathbb{Q}}$ *is defined as* $f(1, 1) = \infty$ *and* $f(0, 1) = f(0, 0) = f(1, 0) = 0$ *and* $u \colon D \to \overline{\mathbb{Q}}$ *is simply* $u(x) = 1 - x$*. Then the* MAX-INDEPENDENT-SET *graph problem is captured in* $\mathrm{VCSP}(\Gamma_{mis})$ *(binary constraints correspond to edges and the unary one is applied to every vertex; the independent set is then the subset of vertices labelled with 1). In fact, the opposite also holds (and is easy to see); any instance of* $\mathrm{VCSP}(\Gamma_{mis})$ *can be reduced to an instance of* MAX-INDEPENDENT-SET*.*

In our final example, we will construct a valued constraint language that corresponds to the (s, t)-MIN-CUT problem on directed graphs. As we define the cost functions, keep in mind that label 0 will correspond to the source component and label 1 to the sink component.

Example 2.1.8. *Fix* $D = \{0, 1\}$*. For any cost* $w \in \overline{\mathbb{Q}}, w \geq 0$*, let* $f_w \colon D^2 \to \overline{\mathbb{Q}}$ *be given by*

$$f_w(x, y) = \begin{cases} w & \text{if } x = 0,\ y = 1, \\ 0 & \text{otherwise.} \end{cases}.$$

and also for a fixed $d \in D$ *define* $u_w^d \colon D \to \overline{\mathbb{Q}}$ *by*

$$u_w(x) = \begin{cases} w & \text{if } x = d, \\ 0 & \text{otherwise.} \end{cases}$$

We denote by Γ_{cut} *the set of all possible* f_w *and* u_w^d*. Now the* (s,t)-MIN-CUT *can be modeled by placing* u_∞^1 *on the source and* u_∞^0 *on the sink and transforming the (weighted) graph edges into binary constraints* f_w*.*

Interestingly, also any instance of $\mathrm{VCSP}(\Gamma_{cut})$ *can be reduced to a* (s,t)-MIN-CUT *instance [35].*

For more examples of capturing well-known problems or subsuming previously studied frameworks see [55].

The main goal of our line of research is to classify the complexity of problems $\mathrm{VCSP}(\Gamma)$. Problems $\mathrm{CSP}(\Gamma)$ and $\mathrm{VCSP}(\Gamma)$ are called tractable if, for each finite $\Gamma' \subseteq \Gamma$, $\mathrm{VCSP}(\Gamma')$ is tractable. Also, $\mathrm{VCSP}(\Gamma)$ is called NP-hard if, for some finite $\Gamma' \subseteq \Gamma$, $\mathrm{VCSP}(\Gamma')$ is NP-hard. One advantage of defining tractability in terms of finite subsets is that the tractability of a valued constraint language is independent of whether the cost functions are represented explicitly (say, via full tables of values, or via tables for the finite-valued parts) or implicitly (via oracles). Following [12], we say that $\mathrm{VCSP}(\Gamma)$ is *globally tractable* if there is a polynomial-time algorithm solving $\mathrm{VCSP}(\Gamma)$, assuming all functions in instances are given by full tables of values. For CSPs, there is no example of $\mathrm{CSP}(\Gamma)$ that is tractable, but not globally tractable, and it is conjectured in [12] that no such $\mathrm{CSP}(\Gamma)$ exists.

Polymorphisms, Expressibility, Cores

Let $\mathcal{O}_D^{(m)}$ denote the set of all operations $g : D^m \to D$ and let $\mathcal{O}_D = \cup_{m \geq 1} \mathcal{O}_D^{(m)}$. When D is clear from the context, we will sometimes write simply $\mathcal{O}^{(m)}$ and $\mathcal{O}$.

Any language Γ defined on D can be associated with a set of operations on D, known as the polymorphisms of Γ, which allow one to combine (often in a useful way) several feasible assignments into a new one.

Definition 2.1.9. An operation $g \in \mathcal{O}_D^{(m)}$ is a *polymorphism* of a cost function $f \in \mathcal{F}_D$ if, for any $x^1, x^2, \ldots, x^m \in \operatorname{dom} f$, we have that $g(x^1, x^2, \ldots, x^m) \in \operatorname{dom} f$ where g is applied component-wise.

For any valued constraint language Γ over a set D, we denote by $\operatorname{Pol}(\Gamma)$ the set of all operations on D which are polymorphisms of every $f \in \Gamma$.

Example 2.1.10. *Let $f \in \mathcal{F}_{\{0,1\}}^{(n)}$ be such that $f(1, \ldots, 1, 0) = \infty$ and $f(a_1, \ldots, a_n) = 0$ otherwise. It corresponds to the Horn clause $(x_1 \vee \ldots \vee x_{n-1} \vee \overline{x_n})$. Then it is well known and easy to see that the binary operation $\min \in \mathcal{O}_{\{0,1\}}$ is a polymorphism of f.*

Clearly, if g is a polymorphism of a cost function f, then g is also a polymorphism of $\operatorname{dom} f$. For $\{0, \infty\}$-valued functions, which naturally correspond to relations, the notion of a polymorphism defined above coincides with the standard notion of a polymorphism for relations. Note that the projections (aka dictators), i.e. operations of the form $e_n^i(x_1, \ldots, x_n) = x_i$, are polymorphisms of all valued constraint languages. Polymorphisms play the key role in the algebraic approach to the CSP, but, for VCSPs, more general constructs are necessary, which we now define.

Definition 2.1.11. An m-ary *fractional operation* ω on D is a probability distribution on $\mathcal{O}_D^{(m)}$. The support of ω is defined as $\operatorname{supp}(\omega) = \{g \in \mathcal{O}_D^{(m)} \mid \omega(g) > 0\}$.

Definition 2.1.12. A m-ary fractional operation ω on D is said to be a *fractional polymorphism* of a cost function $f \in \mathcal{F}_D$ if, for any $x^1, x^2, \ldots, x^m \in \operatorname{dom} f$, we have

$$\sum_{g \in \operatorname{supp}(\omega)} \omega(g) f(g(x^1, \ldots, x^m)) \leq \frac{1}{m}(f(x^1) + \ldots + f(x^m)). \tag{2.2}$$

For a constraint language Γ, $\operatorname{fPol}(\Gamma)$ will denote the set of all fractional operations that are fractional polymorphisms of each function in Γ. Also, let $\operatorname{fPol}^+(\Gamma) = \{g \in \mathcal{O}_D \mid g \in \operatorname{supp}(\omega), \omega \in \operatorname{fPol}(\Gamma)\}$.

The intuition behind the notion of fractional polymorphism is that it allows one to combine several feasible assignments into new feasible assignments so that the expected value of a new assignment (non-strictly) improves the average value of the original assignments.

Example 2.1.13. *Suppose that ω is a binary fractional operation on $D = \{0, 1\}$ such that $\omega(\min) = \omega(\max) = 1/2$. Then it is well-known and easy to check that the finite-valued functions with fractional polymorphism ω are the submodular functions. Moreover, functions with this fractional polymorphism that are not necessarily finite-valued precisely correspond to submodular functions defined on a ring family.*

More examples of fractional polymorphisms can be found in [55, 51, 67].

We remark that, in some papers (e.g., in [15]), fractional polymorphisms (and closely related objects called weighted polymorphisms) are defined as rational-valued functions, which is sufficient for analysing the complexity of VCSPs with finite constraint languages. However, real-valued fractional polymorphisms are necessary to analyse infinite constraint languages [32, 54, 67].

The key observation in the algebraic approach to (V)CSP is that neither the complexity nor the algebraic properties of a language Γ change when functions "expressible" from Γ in a certain way are added to it.

Definition 2.1.14. For a constraint language Γ, let $\langle\Gamma\rangle$ denote the set of all functions $f(x_1, \ldots, x_k)$ such that, for some instance $\mathcal{I}$ of $\mathrm{VCSP}(\Gamma)$ with objective function

$$f_{\mathcal{I}}(x_1, \ldots, x_k, x_{k+1}, \ldots, x_n),$$

we have

$$f(x_1, \ldots, x_k) = \min_{x_{k+1}, \ldots, x_n} f_{\mathcal{I}}(x_1, \ldots, x_k, x_{k+1}, \ldots, x_n).$$

We then say that Γ *expresses* f, and call $\langle\Gamma\rangle$ the *expressive power* of Γ.

Lemma 2.1.15 ([16, 19])**.** *Let $f \in \langle\Gamma\rangle$. Then*

1. *if $\omega \in \mathrm{fPol}(\Gamma)$ then ω is a fractional polymorphism of f and of $\mathrm{dom}\, f$;*
2. *$\mathrm{VCSP}(\Gamma)$ is tractable if and only if $\mathrm{VCSP}(\Gamma \cup \{f, \mathrm{dom}\, f\})$ is tractable;*
3. *$\mathrm{VCSP}(\Gamma)$ is NP-hard if and only if $\mathrm{VCSP}(\Gamma \cup \{f, \mathrm{dom}\, f\})$ is NP-hard.*

The dichotomy problem for VCSPs can be reduced to a class of constraint languages called rigid cores, defined below. Apart from reducing the cases that need to be

considered, this reduction enabled the use of much more powerful results from universal algebra than what can be done without this restriction (see, e.g. [54]).

For a subset $D' \subseteq D$, let $u_{D'}$ be the function defined as follows: $u_{D'}(d) = 0$ if $d \in D'$ and $u_{D'}(d) = \infty$ otherwise. We write u_d for $u_{\{d\}}$. Let $\mathcal{C}_D = \{\{u_d\} \mid d \in D\}$.

Lemma 2.1.16 ([54]). *For any valued constraint language Γ' on a finite set D', there is a subset $D \subseteq D'$ and a valued constraint language Γ on D such that $\mathcal{C}_D \subseteq \Gamma$ and the problems $\mathrm{VCSP}(\Gamma')$ and $\mathrm{VCSP}(\Gamma)$ are polynomial-time equivalent.*

This language Γ is called the *rigid core* of Γ', and it can be obtained from Γ' as follows. Let g' be a unary operation on D' with minimum $|g'(D')|$ among all unary operations $g' \in \mathrm{fPol}^+(\Gamma')$. Then D is set to be $g'(D')$ and Γ is set to be $\{f|_D : f \in \Gamma'\} \cup \mathcal{C}_D$. Thus, the intuition behind moving to the rigid core is that (a) one removes labels from the domain that can always be (uniformly) replaced in any solution to an instance without increasing its value, and (b) one allows constraints of the form u_d that can be used to fix labels for variables, leading to applicability of more powerful algebraic results.

Cyclic and symmetric operations

Several types of operations play a special role in the algebraic approach to (V)CSP.

Definition 2.1.17. An operation $g \in \mathcal{O}_D^{(m)}$, $m \geq 2$, is called

- *idempotent* if $g(x, \ldots, x) = x$ for all $x \in D$;
- *Taylor* if, for each $1 \leq i \leq m$, it satisfies an identity of the form $g(\triangle_1, \triangle_2, \ldots, \triangle_m) = g(\square_1, \square_2, \ldots, \square_m)$ where all $\triangle_j, \square_j$ are in $\{x, y\}$ and $\triangle_i \neq \square_i$.
- *cyclic* if $g(x_1, x_2, \ldots, x_m) = g(x_2, \ldots, x_m, x_1)$ for all $x_1, \ldots, x_m \in D$;
- *symmetric* if $g(x_1, x_2, \ldots, x_m) = g(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(m)})$ for all $x_1, \ldots, x_m \in D$, and any permutation π on $[m]$.

A fractional operation ω is said to be idempotent/cyclic/symmetric if all operations in $\mathrm{supp}(\omega)$ have the corresponding property.

It is well known and easy to see that all polymorphisms and fractional polymorphisms of a rigid core are idempotent.

The following lemma is contained in the proof of Theorem 50 in [54].

Lemma 2.1.18. *Let Γ be a (rigid)*[1] *core on a set D. Then the following are equivalent:*

1. $\mathrm{fPol}^{+}(\Gamma)$ *contains a Taylor operation of arity at least 2;*
2. Γ *has a cyclic fractional polymorphism of (some) arity at least 2;*
3. Γ *has a cyclic fractional polymorphism of every prime arity $p > |D|$.*

The following theorem is Corollary 51 from [54].

Theorem 2.1.19 ([54])**.** *Let Γ be a valued constraint language that is a rigid core. If $\mathrm{fPol}^{+}(\Gamma)$ does not contain a Taylor operation then $\mathrm{VCSP}(\Gamma)$ is NP-hard.*

Kozik and Ochremiak state a conjecture (which they attribute to L. Barto) that the above theorem describes all NP-hard valued constraint languages, and all other languages are tractable. Using Lemma 2.1.18, we restate the original conjecture via cyclic fractional polymorphisms.

Conjecture 2.1.20 ([53])**.** *Let Γ be a valued constraint language that is a rigid core. If Γ has a cyclic fractional polymorphism of arity at least 2, then $\mathrm{VCSP}(\Gamma)$ is tractable.*

Note that, for a finite core Γ (but with fixed D), the above condition can be checked in polynomial time. Indeed, if $p > |D|$ is some fixed prime number, then it is sufficient to check for a cyclic fractional polymorphism of arity p. Such polymorphisms, by definition, are solutions to a system of linear inequalities. Since the number of cyclic operations of arity p on D is constant, the system will have size polynomial in Γ and its feasibility can be decided by linear programming.

For the case when (possibly infinite) Γ consists only of $\{0, \infty\}$-valued functions, $\mathrm{VCSP}(\Gamma)$ is actually a CSP. For such Γ, *any* probability distribution on polymorphisms (of the same arity) is a fractional polymorphism. Then a theorem and a conjecture (the latter now known as the Algebraic CSP Dichotomy Conjecture) equivalent to

[1] Theorem 50 in [54] in fact operates with a weaker notion of a core, which we have not defined. However, in Appendix A.2 we extend the context and will need the full version of Lemma 2.1.18.

Theorem 2.1.19 and Conjecture 2.1.20 were given in [12]. One of several equivalent forms of the Algebraic CSP Dichotomy Conjecture is as follows.

Conjecture 2.1.21 ([12, 3])**.** *Let* Γ *be a valued constraint language that is a rigid core and that consists of* $\{0, \infty\}$*-valued functions. If* Γ *has a cyclic polymorphism of arity at least 2, then* $\mathrm{VCSP}(\Gamma)$ *is tractable. Otherwise,* $\mathrm{VCSP}(\Gamma)$ *is NP-hard.*

In view of this, it is natural to call Conjecture 2.1.20 the *Algebraic VCSP Dichotomy Conjecture*.

Basic LP relaxation

Symmetric operations are known to be closely related to LP-based algorithms for CSP-related problems. One algorithm in particular has been known to solve many VCSPs to optimality. This algorithm is based on the so-called *basic LP relaxation*, or BLP, defined as follows.

Let $\mathbb{M}_n = \{\mu \geq 0 \mid \sum_{x \in D^n} \mu(x) = 1\}$ be the set of probability distributions over labelings in D^n. We also denote $\Delta = \mathbb{M}_1$; thus, Δ is the standard $(|D| - 1)$-dimensional simplex. The corners of Δ can be identified with elements in D. For a distribution $\mu \in \mathbb{M}_n$ and a variable $v \in \{1, \ldots, n\}$, let $\mu_{[v]} \in \Delta$ be the marginal probability of distribution μ for v:

$$\mu_{[v]}(a) \;=\; \sum_{x \in D^n : x_v = a} \mu(x) \qquad \forall a \in D.$$

Given a VCSP instance $\mathcal{I}$ in the form (2.1), we define the value $\mathrm{BLP}(\mathcal{I})$ as follows:

$$\begin{aligned}
\mathrm{BLP}(\mathcal{I}) \;&=\; \min \sum_{t \in T} \sum_{x \in \mathrm{dom}\, f_t} \mu_t(x) f_t(x) && && (2.3)\\
\text{s.t.}\;\; (\mu_t)_{[k]} \;&=\; \alpha_{v(t,k)} && \forall t \in T, k \in \{1, \ldots, n_t\}\\
\mu_t \;&\in\; \mathbb{M}_{n_t} && \forall t \in T\\
\mu_t(x) \;&=\; 0 && \forall t \in T, x \notin \mathrm{dom}\, f_t\\
\alpha_v \;&\in\; \Delta && \forall v \in V
\end{aligned}$$

If there are no feasible solutions then $\mathrm{BLP}(\mathcal{I}) = \infty$. The objective function and all constraints in this system are linear, therefore this is a linear program. Its size is polynomial in the size of $\mathcal{I}$, so $\mathrm{BLP}(\mathcal{I})$ can be found in time polynomial in $|\mathcal{I}|$.

We say that BLP *solves* $\mathcal{I}$ if $\mathrm{BLP}(\mathcal{I}) = \min_{x \in D^n} f_{\mathcal{I}}(x)$, and BLP solves $\mathrm{VCSP}(\Gamma)$ if it solves all instances $\mathcal{I}$ of $\mathrm{VCSP}(\Gamma)$. If BLP solves $\mathrm{VCSP}(\Gamma)$ and Γ is a rigid core, then the optimal solution for every instance can be found by using the standard self-reducibility method. In this method, one goes through the variables in some order, finding $d \in D$ for the current variable v such that instances $\mathcal{I}$ and $\mathcal{I} + u_d(v)$ have the same optimal value (which can be checked by BLP), updating $\mathcal{I} := \mathcal{I} + u_d(v)$, and moving to the next variable. At the end, the instance will have a unique feasible assignment whose value is the optimum of the original instance. Note that in this case $\mathrm{VCSP}(\Gamma)$ is globally tractable.

Theorem 2.1.22 ([51]). *BLP solves* $\mathrm{VCSP}(\Gamma)$ *if and only if, for every* $m > 1$, Γ *has a symmetric fractional polymorphism of arity* m.

Theorem 2.1.23 ([51, 67]). *Let* Γ *be a rigid core constraint language that is finite-valued. If* Γ *has a symmetric fractional polymorphism of arity* 2 *then BLP solves* $\mathrm{VCSP}(\Gamma)$*, and so* $\mathrm{VCSP}(\Gamma)$ *is tractable. Otherwise,* $\mathrm{VCSP}(\Gamma)$ *is NP-hard.*

2.2 Main Result

Definition 2.2.1. Let $\mathcal{I}$ be a VCSP instance over variables V with domain D. The *feasibility instance*, $\mathrm{Feas}(\mathcal{I})$, associated to $\mathcal{I}$ is a CSP instance obtained from $\mathcal{I}$ by replacing each constraint function f_t with $\mathrm{dom}\, f_t$.

For a language Γ, let $\mathrm{Feas}(\Gamma) = \{\mathrm{dom}\, f \mid f \in \Gamma\}$. Then the instances of the problem $\mathrm{CSP}(\mathrm{Feas}(\Gamma))$ are the instances $\mathrm{Feas}(\mathcal{I})$ where $\mathcal{I}$ runs through all instances of $\mathrm{VCSP}(\Gamma)$.

Definition 2.2.2. Let $\mathcal{I}$ be a VCSP instance over variables V with domain D. For each variable $v \in V$, let $D_v = \{d \in D \mid d = \sigma(v)$ for some feasible solution σ for $\mathcal{I}\}$. Then $(1, \infty)$*-minimal instance* $\bar{\mathcal{I}}$ associated with $\mathcal{I}$ is the VCSP instance obtained from $\mathcal{I}$ by adding, for each $v \in V$, the constraint $u_{D_v}(x_v)$.

Note that if Γ is a rigid core and the problem $\mathrm{CSP}(\mathrm{Feas}(\Gamma))$ is tractable, then, for any instance $\mathcal{I}$ of $\mathrm{VCSP}(\Gamma)$, one can construct the associated $(1, \infty)$-minimal instance in polynomial time. Indeed, to find out whether a given $d \in D$ is in D_v, one only needs to decide whether the CSP instance obtained from $\mathrm{Feas}(\mathcal{I})$ by adding the constraint

$u_d(x_v)$ is satisfiable. Since Γ is a rigid core, the latter instance is also an instance of $\mathrm{CSP}(\mathrm{Feas}(\Gamma))$.

If Γ is a rigid core then, for $\mathrm{VCSP}(\Gamma)$ to be tractable, Γ must satisfy the assumption of Conjecture 2.1.20, and also, clearly, the feasibility part of the problem, $\mathrm{CSP}(\mathrm{Feas}(\Gamma))$, must be tractable. Our main result shows that if these necessary conditions are satisfied then $\mathrm{VCSP}(\Gamma)$ is indeed tractable.

Theorem 2.2.3. *Let Γ be a valued constraint language over domain D that is a rigid core. If the following conditions hold then $\mathrm{VCSP}(\Gamma)$ is tractable:*

1. *Γ has a cyclic fractional polymorphism of arity at least 2, and*
2. *$\mathrm{CSP}(\mathrm{Feas}(\Gamma))$ is tractable.*

Otherwise, $\mathrm{VCSP}(\Gamma)$ is not tractable.

In Theorem 2.2.3, the intractability part for (absence of) the first condition follows from Theorem 2.1.19, and it is obvious for the second condition. The tractability part follows from Theorem 2.2.4 below.

Theorem 2.2.4. *Let Γ be an arbitrary language that has a cyclic fractional polymorphism of arity at least 2. If $\mathcal{I}$ is an instance of $\mathrm{VCSP}(\Gamma)$ and $\bar{\mathcal{I}}$ is its associated $(1,\infty)$-minimal instance, then $\mathrm{Opt}(\mathcal{I}) = \mathrm{BLP}(\bar{\mathcal{I}})$.*

Indeed, if Γ is a rigid core satisfying conditions (1) and (2) from Theorem 2.2.3 and $\mathcal{I}$ is an instance of $\mathrm{VCSP}(\Gamma)$ then the equality $\mathrm{Opt}(\mathcal{I}) = \mathrm{BLP}(\bar{\mathcal{I}})$ means that we can efficiently find the optimum value for $\mathcal{I}$ by constructing $\bar{\mathcal{I}}$ (which we can do efficiently because Γ is a rigid core and $\mathrm{CSP}(\mathrm{Feas}(\Gamma))$ is tractable) and then applying BLP to $\bar{\mathcal{I}}$. Then we can find an optimal assignment for $\mathcal{I}$ by self-reduction (see the discussion before Theorem 2.1.22).

Recall the notion of global tractability from Section 2.1. The algorithm that we just described gives the following.

Corollary 2.2.5. *Let Γ be a valued constraint language over domain D that is a rigid core. If*

1. *Γ has a cyclic fractional polymorphism of arity at least 2, and*

2. $\mathrm{CSP}(\mathrm{Feas}(\Gamma))$ *is globally tractable,*

then $\mathrm{VCSP}(\Gamma)$ *is globally tractable.*

It also follows from Theorem 2.2.4 that, for every language Γ that has a cyclic fractional polymorphism of arity at least 2, $\mathrm{VCSP}(\Gamma)$ is polynomial time equivalent to $\mathrm{CSP}(\mathrm{Feas}(\Gamma))$. In particular, *any* complexity classification of CSPs, whether it is the dichotomy as predicted by Conjecture 2.1.21 or anything else, gives a complexity classification of VCSPs.

Let us now discuss how Theorem 2.2.3 can be combined with known CSP complexity classifications to obtain new, previously unknown, VCSP classifications which are tighter than Theorem 2.2.3.

As we explained in the beginning of the chapter, if the Algebraic CSP Dichotomy Conjecture holds, then condition (2) in Theorem 2.2.3 can be omitted and all intractable VCSPs are NP-hard. Since this conjecture holds when $|D| \leq 3$ [10, 63] or when D is arbitrary finite, but Γ contains all unary crisp functions [1, 11], we get the following corollaries.

Corollary 2.2.6. *Let* $|D| \leq 3$ *and let* Γ *be a valued constraint language that is a rigid core on* D*. If* Γ *has a cyclic fractional polymorphism then the problem* $\mathrm{VCSP}(\Gamma)$ *is tractable, otherwise it is NP-hard.*

For the case $|D| = 2$, the tractable cases can be characterised by six specific cyclic fractional polymorphisms [19], and it was shown in [54] that the presence of any cyclic fractional polymorphism (when $|D| = 2$) implies the presence of one of those six. Also, Corollary 2.2.6 generalizes results from [70, 71] where the dichotomy was shown for the special case when $|D| = 3$ and all non-crisp functions in Γ are unary. The specific conditions for tractability in [70, 71] have not been shown to be directly implied by the presence of a cyclic fractional polymorphism, though.

Corollary 2.2.7. *Let* Γ *be a valued constraint language on* D *that contains all unary crisp functions. If* Γ *has a cyclic fractional polymorphism then the problem* $\mathrm{VCSP}(\Gamma)$ *is tractable, otherwise it is NP-hard.*

Corollary 2.2.7 generalizes a result from [70] where the dichotomy was shown for the special case when Γ includes all unary crisp functions and all non-crisp functions

in Γ are unary. Again, the specific condition for tractability in [70] is not known to be directly implied by the presence of a cyclic fractional polymorphism.

It is shown in [54] how Theorem 2.2.3 implies the dichotomy results (including specific conditions for tractability) for the finite-valued case from [67] (Theorem 2.1.23) and for the case when Γ contains all unary functions taking values in $\{0, 1\}$ [52]. The algorithm for the tractable case in [52] is somewhat similar in spirit to our algorithm, and actually inspired the latter.

Let us now explain how Theorem 2.2.3 implies the tractability result from [66] (stated below). An idempotent operation $g \in \mathcal{O}_D$ of arity at least 2 satisfying $g(y, x, x \ldots, x, x) = g(x, y, x, \ldots, x, x) = \ldots = g(x, x, x, \ldots, x, y)$ for all $x, y \in D$ is called a *weak near-unanimity* operation. The tractability result result from [66] states that if $\mathrm{fPol}^+(\Gamma)$ contains weak near-unanimity operations of all but finitely many arities, then $\mathrm{VCSP}(\Gamma)$ is tractable (in fact, via a specific algorithm based on Sherali-Adams hierarchy, which does not follow from our results). This condition on $\mathrm{fPol}^+(\Gamma)$ is well known in the algebraic approach to the CSP, it characterizes (when appropriately formulated) CSPs of bounded width [4]. So assume that $\mathrm{fPol}^+(\Gamma)$ satisfies this condition. Since $\mathrm{fPol}^+(\Gamma) \subseteq \mathrm{Pol}(\Gamma)$, the set $\mathrm{Pol}(\Gamma)$ also contains these operations, so $\mathrm{CSP}(\mathrm{Feas}(\Gamma))$ is tractable by [4]. Moreover, by [3], $\mathrm{fPol}^+(\Gamma)$ then also contains a cyclic operation of arity at least 2. Now (the proof of) Theorem 50 of [54] implies that Γ has a cyclic fractional polymorphism of arity at least 2, and then tractability of $\mathrm{VCSP}(\Gamma)$ follows from Theorem 2.2.3.

We remark that some known VCSP classifications with tighter and more explicit characterisations of tractability can be easily derived from our main result, e.g. the classification for the Boolean case ($|D| = 2$) can be easily derived following the lines of Section 8 of [15]. However, it might take additional effort to derive some others - for example, the dichotomy result from [68] was proved without using our theorem, and it is not known how to derive it from our main result.

2.3 Proof of Theorem 2.2.4: Reduction to a block-finite language

We will prove Theorem 2.2.4 by constructing, from a given (feasible) instance $\mathcal{I}$, a finite valued constraint language Γ' on some finite set D' and an instance $\mathcal{I}'$ of $\mathrm{VCSP}(\Gamma')$ such that $\mathrm{Opt}(\mathcal{I}) = \mathrm{Opt}(\bar{\mathcal{I}}) = \mathrm{Opt}(\mathcal{I}') = \mathrm{BLP}(\mathcal{I}') = \mathrm{BLP}(\bar{\mathcal{I}})$. The first equality is immediate from the definition $\bar{\mathcal{I}}$, the second one will follow trivially from the construction of Γ' and $\mathcal{I}'$, and the last equality holds by Lemma 2.3.1 below, while the key equality $\mathrm{Opt}(\mathcal{I}') = \mathrm{BLP}(\mathcal{I}')$ will follow from the fact that BLP solves $\mathrm{VCSP}(\Gamma')$ that we prove, using Theorem 2.1.22, in Theorem 2.3.4. The construction is inspired by [50], where a similar technique of "lifting" a language was used in a different context.

Let V be the set of variables of instance $\mathcal{I}$, and let

$$f_{\mathcal{I}}(x) \;=\; \sum_{t \in T} f_t(x_{v(t,1)}, \ldots, x_{v(t,n_t)}) \qquad \forall x : V \to D \tag{2.4}$$

be its objective function. For the $(1,\infty)$-minimal instance $\bar{\mathcal{I}}$, the objective function is

$$f_{\bar{\mathcal{I}}}(x) \;=\; \sum_{t \in T} f_t(x_{v(t,1)}, \ldots, x_{v(t,n_t)}) + \sum_{v \in V} u_{D_v}(x_v) \qquad \forall x : V \to D \tag{2.5}$$

Now let $D'_v = \{(v,a) \mid a \in D_v\}$ be a unique copy of D_v. We now define a new language Γ' over domain $D' = \bigcup_{v \in V} D'_v$ as follows:

$$\Gamma' = \bigcup_{t \in T} \left\{ f_t^{\langle v(t,1),\ldots,v(t,n_t)\rangle}, \mathrm{dom}\, f_t^{\langle v(t,1),\ldots,v(t,n_t)\rangle} \right\} \cup \bigcup_{v \in V} \left\{ u_{D'_v} \right\} \cup \{ =_{D'} \}$$

where functions $u_{D'_v}$ are as defined above, $=_{D'}$ is the binary $\{0,\infty\}$-valued function corresponding to the equality relation, and, for an n-ary function f over D and variables $v_1, \ldots, v_n \in V$, we define function $f^{\langle v_1,\ldots,v_n\rangle} : (D')^n \to \overline{\mathbb{Q}}$ as follows:

$$f^{\langle v_1,\ldots,v_n\rangle}(x) = \begin{cases} f(\hat{x}) & \text{if } x = ((v_1,\hat{x}_1),\ldots,(v_n,\hat{x}_n)) \\ \infty & \text{otherwise} \end{cases} \qquad \forall x \in (D')^n$$

The above mentioned instance $\mathcal{I}'$ of $\mathrm{VCSP}(\Gamma')$ is obtained from $\bar{\mathcal{I}}$ by replacing each

function f_t with $f_t^{\langle v(t,1),\dots,v(t,n_t)\rangle}$ and replacing each function u_{D_v} with $u_{D'_v}$.

It is straightforward to check that there is a one-to-one correspondence between the sets of feasible solutions to BLP relaxations for $\mathcal{I}'$ and $\bar{\mathcal{I}}$, and that this correspondence also preserves the values of the solutions.

Lemma 2.3.1. *We have* $\mathrm{BLP}(\mathcal{I}') = \mathrm{BLP}(\bar{\mathcal{I}})$.

Lemma 2.3.2. *If* Γ *has a cyclic fractional polymorphism of arity* $m > 1$ *then* Γ' *has the same property.*

Proof. Let ω be a cyclic fractional polymorphism of Γ. Fix an arbitrary element $d' \in D'$. For each operation $g \in \mathrm{supp}(\omega)$, define the operation g' on D' as follows:

$$g'(x_1,\dots,x_m) = \begin{cases} (v, g(\hat{x}_1,\dots,\hat{x}_m)) & \text{if } x_1=(v,\hat{x}_1),\dots,x_m=(v,\hat{x}_m) \text{ for some } v\in V \\ d' & \text{otherwise} \end{cases}$$

Clearly, each operation g' is cyclic. Consider the fractional operation ω' on D' such that $\omega(g') = \omega(g)$ for all $g \in supp(\omega)$. It is straightforward to check that ω' is a fractional polymorphism of Γ'. □

To prove Theorem 2.2.4, it remains to show that $\mathrm{Opt}(\mathcal{I}') = \mathrm{BLP}(\mathcal{I}')$. We will prove the more general fact that BLP solves $\mathrm{VCSP}(\Gamma')$. The properties of the language Γ' that we use for this (apart from having a cyclic fractional polymorphism) are given below in Definition 2.3.3.

Definition 2.3.3. A finite language Γ is called *block-finite* if its domain D can be partitioned into disjoint subsets $\{D_v \mid v \in V\}$ such that

(a) For any $a \in D_v$ with $v \in V$ there exists a polymorphism $g_a \in \mathcal{O}^{(1)}$ of $\mathrm{Feas}(\Gamma)$ such that $g_a(b) = a$ for all $b \in D_v$.

(b) For any n-ary function $f \in \Gamma$, the relation $\mathrm{dom}\, f$ (viewed as a function $D^n \to \{0,\infty\}$) belongs to Γ. Furthermore, the binary equality relation on D, denoted as $=_D\colon D^2 \to \{0,\infty\}$, also belongs to Γ.

(c) Any n-ary function $f \in \Gamma - \{=_D\}$ satisfies $\mathrm{dom}\, f \subseteq D_{v_1} \times \ldots \times D_{v_n}$ for some $v_1,\dots,v_n \in V$.

It is easy to see that the language Γ' defined in previously in this section is block-finite. It obviously has properties (b) and (c), and it has property (a) because the instance $\mathcal{I}'$ is $(1,\infty)$-minimal. Indeed, if $a=(v,d)\in D'_v$ then, by definition, $\mathcal{I}$ has a feasible solution $\sigma: V\to D$ with $\sigma(v)=d$. Define function g_a as follows: for each $a'=(v',d')\in D'$, set $g_a(a')=(v',\sigma(v'))$. It is easy to check that g_a has the required properties.

From now on, we forget about the original language Γ from the previous section and about the specific language Γ' and work with an arbitrary block-finite language that has a cyclic fractional polymorphism of arity at least 2. For simplicity, we denote our language by Γ. Note that Γ is not necessarily a (rigid) core, but this property is not required in Theorem 2.1.22. By Theorem 2.1.22, in order to prove Theorem 2.2.4, it remains to show the following.

Theorem 2.3.4. *Suppose that a block-finite language Γ admits a cyclic fractional polymorphism ν of arity at least 2. Then, for every $m\geq 2$, Γ admits a symmetric fractional polymorphism $\omega_m^{\mathtt{sym}}$ of arity m.*

In the rest of the chapter we prove Theorem 2.3.4. This will be done in two steps: (i) using the existence of ν, prove the existence of $\omega_2^{\mathtt{sym}}$; (ii) using the existence of $\omega_{m-1}^{\mathtt{sym}}$ for some $m\geq 3$, prove the existence of $\omega_m^{\mathtt{sym}}$. The claim will then follow by induction on m.

Note that for finite-valued languages step (i) was proved in [67] (or rather a very closely related statement), while step (ii) was established in [51]. However, in both cases it was essential that the language is finite-valued. The arguments in [51, 67] seem to break down when infinities are allowed. For example, we were unable to extend the approach in [67] that exploits the connectivity of a certain graph on D. Partial results from this attempt can be found in Appendix A.2. To deal with block-finite languages, we will introduce (in Section 2.5) a new technical tool where we first prove, via Farkas Lemma, the existence of a certain function with special properties in $\langle\Gamma\rangle$.

2.4 A graph of generalized operations

In this section we describe a basic tool that will be used for constructing new fractional polymorphisms, namely a graph of generalized operations introduced in [51].

Let $\mathcal{O}^{(m\to m)}$ be the set of mappings $\mathbf{g} : D^m \to D^m$ and let $\mathbb{1} \in \mathcal{O}^{(m\to m)}$ be the identity mapping. Consider a sequence x of m labelings $x \in [D^n]^m$; this means that $x = (x^1, \ldots, x^m)$ where $x^i \in D^n$ (think of x as an $m \times n$ matrix whose rows are x^1, ..., x^m). For an n-ary function f, we define $f^m(x) = \frac{1}{m}(f(x^1) + \ldots + f(x^m))$ (thus $f^m(x)$ is the average value of f on the rows of x). For a mapping $\mathbf{g} = (g_1, \ldots, g_m) \in \mathcal{O}^{(m\to m)}$, we also denote $x^{\mathbf{g}i} = g_i(x)$ for $i \in [m]$ and $\mathbf{g}(x) = (x^{\mathbf{g}1}, \ldots, x^{\mathbf{g}m})$ (so $g(x)$ is an $m \times n$ matrix where row i is obtained by column-wise application of g_i to x).

A probability distribution ρ over $\mathcal{O}^{(m\to m)}$ will be called a *(generalized) fractional polymorphism of* Γ *of arity* $m \to m$ if each function $f \in \Gamma$ satisfies

$$\sum_{\mathbf{g}\in\mathrm{supp}(\rho)} \rho(\mathbf{g}) f^m(\mathbf{g}(x)) \;\leq\; f^m(x) \qquad \forall x \in [\mathrm{dom}\, f]^m \tag{2.6}$$

We will sometimes represent fractional polymorphisms of arity m and generalised fractional polymorphisms of arity $m \to m$ as vectors in $\mathbb{R}^{\mathcal{O}^{(m)}}$ and $\mathbb{R}^{\mathcal{O}^{(m\to m)}}$, respectively. For $g \in \mathcal{O}^{(m)}$ and $\mathbf{g} \in \mathcal{O}^{(m\to m)}$, we denote the corresponding characteristic vectors by χ_g and $\chi_{\mathbf{g}}$ respectively. It can be checked that a generalized fractional polymorphism ρ of arity $m \to m$ can be converted into a fractional polymorphism ρ' of arity m, as follows:

$$\rho' = \sum_{\mathbf{g}=(g_1,\ldots,g_m)\in\mathrm{supp}(\rho)} \frac{\rho(\mathbf{g})}{m}(\chi_{g_1} + \ldots + \chi_{g_m}).$$

We will use the following construction in several parts of the proof. Assume that we have some probability distribution ω with a finite support such that (i) each element $s \in \mathrm{supp}(\omega)$ corresponds to an element of $\mathcal{O}^{(m\to m)}$ denoted as $\mathbb{1}^s$, and (ii) this distribution satisfies the following property for each $f \in \Gamma$:

$$\sum_{s\in\mathrm{supp}(\omega)} \omega(s) f^m(\mathbb{1}^s(x)) \;\leq\; f^m(x) \qquad \forall x \in [\mathrm{dom}\, f]^m \tag{2.7a}$$

Condition (2.7a) then can be rephrased as saying that vector $\sum_{s\in\mathrm{supp}(\omega)} \omega(s)\chi_{\mathbb{1}^s}$ is a fractional polymorphism of Γ of arity $m \to m$. We will also consider the following condition:

$$\sum_{s\in\mathrm{supp}(\omega)} \omega(s) f(x^{\mathbb{1}^s i}) \;\leq\; f^{m-1}(x_{-i}) \qquad \forall x \in [\mathrm{dom}\, f]^m, i \in [m] \tag{2.7b}$$

where $x_{-i} \in [\mathrm{dom}\, f]^{m-1}$ denotes the sequence of $m-1$ labelings obtained from x by removing the i-th labeling. Note that condition (2.7b) implies (2.7a) (since summing (2.7b) over $i \in [m]$ and dividing by m gives (2.7a)). The second condition will be used only in one of the results; unless noted otherwise, ω is only assumed to satisfy (2.7a).

For a mapping $\mathbf{g} \in \mathcal{O}^{(m\to m)}$ denote $\mathbf{g}^s = \mathbb{1}^s \circ \mathbf{g}$. (This notation is consistent with the earlier one since $\mathbb{1}^s \circ \mathbb{1} = \mathbb{1}^s$ for any s). We use $\mathbf{g}^{s_1\cdots s_k}$ to denote $(\ldots(\mathbf{g}^{s_1})^{\cdots})^{s_k} = \mathbb{1}^{s_k} \circ \ldots \circ \mathbb{1}^{s_1} \circ \mathbf{g}$. Next, define a directed graph $(\mathbb{G}, E)$ as follows:

- $\mathbb{G} = \{\mathbb{1}^{s_1\cdots s_k} \mid s_1, \ldots, s_k \in \mathrm{supp}(\omega), k \geq 0\}$ is the set of all mappings that can be obtained from $\mathbb{1}$ by applying operations from $\mathrm{supp}(\omega)$;
- $E = \{(\mathbf{g}, \mathbf{g}^s) \mid \mathbf{g} \in \mathbb{G}, s \in \mathrm{supp}(\omega)\}$.

This graph can be decomposed into strongly connected components, yielding a directed acyclic graph (DAG) on these components. We define $\mathsf{Sinks}(\mathbb{G}, E)$ to be the set of those strongly connected components $\mathbb{H} \subseteq \mathbb{G}$ of $(\mathbb{G}, E)$ that are sinks of this DAG (i.e. have no outgoing edges). Any DAG has at least one sink, therefore $\mathsf{Sinks}(\mathbb{G}, E)$ is non-empty. We denote $\mathbb{G}^* = \bigcup_{\mathbb{H}\in\mathsf{Sinks}(\mathbb{G},E)} \mathbb{H} \subseteq \mathbb{G}$ and $Range_n(\mathbb{G}^*) = \{\mathbf{g}^*(x) \mid \mathbf{g}^* \in \mathbb{G}^*, x \in [D^n]^m\}$. Also, for a tuple $\hat{x} \in D^m$ we will denote $\mathbb{G}(\hat{x}) = \{\mathbf{g}(\hat{x}) \mid \mathbf{g} \in \mathbb{G}\} \subseteq D^m$.

The following facts can easily be shown (see Appendix A.1).

Proposition 2.4.1. *(a) If* $\mathbf{g}, \mathbf{h} \in \mathbb{G}$ *then* $\mathbf{h} \circ \mathbf{g} \in \mathbb{G}$. *Moreover, if* $\mathbf{g} \in \mathbb{H} \in \mathsf{Sinks}(\mathbb{G}, E)$ *then* $\mathbf{h} \circ \mathbf{g} \in \mathbb{H}$.
(b) Consider connected components $\mathbb{H}', \mathbb{H} \in \mathsf{Sinks}(\mathbb{G}, E)$. *For each* $\mathbf{g}' \in \mathbb{H}'$ *there exists* $\mathbf{g} \in \mathbb{H}$ *satisfying* $\mathbf{g} \circ \mathbf{g}' = \mathbf{g}'$.
(c) For each $x \in Range_n(\mathbb{G}^*)$ *and* $\mathbb{H} \in \mathsf{Sinks}(\mathbb{G}, E)$ *there exists* $\mathbf{g} \in \mathbb{H}$ *satisfying* $\mathbf{g}(x) = x$.

Proposition 2.4.2. *Suppose that* $\hat{x} \in Range_1(\mathbb{G}^*)$ *and* $x \in \mathbb{G}(\hat{x})$.
(a) There holds $x \in Range_1(\mathbb{G}^*)$.
(b) There exists $\mathbf{g} \in \mathbb{G}$ *such that* $\mathbf{g}(x) = \hat{x}$.

We now state main theorems related to the graph $(\mathbb{G}, E)$, that are slight extensions of the results in [51]. Their proofs use the same techniques as [51] and can be found in Appendix A.1.

Theorem 2.4.3. *Let $\widehat{\mathbb{G}}$ be a subset of $\mathbb{G}$ satisfying the following property: for each $\mathbf{g} \in \mathbb{G}$ there exists a path in $(\mathbb{G}, E)$ from $\mathbf{g}$ to some node $\widehat{\mathbf{g}} \in \widehat{\mathbb{G}}$. Then there exists a fractional polymorphism ρ of Γ of arity $m \to m$ with $\mathrm{supp}(\rho) = \widehat{\mathbb{G}}$.*

We will use this result either for the set $\widehat{\mathbb{G}} = \mathbb{G}$ or for the set $\widehat{\mathbb{G}} = \mathbb{G}^*$; clearly, both choices satisfy the condition of the theorem. The first choice gives that Γ admits a fractional polymorphism ρ with $\mathrm{supp}(\rho) = \mathbb{G}$; therefore, if $\mathbf{g} \in \mathbb{G}$, $f \in \langle\Gamma\rangle$ and $x \in [\mathrm{dom}\, f]^m$ then $\mathbf{g}(x) \in [\mathrm{dom}\, f]^m$.

Theorem 2.4.4. *Consider function $f \in \langle\Gamma\rangle$ of arity n and labelings $x \in Range_n(\mathbb{G}^*) \cap [\mathrm{dom}\, f]^m$.*

(a) There holds $f^m(\mathbf{g}(x)) = f^m(x)$ for any $\mathbf{g} \in \mathbb{G}$.

(b) Suppose that condition (2.7b) *holds. Then there exists a probability distribution λ over $\mathbb{G}^*$ (which is independent of f, x) such that $f^\lambda_{i'}(x) = f^\lambda_{i''}(x)$ for any $i', i'' \in [m]$ where*

$$f^\lambda_i(x) = \sum_{\mathbf{g} \in \mathbb{G}^*} \lambda_{\mathbf{g}} f(x^{\mathbf{g}i}) \tag{2.8}$$

2.5 Constructing special functions

In this section, we construct special functions in $\langle\Gamma\rangle$ that play an important role in the proof of Theorem 2.3.4.

For a sequence $x = (x^1, \ldots, x^m) \in D^m$ and a permutation π of $[m]$, we define $x^\pi = (x^{\pi(1)}, \ldots, x^{\pi(m)})$. Similarly, for a mapping $\mathbf{g} = (g_1, \ldots, g_m) \in \mathcal{O}^{(m\to m)}$ define $\mathbf{g}^\pi = (g_{\pi(1)}, \ldots, g_{\pi(m)})$. Let Ω be the set of mappings $\mathbf{g} \in \mathcal{O}^{(m\to m)}$ that satisfy the following condition:

$$\mathbf{g}^\pi(x) = \mathbf{g}(x^\pi) \text{ for any } x \in D^m \text{ and any permutation } \pi \text{ of } [m]. \tag{2.9}$$

Equivalently, $g_{\pi(i)}(x) = g_i(x^\pi)$ for any $i \in [m]$.

Proposition 2.5.1. *If $\mathbf{g}, \mathbf{h} \in \Omega$, then $\mathbf{g} \circ \mathbf{h} \in \Omega$.*

Proof. Just note that

$$(\mathbf{g} \circ \mathbf{h})^{\pi}(x) = \mathbf{g}^{\pi}(\mathbf{h}(x)) = \mathbf{g}(\mathbf{h}^{\pi}(x)) = \mathbf{g}(\mathbf{h}(x^{\pi})) = (\mathbf{g} \circ \mathbf{h})(x^{\pi})$$

for any $x \in D^m$. □

Consider all generalized fractional polymorphisms ω of Γ of arity $m \to m$ satisfying $\mathrm{supp}(\omega) \subseteq \Omega$. At least one such polymorphism exists, namely $\omega = \chi_{\mathbb{1}}$ where $\mathbb{1} \in \mathcal{O}^{(m \to m)}$ is the identity mapping. Among such ω's, pick one with the largest support. It exists due to the following observation: if ω', ω'' are generalized fractional polymorphisms of Γ of arity $m \to m$ then so is the vector $\omega = \frac{1}{2}[\omega' + \omega'']$, and $\mathrm{supp}(\omega) = \mathrm{supp}(\omega') \cup \mathrm{supp}(\omega'')$.

Let us apply the construction of Section 2.4 starting with the chosen distribution ω, where for $\mathbf{g} \in \mathrm{supp}(\omega)$ we define operation $\mathbb{1}^{\mathbf{g}} \in \mathcal{O}^{(m \to m)}$ via $\mathbb{1}^{\mathbf{g}} = \mathbf{g}$. Let the resulting graph be $(\mathbb{G}, E)$. It is straightforward to check that condition (2.7a) holds: it simply expresses the fact that ω is a generalized fractional polymorphsism of Γ of arity $m \to m$.

Proposition 2.5.2. *It holds that* $\mathrm{supp}(\omega) = \mathbb{G}$.

Proof. If $\mathbf{g} \in \mathrm{supp}(\omega)$ then $\mathbf{g} = \mathbb{1}^{\mathbf{g}} \in \mathbb{G}$. Conversely, suppose that $\mathbf{g} \in \mathbb{G}$. We can write $\mathbf{g} = \mathbb{1}^{\mathbf{g}_k} \circ \ldots \circ \mathbb{1}^{\mathbf{g}_1} = \mathbf{g}_k \circ \cdots \circ \mathbf{g}_1$ with $\mathbf{g}_1, \ldots, \mathbf{g}_k \in \mathrm{supp}(\omega) \subseteq \Omega$. Since Ω is closed under composition by Proposition 2.5.1, we get $\mathbf{g} \in \Omega$. By Theorem 2.4.3 there exists a generalized fractional polymorphism ρ with $\mathrm{supp}(\rho) = \mathbb{G}$, and so $\mathbf{g} \in \mathrm{supp}(\rho)$. By maximality of ω we get $\mathbf{g} \in \mathrm{supp}(\omega)$. □

In the remainder of this section we prove the following theorem.

Theorem 2.5.3. *For any* $\hat{x} \in D^m$ *there exists a function* $f \in \langle\Gamma\rangle$ *of arity* m *with* $\arg\min f = \mathbb{G}(\hat{x})$.

Proof. Let Γ^+ be the set of pairs (f, x) with $f \in \Gamma$ and $x \in [\mathrm{dom}\, f]^m$. Let $\Omega' \subseteq \Omega$ be the set of mappings $\mathbf{g} \in \Omega$ that satisfy $\mathbf{g}(x) \in [\mathrm{dom}\, f]^m$ for all $(f, x) \in \Gamma^+$. Note that

$\mathbb{G} = \mathrm{supp}(\omega) \subseteq \Omega'$. By the choice of ω, the following system does not have a solution with rational ρ:

$$\begin{aligned} \rho(\mathbf{g}) &\geq 0 \qquad \forall \mathbf{g} \in \Omega' && \text{(2.10a)} \\ \sum_{\mathbf{g}\in\Omega'} \rho(\mathbf{g}) f^m(x) - \sum_{\mathbf{g}\in\Omega'} \rho(\mathbf{g}) f^m(\mathbf{g}(x)) &\geq 0 \qquad \forall (f,x) \in \Gamma^+ && \text{(2.10b)} \\ \sum_{\mathbf{g}\in\Omega'-\mathbb{G}} -\rho(\mathbf{g}) &< 0 && \text{(2.10c)} \end{aligned}$$

Next, we use the following well-known result (see, e.g. [64]).

Lemma 2.5.4 (Farkas Lemma)**.** *Let A be a $p \times q$ matrix and b be a p-dimensional vector. Then exactly one of the following is true:*

- *There exists $\lambda \in \mathbb{R}^q$ such that $A\lambda = b$ and $\lambda \geq 0$.*
- *There exists $\mu \in \mathbb{R}^p$ such that $\mu^T A \geq 0$ and $\mu^T b < 0$.*

If A and b are rational then λ and μ can also be chosen in $\mathbb{Q}^q$ and $\mathbb{Q}^p$, respectively.

By this lemma, the following system has a solution with rational $\lambda \geq 0$:

$$\begin{aligned} \lambda(\mathbf{g}) + \sum_{(f,x)\in\Gamma^+} \lambda(f,x)(f^m(x) - f^m(\mathbf{g}(x)) &= 0 \qquad \forall \mathbf{g} \in \mathbb{G} && \text{(2.11a)} \\ \lambda(\mathbf{g}) + \sum_{(f,x)\in\Gamma^+} \lambda(f,x)(f^m(x) - f^m(\mathbf{g}(x)) &= -1 \qquad \forall \mathbf{g} \in \Omega' - \mathbb{G} && \text{(2.11b)} \end{aligned}$$

We will now define several instances of $\mathrm{VCSP}(\Gamma)$ where it will be convenient to use constraints with rational positive weights; these weights can always be made integer by multiplying the instances by an appropriate positive integer, which would not affect the reasoning, but make notation cumbersome.

We will define a Γ-instance $\mathcal{I}$ with $m|D|^m$ variables $\mathcal{V} = \{(i,z) \,|\, i \in [m], z \in D^m\}$. The labelings $\mathcal{V} \to D$ for this instance can be identified with mappings $\mathbf{g} = (g_1, \ldots, g_m) \in \mathcal{O}^{(m\to m)}$, if we define $\mathbf{g}(i,z) = g_i(z)$ for the coordinate $(i,z) \in \mathcal{V}$. We define the cost function of $\mathcal{I}$ as follows:

$$f_{\mathcal{I}}(\mathbf{g}) = \sum_{(f,x)\in\Gamma^+, \lambda(f,x)\neq 0} \lambda(f,x) f^m(\mathbf{g}(x)) \qquad \forall \mathbf{g} \in \mathcal{O}^{(m\to m)} \qquad \text{(2.12)}$$

From (2.11) we get

$$f_{\mathcal{I}}(\mathbb{1}) \;=\; f_{\mathcal{I}}(\mathbf{g}) - \lambda(\mathbf{g}) \;\leq\; f_{\mathcal{I}}(\mathbf{g}) \;<\; \infty \qquad \forall \mathbf{g} \in \mathbb{G} \tag{2.13a}$$

$$f_{\mathcal{I}}(\mathbb{1}) \;<\; f_{\mathcal{I}}(\mathbf{g}) - \lambda(\mathbf{g}) \;\leq\; f_{\mathcal{I}}(\mathbf{g}) \;<\; \infty \qquad \forall \mathbf{g} \in \Omega' - \mathbb{G} \tag{2.13b}$$

Furthermore, $f^m(\cdot)$ is invariant with respect to permuting its arguments, and thus

$$f_{\mathcal{I}}(\mathbf{g}) \;=\; f_{\mathcal{I}}(\mathbf{g}^\pi) \qquad \forall \mathbf{g} \in \mathcal{O}^{(m\to m)},\ \text{permutation } \pi \text{ of } [m] \tag{2.13c}$$

Let T be the set of tuples (i, j, x, y) where $i, j \in [m]$, $x, y \in D^m$ and $i = \pi(j)$, $y = x^\pi$ for some permutation π of $[m]$. Define another Γ-instance $\mathcal{I}'$ with variables $\mathcal{V}$ and the cost function

$$f_{\mathcal{I}'}(\mathbf{g}) = f_{\mathcal{I}}(\mathbf{g}) \;+ \sum_{(i,j,x,y)\in T} =_D (\mathbf{g}(i,x), \mathbf{g}(j,y)) + \sum_{(f,x)\in\Gamma^+} (\operatorname{dom} f)^m(\mathbf{g}(x)) \quad \forall \mathbf{g} \in \mathcal{O}^{(m\to m)} \tag{2.14}$$

where $=_D$ is the equality relation on D. The instance $\mathcal{I}'$ is a Γ-instance because of condition (b) in the definition of a block-finite language. Note that the second term in (2.14) for $\mathbf{g} = (g_1, \ldots, g_m)$ equals 0 if $g_{\pi(j)}(x) = g_j(x^\pi)$ for all $j \in [m]$, $x \in D^m$ and permutation π of $[m]$. Otherwise the second term equals ∞. In other words, the second term is zero if and only if mapping $\mathbf{g}$ satisfies condition (2.9), i.e. if and only if $\mathbf{g} \in \Omega$. Similarly, the third term in (2.14) is zero if $\mathbf{g} \in \Omega'$, and ∞ if $\mathbf{g} \in \Omega - \Omega'$. We obtain that

$$f_{\mathcal{I}'}(\mathbb{1}) \;\leq\; f_{\mathcal{I}'}(\mathbf{g}) \;<\; \infty \qquad \forall \mathbf{g} \in \mathbb{G} \tag{2.15a}$$

$$f_{\mathcal{I}'}(\mathbb{1}) \;<\; f_{\mathcal{I}'}(\mathbf{g}) \;<\; \infty \qquad \forall \mathbf{g} \in \Omega' - \mathbb{G} \tag{2.15b}$$

$$f_{\mathcal{I}'}(\mathbf{g}) \;=\; \infty \qquad \forall \mathbf{g} \in \mathcal{O}^{(m\to m)} - \Omega' \tag{2.15c}$$

These equations imply that $\mathbb{1} \in \arg\min f_{\mathcal{I}'} \subseteq \mathbb{G}$. We will show next that $\arg\min f_{\mathcal{I}'} = \mathbb{G}$.

For an index $k \in \mathbb{Z}$ let $\bar{k} \in [m]$ be the unique index with $\bar{k} - k = 0 \pmod m$. Let π_k be the cyclic permutation of $[m]$ with $\pi_k(1) = \bar{k}$. In particular, π_1 is the identity permutation. Also, for $k \in \mathbb{Z}$ let $e^k \in \mathcal{O}^{(m)}$ be the projection to the $\bar{k}$-th coordinate. For a mapping $\mathbf{g} = (g_1, \ldots, g_m) \in \mathcal{O}^{(m\to m)}$ and a tuple $z \in D^m$ we will denote $\mathbf{g}(k, z) = \mathbf{g}(\bar{k}, z) \in D$.

From the definition, for any permutation π of $[m]$ and any $(i,z) \in \mathcal{V}$ we have $\mathbf{g}^{\pi}(i,z) = (g_{\pi(1)}, \dots, g_{\pi(m)})(i,z) = g_{\pi(i)}(z) = \mathbf{g}(\pi(i),z)$. In particular, $\mathbf{g}^{\pi_j}(i,z) = \mathbf{g}(i+j-1,z)$.

From (2.13c) we have $f_{\mathcal{I}}(\mathbb{1}^{\pi_1}) = \ldots = f_{\mathcal{I}}(\mathbb{1}^{\pi_m}) = f_{\mathcal{I}}(\mathbb{1})$. Recall that $f_{\mathcal{I}} \in \langle\Gamma\rangle$ admits a generalized fractional polymorphism ω with $\mathrm{supp}(\omega) = \mathbb{G}$. Applying this polymorphism gives

$$\sum_{\mathbf{g}\in\mathrm{supp}(\omega)} \omega(\mathbf{g}) f_{\mathcal{I}}^m(\mathbf{g}(\mathbb{1}^{\pi_1}, \dots, \mathbb{1}^{\pi_m})) \le f_{\mathcal{I}}^m(\mathbb{1}^{\pi_1}, \dots, \mathbb{1}^{\pi_m}) = f_{\mathcal{I}}(\mathbb{1}) \tag{2.16}$$

Here we view $\mathbb{1}^{\pi_1}, \dots, \mathbb{1}^{\pi_m}$ (and later $\mathbf{g}^{\pi_1}, \dots, \mathbf{g}^{\pi_m}$) as labelings for the instance $\mathcal{I}$, while $\mathbf{g}$ is a mapping in $\mathcal{O}^{(m\to m)}$ acting on the first m labelings coordinate-wise. We claim that $\mathbf{g}(\mathbb{1}^{\pi_1}, \dots, \mathbb{1}^{\pi_m}) = (\mathbf{g}^{\pi_1}, \dots, \mathbf{g}^{\pi_m})$ for each $\mathbf{g} = (g_1, \dots, g_m) \in \mathrm{supp}(\omega)$. Indeed, we need to show that $g_j(\mathbb{1}^{\pi_1}, \dots, \mathbb{1}^{\pi_m}) = \mathbf{g}^{\pi_j}$ for each $j \in [m]$. Let us prove this for coordinate $(i,z) \in \mathcal{V}$. We can write

$$\begin{aligned} g_j(\mathbb{1}^{\pi_1}(i,z), \dots, \mathbb{1}^{\pi_m}(i,z)) &= g_j(e^i(z), \dots, e^{i+m-1}(z)) \\ &= g_j(z^{\pi_i}) = g_{\pi_i(j)}(z) = \mathbf{g}(i+j-1,z) = \mathbf{g}^{\pi_j}(i,z) \end{aligned}$$

which proves the claim. We can now rewrite (2.16) as follows:

$$\sum_{\mathbf{g}\in\mathrm{supp}(\omega)} \omega(\mathbf{g}) f_{\mathcal{I}}^m(\mathbf{g}^{\pi_1}, \dots, \mathbf{g}^{\pi_m}) \le f_{\mathcal{I}}(\mathbb{1}) \tag{2.17}$$

Using (2.13c) and the fact that $f_{\mathcal{I}}(\mathbf{g}) = f_{\mathcal{I}'}(\mathbf{g})$ for each $\mathbf{g} \in \mathrm{supp}(\omega) = \mathbb{G}$, we obtain

$$\sum_{\mathbf{g}\in\mathrm{supp}(\omega)} \omega(\mathbf{g}) f_{\mathcal{I}'}(\mathbf{g}) \le f_{\mathcal{I}'}(\mathbb{1}) \tag{2.18}$$

Since $\mathbb{1} \in \arg\min f_{\mathcal{I}'}$, we conclude that $\mathbb{G} = \mathrm{supp}(\omega) \subseteq \arg\min f_{\mathcal{I}'}$. Therefore, $\arg\min f_{\mathcal{I}'} = \mathbb{G}$.

We can finally prove Theorem 2.5.3. We define function $f \in \langle\Gamma\rangle$ with m variables as follows:

$$f(x) = \min_{\mathbf{g}\in\mathcal{O}^{(m\to m)}:\mathbf{g}(\hat{x})=x} f_{\mathcal{I}'}(\mathbf{g}) \qquad \forall x \in D^m$$

Consider tuple $x \in D^m$. We have $x \in \arg\min f$ if and only if there exists $\mathbf{g} \in \arg\min f_{\mathcal{I}'} = \mathbb{G}$ with $\mathbf{g}(\hat{x}) = x$. The latter condition holds if and only if $x \in \mathbb{G}(\hat{x})$. □

2.6 Proof of Theorem 2.3.4

We will prove the following result.

Theorem 2.6.1. *Assume that one of the following holds:*

(a) $m = 2$ *and* Γ *admits a cyclic fractional polymorphism of arity at least 2.*

(b) $m \geq 3$ *and* Γ *admits a symmetric fractional polymorphism of arity* $m - 1$*.*

Let $f \in \langle\Gamma\rangle$ *be a function of arity* m *with* $\arg\min f = \mathbb{G}(\hat{x})$*, where* $\hat{x} \in Range_1(\mathbb{G}^*)$*. Then for every distinct pair of indices* $i, j \in [m]$ *there exists* $x \in \arg\min f$ *with* $x_i = x_j$*.*

We claim that this will imply Theorem 2.3.4. Indeed, we can use the following observation.

Proposition 2.6.2. *Suppose that* $\hat{x} \in Range_1(\mathbb{G}^*)$*, and there exists* $x \in \mathbb{G}(\hat{x})$ *with* $x_i = x_j$ *for some* $i, j \in [m]$*. Then* $\hat{x}_i = \hat{x}_j$*.*

Proof. By Proposition 2.4.2(b), there exists $\mathbf{g} \in \mathbb{G}$ such that $\mathbf{g}(x) = \hat{x}$. Let π be the permutation of $[m]$ that swaps i and j. By the choice of x, we have $x^\pi = x$. We can write $\hat{x}_j = g_j(x) = g_{\pi(i)}(x) = g_i(x^\pi) = g_i(x) = \hat{x}_i$. This proves the claim. □

Corollary 2.6.3. *If the precondition of Theorem 2.6.1 holds, then* Γ *admits a symmetric fractional polymorphism of arity* m*.*

Proof. Using Theorem 2.5.3, Theorem 2.6.1 and Proposition 2.6.2, we conclude that for any $\hat{x} \in Range_1(\mathbb{G}^*)$ we have $\hat{x}_1 = \ldots = \hat{x}_m$. Indeed, by Theorem 2.5.3 there exists a function $f \in \langle\Gamma\rangle$ with $\mathbb{G}(\hat{x}) = \arg\min f$. Theorem 2.6.1 implies that the precondition of Proposition 2.6.2 holds for any distinct pair of indices $i, j \in [m]$, and therefore $\hat{x}_i = \hat{x}_j$.

By Theorem 2.4.3, there exists a generalized fractional polymorphism ρ of Γ of arity $m \to m$ with $\mathrm{supp}(\rho) = \mathbb{G}^*$. Vector $\sum_{\mathbf{g}=(g_1,\ldots,g_m)\in\mathbb{G}^*} \rho(\mathbf{g})\frac{1}{m}[\chi_{g_1} + \ldots + \chi_{g_m}]$ is then an m-ary fractional polymorphism of Γ; all operations in its support are symmetric because $\mathbb{G}^* \subseteq \Omega$ and $\hat{x}_1 = \ldots = \hat{x}_m$ for any $\hat{x} \in Range_1(\mathbb{G}^*)$. □

It remains to prove Theorem 2.6.1. A proof of parts (a) and (b) of Theorem 2.6.1 is given later in this section. In both parts we will need the following result; it exploits the fact that Γ is block-finite.

Lemma 2.6.4. *Suppose that* $\hat{x} \in Range_1(\mathbb{G}^*)$*,* $x \in \mathbb{G}(\hat{x})$ *and* f *is an* m*-ary function in* $\langle\Gamma\rangle$ *with* $\arg\min f = \mathbb{G}(\hat{x})$*. Then* $(a, \ldots, a) \in \operatorname{dom} f$ *for any* $a \in \{x_1, \ldots, x_m\}$*.*

Proof. We say that a tuple $z \in D^m$ is *proper* if $z_1, \ldots, z_m \in D_v$ for some $v \in V$. We will show that x is proper; the lemma will then follow from condition (a) from the definition of a block-finite language and the fact that $x \in \operatorname{dom} f$.

Fix an arbitrary element $a \in D$, and define mapping $\mathbf{g} \in \mathcal{O}^{(m\to m)}$ as follows:

$$\mathbf{g}(z) = \begin{cases} z & \text{if } z \text{ is proper} \\ (a, \ldots, a) & \text{otherwise} \end{cases}$$

We claim that $\mathbf{g} \in \Omega$. Indeed, consider $z \in D^m$. If $\mathbf{g}(z) = z$, the condition (2.9) holds trivially. Otherwise, we can easily check that

$$\mathbf{g}^\pi(z) = (a, \ldots, a)^\pi = (a, \ldots, a) = \mathbf{g}(z^\pi)$$

and so the condition (2.9) holds either way.

Let us now show that the vector $\rho = \chi_{\mathbf{g}}$ is a generalized fractional polymorphism of Γ of arity $m \to m$. Checking inequality (2.6) for binary equality relation $f = (=_D)$ is straighforward. Consider function $f \in \Gamma - \{=_D\}$. Since Γ is block-finite, we have $\operatorname{dom} f \subseteq D_{v_1} \times \ldots \times D_{v_n}$ for some $v_1, \ldots, v_n \in V$. This implies that for any $x \in [\operatorname{dom} f]^m$ we have $\mathbf{g}(x) = x$ (this can be checked coordinate-wise). Therefore, we have an equality in (2.6).

By the results above we obtain that $\mathbf{g} \in \mathbb{G}$. We are now ready to prove that x is proper. Suppose that this is not true, then $\mathbf{g}(x) = (a, \ldots, a)$. We have $\hat{x} \in Range_1(\mathbb{G}^*)$ and $x \in \mathbb{G}(\hat{x})$, so by Proposition 2.4.2(a) we conclude that $x \in Range_1(\mathbb{G}^*)$. We also have $(a, \ldots, a) \in \mathbb{G}(x)$, so Proposition 2.6.2 gives that $x_1 = \ldots = x_m$. This means that x is proper, which contradicts the earlier assumption. □

Case $m = 2$: proof of Theorem 2.6.1(a)

We start with the following observation.

Proposition 2.6.5. *If* $(a, b) \in \mathbb{G}(\hat{x})$ *then* $(b, a) \in \mathbb{G}(\hat{x})$*.*

Proof. Consider mapping $\bar{\mathbb{1}} = (e_2^2, e_2^1)$, where $e_2^k \in \mathcal{O}^{(2)}$ is the projection to the the k-th variable. It can be checked that $\bar{\mathbb{1}} \in \Omega$, and $\chi_{\bar{\mathbb{1}}}$ is a generalized fractional polymorphism of Γ of arity $2 \to 2$. Therefore, $\bar{\mathbb{1}} \in \mathbb{G}$.

We have $(a, b) = \mathbf{g}(\hat{x})$ for some $\mathbf{g} \in \mathbb{G}$. We also have $(b, a) = (\bar{\mathbb{1}} \circ \mathbf{g})(\hat{x})$ and $\bar{\mathbb{1}} \circ \mathbf{g} \in \mathbb{G}$, and therefore $(b, a) \in \mathbb{G}(\hat{x})$. □

Denote $A = \{x_1 \mid x \in \mathbb{G}(\hat{x})\} \subseteq D$, and let a be an element in A that minimizes $f(a, a)$. Note that $(a, a) \in \operatorname{dom} f$ by Lemma 2.6.4. Condition $\arg\min f = \mathbb{G}(\hat{x})$ and Proposition 2.6.5 imply that $(a, b), (b, a) \in \arg\min f$ for some $b \in A$. By assumption, Γ admits a cyclic fractional polymorphism ν of some arity $r \geq 2$. Let us apply it to tuples $(a, b), (b, a), (a, a), \ldots, (a, a)$, where (a, a) is repeated $r - 2$ times:

$$\sum_{h \in \operatorname{supp}(\nu)} \nu(h) f(h(a, b, a, \ldots, a), h(b, a, a, \ldots, a)) \leq \frac{2}{r} f(a, b) + \frac{r-2}{r} f(a, a) \tag{2.19}$$

We have $h(a, b, a, \ldots, a) = h(b, a, a, \ldots, a)$ since ν is cyclic; denote this element as a_h. We claim that $a_h \in A$ for any $h \in \operatorname{supp}(\nu)$. Indeed, consider a unary function $u_A(x_1) = \min_{x_2} f(x_1, x_2)$. It can be checked that $\arg\min u_A = A$. Then the presence of u_A in $\langle\Gamma\rangle$ implies that after applying ν to $(a, b, a, \ldots, a)$ one gets

$$\sum_{h \in \operatorname{supp}(\nu)} \nu(h) u_A(a_h) \leq \frac{r-1}{r} u_A(a) + \frac{1}{r} u_A(b) = \min u_A$$

and thus indeed $a_h \in \arg\min u_A = A$ for any $h \in \operatorname{supp}(\nu)$.

By the choice of a we have $f(a, a) \leq f(a_h, a_h)$ for any $h \in \operatorname{supp}(\nu)$. From (2.19) we thus get

$$f(a, a) \leq \frac{2}{r} f(a, b) + \frac{r-2}{r} f(a, a) \tag{2.20}$$

and so $f(a, a) \leq f(a, b)$, implying $(a, a) \in \arg\min f$.

Case $m \geq 3$: proof of Theorem 2.6.1(b)

We define binary function $\bar{f} \in \langle\Gamma\rangle$ as follows: $\bar{f}(a, b) = \min_{x \in D^m : x_i = a, x_j = b} f(x)$.

If $z = (z_1, \ldots, z_m)$ is some sequence of size m and k is an index in $[m]$ then we will use z_{-k} to denote the subsequence of z of size $m - 1$ obtained by deleting the k-th

element.

Let $\tilde{\omega}$ be a symmetric fractional polymorphism of Γ of arity $m-1$. Following the construction in [51], we define graph $(\tilde{\mathbb{G}}, \tilde{E})$ as described in Section 2.4, starting with the distribution $\tilde{\omega}$ where for $s \in \operatorname{supp}(\tilde{\omega})$ mapping $\mathbb{1}^s \in \mathcal{O}^{(m\to m)}$ is defined as follows:

$$\mathbb{1}^s(x) = (s(x_{-1}), \ldots, s(x_{-m})) \qquad \forall x \in D^m$$

It can be checked that if $\mathbf{g} = (g_1, \ldots, g_m) \in \tilde{\mathbb{G}}$ and $s \in \operatorname{supp}(\tilde{\omega})$ then $\mathbf{g}^s = (s \circ \mathbf{g}_{-1}, \ldots, s \circ \mathbf{g}_{-m})$. It can also be checked that condition (2.7b) holds for any $f \in \Gamma$: it corresponds to the fractional polymorphism $\tilde{\omega}$ applied to $m-1$ tuples $x_{-i} \in [\operatorname{dom} f]^{m-1}$.

Proposition 2.6.6. *There holds* $\tilde{\mathbb{G}} \subseteq \mathbb{G}$.

Proof. We claim that $\mathbb{1}^s \in \Omega$ for any $s \in \operatorname{supp}(\tilde{\omega})$. Indeed, for a permutation π of $[m]$ and $x \in D^m$ we can write

$$\mathbb{1}^s(x^\pi) = (s(x^\pi_{-1}), \ldots, s(x^\pi_{-m})) = (s(x_{-\pi(1)}), \ldots, s(x_{-\pi(m)})) = (\mathbb{1}^s)^\pi(x),$$

where the second equality uses that s is symmetric. Since each $\mathbf{g} \in \tilde{\mathbb{G}}$ has the form $\mathbf{g} = \mathbb{1}^{s_k} \circ \ldots \circ \mathbb{1}^{s_1}$ for some $s_1, \ldots, s_k \in \operatorname{supp}(\tilde{\omega})$ and Ω is closed under composition by Proposition 2.5.1, we get $\tilde{\mathbb{G}} \subseteq \Omega$.

Applying Theorem 2.4.3 (with $\tilde{\mathbb{G}}$ as both $\mathbb{G}$ and $\widehat{\mathbb{G}}$), we obtain a generalized fractional polymorphism $\tilde{\rho}$ with $\operatorname{supp}(\tilde{\rho}) = \tilde{\mathbb{G}} \subseteq \Omega$. By maximality of ω we get the desired $\tilde{\mathbb{G}} = \operatorname{supp}(\tilde{\rho}) \subseteq \operatorname{supp}(\omega) = \mathbb{G}$. □

For each $\mathbf{g} \in \tilde{\mathbb{G}}$ and $k \in [m]$ let us define labeling $x^{[\mathbf{g}k]} \in D^2$ as follows: set $x = \mathbf{g}(\hat{x})$, and then

- If $k = i$, set $x^{[\mathbf{g}k]} = (x_i, x_j)$. We have $x^{[\mathbf{g}i]} \in \arg\min \bar{f}$ since $x \in \mathbb{G}(\hat{x}) = \arg\min f$.
- If $k = j$, set $x^{[\mathbf{g}k]} = (x_j, x_i)$.
- If $k \neq i$ and $k \neq j$, set $x^{[\mathbf{g}k]} = (x_k, x_k)$. We have $x^{[\mathbf{g}k]} \in \operatorname{dom} \bar{f}$ by Lemma 2.6.4.

Proposition 2.6.7. *Suppose that* $\mathbf{g} \in \tilde{\mathbb{G}}$ *and* $\mathbf{g}^s = \mathbf{h}$ *where* $s \in \operatorname{supp}(\tilde{\omega})$ *(so that* $\mathbf{h} \in \tilde{\mathbb{G}}$*). Then*

$$\mathbb{1}^s(x^{[\mathbf{g}1]}, \ldots, x^{[\mathbf{g}m]}) = (x^{[\mathbf{h}1]}, \ldots, x^{[\mathbf{h}m]}).$$

Proof. Denote $x = \mathbf{g}(\hat{x})$ and $y = \mathbf{h}(\hat{x})$. We have $y = \mathbb{1}^s(x)$, or $y_k = s(x_{-k})$ for any $k \in [m]$. Also,

$$x^{[\mathbf{g}k]} = \begin{cases} (x_i, x_j) & \text{if } k = i \\ (x_j, x_i) & \text{if } k = j \\ (x_k, x_k) & \text{if } k \neq i \text{ and } k \neq j \end{cases} \qquad x^{[\mathbf{h}k]} = \begin{cases} (y_i, y_j) & \text{if } k = i \\ (y_j, y_i) & \text{if } k = j \\ (y_k, y_k) & \text{if } k \neq i \text{ and } k \neq j \end{cases}$$

It can be checked coordinate-wise (using that s is symmetric) that

$$x^{[\mathbf{h}k]} = s((x^{[\mathbf{g}1]}, \dots, x^{[\mathbf{g}m]})_{-k})$$

for any $k \in [m]$. This gives the claim. □

Denote $\tilde{\mathbb{G}}^* = \bigcup_{\mathbb{H} \in \mathsf{Sinks}(\tilde{\mathbb{G}}, \tilde{E})} \mathbb{H} \subseteq \tilde{\mathbb{G}}$. Let us fix an arbitrary $\tilde{\mathbf{g}} \in \tilde{\mathbb{G}}^*$, and define $\tilde{x} = (x^{[\tilde{\mathbf{g}}1]}, \dots, x^{[\tilde{\mathbf{g}}m]}) \in [D^2]^m$.

Proposition 2.6.8. *For any $\mathbf{g} \in \tilde{\mathbb{G}}$ there holds $\mathbf{g} \circ \tilde{\mathbf{g}} \in \tilde{\mathbb{G}}$. Furthermore, $\mathbf{g}(\tilde{x}) = (x^{[(\mathbf{g} \circ \tilde{\mathbf{g}})1]}, \dots, x^{[(\mathbf{g} \circ \tilde{\mathbf{g}})m]})$.*

Proof. The first claim is by Proposition 2.4.1(a); let us show the second one. Let $d(\mathbb{1}, \mathbf{g})$ be the shortest distance from $\mathbb{1}$ to $\mathbf{g}$ in the graph $(\tilde{\mathbb{G}}, \tilde{E})$. (By the definition of this graph, we have $0 \leq d(\mathbb{1}, \mathbf{g}) < \infty$ for any $\mathbf{g} \in \tilde{\mathbb{G}}$, and $\mathbb{1} \in \tilde{\mathbb{G}}$.) We will use induction on $d(\mathbb{1}, \mathbf{g})$. The base case $d(\mathbb{1}, \mathbf{g}) = 0$ (i.e. $\mathbf{g} = \mathbb{1}$) holds by construction. Suppose that the claim holds for all mappings $\mathbf{g} \in \tilde{\mathbb{G}}$ with $d(\mathbb{1}, \mathbf{g}) = k \geq 0$, and consider mapping $\mathbf{h} \in \tilde{\mathbb{G}}$ with $d(\mathbb{1}, \mathbf{h}) = k + 1$. There must exist mapping $\mathbf{g} \in \tilde{\mathbb{G}}$ and operation $s \in \mathrm{supp}(\tilde{\omega})$ such that $d(\mathbb{1}, \mathbf{g}) = k$ and $\mathbf{g}^s = \mathbf{h}$. Observe that $(\mathbf{g} \circ \tilde{\mathbf{g}})^s = \mathbb{1}^s \circ \mathbf{g} \circ \tilde{\mathbf{g}} = \mathbf{g}^s \circ \tilde{\mathbf{g}} = \mathbf{h} \circ \tilde{\mathbf{g}}$. We can thus write

$$\mathbf{h}(\tilde{x}) = (\mathbb{1}^s \circ \mathbf{g})(\tilde{x}) = \mathbb{1}^s(\mathbf{g}(\tilde{x})) \overset{(1)}{=} \mathbb{1}^s(x^{[(\mathbf{g} \circ \tilde{\mathbf{g}})1]}, \dots, x^{[(\mathbf{g} \circ \tilde{\mathbf{g}})m]}) \overset{(2)}{=} (x^{[(\mathbf{h} \circ \tilde{\mathbf{g}})1]}, \dots, x^{[(\mathbf{h} \circ \tilde{\mathbf{g}})m]})$$

where (1) holds by the induction hypothesis and (2) is by Proposition 2.6.7. □

Proposition 2.6.9. *There holds $\tilde{x} \in Range_2(\tilde{\mathbb{G}}^*) \cap [\mathrm{dom}\, \bar{f}]^m$.*

Proof. By Proposition 2.4.1(b) there exists $\mathbf{g} \in \tilde{\mathbb{G}}^*$ with $\mathbf{g} \circ \tilde{\mathbf{g}} = \tilde{\mathbf{g}}$. Using Proposition 2.6.8, we can write $\mathbf{g}(\tilde{x}) = (x^{[(\mathbf{g}\circ\tilde{\mathbf{g}})1]}, \ldots, x^{[(\mathbf{g}\circ\tilde{\mathbf{g}})m]}) = (x^{[\tilde{\mathbf{g}}1]}, \ldots, x^{[\tilde{\mathbf{g}}m]}) = \tilde{x}$. This shows that $\tilde{x} \in Range_2(\tilde{\mathbb{G}}^*)$.

Now let us show $x^{[\tilde{\mathbf{g}}k]} \in \operatorname{dom} \bar{f}$ for each $k \in [m]$. It suffices to prove it for $k = j$ (for other indices k the claim holds by construction). We have $\tilde{\mathbf{g}} \in \mathbb{H}$ for some strongly connected component $\mathbb{H} \in \mathsf{Sinks}(\tilde{\mathbb{G}}, \tilde{E})$. There is a path from $\tilde{\mathbf{g}}$ to $\tilde{\mathbf{g}}$ in $(\mathbb{H}, E[\mathbb{H}])$, therefore there exists mapping $\mathbf{h} \in \mathbb{H} \subseteq \tilde{\mathbb{G}}^*$ and $s \in \operatorname{supp}(\tilde{\omega})$ with $\mathbf{h}^s = \tilde{\mathbf{g}}$. Define $x = (x^{[\mathbf{h}1]}, \ldots, x^{[\mathbf{h}m]})$, then by Proposition 2.6.7 we have $\mathbb{1}^s(x) = \tilde{x}$. In particular, $x^{[\tilde{\mathbf{g}}j]} = s(x_{-j})$. Also, we have $x_{-j} \in [\operatorname{dom} \bar{f}]^{m-1}$ by construction. Since Γ admits $\tilde{\omega}$ and $s \in \operatorname{supp}(\tilde{\omega})$, we conclude that $x^{[\tilde{\mathbf{g}}j]} \in \operatorname{dom} \bar{f}$. □

Pick $k \in [m] - \{i, j\}$. By Theorem 2.4.4(b) we obtain that there exists a probability distribution λ over $\tilde{\mathbb{G}}^*$ such that $\bar{f}_i^{\lambda}(\tilde{x}) = \bar{f}_k^{\lambda}(\tilde{x})$. Using Proposition 2.6.8, we can rewrite this condition as

$$\sum_{\mathbf{g} \in \tilde{\mathbb{G}}^*} \lambda_{\mathbf{g}} \bar{f}(x^{[(\mathbf{g}\circ\tilde{\mathbf{g}})i]}) = \sum_{\mathbf{g} \in \tilde{\mathbb{G}}^*} \lambda_{\mathbf{g}} \bar{f}(x^{[(\mathbf{g}\circ\tilde{\mathbf{g}})k]})$$

Every tuple $x^{[(\mathbf{g}\circ\tilde{\mathbf{g}})i]}$ on the LHS belongs to $\arg\min \bar{f}$. Therefore, every tuple $x^{[(\mathbf{g}\circ\tilde{\mathbf{g}})k]}$ on the RHS corresponding to mapping $\mathbf{g} \in \tilde{\mathbb{G}}^*$ with $\lambda_{\mathbf{g}} > 0$ also belongs to $\arg\min \bar{f}$.

We proved that there exists $x \in \arg\min f$ with $x_i = x_j$.

3 Generalizing Edmonds' Algorithm

3.1 Preliminaries

Let us begin by giving a specialized definition Boolean CSP, so that we do not need to rely on (much) more general Definitions 2.1.1, 2.1.2, 2.1.3.

Definition 3.1.1. A Boolean CSP *instance* I is a pair $(V, \mathcal{C})$ where V is the set of *variables* and $\mathcal{C}$ the set of *constraints* of I. A k-ary constraint $C \in \mathcal{C}$ is a pair (σ, R_C) where $\sigma \subseteq V$ is a set of size k (called the *scope* of C) and $R_C \subseteq \{0,1\}^\sigma$ is a relation on $\{0,1\}$. A solution to I is a mapping $\hat{f} : V \to \{0,1\}$ such that for every constraint $C = (\sigma, R_C) \in \mathcal{C}$, $\hat{f}$ restricted to σ lies in R_C.

If all constraint relations of I come from a set of relations Γ (called the *constraint language*), we say that I is a Γ-instance. For Γ fixed, we will denote the problem of deciding if a Γ-instance given on input has a solution by $\mathrm{CSP}(\Gamma)$.

Note that the above definition is not fully general in the sense that it *does not allow one variable to occur multiple times in a constraint*; we have chosen to define Boolean CSP in this way to make our notation a bit simpler. This can be done without loss of generality as long as Γ contains the equality constraint (i.e. $\{(0,0),(1,1)\}$): If a variable, say v, occurs in a constraint multiple times, we can add extra copies of v to our instance and join them together by the equality constraint to obtain a slightly larger instance that satisfies our definition.

For brevity of notation, we will often not distinguish a constraint $C \in \mathcal{C}$ from its constraint relation R_C; the exact meaning of C will always be clear from the context.

Even though in principle different constraints can have the same constraint relation, our notation would get cumbersome if we wrote R_C everywhere.

The main point of interest is classifying the computational complexity of $\mathrm{CSP}(\Gamma)$. Constraints of an instance are specified by lists of tuples in the corresponding relations and thus those lists are considered to be part of the input. We will say that Γ contains the unary constant relations if $\{(0)\}, \{(1)\} \in \Gamma$ (these relations allow us to fix the value of a certain variable to 0 or 1).

For Boolean CSPs (where variables are assigned Boolean values), the complexity classification of $\mathrm{CSP}(\Gamma)$ due to Schaefer has been known for a long time [63]. Our main focus is on restricted forms of the CSP. In particular, we are interested in structural restrictions, i.e. in restrictions on the constraint network. With such limitation the Boolean case becomes complicated again. (As a side note, we expect similar problems for larger domains to be very hard to classify. For example, Dvořák and Kupec note that one can encode four-coloring of planar graphs as a class of planar CSPs. Such CSPs are always satisfiable but for a highly non-trivial reason, namely the Four color theorem.)

A natural structural restriction would be to limit the number of constraints in whose scope a variable can lie. When $k \geq 3$ and Γ contains all unary constants, then $\mathrm{CSP}(\Gamma)$ with each variable in at most k constraints is polynomial time equivalent to unrestricted $\mathrm{CSP}(\Gamma)$, see [24, Theorem 2.3]. This leaves instances with at most two occurrences per variable in the spotlight. To make our arguments clearer, we will assume that each variable occurs exactly in two constraints (following [29], we can reduce decision CSP instances with at most two appearances of each variable to instances with exactly two appearances by taking two copies of the instance and identifying both copies of v whenever v is a variable that originally appeared in only a single constraint).

Definition 3.1.2 (Edge CSP)**.** Let Γ be a constraint language. Then the problem $\mathrm{CSP}_{\textsf{EDGE}}(\Gamma)$ is the restriction of $\mathrm{CSP}(\Gamma)$ to those instances in which every variable is present in exactly two constraints.

Perhaps a more natural way to look at an instance I of an edge CSP is to consider a graph whose edges correspond to variables of I and nodes to constraints of I. Constraints (nodes) are incident with variables (edges) they interact with. In this (multi)graph, we are looking for a satisfying Boolean edge labeling. Viewed like this,

edge CSP becomes a counterpart to the usual CSP where variables are typically identified with nodes and constraints with (hyper)edges. The idea of "switching" the role of (hyper)edges and vertices already appeared in the counting CSP community under the name Holant problems [13].

This type of CSP is sometimes called "binary CSP" in the literature [26]. However, this term is very commonly used for CSPs whose all constraints have arity at most two [69]. In order to resolve this confusion (and for the reasons described in the previous paragraph), we propose the term "edge CSP".

As we said above, we we will only consider Boolean edge CSP, often omitting the word "Boolean" for space reasons. The following Boolean-specific definitions will be useful to us:

Definition 3.1.3. Let $f\colon V \to \{0,1\}$ and $v \in V$. We will denote by $f \oplus v$ the mapping $V \to \{0,1\}$ that agrees with f on $V \setminus \{v\}$ and has value $1 - f(v)$ on v. For a set $S = \{s_1, \dots, s_k\} \subseteq V$ we let $f \oplus S = f \oplus s_1 \oplus \cdots \oplus s_k$. Also for $f, g\colon V \to \{0,1\}$ let $f \,\Delta\, g \subseteq V$ be the set of variables v for which $f(v) \neq g(v)$.

Definition 3.1.4. Let V be a set. A non-empty subset M of $\{0,1\}^V$ is called a Δ-*matroid* if whenever $f, g \in M$ and $v \in f \,\Delta\, g$, then there exists $u \in f \,\Delta\, g$ such that $f \oplus \{u, v\} \in M$. If moreover, the parity of the number of ones over all tuples of M is constant, we have an *even* Δ-*matroid* (note that in that case we never have $u = v$ so $f \oplus \{u, v\}$ reduces to $f \oplus u \oplus v$).

The strongest hardness result on Boolean edge CSP is from Feder.

Theorem 3.1.5 ([29])**.** *If Γ is a constraint language containing unary constant relations such that $\mathrm{CSP}(\Gamma)$ is NP-Hard and there is $R \in \Gamma$ which is not a Δ-matroid, then $\mathrm{CSP}_{\mathsf{EDGE}}(\Gamma)$ is NP-Hard.*

Tractability was shown for special classes of Δ-matroids, namely binary [34, 24], co-independent [29], compact [41], and local [24] (see the definitions in the respective papers). All the proposed algorithms are based on variants of searching for augmenting paths. In this work we propose a more general algorithm that involves both augmentations and contractions. In particular, we prove the following.

Theorem 3.1.6. *If Γ contains only even Δ-matroid relations, then $\mathrm{CSP}_{\mathsf{EDGE}}(\Gamma)$ can be solved in polynomial time.*

Our algorithm will in fact be able to solve even a certain optimization version of the edge CSP (corresponding to finding a maximum matching). This is discussed in detail in Section 3.4.

In Section 3.6 we show that if a class of Δ-matroids is *efficiently coverable*, then it defines a tractable CSP. The whole construction is similar to, but more general than, $\mathcal{C}$-zebra Δ-matroids introduced in [30]. We note here also that the class of coverable Δ-matroids is natural in the sense of being closed under direct products and identifying variables (in other words, gadget constructions).

Definition 3.1.7. Let M be a Δ-matroid. We say that $\alpha, \beta \in M$ are *even-neighbors* if there exist distinct variables $u, v \in V$ such that $\beta = \alpha \oplus u \oplus v$ and $\alpha \oplus u \notin M$. We say we can reach $\gamma \in M$ from $\alpha \in M$ if there is a chain $\alpha = \beta_0, \beta_1, \ldots, \beta_n = \gamma$ where each pair β_i, β_{i+1} are even-neighbors.

Definition 3.1.8. We say that M is *coverable* if for every $\alpha \in M$ there exists M_α such that:

1. M_α is an even Δ-matroid (over the same ground set as M),
2. M_α contains all $\beta \in M$ that can be reached from α (including α itself),
3. whenever $\gamma \in M$ can be reached from α and $\gamma \oplus u \oplus v \in M_\alpha \setminus M$, then $\gamma \oplus u, \gamma \oplus v \in M$.

It is easy to see that every even Δ-matroid M is coverable. We simply take $M_\alpha = M$ for every $\alpha \in M$.

In our algorithm, we will need to have access to the sets M_α, so we need to assume that all our Δ-matroids, in addition to being coverable, come from a class of Δ-matroids where the sets M_α can be determined quickly. This is what *efficiently coverable* means (for a formal definition see Definition 3.6.1).

The following theorem is a strengthening of a result from [30]:

Theorem 3.1.9. *If Γ contains only efficiently coverable Δ-matroid constraints, then $\mathrm{CSP}_{\mathsf{EDGE}}(\Gamma)$ can be solved in polynomial time.*

Again, the algorithm will solve even a certain optimization version of the edge CSP.

As we show in Appendix A.5, efficiently coverable Δ-matroid classes include numerous known tractable classes of Δ-matroids: $\mathcal{C}$-zebra Δ-matroids [30] for any $\mathcal{C}$ subclass of even Δ-matroids (where we assume, just like in [30], that we are given the zebra representations on input) as well as co-independent [29], compact [41], local [24], and binary [34, 24] Δ-matroids. To our best knowledge these are all the known tractable classes and according to [24] the classes other than $\mathcal{C}$-zebras are pairwise incomparable.

3.2 Implications

In this section we explain how our result implies full complexity classification of planar Boolean CSPs.

Definition 3.2.1. Let Γ be a constraint language. Then $\mathrm{CSP}_{\text{PLANAR}}(\Gamma)$ is the restriction of $\mathrm{CSP}(\Gamma)$ to the set of instances for which there exists a planar graph $G(V, E)$ such that $v_1, \ldots, v_k$ is a face of G (with nodes listed in counter-clockwise order) if and only if there is a unique constraint imposed on the tuple of variables $(v_1, \ldots, v_k)$.

It is also noted in [26] that checking whether an instance has a planar representation, as well as finding it, can be done efficiently (see e.g.. [38]) and hence it does not matter if we are given a planar drawing of G as a part of the input or not. The planar restriction does lead to new tractable cases, for example planar NAE-3-SAT (Not-All-Equal 3-Satisfiability) [60]. For more results on planar CSPs, particularly related to approximation, see [22].

Definition 3.2.2. A relation R is called *self-complementary* if for all $T \in \{0,1\}^n$ we have $T \in R$ if and only if $T \oplus \{1, 2, \ldots, n\} \in R$ (i.e. R is invariant under simultaneous flipping of all entries of a tuple).

Definition 3.2.3. For a tuple of Boolean variables $T = (x_1, \ldots, x_n)$, let $dT = (x_1 \oplus x_2, \ldots, x_n \oplus x_1)$. For a relation R and a set of relations Γ, let $dR = \{dT \colon T \in R\}$ and $d\Gamma = \{dR \colon R \in \Gamma\}$.

Since self-complementary relations don't change when we flip all their coordinates, we can describe a self-complementary relation by looking at the differences of neighboring coordinates; this is exactly the meaning of dR. Note that these differences are realized over edges of the given planar graph.

Knowing this, it is not so difficult to imagine that via switching to the planar dual of G, one can reduce a planar CSP instance to some sort of edge CSP instance. This is in fact part of the following theorem from [26]:

Theorem 3.2.4. *Let Γ be such that $\mathrm{CSP}(\Gamma)$ is NP-Hard. Then:*

1. *If there is $R \in \Gamma$ that is not self-complementary, then $\mathrm{CSP}_{\mathsf{PLANAR}}(\Gamma)$ is NP-Hard.*
2. *If every $R \in \Gamma$ is self-complementary and there exists $R \in \Gamma$ such that dR is not even Δ-matroid, then $\mathrm{CSP}_{\mathsf{PLANAR}}(\Gamma)$ is NP-Hard.*
3. *If every $R \in \Gamma$ is self-complementary and whose dR is an even Δ-matroid, then $\mathrm{CSP}_{\mathsf{PLANAR}}(\Gamma)$ is polynomial-time reducible to*

$$\mathrm{CSP}_{\mathsf{EDGE}}(d\Gamma \cup \{EVEN_1, EVEN_2, EVEN_3\})$$

where $EVEN_i = \{(x_1, \dots x_i) \colon x_1 \oplus \cdots \oplus x_i = 0\}$.

Using Theorem 3.1.6, we can finish this classification:

Theorem 3.2.5 (Dichotomy for $\mathrm{CSP}_{\mathsf{PLANAR}}$)**.** *Let Γ be a constraint language. Then $\mathrm{CSP}_{\mathsf{PLANAR}}(\Gamma)$ is solvable in polynomial time if either*

1. *$\mathrm{CSP}(\Gamma)$ is solvable in polynomial time or;*
2. *Γ contains only self-complementary relations R such that dR is an even Δ-matroid.*

Otherwise, $\mathrm{CSP}_{\mathsf{PLANAR}}(\Gamma)$ is NP-Hard.

Proof. By Theorem 3.2.4 the only unresolved case reduces to solving

$$\mathrm{CSP}_{\mathsf{EDGE}}(d\Gamma \cup \{EVEN_1, EVEN_2, EVEN_3\}).$$

Since the relations $EVEN_i$ are even Δ-matroids for every i, this is polynomial-time solvable thanks to Theorem 3.1.6. □

3.3 Even $\triangle$-matroids and matchings

In this section we highlight the similarities and dissimilarities between even Δ-matroid CSPs and matching problems. These similarities will guide us on our way through the rest of the chapter.

Example 3.3.1. *For $n \in \mathbb{N}$ consider the "perfect matching" relation $M_n \subseteq \{0,1\}^n$ containing precisely the tuples in which exactly one coordinate is set to one and all others to zero. Note that M_n is an even Δ-matroid for all n. Then the instance I of $\mathrm{CSP}_{\mathsf{EDGE}}(\{M_n : n \in \mathbb{N}\})$ (represented in Figure 3.1) is equivalent to deciding whether the graph of the instance has a perfect matching (every node is adjacent to precisely one edge with label 1).*

One may also construct an equivalent instance I' by "merging" some parts of the graph (in Figure 3.1 those are X and Y) to single constraint nodes. The constraint relations imposed on the "supernodes" record sets of outgoing edges which can be extended to a perfect matching on the subgraph induced by the "supernode". For example, in the instance I' the constraints imposed on X and Y would be (with variables ordered as in Figure 3.1):

$$X = \{10000, 01000, 00100, 00010, 11001, 10101, 10011\},$$
$$Y = \{001, 010, 100, 111\}.$$

It is easy to check that both X and Y are even Δ-matroids.

One takeaway from this example is that any algorithm that solves edge CSP for the even Δ-matroid case has to work for perfect matchings in graphs as well. Another is the construction of even Δ-matroids X and Y which can be generalized as follows.

Definition 3.3.2 (Matching realizable relations)**.** Let G be a graph and let $v_1, \ldots, v_a \in V(G)$ be distinct nodes of G. For an a-tuple $T = (x_1, \ldots, x_a) \in \{0,1\}^a$, we denote by G_T the graph obtained from G by deleting all nodes v_i such that $x_i = 1$. Then we can define

$$M(G, v_1, \ldots, v_a) = \{T \in \{0,1\}^a : G_T \text{ has a perfect matching}\}.$$

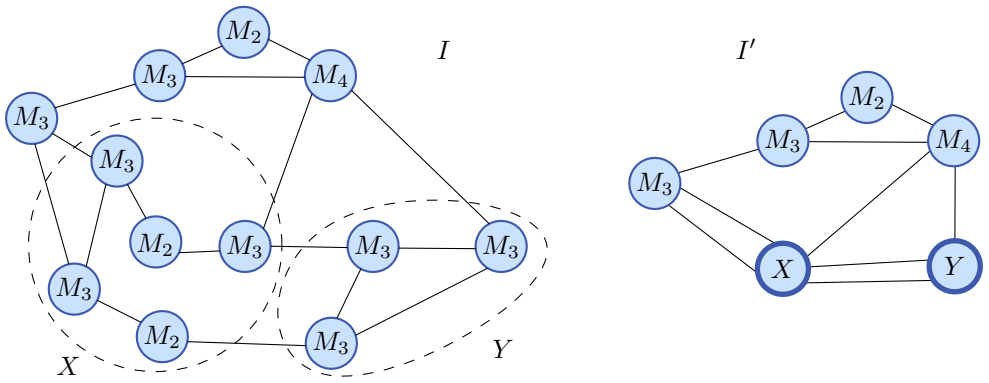

Figure 3.1: On the left we see an instance I that is equivalent to testing for perfect matching of the given graph. On the right is an equivalent instance I' with contracted "supernodes" X and Y.

We say that a relation $R \in \{0,1\}^a$ is *matching realizable* if $R = M(G, v_1, \ldots, v_a)$ for some graph G and nodes $v_1, \ldots, v_a \in V(G)$.

Every matching realizable relation is an even Δ-matroid [26]. Also, it should be clear from the definition and the preceding example that $\mathrm{CSP}_{\mathsf{EDGE}}(\Gamma)$ is tractable if Γ contains only matching realizable relations (assuming we know the graph G and the nodes $v_1, \ldots, v_a$ for each relation): One can simply replace each constraint node with the corresponding graph and then test for existence of perfect matching.

The authors of [26] also verify that every even Δ-matroid of arity at most 5 is matching realizable. However, as we prove in Appendix A.3, this is not true for higher arities.

Proposition 3.3.3. *There exists an even Δ-matroid of arity 6 which is not matching realizable.*

Proposition 3.3.3 shows that we cannot hope to simply replace the constraint nodes by graphs and run the Edmonds' algorithm. The Δ-matroid constraints can exhibit new and more complicated behavior than just matchings in graphs, as we shall soon see. In fact, there is a known exponential lower bound for solving edge CSP with matroid constraints (matroids being special cases of even Δ-matroids) given by oracles (i.e. not lists of tuples) [7], which rules out any polynomial time algorithm that would work in the oracle model. In particular, we are convinced that our method of contracting blossoms can not be significantly simplified while still staying polynomial time computable.

3.4 Algorithm

Setup

We can draw edge CSP instances as constraint graphs: The *constraint graph* $G_I = (V \cup \mathcal{C}, \mathcal{E})$ of I is a bipartite graph with partitions V and $\mathcal{C}$. There is an edge $\{v, C\} \in \mathcal{E}$ if and only if v belongs to the scope of C. Throughout the rest of the chapter we use lower-case letters for variable nodes in V ($u, v, x, y, \ldots$) and upper-case letters for constraint nodes in $\mathcal{C}$ ($A, B, C, \ldots$). Since we are dealing with edge CSP, the degree of each node $v \in V$ in G_I is exactly two and since we don't allow a variable to appear in a constraint twice, G_I has no multiple edges. For such instances I we introduce the following terminology and notation.

Definition 3.4.1. An *edge labeling* of I is a mapping $f : \mathcal{E} \to \{0, 1\}$. For a constraint $C \in \mathcal{C}$ with the scope σ we will denote by $f(C)$ the tuple in $\{0, 1\}^\sigma$ such that $f(C)(v) = f(\{v, C\})$ for all $v \in \sigma$. Edge labeling f will be called *valid* if $f(C) \in C$ for all $C \in \mathcal{C}$.

Variable $v \in V$ is called *consistent* in f if $f(\{v, A\}) = f(\{v, B\})$ for the two distinct edges $\{v, A\}, \{v, B\} \in \mathcal{E}$ of G_I. Otherwise, v is *inconsistent* in f.

A valid edge labeling f is *optimal* if its number of inconsistent variables is minimal among all valid edge labelings of I. Otherwise f is called *non-optimal*.

Note that I has a solution if and only if an optimal edge labeling f of I has no inconsistent variables.

The main theorem we prove is the following strengthening of Theorem 3.1.6.

Theorem 3.4.2. *Given an edge CSP instance I with even Δ-matroid constraints, an optimal edge labeling f of I can be found in time polynomial in $|I|$.*

Walks and blossoms When studying matchings in a graph, paths and augmenting paths are important. We will use analogous objects, called f-walks and augmenting f-walks, respectively.

Definition 3.4.3. A *walk* q of length k in the instance I is a sequence $q_0 C_1 q_1 C_2 \ldots C_k q_k$ where the variables q_{i-1}, q_i lie in the scope of the constraint C_i, and each edge $\{v, C\} \in \mathcal{E}$

is traversed at most once: vC and Cv occur in q at most once, and they do not occur simultaneously.

Note that q can be viewed as a walk in the graph G_I that starts and ends at nodes in V. Since each node $v \in V$ has degree two in G_I, the definitions imply that v can be visited by q at most once, with a single exception: we may have $q_0 = q_k = v$, with $q = vC \dots Dv$ where $C \neq D$. We allow walks of length 0 (i.e. single vertex walks) for formal reasons.

A *subwalk* of q, denoted by $q_{[i,j]}$, is the walk $q_i C_{i+1} \dots C_j q_j$ (again, we need to start and end in a variable). The inverse walk to q, denoted by q^{-1}, is the sequence $q_k C_k \dots q_1 C_1 q_0$. Given two walks p and q such that the last node of p is the first node of q, we define their concatenation pq in the natural way. If $p = \alpha_1 \dots \alpha_k$ and $q = \beta_1 \dots \beta_\ell$ are sequences of nodes of a graph where α_k and β_1 are different but adjacent, we will denote the sequence $\alpha_1 \dots \alpha_k \beta_1 \dots \beta_\ell$ also by pq (or sometimes as p, q).

If f is an edge labeling of I and q a walk in I, we denote by $f \oplus q$ the mapping that takes f and flips the values on all variable-constraint edges encountered in q, i.e.

$$(f \oplus q)(\{v, C\}) = \begin{cases} 1 - f(\{v, C\}) & \text{if } q \text{ contains } vC \text{ or } Cv \\ f(\{v, C\}) & \text{otherwise} \end{cases} \tag{3.1}$$

Definition 3.4.4. Let f be a valid edge labeling of an instance I. A walk

$$q = q_0 C_1 q_1 C_2 \dots C_k q_k$$

with $q_0 \neq q_k$ will be called an *f-walk* if

1. variables $q_1, \dots, q_{k-1}$ are consistent in f, and
2. $f \oplus q_{[0,i]}$ is a valid edge labeling for any $i \in [1, k]$.

If in addition variables q_0 and q_k are inconsistent in f then q will be called an *augmenting f-walk*.

Later we will show that a valid edge labeling f is non-optimal if and only if there exists an augmenting f-walk. Note that one direction is straightforward: If p is an augmenting f-walk, then $f \oplus p$ is valid and has 2 fewer inconsistent variables than f.

Another structure used by the Edmonds' algorithm [27] for matchings is a *blossom*. The precise definition of a blossom in our setting (Definition 3.5.9) is a bit technical. Informally, an f-blossom is a walk $b = b_0 C_1 b_1 C_2 \dots C_k b_k$ with $b_0 = b_k$ such that:

1. variable $b_0 = b_k$ is inconsistent in f while variables $b_1, \dots, b_{k-1}$ are consistent, and
2. $f \oplus b_{[i,j]}$ is a valid edge labeling for any non-empty proper subinterval $[i, j] \subsetneqq [0, k]$,
3. there are no bad shortcuts inside b (we will make this precise later).

Algorithm description

We are given an instance I of edge CSP with even Δ-matroid constraints together with an edge labeling f and we want to either show that f is optimal or improve it. Our algorithm will explore the graph $(V \cup \mathcal{C}, \mathcal{E})$ building a directed forest T. Each variable node $v \in V$ will be added to T at most once. Constraint nodes $C \in \mathcal{C}$, however, can be added to T multiple times. To tell the copies of C apart (and to keep track of the order in which we built T), we will mark each C with a timestamp $t \in \mathbb{N}$; the resulting node of T will be denoted as $C^t \in \mathcal{C} \times \mathbb{N}$. Thus, the forest will have the form $T = (V(T) \cup \mathcal{C}(T), E(T))$ where $V(T) \subseteq V$ and $\mathcal{C}(T) \subseteq \mathcal{C} \times \mathbb{N}$.

The roots of the forest T will be the inconsistent nodes of the instance (for current f); all non-root nodes in $V(T)$ will be consistent. The edges of T will be oriented towards the leaves. Thus, each non-root node $\alpha \in V(T) \cup \mathcal{C}(T)$ will have exactly one parent $\beta \in V(T) \cup \mathcal{C}(T)$ with $\beta\alpha \in E(T)$. For a node $\alpha \in V(T) \cup \mathcal{C}(T)$ let $\mathrm{walk}(\alpha)$ be the the unique path in T from a root to α. Note that $\mathrm{walk}(\alpha)$ is a subgraph of T. Sometimes we will treat walks in T as sequences of nodes in $V \cup \mathcal{C}$ discussed in Section 3.4 (i.e. with timestamps removed); such places should be clear from the context.

We will grow the forest T in a greedy manner as shown in Algorithm 1. The structure of the algorithm resembles that of the Edmonds' algorithm for matchings [27], with the following important distinctions: First, in the Edmonds' algorithm each "constraint node" (i.e. each node of the input graph) can be added to the forest at most once, while in Algorithm 1 some constraints $C \in \mathcal{C}$ can be added to T and "expanded" multiple times (i.e. $E(T)$ may contain edges $C^s u$ and $C^t w$ added at distinct timestamps $s \neq t$). This is because we allow more general constraints. In particular, if C is a "perfect matching"

Algorithm 1 Improving a given edge labeling

Input: Instance I, valid edge labeling f of I.
Output: A valid edge labeling g of I with fewer inconsistent variables than f, or "No" if no such g exists.

1. Initialize T as follows: set timestamp $t = 1$, and for each inconsistent variable $v \in V$ of I add v to T as an isolated root.
2. Pick an edge $\{v, C\} \in \mathcal{E}$ such that $v \in V(T)$ but there is no s such that $vC^s \in E(T)$ or $C^s v \in E(T)$. (If no such edge exists, then output "No" and terminate.)
3. Add new node C^t to T together with the edge vC^t.
4. Let W be the set of all variables $w \neq v$ in the scope of C such that $f(C) \oplus v \oplus w \in C$. For each $w \in W$ do the following (see Figure 3.2):
   (a) If $w \notin V(T)$, then add w to T together with the edge $C^t w$.
   (b) Else if w has a parent of the form C^s for some s, then do nothing.
   (c) Else if v and w belong to different trees in T (i.e. originate from different roots), then we have found an augmenting f-walk. Let $p = \mathrm{walk}(C^t)\,\mathrm{walk}(w)^{-1}$, output $f \oplus p$ and exit.
   (d) Else if v and w belong to the same tree in T, then we have found a blossom. Form a new instance I^b and new valid edge labeling f^b of I^b by *contracting* this blossom. Solve this instance recursively, use the resulting improved edge labeling for I^b (if it exists) to compute an improved valid edge labeling for I, and terminate. All details are given in Sec. 3.4.
5. Increase the timestamp t by 1 and goto step 2.

constraint (i.e. $C = \{(a_1, \ldots, a_k) \in \{0,1\}^k \ : \ a_1 + \ldots + a_k = 1\}$) then Algorithm 1 will expand it at most once. (We will not use this fact, and thus omit the proof.)

Note that even when we enter a constraint node for the second or third time, we "branch out" based on transitions vCw available before the first visit, even though the tuple of C might have changed in the meantime. This could cause one to doubt that Algorithm 1 works at all.

A vague answer to this objection is that we grow T very carefully: While the Edmonds' algorithm does not impose any restrictions on the order in which the forest is grown, we require that all valid children $w \in W$ be added to T simultaneously when exploring edge $\{v, C\}$ in step 4. Informally speaking, this will guarantee that forest T does not have "shortcuts", a property that will be essential in the proofs. The possibility of having shortcuts is something that is not present in graph matchings and is one of the properties

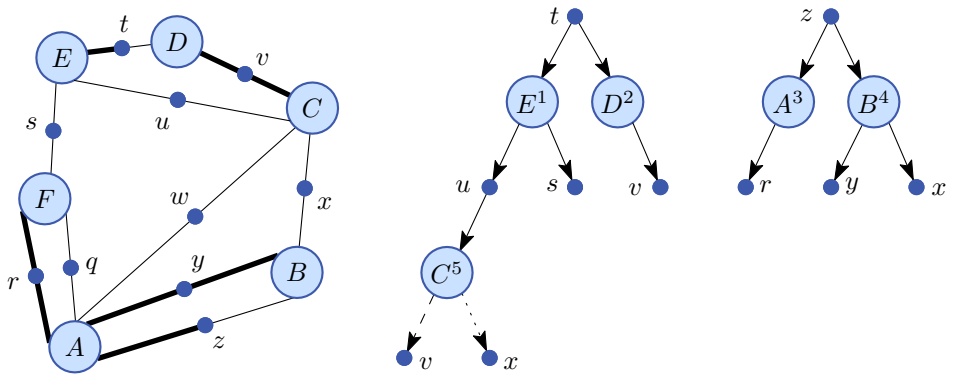

Figure 3.2: A possible run of Algorithm 1 on the instance I' from Example 3.3.1 (with renamed constraint nodes) where the edge labeling f is marked by thick (1) and thin (0) half-edges. We see that the algorithm finds a blossom when it hits the variable v the second time in the same tree. However, had we first processed the transition Cx (which we could have done), we would have found an augmenting path $p = \mathrm{walk}(C^5)\,\mathrm{walk}(x)^{-1}$ (where $\mathrm{walk}(x)^{-1}$ ends in z).

of even Δ-matroids responsible for the considerable length of the correctness proofs.

In the following theorem, we collect all pieces we need to show that Algorithm 1 is correct and runs in polynomial time:

Theorem 3.4.5. *If I is a CSP instance, f is a valid edge labeling of I, and we run Algorithm 1, then the following is true:*

1. *The mapping $f \oplus p$ from step 4c is a valid edge labeling of I with fewer inconsistencies than f.*
2. *When contracting a blossom as described Section 3.4 I^b is an edge CSP instance with even Δ-matroid constraints and f^b is a valid edge labeling to I^b.*
3. *The recursion in 4d will occur at most $O(|V|)$ many times.*
4. *In step 4d, f^b is optimal for I^b if and only if f is optimal for I. Moreover, given a valid edge labeling g^b of I^b with fewer inconsistent variables than f^b, we can in polynomial time output a valid edge labeling g of I with fewer inconsistent variables than f.*
5. *If the algorithm answers "No" then f is optimal.*

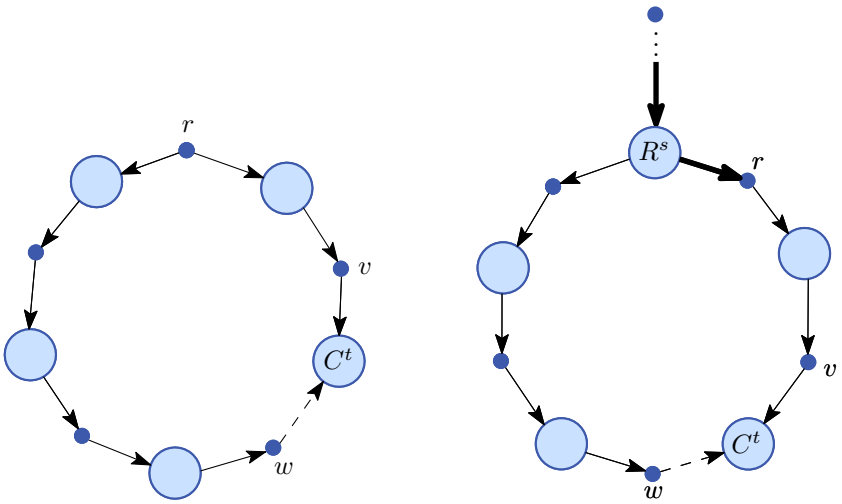

Figure 3.3: The two cases of step 4d. On the left, $\alpha = r$ is a variable, while on the right $\alpha = R^s$ is a constraint and the thick edges denote $p = \mathrm{walk}(r)$. The dashed edges are orientations of edges from $\mathcal{E}$ that are not in the digraph T, but belong to the blossom.

Contracting a blossom (step 4d)

We now elaborate step 4d of Algorithm 1. First, let us describe how to obtain the blossom b. Let $\alpha \in V(T) \cup \mathcal{C}(T)$ be the lowest common ancestor of nodes v and w in T. Two cases are possible.

1. $\alpha = r \in V(T)$. Variable node r must be inconsistent in f because it has outdegree two and thus is a root of one of the trees in the forest. We let $b = \mathrm{walk}(C^t)\,\mathrm{walk}(w)^{-1}$ in this case.

2. $\alpha = R^s \in \mathcal{C}(T)$. Let r be the child of R^s in T that is an ancestor of v. Replace edge labeling f with $f \oplus \mathrm{walk}(r)$ (variable r then becomes inconsistent). Now define walk $b = pq^{-1}r$ where p is the walk from r to C^t in T and q is the walk from R^s to w in T (see Figure 3.3).

Lemma 3.4.6 (To be proved in Section 3.5)**.** *Assume that Algorithm 1 reaches step 4d and one of the cases described in the above paragraph occurs. Then:*

1. *in the case 2 the edge labeling $f \oplus \mathrm{walk}(r)$ is valid, and*

2. *in both cases the walk b is an f-blossom (for the new edge labeling f, in the second case). (Note that we have not formally defined f-blossoms yet; they require some machinery that will come later – see Definition 3.5.9.)*

To summarize, at this point we have a valid edge labeling f of instance I and an f-blossom $b = b_0 C_1 b_1 \dots C_k b_k$. Let us denote by L the set of constraints in the blossom, i.e. $L = \{C_1, \dots, C_k\}$.

We construct a new instance I^b and its valid edge labeling f^b by *contracting the blossom* b as follows: we take I, add one $|L|$-ary constraint N to I, delete the variables $b_1, \dots, b_k$, and add new variables $\{v_C \colon C \in L\}$ (see Figure 3.4). The scope of N is $\{v_C \colon C \in L\}$ and the matroid of N consists of exactly those maps $\alpha \in \{0,1\}^L$ that send one v_C to 1 and the rest to 0.

In addition to all this, we replace each blossom constraint $D \in L$ by the constraint D^b whose scope is $\sigma \setminus \{b_1, \dots, b_k\} \cup \{v_D\}$ where σ is the scope of D. The constraint relation of D^b consists of all maps β for which there exists $\alpha \in D$ such that α agrees with β on $\sigma \setminus \{b_1, \dots, b_k\}$ and one of the following occurs (see Figure 3.5):

1. $\beta(v_D) = 0$ and α agrees with $f(D)$ on all variables in $\{b_1, \dots, b_k\} \cap \sigma$, or
2. $\beta(v_D) = 1$ and there is exactly one variable $z \in \{b_1, \dots, b_k\} \cap \sigma$ such that $\beta(z) \neq f(D)(z)$.

The lemma below follows from a more general result shown e.g. in [29, Theorem 4] (there is also a straightforward elementary proof which we include in Section 3.5).

Lemma 3.4.7. *Each C_i^b is an even Δ-matroid.*

We define the edge labeling f^b of I^b as follows: for constraints $A \notin \{C_1, \dots, C_k, N\}$ we set $f^b(A) = f(A)$. For each $C \in L$, we let $f^b(C^b)(v) = f(C)(v)$ when $v \neq v_C$, and $f^b(C^b)(v_C) = 0$. Finally, we let $f^b(N)(v_C) = 1$ for $C = C_1$ and $f^b(N)(v_C) = 0$ for all other Cs. (The last choice is arbitrary; initializing $f^b(N)$ with any other tuple in N would work as well).

It is easy to check that f^b is valid for I^b. Furthermore, v_{C_1} is inconsistent in f^b while for each $C \in L \setminus \{C_1\}$ the variable v_C is consistent.

Observation 3.4.8. *In the situation described above, the instance I^b will have at most as many variables as I and one constraint more than I. Edge labelings f and f^b have the same number of inconsistent variables.*

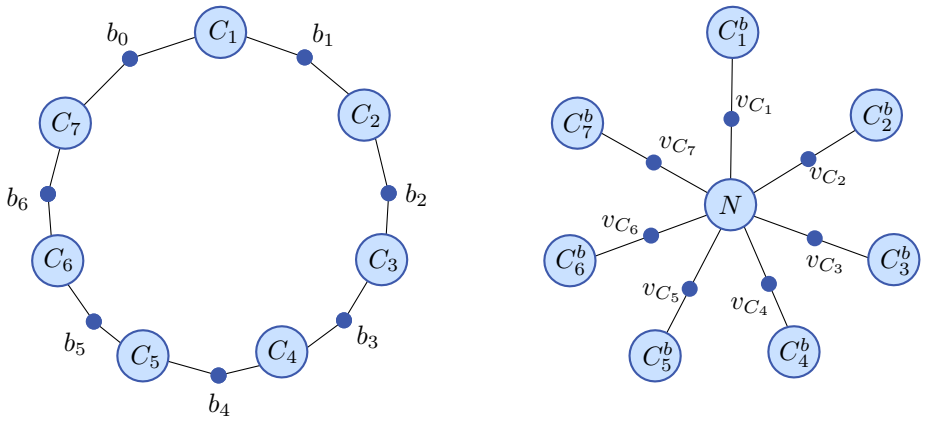

Figure 3.4: A blossom (left) and a contracted blossom (right) in the case when all constraints $C_1, \ldots, C_k$ are distinct. If some constraints appear in the blossom multiple times then the number of variables v_{C_i} will be smaller than k (see Figure 3.5).

Corollary 3.4.9 (Theorem 3.4.5(3))**.** *When given an instance I, Algorithm 1 will recursively call itself $O(|V|)$ many times.*

Proof. Since $\mathcal{C}$ and V are partitions of G_I and the degree of each $v \in V$ is two, the number of edges of G_I is $2|V|$. From the other side, the number of edges of G_I is equal to the sum of arities of all constraints in I. Since we never consider constraints with empty scopes, the number of constraints of an instance is at most double the number of variables of the instance.

Since each contraction adds one more constraint and never increases the number of variables, it follows that there can not be a sequence of consecutive contractions longer than $2|V|$, which is $O(|V|)$. □

The following two lemmas, which we prove in Section 3.5, show why the procedure works. In both lemmas, we let (I, f) and (I^b, f^b) denote the instance and the valid edge labeling before and after the contraction, respectively.

Lemma 3.4.10. *In the situation described above, if f^b is optimal for I^b, then f is optimal for I.*

Lemma 3.4.11. *In the situation described above, if we are given a valid edge labeling g^b of I^b with fewer inconsistencies than f^b, then we can find in polynomial time a valid edge labeling g of I with fewer inconsistencies than f.*

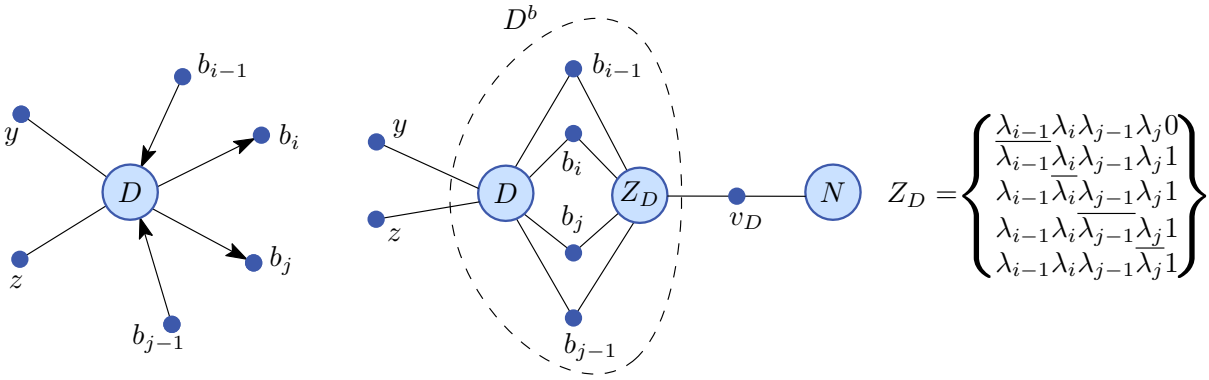

Figure 3.5: Modification of a constraint node D that appears in a blossom b twice, i.e. when we have $b = \ldots b_{i-1} D b_i \ldots b_{j-1} D b_j \ldots$ (and so $D = C_i = C_j$). Variables y and z are not part of the walk. The construction of D^b described in the text can be alternatively viewed as attaching a "gadget" constraint Z_D as shown in the figure. Here Z_D is an even Δ-matroid with five tuples that depend on the values $\lambda_k = f(\{b_k, D\})$ and $\overline{\lambda_k} = 1 - \lambda_k$.

Time complexity of Algorithm 1

To see that Algorithm 1 runs in time polynomial in the size of I, consider first the case when step 4d does happen. In this case, the algorithm runs in time polynomial in the size of I, since it essentially just searches through the graph G_I.

Moreover, from the description of contracting a blossom in part 3.4, it is easy to see that one can compute I^b and f^b from I and f in polynomial time and that I^b is not significantly larger than I: I^b has at most as many variables as I and the contracted blossom constraints C^b are not larger than the original constraints C. Finally, I^b does have one brand new constraint N, but N contains only $O(|V|)$ many tuples. Therefore, we have $|I^b| \leq |I| + O(|V|)$ where $|V|$ does not change. By Corollary 3.4.9, there will be at most $O(|V|)$ contractions in total, so the size of the final instance $I^\star$ is at most $|I| + O(|V|^2)$, which is easily polynomial in $|I|$.

All in all, Algorithm 1 will give its answer in time polynomial in $|I|$.

3.5 Proofs

In this section, we flesh out detailed proofs of the statements we gave above. In the whole section, I will be an instance of a Boolean edge CSP whose constraints are even Δ-matroids.

In Sec. 3.5 we establish some properties of f-walks, and show in particular that a valid edge labeling f of I is non-optimal if and only if there exists an augmenting f-walk in I. In Sec. 3.5 we introduce the notion of an *f-DAG*, prove that the forest T constructed during the algorithm is in fact an f-DAG, and describe some tools for manipulating f-DAGs. Then in Sec. 3.5 we analyze augmentation and contraction operations, namely prove Theorem 3.4.5(1) and Lemmas 3.4.6, 3.4.10, 3.4.11 (which imply Theorem 3.4.5(2, 4). Finally, in Sec. 3.5 we prove Theorem 3.4.5(5).

For edge labelings f, g, let $f \Delta g \subseteq \mathcal{E}$ be the set of edges in $\mathcal{E}$ on which f and g differ.

Observation 3.5.1. *If f and g are valid edge labelings of instance I then they have the same number of inconsistencies modulo 2.*

Proof. We use induction on $|f \Delta g|$. The base case $|f \Delta g| = 0$ is trivial. For the induction step let us consider valid edge labelings f, g with $|f \Delta g| \geq 1$. Pick an edge $\{v, C\} \in f \Delta g$. By the property of even Δ-matroids there exists another edge $\{w, C\} \in f \Delta g$ with $w \neq v$ such that $f(C) \oplus v \oplus w \in C$. Thus, edge labeling $f^\star = f \oplus (vCw)$ is valid. Clearly, f and $f^\star$ have the same number of inconsistencies modulo 2. By the induction hypothesis, the same holds for edge labelings $f^\star$ and g (since $|f^\star \Delta g| = |f \Delta g| - 2$). This proves the claim. □

The properties of f-walks

Let us begin with some results on f-walks that will be of use later. The following lemma is a (bit more technical) variant of the well known property of labelings proven in [24, Theorem 3.6]:

Lemma 3.5.2. *Let f, g be valid edge labelings of I such that g has fewer inconsistencies than f, and x be an inconsistent variable in f. Then there exists an augmenting f-walk that begins in a variable different from x. Moreover, such a walk can be computed in polynomial time given I, f, g, and x.*

Proof. Our algorithm will proceed in two stages. First, we repeatedly modify the edge labeling g using the following procedure:

(1) Pick a variable $v \in V$ which is consistent in f, but not in g. (If no such v exists then go to the next paragraph). By the choice of v, there exists a unique edge $\{v, C\} \in f \,\Delta\, g$. Pick a variable $w \neq v$ in the scope of C such that $\{w, C\} \in f \,\Delta\, g$ and $g(C) \oplus v \oplus w \in C$ (it exists since C is an even Δ-matroid). Replace g with $g \oplus (vCw)$, then go to the beginning and repeat.

It can be seen that g remains a valid edge labeling, and the number of inconsistencies in g never increases. Furthermore, each step decreases $|f \,\Delta\, g|$ by 2, so this procedure must terminate after at most $O(|\mathcal{E}|) = O(|V|)$ steps.

We now have valid edge labelings f, g such that f has more inconsistencies than g, and variables consistent in f are also consistent in g. Since the parity of number of inconsistencies in f and g is the same, f has at least two more inconsistent variables than g; one of them must be different from x.

In the second stage we will maintain an f-walk p and the corresponding valid edge labeling $f^\star = f \oplus p$. To initialize, pick a variable $r \in V \setminus \{x\}$ which is consistent in g but not in f, and set $p = r$ and $f^\star = f$. We then repeatedly apply the following step:

(2) Let v be the endpoint of p. The variable v is consistent in g but not in $f^\star$, so there is a unique edge $\{v, C\} \in f^\star \,\Delta\, g$. Pick a variable $w \neq v$ in the scope of C such that $\{w, C\} \in f^\star \,\Delta\, g$ and $f^\star(C) \oplus v \oplus w \in C$ (it exists since C is an even Δ-matroid). Append vCw to the end of p, and accordingly replace $f^\star$ with $f^\star \oplus (vCw)$ (which is valid by the choice of w). As a result of this update, edges $\{v, C\}$ and $\{w, C\}$ are removed from $f^\star \,\Delta\, g$.

If w is inconsistent in f, then output p (which is an augmenting f-walk) and terminate. Otherwise w is consistent in f (and thus in g) but not in $f^\star$; in this case, go to the beginning of (2) and repeat.

Each step decreases $|f^\star \,\Delta\, g|$ by 2, so this procedure must terminate after at most $O(|\mathcal{E}|) = O(|V|)$ steps. To see that p is indeed a walk, observe that the starting node r has exactly one incident edge in the graph $(V \cup \mathcal{C}, f^\star \,\Delta\, g)$. Since this edge is immediately removed from $f^\star \,\Delta\, g$, we will never encounter the variable r again during the procedure. □

Invariants of Algorithm 1: f-DAGs

In this section we examine the properties of the forest T as generated by Algorithm 1. For future comfort, we will actually allow T to be a bit more general than what appears in Algorithm 1 – our T can be a directed acyclic digraph (DAG):

Definition 3.5.3. Let I be a Boolean edge CSP instance and f a valid edge labeling of I. We will call a directed graph T an *f-DAG* if $T = (V(T) \cup \mathcal{C}(T), E(T))$ where $V(T) \subseteq V$, $\mathcal{C}(T) \subseteq \mathcal{C} \times \mathbb{N}$, and the following conditions hold:

1. Edges of $E(T)$ have the form vC^t or C^tv where $\{v, C\} \in \mathcal{E}$ and $t \in \mathbb{N}$.
2. For each $\{v, C\} \in \mathcal{E}$ there is at most one $t \in \mathbb{N}$ such that vC^t or C^tv appears in $E(T)$. Moreover, vC^t and C^tv are never both in $E(T)$.
3. Each node $v \in V(T)$ has at most one incoming edge. (Note that by the previous properties, the node v can have at most two incident edges in T.)
4. Timestamps t for nodes $C^t \in \mathcal{C}(T)$ are all distinct (and thus give a total order on $\mathcal{C}(T)$). Moreover, this order can be extended to a total order $\prec$ on $V(T) \cup \mathcal{C}(T)$ such that $\alpha \prec \beta$ for each edge $\alpha\beta \in E(T)$. (So in particular the digraph T is acyclic.)
5. If T contains edges uC^t and one of vC^t or C^tv, then $f(C) \oplus u \oplus v \in C$.
6. ("No shortcuts" property) If T contains edges uC^s and one of vC^t or C^tv where $s < t$, then $f(C) \oplus u \oplus v \notin C$.

It is easy to verify that any subgraph of an f-DAG is also an f-DAG. If T is an f-DAG, then we denote by $f \oplus T$ the edge labeling we obtain from f by flipping the value of all $f(\{v, C\})$ such that $vC^t \in E(T)$ or $C^tv \in E(T)$ for some timestamp t. We will need to show that $f \oplus T$ is a valid edge labeling for nice enough f-DAGs T.

The following lemma shows the promised invariant property:

Lemma 3.5.4. *Let us consider the structure T during the run of Algorithm 1 with the input I and f. At any moment during the run, the forest T is an f-DAG.*

Moreover, if steps 4c or 4d are reached, then the digraph $T^\star$ obtained from T by removing all edges outgoing from C^t (defined by step 3) and adding the edge wC^t is also an f-DAG.

Proof. Obviously, an empty T is an f-DAG, as is the initial T consisting of inconsistent variables and no edges. To verify that T remains an f-DAG during the whole run of Algorithm 1, we need to make sure that neither adding vC^t in step 3, nor adding C^tw in step 4a violates the properties of T. Let us consider step 3 first. By the choice of v and C^t, we immediately get that properties (1), (2), (3), and (4) all hold even after we have added vC^t to T (we can order the nodes by the order in which they were added to T). Since there is only one edge incident with C^t, property (5) holds as well. Finally, the only way the no shortcuts property (i.e. property (6)) could fail would be if there were some u and s such that $uC^s \in E(T)$ and $f(C) \oplus u \oplus v \in C$. But then, after the node C^s got added to T, we should have computed the set W of variables w such that $f(C) \oplus u \oplus w$ (step 4) and v should have been in $W \setminus V(T)$ at that time, i.e. we should have added the edge C^sv before, a contradiction. The analysis of step 4a is similar.

Assume now that Algorithm 1 has reached one of steps 4c or 4d and consider the DAG $T^\star$ that we get from T by removing all edges of the form C^tz and adding the edge wC^t. Note that the node C^t is the only node with two incoming edges. The only three properties that going from T to $T^\star$ could possibly affect are (2), (5) and (6). Were (2) violated, we would have $C^sw \in E(T)$ already, and so step 4b would be triggered instead of steps 4c or 4d. For property (5), the only new pair of edges to consider is vC^t and wC^t for which we have $f(C) \oplus v \oplus w \in C$. Finally, if property (6) became violated after adding the edge wC^t then there were a u and $s < t$ such that $uC^s \in E(T)$ and $f(C) \oplus u \oplus w \in C$. Node C^s must have been added after w, or else we would have $C^sw \in E(T)$. Also, w cannot have a parent of the form C^k (otherwise step 4b would be triggered for w when expanding C^t). But then one of steps 4d or 4c would be triggered at timestamp s already when we tried to expand C^s, a contradiction. □

We will use the following two lemmas to prove that $f \oplus p$ is a valid edge labeling of I for various paths p that appear in steps 4c and 4d.

Lemma 3.5.5. *Let T be an f-DAG, and C^s be the constraint node in $\mathcal{C}(T)$ with the smallest timestamp s. Suppose that C^s has exactly two incident edges, namely incoming edge uC^s where u does not have other incident edges besides uC^s and another edge C^sv (see Figure 3.6). Let $f^\star = f \oplus (uCv)$ and let $T^\star$ be the DAG obtained from T by removing nodes u, C^s and the two edges incident to C^s. Then $f^\star$ is a valid edge labeling of I and $T^\star$ is an $f^\star$-DAG.*

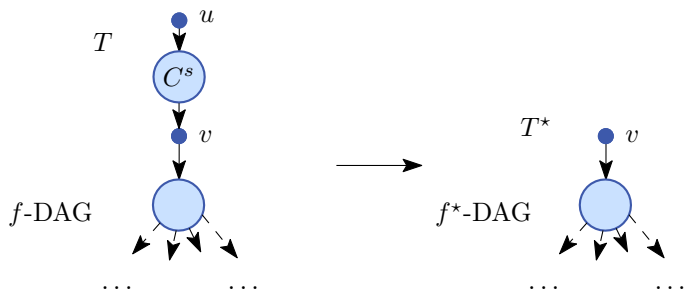

Figure 3.6: An f-DAG T on the left turns into $f^\star$-DAG $T^\star$ on the right; the setting from Lemma 3.5.5.

Proof. Since $T^\star$ is a subgraph of T, it immediately follows that $T^\star$ satisfies the properties (1), (2), (3), and (4) from the definition of an f-DAG all hold.

Let us show that $T^\star$ has property (5). Consider a constraint node $C^t \in \mathcal{C}(T^\star)$ with $t > s$ (nothing has changed for other constraint nodes in $\mathcal{C}(T^\star)$), and suppose that $T^\star$ contains edges xC^t and one of yC^t or $C^t y$. If $x = y$, the situation is trivial, so assume that u, v, x, y are all distinct variables. We need to show that $f^\star(C) \oplus x \oplus y \in C$. The constraint C contains the tuples $f(C) \oplus u \oplus v$ and $f(C) \oplus x \oplus y$ (by condition (5) for T), but the no shortcuts property prohibits the tuples $f(C) \oplus u \oplus x$ and $f(C) \oplus u \oplus y$ from lying in C. Therefore, applying the even Δ-matroid property on $f(C) \oplus u \oplus v$ and $f(C) \oplus x \oplus y$ in the variable u we get that C must contain $f(C) \oplus u \oplus v \oplus x \oplus y$, so we have $f^\star(C) \oplus x \oplus y \in C$.

Now let us prove that $T^\star$ and $f^\star$ have the "no shortcuts" property. Consider constraint nodes C^k, C^ℓ in $\mathcal{C}(T^\star)$ with $s < k < \ell$ (since nothing has changed for constraint nodes other than C), and suppose that $T^\star$ contains edges xC^k and one of yC^ℓ or $C^\ell y$, where again u, v, x, y are all distinct variables. We need to show that $f^\star(C) \oplus x \oplus y \notin C$, or equivalently that $f(C) \oplus u \oplus v \oplus x \oplus y \notin C$.

Assume that it is not the case. Apply the even Δ-matroid property to tuples $f(C) \oplus u \oplus v \oplus x \oplus y$ and $f(C)$ (which are both in C) in coordinate v. We get that either $f(C) \oplus x \oplus y \in C$, or $f(C) \oplus u \oplus x \in C$, or $f(C) \oplus u \oplus y \in C$. This contradicts the "no shortcuts" property for the pair (C^k, C^ℓ), or (C^s, C^k), or (C^s, C^ℓ), respectively, and we are done. □

Corollary 3.5.6. *Let I be an edge CSP instance and f be a valid edge labeling.*

1. *Let T be an f-DAG that consists of two directed paths $x_0 C_1^{t_1} x_1 \dots x_{k-1} C_k^{t_k}$ and*

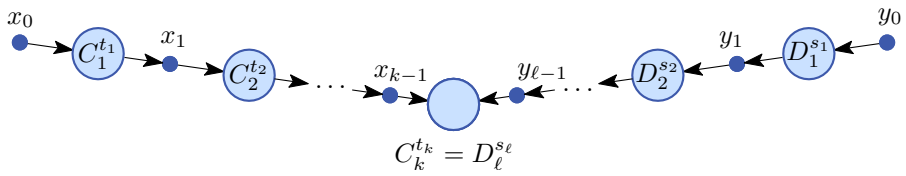

Figure 3.7: Two meeting paths from Corollary 3.5.6.

$y_0 D_1^{s_1} \dots y_{\ell-1} D_\ell^{s_\ell}$ *that are disjoint everywhere except at the constraint* $C_k^{t_k} = D_\ell^{s_\ell}$ *(see Figure 3.7). Then* $f \oplus T$ *is a valid edge labeling of* I*.*

2. *Let* T *be an* f*-DAG that consists of a single directed path* $x_0 C_1^{t_0} x_1 \dots x_{k-1} C_k^{t_k} x_k$*. Then* $f \oplus T$ *is a valid edge labeling of* I*.*

Proof. We will prove only part (a); the proof of part (b) is completely analogous. We proceed by induction on $k + \ell$. If $k = \ell = 1$, T consists only of the two edges $x_0 C^t$ and $y_0 C^t$ (where C^t is an abbreviated name for $C_1^{t_1} = D_1^{s_1}$). Then the fact that $f \oplus (x_0 C y_0)$ is a valid edge labeling follows from the property (5) of f-DAGs.

If we are now given an f-DAG T of the above form, then we compare t_1 and s_1. Since the situation is symmetric, we can assume without loss of generality that $s_1 > t_1$. We then use Lemma 3.5.5 for $x_1 C_1^{t_1} x_2$ (there is a x_2 since $t_k > s_1 > t_1$), obtaining the $(f \oplus (x_1 C_1 x_2))$-DAG $T^\star$ that consists of two directed paths $x_2 \dots x_k C^{t_k}$ and $y_1 D_1^{s_1} \dots y_\ell D_\ell^{s_\ell}$. Since $T^\star$ is shorter than T, the induction hypothesis gets us that $f \oplus (x_1 C_1 x_2) \oplus T^\star = f \oplus T$ is a valid edge labeling. □

Lemma 3.5.7. *Let* T *be an* f*-DAG, and* C^s *be the constraint node in* $\mathcal{C}(T)$ *with the smallest timestamp* s*. Suppose that* C^s *has exactly one incoming edge* uC^s*, and* u *does not have other incident edges besides* uC^s*. Suppose also that* C^s *has an outgoing edge* $C^s v$*. Let* $f^\star = f \oplus (uCv)$*, and* $T^\star$ *be the DAG obtained from* T *by removing the edge* uC^s *together with* u *and reversing the orientation of edge* $C^s v$ *(see Figure 3.8). Then* $f^\star$ *is a valid edge labeling of* I *and* $T^\star$ *is an* $f^\star$*-DAG.*

Proof. It is easy to verify that $T^\star$ satisfies the properties (1), (2) and (3). To see property (4), just take the linear order on nodes of T and change the position of v so that it is the new minimal element in this order (v has no incoming edges in $T^\star$).

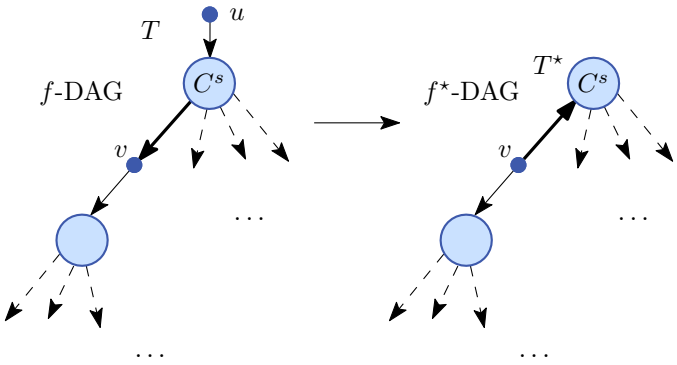

Figure 3.8: An f-DAG T turns into an $f^\star$-DAG $T^\star$ (see Lemma 3.5.7).

Let us prove that property (5) of Definition 3.5.3 is preserved. First, consider constraint node C^s. Suppose that $T^\star$ contains one of xC^s or $C^s x$ with $x \neq v$. We need to show that $f^\star(C) \oplus v \oplus x \in C$, or equivalently $f(C) \oplus u \oplus x \in C$ (since $f^\star(C) \oplus v = f(C) \oplus (u \oplus v) \oplus v = f(C) \oplus u$). This claim holds by property (5) of Definition 3.5.3 for T.

Now consider a constraint node $C^t \in \mathcal{C}(T^\star)$ with $t > s$, and suppose that $T^\star$ contains edges xC^t and one of yC^t or $C^t y$. We need to show that $f^\star(C) \oplus x \oplus y \in C$, or equivalently that $f(C) \oplus u \oplus v \oplus x \oplus y \in C$. For that we can simply repeat word-by-word the argument used in the proof of Lemma 3.5.5.

Now let us prove that the "no shortcuts" property is preserved. First, consider a constraint node C^t in $\mathcal{C}(T^\star)$ with $t > s$, and suppose that $T^\star$ contains one of xC^t or $C^t x$. We need to show that $f^\star(C) \oplus v \oplus x \notin C$, or equivalently $f(C) \oplus u \oplus x \notin C$. This claim holds by the "no shortcuts" property for T. Now consider constraint nodes C^k, C^ℓ in $\mathcal{C}(T^\star)$ with $s < k < \ell$, and suppose that $T^\star$ contains edges xC^k and one of yC^ℓ or $C^\ell y$. Note that u, v, x, y are all distinct variables. We need to show that $f^\star(C) \oplus x \oplus y \notin C$, or equivalently that $f(C) \oplus u \oplus v \oplus x \oplus y \notin C$. For that we can simply repeat word-by-word the argument used to show the no shortcuts property in the proof of Lemma 3.5.5. □

Analysis of augmentations and contractions

First, we prove the correctness of the augmentation operation.

Proposition 3.5.8 (Theorem 3.4.5(1) restated)**.** *The mapping $f \oplus p$ from step 4c is a valid edge labeling of I with fewer inconsistencies than f.*

Proof. Let T_1 be the f-DAG constructed during the run of Algorithm 1; let T_2 be the DAG obtained from T_1 by adding the edge wC^t. By Lemma 3.5.4, T_2 is a subgraph of an f-DAG, so it is an f-DAG itself. Let T_3 be the subgraph of T_2 (and thus also an f-DAG) induced by the nodes in p. It is easy to verify that T_3 consists of two directed paths that share their last node. Therefore, by Corollary 3.5.6, we get that $f \oplus T_3 = f \oplus p$ is a valid edge labeling of I. □

In the remainder of this section we show the correctness of the contraction operation by proving Lemmas 3.4.6, 3.4.10, 3.4.11. Let us begin by giving a full definition of a blossom:

Definition 3.5.9. Let f be a valid edge labeling. An *f-blossom* is any walk $b = b_0C_1b_1C_2 \dots C_kb_k$ with $b_0 = b_k$ such that:

1. variable $b_0 = b_k$ is inconsistent in f while variables $b_1, \dots, b_{k-1}$ are consistent, and
2. there exists $\ell \in [1, k]$ and timestamps $t_1, \dots, t_k$ such that the DAG consisting of two directed paths $b_0C_1^{t_1} \dots b_{\ell-1}C_\ell^{t_\ell}$ and $b_kC_k^{t_k}b_{k-1} \dots b_\ell C_\ell^{t_\ell}$ is an f-DAG.

Lemma 3.5.10. *Let b be a blossom. Then $b_{[i,j]}$ is an f-walk for any non-empty proper subinterval $[i, j] \subsetneq [0, k]$.*

Proof. Let us denote the f-DAG from the definition of a blossom by B. By taking an appropriate subgraph of B and applying Corollary 3.5.6 we get that $f \oplus b_{[i,j]}$ is valid for any non-empty subinterval $[i, j] \subsetneq [0, k]$. Since the set of these intervals is downward closed, $b_{[i,j]}$ is in fact an f-walk. □

Lemma 3.5.11 (Lemma 3.4.6 restated)**.** *Assume that Algorithm 1 reaches step 4d and one of the cases described at the beginning of Section 3.4 occurs. Then:*

1. *in the case 2 the edge labeling $f \oplus \mathrm{walk}(r)$ is valid, and*
2. *in both cases the walk b is an f-blossom (for the new edge labeling f, in the second case).*

Proof. Let T be the forest at the moment of contraction, $T^\dagger$ be the subgraph of T containing only paths $\mathrm{walk}(C^t)$ and $\mathrm{walk}(w)$, and $T^\star$ be the graph obtained from $T^\dagger$ by

adding the edge wC^t. By Lemma 3.5.4, graph $T^\star$ is an f-DAG (we also need to observe that any subgraph of an f-DAG is again an f-DAG).

If the lowest common ancestor of u and v in T is a variable node $r \in V(T)$ (i.e. we have case 1 from Section 3.4), then the f-DAG $T^\star$ consists of two directed paths from r to the constraint C and it is easy to verify that when we let b to be one of these paths followed by the other in reverse, we get a blossom.

Now consider case 2, i.e. when the lowest common ancestor of u and v in T is a constraint node $R^s \in \mathcal{C}(T)$. Note that $T^\star$ has the unique source node u (that does not have incoming edges), and u has an outgoing edge uD^t where D^t is the constraint node with the smallest timestamp in $T^\star$. Let us repeat the following operation while $D^t \neq R^s$: Replace f with $f \oplus (uD^t z)$ where z is the unique out-neighbor of D^t in $T^\star$, and simultaneously modify $T^\star$ by removing nodes u, D^t and edges $uD^t, D^t z$. By Lemma 3.5.5 f remains a valid edge labeling throughout this process, and $T^\star$ remains an f-DAG (for the latest f).

We get to the point that the unique in-neighbor u of R^s is the source node of $T^\star$. Replace f with $f \oplus (uR^s r)$, and simultaneously modify $T^\star$ by removing node u together with the edge uR^s and reversing the orientation of edge $R^s r$. The new f is again valid, and the new $T^\star$ is an f-DAG by Lemma 3.5.7. This means that the resulting walk b is an f-blossom for the new f. □

Lemma 3.5.12 (Lemma 3.4.7 restated)**.** *Each D^b constructed in Section 3.4 is an even Δ-matroid.*

Proof. We will denote by σ the scope of D. Let $\beta, \beta' \in D^b$ and assume that $\beta(v) \neq \beta'(v)$ for some v. We will show that there is a $u \neq v$ such that $\beta(u) \neq \beta'(u)$ and $\beta \oplus v \oplus u \in D^b$.

Assume first that v is different from v_D. From the definition of D^b, we know that there exist $\alpha, \alpha' \in D$ that witness $\beta, \beta' \in D^b$. In particular, $\alpha(v) \neq \alpha'(v)$. Since D is an even Δ-matroid, there exists a variable w such that $\alpha(w) \neq \alpha'(w)$ and $\alpha \oplus v \oplus w \in D$. If w is not in $\{b_1, \ldots, b_k\}$, we get $\beta(w) \neq \beta'(w)$ and $\beta \oplus v \oplus w \in D^b$, while if $w \in \{b_1, \ldots, b_k\}$, we get $\beta \oplus v \oplus v_D \in D^b$. If $\beta(v_D) \neq \beta'(v_D)$, we are done. Otherwise, α and α' disagree exactly on two variables from $\{b_1, \ldots, b_k\}$, one of which is w; let t be the other variable. The Δ-matroid property used on $\alpha \oplus v \oplus w$ and α' gives us that there is a $z \notin \{b_1, \ldots, b_k\}$

such that $\alpha(z) \neq \alpha'(z)$ and $\alpha \oplus v \oplus w \oplus t \oplus z \in D$. From this, it easily follows that $\beta \oplus v \oplus z \in D^b$.

In the case $v = v_D$, we again consider $\alpha, \alpha' \in D$ as above. Since $\beta(v_D) \neq \beta'(v_D)$, we know that α and α' disagree at exactly one variable b_i. Using the Δ-matroid property on α, α' with the variable b_i, we obtain a variable $w \neq b_i$ such that α and α' disagree on w and $\alpha \oplus b_i \oplus w \in D$. Then w needs to be from $\sigma \setminus \{b_1, \ldots, b_k\}$, so $\beta(w) \neq \beta'(w)$ and $\alpha \oplus b_i \oplus w$ witnesses that $\beta \oplus v_D \oplus w \in D^b$ □

Finally, we prove two lemmas showing that if we contract a blossom b in instance I to obtain the instance I^b and the edge labeling f^b, then f^b is optimal if and only if f^b is optimal for I^b.

Lemma 3.5.13 (Lemma 3.4.10 restated)**.** *In the situation described above, if f^b is optimal for I^b, then f is optimal for I.*

Proof. Assume that f is not optimal for I, so there exists a valid edge labeling g with fewer inconsistencies than f. Then by Lemma 3.5.2 there exists an augmenting f-walk p in I that starts at some node other than b_k. Denote by p^b the sequence obtained from p by replacing each C_i from the blossom by C_i^b. Observe that if p does not contain the variables $b_1, \ldots, b_k$, then p is an f-walk if and only if p^b is an f^b-walk, so the only interesting case is when p enters the set $\{b_1, \ldots, b_k\}$.

We will proceed along p and consider the first i such that there is a blossom constraint D and an index j for which $p_{[0,i]}Db_j$ is an f-walk (i.e. we can enter the blossom from p).

If $D = C_1$, then it follows from the definition of C_1^b that $p_{[0,i]}C_1^b v_{C_1}$ is an augmenting f^b-walk in I^b, while if $D \neq C_1$, then $p_{[0,i]}D^b v_{C_D} N v_{C_1}$ is an augmenting f^b-walk. In both cases, we get an augmenting walk for f^b, and so f^b was not optimal. □

To show the other direction, we will first prove the following result.

Lemma 3.5.14. *Let q be an f-walk and T an f-DAG such that there is no proper prefix $q^\star$ of q and no edge vC^s or $C^s v$ of T such that $q^\star Cv$ would be an f-walk. Then T is a $(f \oplus q)$-DAG.*

Proof. We proceed by induction on the length of q. If q has length 0, the claim is trivial. Otherwise, let $q = xCyq^\dagger$ for some $q^\dagger$. Note that $q^\dagger$ is trivially an $(f \oplus (xCy))$-walk. We

verify that T is an $(f \oplus (xCy))$-DAG, at which point it is straightforward to apply the induction hypothesis with $f \oplus xCy$ and $q^\dagger$ to show that T is an $(f \oplus q)$-DAG.

We choose the timestamp t to be smaller than any of the timestamps appearing in T and construct the DAG $T^\dagger$ from T by adding the nodes x, y, C^t and edges xC^t and $C^t y$. It is easy to see that $T^\dagger$ is an f-DAG – the only property that might possibly fail is the no shortcuts property. However, since the timestamp of C^t is minimal, were the no shortcuts property violated, T would have to contain an edge of the form vC^s or $C^s v$ such that $f(C) \oplus x \oplus v \in C$. But in that case, we would have the f-walk xCv, contradicting our assumption on prefixes of q.

It follows that $T^\dagger$ is an f-DAG and we can use Lemma 3.5.5 with the constraint C^t and edges xC^t and $C^t y$ to show that T is an $(f \oplus (xC^t y))$-DAG, concluding the proof. □

Lemma 3.5.15 (Lemma 3.4.11 restated)**.** *In the situation described above, if we are given a valid edge labeling g^b of I^b with fewer inconsistencies than f^b, then we can find in polynomial time a valid edge labeling g of I with fewer inconsistencies than f.*

Proof. Our overall strategy here is to take an inconsistency from the outside of the blossom b and bring it into the blossom. We begin by showing how to get a valid edge labeling f' for I with an inconsistent variable just one edge away from b.

Using Lemma 3.5.2, we can use g^b and f^b to find in polynomial time an augmenting f^b-walk p^b that does not begin at the inconsistent variable v_{C_1}. If p^b does not contain any of the variables $v_{C_1}, \ldots, v_{C_k}$, then we can just output the walk p obtained from p^b by replacing each C_i^b by C_i and be done. Assume now that some v_C appears in p^b. We choose the f^b-walk r^b so that $r^b C^b v_C$ is the shortest prefix of p^b that ends with some blossom variable v_C. By renaming all C^bs in r^b to Cs, we get the walk r. It is straightforward to verify that r is an f-walk and that $rC_i b_i$ or $rC_i b_{i-1}$ is an f-walk for some $i \in [1, k]$. Let q be the shortest prefix of r such that one of $qC_i b_i$ or $qC_i b_{i-1}$ is an f-walk for some $i \in [1, k]$.

Recall that the blossom b originates from an f-DAG B. The minimality of q allows us to apply Lemma 3.5.14 and obtain that B is also an $(f \oplus q)$-DAG. Let $f' = f \oplus q$ and let x be the last variable in q. It is easy to see that f' is a valid edge labeling with exactly as many inconsistent variables as f. Moreover x is inconsistent in f' and there is an

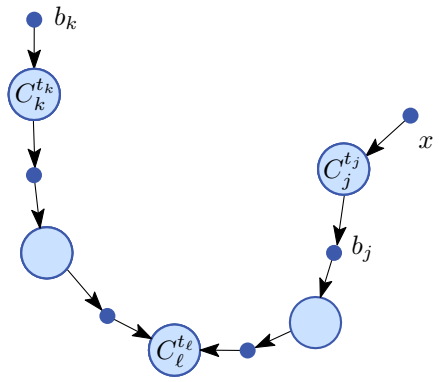

Figure 3.9: The f'-DAG B' constructed using xC_jb_j (where $j < \ell$).

index i such that at least one of xC_ib_i or xC_ib_{i-1} is an f'-walk. We will now show how to improve f'.

If the constraint C_i appears only once in the blossom b, it is easy to verify (using Lemma 3.5.10) that one of $xC_ib_{[i,k]}$ or $xC_ib_{[0,i-1]}{}^{-1}$ is an augmenting f'-walk. However, since the constraint C_i might appear in the blossom several times, we have to come up with a more elaborate scheme. The blossom b comes from an f'-DAG B in which some node $C_\ell^{t_\ell}$ is the node with the maximal timestamp (for a suitable $\ell \in [1, k]$). Assume first that there is a $j \in [\ell, k]$ such that xC_jb_j is an f'-walk. In that case, we take maximal such j and consider the DAG B' we get by adding the edge $C_j^{t_j}x$ to the subgraph of B induced by the nodes $C_j^{t_j}, b_j, C_{j+1}^{t_{j+1}}, \ldots, C_k^{t_k}, b_k$.

It is routine to verify that B' is an f'-DAG; the only thing that could possibly fail is the no shortcuts property involving $C_j^{t_j}$. However, $C_j^{t_j}$ has maximal timestamp in B' and there is no $i > j$ such that $f'(C_j) \oplus x \oplus b_i \in C_j$.

Using Corollary 3.5.6, we get that $f' \oplus B'$ is a valid edge labeling which has fewer inconsistencies than f', so we are done. In a similar way, we can improve f' when there exists a $j \in [1, \ell]$ such that xC_jb_{j-1} is an f'-walk.

If neither of the above cases occurs, then we take j such that the timestamp t_j is maximal and either xC_jb_j or xC_jb_{j-1} is an f'-walk. Without loss of generality, let xC_jb_j be an f'-walk. Then $j < \ell$ and we consider the DAG B' we get from the subgraph of B induced by $C_j^{t_j}, b_j, C_{j+1}^{t_{j+1}}, \ldots, C_k^{t_k}, b_k$ by adding the edge $xC_j^{t_j}$ (see Figure 3.9). As before, the only way B' can not be an f'-DAG is if the no shortcuts property fails, but that is impossible: we chose j so that t_j is maximal, so an examination of the makeup of B' shows that the only bad thing that could possibly happen is if there were an index $i \geq \ell$

such that $C_i = C_j$, we had in B the edge $b_i C_i^{t_i}$, and $f'(C_i) \oplus b_i \oplus x \in C_i$. But then we would have the f'-walk xC_ib_i for $i \geq \ell$ and the procedure from the previous paragraph would apply. Using Corollary 3.5.6, we again see that $f' \oplus B'$ is a valid edge labeling with fewer inconsistencies than f.

It is easy to verify that finding q, calculating $f' = f \oplus q$, finding an appropriate j and augmenting f' can all be done in time polynomial in the size of the instance. □

Proof of Theorem 3.4.5(5)

In this section we will prove that if the algorithm answers "No" then f is an optimal edge labeling.

Lemma 3.5.16. *Suppose that Algorithm 1 outputs "No" in step 2, without ever visiting steps 4c and 4d. Then f is optimal.*

Proof. Let T be the forest upon termination, and denote

$$\overline{E}(T) = \{Cv \colon C^t v \in E(T) \text{ for some } t\} \cup \{vC \colon vC^t \in E(T) \text{ for some } t\}.$$

Inspecting Algorithm 1, one can check that $\overline{E}(T)$ has the following properties:

1. If v is an inconsistent variable in f and $\{v, C\} \in \mathcal{E}$, then $vC \in \overline{E}(T)$.
2. If $Cv \in \overline{E}(T)$ and $\{v, D\} \in \mathcal{E}, D \neq C$, then $vD \in \overline{E}(T)$.
3. If $vC \in \overline{E}(T)$, then $Cv \notin \overline{E}(T)$.
4. Suppose that $vC \in \overline{E}(T)$ and $f(C) \oplus v \oplus w \in C$ where v, w are distinct nodes in the scope of constraint C. Then $Cw \in \overline{E}(T)$.

An f-walk p will be called *bad* if it starts at a variable node which is inconsistent in f, and contains an edge $Cv \notin \overline{E}(T)$; otherwise p is *good*. Clearly, any augmenting f-walk is bad: its last edge Cv satisfies $vC \in \overline{E}(T)$ by property 1, and thus $Cv \notin \overline{E}(T)$. Thus, if f is not optimal, then by Lemma 3.5.2 there exists at least one bad f-walk. Let p be a shortest bad f-walk. Write $p = p^\star(vCw)$ where $p^\star$ ends at v. By minimality of p, $p^\star$ is good and $Cw \notin \overline{E}(T)$. Using properties (1) or (2), we obtain that $vC \in \overline{E}(T)$ (and therefore $Cv \notin \overline{E}(T)$).

Let q be the shortest prefix of $p^\star$ (also an f-walk) such that $f \oplus q \oplus (vCw)$ is valid (at least one such prefix exists, namely $q = p^\star$). The walk q must be of positive length (otherwise the precondition of property (4) would hold, and we would get $Cw \in \overline{E}(T)$, a contradiction). Also, the last constraint node in q must be C, otherwise we could have taken a shorter prefix (namely one ending at C). Thus, we can write $q = q^\star(xCy)$ where $q^\star$ ends at x. Note that, since p is a walk, the variables x, y, v, w are (pairwise) distinct.

We shall write $g = f \oplus q^\star$. Let us apply the even Δ-matroid property to tuples $g(C) \oplus x \oplus y \oplus v \oplus w$ and $g(C)$ (which are both in C) in coordinate y. We get that either $g(C) \oplus v \oplus w \in C$, or $g(C) \oplus x \oplus v \in C$, or $g(C) \oplus x \oplus w \in C$. In the first case we could have chosen $q^\star$ instead of q – a contradiction to the minimality of q. In the other two cases $q^\star(xCu)$ is an f-walk for some $u \in \{v, w\}$. But then from $Cu \notin \overline{E}(T)$ we get that $q^\star(xCu)$ is a bad walk – a contradiction to the minimality of p. □

Corollary 3.5.17 (Theorem 3.4.5(5))**.** *If Algorithm 1 answers "No", then the edge labeling f is optimal.*

Proof. Algorithm 1 can answer "No" for two reasons: either the forest T can not be grown further and neither an augmenting path nor a blossom are found, or the algorithm finds a blossom b, contracts it and then concludes that f^b is optimal for I^b. We proceed by induction on the number of contractions that have occurred during the run of the algorithm.

The base case, when there were no contractions, follows from Lemma 3.5.16. The induction step is an easy consequence of Lemma 3.4.10: If we find b and the algorithm answers "No" when run on f^b and I^b then, by the induction hypothesis, f^b is optimal for I^b, and by Lemma 3.4.10 f is optimal for I. □

3.6 Extending the algorithm to efficiently coverable Δ-matroids

In this section we extend Algorithm 1 from even Δ-matroids to a wider class of so-called efficiently coverable Δ-matroids. The idea of the algorithm is similar to what [30] previously did for $\mathcal{C}$-zebra Δ-matroids, but our method covers a larger class of Δ-matroids.

Let us begin by giving a formal definition of efficiently coverable Δ-matroids.

Definition 3.6.1. We say that a class of Δ-matroids Γ is *efficiently coverable* if there is an algorithm that, given input $M \in \Gamma$ and $\alpha \in M$, lists in polynomial time a set M_α so that the system $\{M_\alpha \colon \alpha \in M\}$ satisfies the conditions of Definition 3.1.8.

Before we go on, we would like to note that coverable Δ-matroids are closed under gadgets, i.e. the "supernodes" shown in Figure 3.1. Taking gadgets is a common construction in the CSP world, so being closed under gadgets makes coverable Δ-matroids a very natural class to study.

Definition 3.6.2. Given $M \subseteq \{0,1\}^U$ and $N \subseteq \{0,1\}^V$ with U, V disjoint sets of variables, we define the *direct product* of M and N as

$$M \times N = \{(\alpha, \beta) \colon \alpha \in M, \beta \in N\} \subseteq \{0,1\}^{U \cup V}.$$

If $w_1, w_2 \in U$ are distinct variables of M, then the Δ-matroid obtained from M by identifying w_1 and w_2 is

$$\begin{aligned} M_{w_1=w_2} &= \{\beta_{\restriction U \setminus \{w_1,w_2\}} \colon \beta \in M,\ \beta(w_1) = \beta(w_2)\} \\ &\subseteq \{0,1\}^{U \setminus \{w_1,w_2\}} \end{aligned}$$

It is easy to verify that a direct product of two (even) Δ-matroids is an (even) Δ-matroid. Showing that identifying two variables in an (even) Δ-matroid yields an (even) Δ-matroid is a bit more difficult, but still straightforward.

If a matroid P is obtained from some matroids $M_1, \ldots, M_k$ by a sequence of direct products and identifying variables, we say that P is *gadget-constructed* from $M_1, \ldots, M_k$ (a gadget is an edge CSP instance with some variables present in only one constraint – these are the "output variables").

Theorem 3.6.3. *The class of coverable Δ-matroids is closed under:*

1. *Direct products,*
2. *identifying pairs of variables, and*
3. *gadget constructions.*

Proof. 1. Let $M \subseteq \{0,1\}^U$ and $N \subseteq \{0,1\}^V$ be two coverable Δ-matroids. We claim that if (α, β) and (γ, δ) are even-neighbors in $M \times N$, then either $\alpha = \gamma$ and β is an even-neighbor of δ in N, or $\beta = \delta$ and α is an even-neighbor of γ in M. This is straightforward to verify: Without loss of generality let us assume that $u \in U$ is a variable of M such that $(\alpha, \beta) \oplus u \notin M \times N$, and let v be the variable such that $(\alpha, \beta) \oplus u \oplus v = (\gamma, \delta)$. Since we are dealing with a direct product, we must have $\alpha \oplus u \notin M$ and in order for $(\alpha, \beta) \oplus u \oplus v$ to lie in $M \times N$, we must have $\alpha \oplus u \oplus v \in M$. But then $\delta = \beta$ and $\alpha \oplus u \oplus v = \gamma$ is an even-neighbor of α.

Let $\alpha \in M$, $\beta \in N$ and let M_α, N_β be the even Δ-matroids from Definition 3.1.8 for M and N. From the above paragraph, it follows by induction that whenever (γ, δ) is reachable from (α, β), then $(\gamma, \delta) \in M_\alpha \times N_\beta$. Since each $M_\alpha \times N_\beta$ is an even Δ-matroid, the direct product $M \times N$ satisfies the first two parts of Definition 3.1.8.

It remains to show that if we can reach $(\gamma, \delta) \in M \times N$ from $(\alpha, \beta) \in M \times N$ and $(\gamma, \delta) \oplus u \oplus v \in M_\alpha \times N_\beta \setminus M \times N$, then $(\gamma, \delta) \oplus u, (\gamma, \delta) \oplus v \in M \times N$. By the first paragraph of this proof, we can reach γ from α in M and δ from β in N. Moreover, both u and v must lie in the same set U or V, for otherwise we would have that $(\gamma \oplus v, \delta \oplus u)$ or $(\gamma \oplus u, \delta \oplus v)$ lies in $M_\alpha \times N_\beta$, a contradiction with M_α being an even Δ-matroid. So let (again without loss of generality) $u, v \in V$. Then $\delta \oplus u \oplus v \in N_\beta \setminus N$. Since N is coverable and δ is reachable from β, we get $\delta \oplus v, \delta \oplus u \in N$, giving us $(\gamma, \delta) \oplus u, (\gamma, \delta) \oplus v \in M \times N$ and we are done.

2. Let $M \subseteq \{0,1\}^U$ be coverable and $w_1 \neq w_2$ be two variables.

Similarly to the previous item, the key part of the proof is to show that the relation of being reachable survives identifying w_1 and w_2: More precisely, take $\alpha, \gamma \in M_{w_1=w_2}$ and $\beta \in M$ such that $\beta(w_1) = \beta(w_2)$ and $\alpha = \beta_{\restriction U \setminus \{w_1, w_2\}}$ (i.e. β witnesses $\alpha \in M_{w_1=w_2}$). Assume that we can reach γ from α. Then we can reach from β a tuple $\delta \in M$ such that $\delta(w_1) = \delta(w_2)$ and $\gamma = \delta_{\restriction U \setminus \{w_1, w_2\}}$.

Since we can proceed by induction, it is enough to prove this claim in the case when α, γ are even-neighbors. So assume that there exist variables u and v such that $\gamma = \alpha \oplus u \oplus v$ and $\alpha \oplus u \notin M_{w_1=w_2}$. From the latter, it follows that $\beta \oplus u, \beta \oplus u \oplus w_1 \oplus w_2 \notin M$. Knowing all this, we see that if $\beta \oplus u \oplus v \in M$, the tuple $\beta \oplus u \oplus v$ is an even-neighbor of β and we are done.

Assume thus that $\beta \oplus u \oplus v \notin M$. Let δ be the tuple of M witnessing $\gamma \in M_{w_1=w_2}$. Since $\beta \oplus u \oplus v \notin M$, we get $\delta = \beta \oplus u \oplus v \oplus w_1 \oplus w_2$. Since $\beta \oplus u, \beta \oplus u \oplus v \notin M$, the Δ-matroid property applied on β and δ in the variable u gives us (without loss of generality) that $\beta \oplus u \oplus w_1 \in M$. But then β is an even-neighbor of $\beta \oplus u \oplus w_1$ in M, which is an even neighbor (via the variable w_2 – recall that $\beta \oplus u \oplus w_1 \oplus w_2 \notin M$) of δ and so we can reach δ from β, proving the claim.

Assume now that M is coverable. We want to show that the sets $(M_\beta)_{w_1=w_2}$ where β ranges over M cover $M_{w_1=w_2}$. Choose $\alpha \in M_{w_1=w_2}$ and let $\beta \in M$ be the witness for $\alpha \in M_{w_1=w_2}$. We claim that the even Δ-matroid $(M_\beta)_{w_1=w_2}$ contains all members of $M_{w_1=w_2}$ that can be reached from α. Indeed, whenever γ can be reached from α, some $\delta \in M$ that witnesses $\gamma \in M_{w_1=w_2}$ can be reached from β, so $\delta \in M_\beta$ and $\gamma \in (M_\beta)_{w_1=w_2}$.

To finish the proof, take $\beta \in M$ witnessing $\alpha \in M_{w_1=w_2}$ and $\gamma \in M_{w_1=w_2}$ that is reachable from α and satisfies $\gamma \oplus u \oplus v \in (M_\beta)_{w_1=w_2} \setminus M_{w_1=w_2}$ for a suitable pair of variables u, v. Take a $\delta \in M$ that witnesses $\gamma \in M_{w_1=w_2}$ and is reachable from β (we have shown above that such a δ exists). Since $\gamma \oplus u \oplus v \in (M_\beta)_{w_1=w_2} \setminus M_{w_1=w_2}$, we know that neither $\delta \oplus u \oplus v$ nor $\delta \oplus u \oplus v \oplus w_1 \oplus w_2$ lies in M, but at least one of these two tuples lies in M_β. If $\delta \oplus u \oplus v \in M_\beta$, we just use coverability of M to get $\delta \oplus u, \delta \oplus v \in M$, which translates to $\gamma \oplus u, \gamma \oplus v \in M_{w_1=w_2}$. If this is not the case, we know that $\delta, \delta \oplus u \oplus v \oplus w_1 \oplus w_2 \in M_\beta$ and $\delta \oplus u \oplus v \notin M_\beta$. We show that in this situation we have $\gamma \oplus u \in M_{w_1=w_2}$; the proof of $\gamma \oplus v \in M_{w_1=w_2}$ is analogous.

Using the even Δ-matroid property of M_β on δ and $\delta \oplus u \oplus v \oplus w_1 \oplus w_2$ in the variable u, we get that without loss of generality $\delta \oplus u \oplus w_1 \in M_\beta$ (recall that $\delta \oplus u \oplus v \notin M_\beta$). If $\delta \oplus u \oplus w_1 \notin M$, we can directly use coverability of M on δ to get that $\delta \oplus u \in M$, resulting in $\gamma \oplus u \in M_{w_1=w_2}$. If, on the other hand, $\delta \oplus u \oplus w_1 \in M$ and $\delta \oplus u \notin M$, then $\delta \oplus u \oplus w_1$ is reachable from β, so we can use coverability of M on $\delta \oplus u \oplus w_1 \in M$ and $\delta \oplus u \oplus v \oplus w_1 \oplus w_2 \in M_\beta \setminus M$ to get $\delta \oplus u \oplus w_1 \oplus w_2 \in M$, which again results in $\gamma \oplus u \in M_{w_1=w_2}$, finishing the proof.

3. This follows from the previous two points as any gadget construction is equivalent to a sequence of products followed by identifying variables.

□

Returning to edge CSP, the main notions from the even Δ-matroid case translate to the efficiently coverable Δ-matroid case easily. The definitions of valid, optimal, and non-optimal edge labeling may remain intact for coverable Δ-matroids, but we need to adjust our definition of a walk, which will now be allowed to end in a constraint.

Definition 3.6.4 (Walk for general Δ-matroids)**.** A *walk* q of length k or $k + 1/2$ in the instance I is a sequence $q_0C_1q_1C_2 \dots C_kq_k$ or $q_0C_1q_1C_2 \dots C_{k+1}$, respectively, where the variables q_{i-1}, q_i lie in the scope of the constraint C_i, and each edge $\{v, C\} \in \mathcal{E}$ is traversed at most once: vC and Cv occur in q at most once, and they do not occur simultaneously.

Given an edge labeling f and a walk q, we define the edge labeling $f \oplus q$ in the same way as before (see eq. (3.1)). We also extend the definitions of an f-walk and an augmenting f-walk for a valid edge labeling f: A walk q is an f-walk if $f \oplus q^\star$ is a valid edge labeling whenever $q^\star = q$ or $q^\star$ is a prefix of q that ends at a variable. An f-walk is called augmenting if: (1) it starts at a variable inconsistent in f, (2) it ends either at a different inconsistent variable or in a constraint, and (3) all variables inside of q (ie. not endpoints) are consistent in f. Note that if f is a valid edge labeling for which there is an augmenting f-walk, then f is non-optimal (since $f \oplus q$ is a valid edge labeling with 1 or 2 fewer inconsistent variables).

The main result of this section is tractability of efficiently coverable Δ-matroids.

Theorem 3.6.5 (Theorem 3.1.9 restated)**.** *Given an edge CSP instance I with efficiently coverable Δ-matroid constraints, an optimal edge labeling f of I can be found in time polynomial in $|I|$.*

The rough intuition of the algorithm for improving coverable Δ-matroid edge CSP instances is the following. When dealing with general Δ-matroids, augmenting f-walks may also end in a constraint – let us say that I has the augmenting f-walk q that ends in a constraint C. In that case, the parity of $f(D)$ and $(f \oplus p)(D)$ is the same for all $D \neq C$. If we guess the correct C (in fact, we will try all options) and flip its parity, we can, under reasonable conditions, find this augmentation via the algorithm for even Δ-matroids.

Not all Δ-matroids M are coverable (see Appendix A.4 for counterexample). However, we will show below how to efficiently cover many previously considered classes of

Δ-matroids. These would be co-independent [29], compact [41], local [24], and binary [34, 24] Δ-matroids.

Proposition 3.6.6. *The classes of co-independent, local, compact, and binary Δ-matroids are efficiently coverable.*

For the proof of this proposition as well as (some of) the definitions, we refer the reader to Appendix A.5.

The algorithm

The following lemma is a straightforward generalization of Lemma 3.5.2.

Lemma 3.6.7. *Let f, g be valid edge labelings of instance I (with general Δ-matroid constraints) such that g has fewer inconsistencies than f. Then we can, given f and g, compute in polynomial time an augmenting f-walk p (possibly ending in a constraint, in the sense of Definition 3.6.4).*

Proof. We proceed in two stages like in the proof of Lemma 3.5.2: First we modify g so that any variable consistent in f is consistent in g, then we look for the augmenting f-walk in $f \Delta g$. The only difference over Lemma 3.5.2 is that our g-walks and f-walks can now end in a constraint as well as in a variable.

First, we repeatedly modify the edge labeling g using the following procedure:

(1) Pick a variable $v \in V$ which is consistent in f, but not in g. (If no such v exists then go to the next paragraph). By the choice of v, there exists a unique edge $\{v, C\} \in f \Delta g$. If $g(C) \oplus v \in C$, replace g with $g \oplus vC$, then go to the beginning and repeat. Otherwise, pick variable $w \neq v$ in the scope of C such that $\{w, C\} \in f \Delta g$ and $g(C) \oplus v \oplus w \in C$ (it exists since C is a Δ-matroid and $g(C) \oplus v \notin C$). Replace g with $g \oplus (vCw)$ and then also go to the beginning and repeat.

It can be seen that g remains a valid edge labeling, and the number of inconsistencies in g never increases. Furthermore, each step decreases $|f \Delta g|$, so this procedure must terminate after at most $O(|\mathcal{E}|) = O(|V|)$ steps.

We now have valid edge labelings f, g such that f has more inconsistencies than g, and variables consistent in f are also consistent in g. In the second stage we will

maintain an f-walk p and the corresponding valid edge labeling $f^\star = f \oplus p$. To initialize, pick a variable $r \in V$ which is consistent in g but not in f, and set $p = r$ and $f^\star = f$. We then repeatedly apply the following step:

2. Let v be the endpoint of p. The variable v is consistent in g but not in $f^\star$, so there must exist a unique edge $\{v, C\} \in f^\star \Delta g$. If $f^\star(C) \oplus v \in C$, then output pC (an augmenting f-walk). Otherwise, pick variable $w \neq v$ in the scope of C such that $\{w, C\} \in f^\star \Delta g$ and $f^\star(C) \oplus v \oplus w \in C$ (it exists since C is a Δ-matroid and $f^\star(C) \oplus v \notin C$). Append vCw to the end of p, and accordingly replace $f^\star$ with $f^\star \oplus (vCw)$ (which is valid by the choice of w). As a result of this update, edges $\{v, C\}$ and $\{w, C\}$ are removed from $f^\star \Delta g$.

   If w is inconsistent in f, then output p (which is an augmenting f-walk) and terminate. Otherwise w is consistent in f (and thus in g) but not in $f^\star$; in this case, go to the beginning and repeat.

It is easy to verify that the p being produced is an f-walk. Also, each step decreases $|f^\star \Delta g|$ by 2, so this procedure must terminate after at most $O(|\mathcal{E}|) = O(|V|)$ steps and just like in the case of even Δ-matroids, the only way to terminate is to find an augmentation. □

Definition 3.6.8. Let f be a valid edge labeling of instance I with coverable Δ-matroid constraints. For a constraint $C \in \mathcal{C}$ and a Δ-matroid $C' \subseteq C$, we will denote by $I(f, C, C')$ the instance obtained from I by replacing the constraint relation of C by C' and the constraint relation of each $D \in \mathcal{C} \setminus \{C\}$ by the even Δ-matroid $D_{f(D)}$ (that comes from the covering).

Observe that f induces a valid edge labeling for $I(f, C, C)$. Moreover, if we choose $\alpha \in C$, then $I(f, C, \{\alpha\})$ is an edge CSP instance with even Δ-matroid constraints and hence we can find its optimal edge labeling by Algorithm 1 in polynomial time.

Lemma 3.6.9. *Let f be a non-optimal valid edge labeling of instance I with coverable Δ-matroid constraints. Then there exist $C \in \mathcal{C}$ and $\alpha \in C$ such that the optimal edge labeling for $I(f, C, \{\alpha\})$ has fewer inconsistencies than f.*

Proof. If f is non-optimal for I, then by Lemma 3.6.7 there exists an augmenting f-walk q in I. Take q such that no proper prefix of q is augmenting (i.e. we can not end early in a constraint). Let C be the last constraint in the walk and let $\alpha = (f \oplus q)(C)$.

We claim that $f \oplus q$ is also a valid edge labeling for the instance $I(f, C, \{\alpha\})$. This is enough to prove the statement, since this labeling of $I(f, C, \{\alpha\})$ has fewer inconsistencies than f. Since we choose α so that $(f \oplus q)(C) = \alpha$, we only need to consider constraints different from C. Assume that p is the shortest prefix of q such that $(f \oplus p)(D)$ is not reachable from $D_{f(D)}$ for some $D \neq C$ (if there is no such thing, then $(f \oplus q)(D) \in D_{f(D)}$ for all $D \neq C$). We let $p = p^\star xDy$. Since $(f \oplus p^\star)(D)$ is reachable from $f(D)$, but $(f \oplus p^\star)(D) \oplus x \oplus y$ is not, we must have $(f \oplus p^\star)(D) \oplus x \in D$. But then $p^\star xD$ is an augmenting f-walk in I that is shorter than p, a contradiction with the choice of p. □

Lemma 3.6.10. *Let f be a valid assignment for the instance I with coverable Δ-matroid constraints and let $C \in \mathcal{C}$ and $\alpha \in C$ be such that there exists a valid edge labeling g for the instance $I(f, C, \{\alpha\})$ with fewer inconsistencies than f. Then there exists an augmenting f-walk for I and it can be computed in polynomial time given g.*

Proof. We begin by noticing that both f and g are valid edge labelings for the instance $I(f, C, C)$. Since g has fewer inconsistencies than f, by Lemma 3.6.7 we can compute an f-walk q which is augmenting in $I(f, C, C)$. It is easy to examine q and check if some proper prefix of q is an augmenting f-walk for I (ending in a constraint). If that happens we are done, so let us assume that this is not the case. We will show that then q itself must be an augmenting f-walk for I.

First assume that every prefix of q with integral length is an f-walk in I. Then either q is of integral length and we are done (q is its own prefix), or q ends in a constraint. If it is the latter, q must end in C, since that is the only constraint of $I(f, C, C)$ that is not forced to be an even Δ-matroid. But the constraint relation C is the same for both I and $I(f, C, C)$, so flipping the last edge of q is allowed in I.

Let now p be the shortest prefix of q with integral length which is not an f-walk in I. We can write $p = p^\star xDy$ for suitable x, y, D. The constraint relation of D must be different in I and $I(f, C, C)$, so $D \neq C$. By the choice of p, for any prefix r of $p^\star$ of integral length we have $(f \oplus r)(D) \in D$ and moreover the tuple $(f \oplus r)(D)$ is reachable

from $f(D)$. (If not, take the shortest counterexample r. Obviously, $r = r^\star uDv$ for some variables u, v and a suitable $r^\star$. Since $(f \oplus r^\star)(D) \in D$ is reachable from $f(D)$ and $(f \oplus r^\star)(D) \oplus u \oplus v$ is not, we get $(f \oplus r^\star)(D) \oplus u \in D$ and $r^\star uD$ is augmenting in I, which is a contradiction.) This holds also for $r = p^\star$, so $(f \oplus p^\star)(D)$ is reachable from $f(D)$.

To finish the proof, let $\beta^\star = (f \oplus p^\star)(D)$ and $\beta = (f \oplus p)(D)$. We showed that $\beta^\star \in D$ is reachable from $f(D)$. Also, $\beta^\star \oplus x \oplus y = \beta \in D_{f(D)} \setminus D$. Then by the definition of coverable Δ-matroids we have $\beta^\star \oplus x \in D$. Thus $p^\star xD$ is an augmenting f-walk in I and we are done.

It is easy to see that all steps of the proof can be made algorithmic. □

Now the algorithm is very simple to describe. Set some valid edge labeling f and repeat the following procedure. For all pairs (C, α) with $\alpha \in C$ and $C \in \mathcal{C}$, call Algorithm 1 on the instance $I(f, C, \{\alpha\})$ (computing the instance $I(f, C, \{\alpha\})$ can be done in polynomial time because all constraints of I come from an efficiently coverable class). If for some (C, α) we obtained an edge labeling of $I(f, C, \{\alpha\})$ with fewer inconsistencies, use Lemma 3.6.10 to get an augmenting f-walk for I. Otherwise, we have proved that the original f was optimal.

The algorithm is correct due to Lemma 3.6.9. The running time is polynomial because there are at most $|I|$ pairs (C, α) such that $\alpha \in C$ and at most $|I|$ inconsistencies in the initial edge labeling, so the (polynomial) Algorithm 1 gets called at most $|I|^2$ times.

Even-zebras are coverable (but not vice versa)

The paper [30] introduces several classes of zebra Δ-matroids. For simplicity, we will consider only one of them: $\mathcal{C}$-zebras.

Definition 3.6.11. Let $\mathcal{C}$ be a subclass of even Δ-matroids. A Δ-matroid M is a $\mathcal{C}$-zebra if for every $\alpha \in M$ there exists an even Δ-matroid M_α in $\mathcal{C}$ that contains all tuples in M of the same parity as α and such that for every $\beta \in M$ and every $u, v \in V$ such that $\beta \oplus u \oplus v \in M_\alpha \setminus M$ we have $\beta \oplus v, \beta \oplus u \in M$.

In [30], the authors show a result very much similar to Theorem 3.1.9, but for $\mathcal{C}$-zebras: In our language, the result states that if one can find optimal labelings for

$\mathrm{CSP}_{\mathsf{EDGE}}(\mathcal{C})$ in polynomial time, then the same is true for the edge CSP with $\mathcal{C}$-zebra constraints. In the rest of this section, we show that coverable Δ-matroids properly contain the class of $\mathcal{C}$-zebras with $\mathcal{C}$ equal to all even Δ-matroids (this is the largest $\mathcal{C}$ allowed in the definition of $\mathcal{C}$-zebras) – we will call this class *even-zebras* for short.

Observation 3.6.12. *Let M be an even-zebra. Then M is coverable.*

Proof. Given $\alpha \in M$, we can easily verify that the matroids M_α satisfy all conditions of the definition of coverable Δ-matroids: Everything reachable from α has the same parity as α and the last condition from the definition of even-zebras is identical to coverability. □

Moreover, it turns out that the inclusion is proper: There exists a Δ-matroid that is coverable, but is not an even-zebra.

Let us take $M = \{(0,0,0),\ (1,1,0),\ (1,0,1),\ (0,1,1),\ (1,1,1)\}$ and consider $N = M \times M$. It is easy to verify that M is a Δ-matroid that is an even-zebra with the sets M_α equal to $\{(0,0,0),\ (1,1,0),\ (1,0,1),\ (0,1,1)\}$ and $\{(1,1,1)\}$, respectively, and thus M is coverable.

Since coverable Δ-matroids are closed under direct products, N is also coverable. However, N is not an even-zebra: Assume that there exists a set N_α that contains all tuples of N of odd parity and satisfies the zebra condition. Then the two tuples $(1,1,1,0,0,0)$ and $(1,1,0,1,1,1)$ of N belong to N_α. Since N_α is an even Δ-matroid, switching in the third coordinate yields that N contains the tuple $(1,1,0,1,0,0)$ (this is without loss of generality; the other possibilities are all symmetric). This tuple is not a member of N, yet we got it from $(1,1,1,0,0,0) \in N$ by switching the third and fourth coordinate. So in order for the zebra property to hold, we need $(1,1,1,1,0,0) \in N$, a contradiction.

The above example also shows that even-zebras, unlike coverable Δ-matroids, are not closed under direct products.

3.7 Discussion

We have given an algorithm, that solves Boolean edge CSPs for all even Δ-matroids and a large gadget-closed class of Δ-matroids. However, many follow-up questions remain.

Open Problem 3.7.1. *Is* $\mathrm{CSP}_{\mathsf{EDGE}}(\Gamma)$ *solvable in polynomial time if all relations in* Γ *are* Δ*-matroids?*

We believe that the tractability can be pushed beyond efficiently coverable Δ-matroids but at the same time do not have enough evidence to phrase Open Problem 3.7.1 as a conjecture.

In that regard, further investigation of Algorithm 1 could be useful. With minor modifications this algorithm can be also phrased to work with Δ-matroid constraints. We were able to show its correctness for a certain subclass of Δ-matroids but at the same time lifting augmenting f-walks broke terribly, especially when multiple contractions were involved. Non coverable Δ-matroid from Appendix A.4 was a particularly rich source of counterexamples.

Open Problem 3.7.2. *What is the power of Algorithm 1, when naturally extended to* Δ*-matroids?*

Algorithm 1 has one additional undesirable feature. While Edmonds' algorithm has implementations that maintain one forest through all levels of contraction, our Algorithm 1 is not able to do so. It seems a different contraction mechanism would be needed.

Open Problem 3.7.3. *Can Algorithm 1 be modified as to maintain one forest through all levels of contraction and uncontraction?*

A positive answer to this question would be beneficial not only for efficiency reasons but it would also unlock the opportunity to move towards weighted edge CSP by mimicking the primal-dual algorithm for weighted matching [57].

Open Problem 3.7.4. *Is there a framework for weighted edge CSP and an algorithm that would extend maximum weight perfect matching algorithm in analogous fashion as how Algortihm 1 generalizes the unweighted version?*

Bibliography

[1] L. Barto. The dichotomy for conservative constraint satisfaction problems revisited. In *Proceedings of the 26th IEEE Symposium on Logic in Computer Science (LICS'11)*, pages 301–310. IEEE Computer Society, 2011.

[2] L. Barto. The collapse of the bounded width hierarchy. *Journal of Logic and Computation*, 26(3):923–943, 2016.

[3] L. Barto and M. Kozik. Absorbing subalgebras, cyclic terms and the constraint satisfaction problem. *Logical Methods in Computer Science*, 8(1):1–26, 2012.

[4] L. Barto and M. Kozik. Constraint satisfaction problems solvable by local consistency methods. *Journal of the ACM*, 61(1):Article 3, 2014.

[5] L. Barto and M. Kozik. Robustly solvable constraint satisfaction problems. *SIAM Journal on Computing*, 45(4):1646–1669, 2016.

[6] L. Barto, M. Kozik, and T. Niven. The CSP dichotomy holds for digraphs with no sources and no sinks (a positive answer to a conjecture of Bang-Jensen and Hell). *SIAM Journal on Computing*, 38(5):1782–1802, 2009.

[7] A. Bouchet. Multimatroids I. coverings by independent sets. *SIAM Journal on Discrete Mathematics*, 10(4):626–646, 1997.

[8] Y. Boykov, O. Veksler, and R. Zabih. Markov random fields with efficient approximations. In *Computer Vision and Pattern Recognition*, pages 648–655. IEEE Computer Society, 1998.

[9] J. Brown-Cohen and P. Raghavendra. Correlation Decay and Tractability of CSPs. In *43rd International Colloquium on Automata, Languages, and Programming*

(ICALP 2016), volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 79:1–79:13, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[10] A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *Journal of the ACM*, 53(1):66–120, 2006.

[11] A. Bulatov. Complexity of conservative constraint satisfaction problems. *ACM Transactions on Computational Logic*, 12(4):Article 24, 2011.

[12] A. Bulatov, P. Jeavons, and A. Krokhin. Classifying the complexity of constraints using finite algebras. *SIAM Journal on Computing*, 34(3):720–742, 2005.

[13] J.-Y. Cai, P. Lu, and M. Xia. Computational complexity of holant problems. *SIAM J. Comput.*, 40(4):1101–1132, July 2011.

[14] S. O. Chan, J. R. Lee, P. Raghavendra, and D. Steurer. Approximate constraint satisfaction requires large LP relaxations. *J. ACM*, 63(4):34:1–34:22, 2016.

[15] D. Cohen, M. Cooper, P. Creed, P. Jeavons, and S. Živný. An algebraic theory of complexity for discrete optimisation. *SIAM Journal on Computing*, 42(5):1915–1939, 2013.

[16] D. Cohen, M. Cooper, and P. Jeavons. An algebraic characterisation of complexity for valued constraints. In *CP'06*, volume 4204 of *LNCS*, pages 107–121, 2006.

[17] D. Cohen, M. Cooper, P. Jeavons, and A. Krokhin. Supermodular functions and the complexity of Max CSP. *Discrete Applied Mathematics*, 149(1-3):53–72, 2005.

[18] D. Cohen and P. Jeavons. The complexity of constraint languages. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, chapter 8. Elsevier, 2006.

[19] D. A. Cohen, M. C. Cooper, P. G. Jeavons, and A. A. Krokhin. The Complexity of Soft Constraint Satisfaction. *Artificial Intelligence*, 170(11):983–1016, 2006.

[20] Y. Crama and P. Hammer. *Boolean Functions – Theory, Algorithms and Applications*. Cambridge University Press, 2011.

[21] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*, volume 7 of *SIAM Monographs on Discrete Mathematics and Applications*. 2001.

[22] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[23] N. Creignou, P. Kolaitis, and H. Vollmer, editors. *Complexity of Constraints*, volume 5250 of *LNCS*. Springer, 2008.

[24] V. Dalmau and D. Ford. *Mathematical Foundations of Computer Science 2003: 28th International Symposium, MFCS 2003, Bratislava, Slovakia, August 25-29, 2003. Proceedings*, chapter Generalized Satisfiability with Limited Occurrences per Variable: A Study through Delta-Matroid Parity, pages 358–367. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[25] V. Deineko, P. Jonsson, M. Klasson, and A. Krokhin. The approximability of Max CSP with fixed-value constraints. *Journal of the ACM*, 55(4):Article 16, 2008.

[26] Z. Dvořák and M. Kupec. On planar Boolean CSP. In *ICALP '15*, volume 9134 of *Lecture Notes in Computer Science*, pages 432–443. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[27] J. Edmonds. Path, trees, and flowers. *Canadian J. Math.*, 17:449–467, 1965.

[28] A. Ene, J. Vondrák, and Y. Wu. Local distribution and the symmetry gap: Approximability of multiway partitioning problems. In *SODA*, pages 306–325, 2013. Update at arXiv1503.03905.

[29] T. Feder. Fanout limitations on constraint systems. *Theoretical Computer Science*, 255(1–2):281–293, 2001.

[30] T. Feder and D. Ford. Classification of bipartite Boolean constraint satisfaction through delta-matroid intersection. *SIAM Journal on Discrete Mathematics*, 20(2):372–394, 2006.

[31] T. Feder and M. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM Journal on Computing*, 28:57–104, 1998.

[32] P. Fulla and S. Živný. A galois connection for weighted (relational) clones of infinite size. *ACM Trans. Comput. Theory*, 8(3):9:1–9:21, May 2016.

[33] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[34] J. F. Geelen, S. Iwata, and K. Murota. The linear delta-matroid parity problem. *Journal of Combinatorial Theory, Series B*, 88(2):377 – 398, 2003.

[35] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. *J. ACM*, 35(4):921–940, Oct. 1988.

[36] G. Gottlob, G. Greco, and F. Scarcello. Tractable optimization problems through hypergraph-based structural restrictions. In *Proceedings of ICALP'09*, pages 16–30, 2009.

[37] J. Håstad. Every 2-CSP allows nontrivial approximation. *Computational Complexity*, 17(4):549–566, 2008.

[38] J. Hopcroft and R. Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, Oct. 1974.

[39] A. Huber, A. Krokhin, and R. Powell. Skew bisubmodularity and valued CSPs. *SIAM Journal on Computing*, 43(3):1064–1084, 2014.

[40] P. Idziak, P. Markovic, R. McKenzie, M. Valeriote, and R. Willard. Tractability and learnability arising from algebras with few subpowers. *SIAM Journal on Computing*, 39(7):3023–3037, 2010.

[41] G. Istrate. Looking for a version of Schaefer's dichotomy theorem when each variable occurs at most twice. Technical report, University of Rochester, Rochester, NY, USA, 1997.

[42] P. Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200:185–204, 1998.

[43] P. Jeavons, D. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the ACM*, 44:527–548, 1997.

[44] P. Jonsson, M. Klasson, and A. Krokhin. The approximability of three-valued Max CSP. *SIAM Journal on Computing*, 35(6):1329–1349, 2006.

[45] P. Jonsson, F. Kuivinen, and J. Thapper. Min CSP on four elements: Moving beyond submodularity. In *Proceedings of CP'11*, pages 438–453, 2011.

[46] P. Jonsson and G. Nordh. Introduction to the Maximum Solution problem. In *Complexity of Constraints*, volume 5250 of *LNCS*, pages 255–282. 2008.

[47] A. Kazda, V. Kolmogorov, and M. Rolínek. Even delta-matroids and the complexity of planar Boolean CSPs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'17)*, pages 307–326. SIAM, 2017.

[48] P. Kolaitis and M. Vardi. Conjunctive-query containment and constraint satisfaction. *Journal of Computer and System Sciences*, 61:302–332, 2000.

[49] V. Kolmogorov, A. Krokhin, and M. Rolínek. The complexity of general-valued csps. In *Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, FOCS '15, pages 1246–1258, Washington, DC, USA, 2015. IEEE Computer Society.

[50] V. Kolmogorov, M. Rolínek, and R. Takhanov. Effectiveness of structural restrictions for hybrid csps. In *Algorithms and Computation: 26th International Symposium, ISAAC 2015, Nagoya, Japan, December 9-11, 2015, Proceedings*, pages 566–577, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[51] V. Kolmogorov, J. Thapper, and S. Živný. The power of linear programming for general-valued CSPs. *SIAM Journal on Computing*, 44(1):1—-36, 2015.

[52] V. Kolmogorov and S. Živný. The complexity of conservative valued CSPs. *Journal of the ACM*, 60(2):Article 10, 2013.

[53] M. Kozik and J. Ochremiak. Algebraic properties of valued constraint satisfaction problem. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 846–858, 2015.

[54] M. Kozik and J. Ochremiak. Algebraic properties of valued constraint satisfaction problem. arXiv1403.0476, 2015.

[55] A. Krokhin and S. Živný. The complexity of valued CSPs. To appear in The Constraint Satisfaction Problem: Complexity and Approximability, 2017.

[56] S. L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

[57] L. Lovász and M. Plummer. *Matching Theory*. AMS Chelsea Publishing Series. American Mathematical Soc., 2009.

[58] D. Marx. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. *J. ACM*, 60(6):Article 42, 2013.

[59] U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Information Sciences*, 7:95–132, 1974.

[60] B. M. E. Moret. Planar NAE3SAT is in P. *SIGACT News*, 19(2):51–54, June 1988.

[61] P. Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *STOC'08*, pages 245–254, 2008.

[62] F. Rossi, P. van Beek, and T. Walsh, editors. *Handbook of Constraint Programming*. Elsevier, 2006.

[63] T. Schaefer. The complexity of satisfiability problems. In *STOC'78*, pages 216–226, 1978.

[64] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1986.

[65] R. Takhanov. A dichotomy theorem for the general minimum cost homomorphism problem. In *STACS'10*, pages 657–668, 2010.

[66] J. Thapper and S. Živný. Sherali-Adams relaxations for valued CSPs. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 1058–1069, 2015.

[67] J. Thapper and S. Živný. The complexity of finite-valued CSPs. In *Proceedings of the 45th ACM Symposium on the Theory of Computing (STOC'13)*, pages 695–704. ACM, 2013.

[68] J. Thapper and S. Živný. The power of Sherali-Adams relaxations for general-valued CSPs. arXiv1606.02577, 2016.

[69] E. P. K. Tsang. *Foundations of constraint satisfaction*. Computation in cognitive science. Academic Press, London and San Diego, 1993.

[70] H. Uppman. The Complexity of Three-Element Min-Sol and Conservative Min-Cost-Hom. In *Proceedings of the 40th International Colloquium on Automata, Languages, and Programming (ICALP'13)*, volume 7965 of *Lecture Notes in Computer Science*, pages 804–815. Springer, 2013.

[71] H. Uppman. Computational complexity of the extended minimum cost homomorphism problem on three-element domains. In *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014)*, pages 651–662, 2014.

[72] M. Wainwright and M. Jordan. Graphical models, exponential families, and variational inferences. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

A Appendices

A.1 Proofs for Section 2.4

In this section we prove the properties of graph $(\mathbb{G}, E)$ stated in Section 2.4.

Proof of Proposition 2.4.1

Part (a) We have $\mathbf{g} = \mathbb{1}^{s_1 \cdots s_k}$ and $\mathbf{h} = \mathbb{1}^{s_{k+1} \cdots s_\ell}$ for some $s_1, \ldots, s_\ell \in \mathrm{supp}(\omega)$ and $0 \leq k \leq \ell$. Therefore, $\mathbf{h} \circ \mathbf{g} = [\mathbb{1}^{s_\ell} \circ \ldots \circ \mathbb{1}^{s_{k+1}}] \circ [\mathbb{1}^{s_k} \circ \ldots \circ \mathbb{1}^{s_1}] = \mathbb{1}^{s_1 \cdots s_\ell} \in \mathbb{G}$. Also, $\mathbf{h} \circ \mathbf{g} = \mathbf{g}^{s_{k+1} \cdots s_\ell}$, and so there is path from $\mathbf{g}$ to $\mathbf{h} \circ \mathbf{g}$ in $(\mathbb{G}, E)$. Since no edges leave the strongly connected component $\mathbb{H}$, we obtain that if $\mathbf{g} \in \mathbb{H}$ then $\mathbf{h} \circ \mathbf{g} \in \mathbb{H}$.

Part (b) Pick $\hat{\mathbf{g}} \in \mathbb{H}$. Since $\mathbb{H}'$ is strongly connected, there is a path from $\hat{\mathbf{g}} \circ \mathbf{g}' \in \mathbb{H}^*$ to $\mathbf{g}' \in \mathbb{H}'$ in $(\mathbb{G}, E)$, i.e. $\mathbf{g}' = [\hat{\mathbf{g}} \circ \mathbf{g}']^{s_1 \cdots s_k} = \mathbf{h} \circ \hat{\mathbf{g}} \circ \mathbf{g}'$ where $\mathbf{h} = \mathbb{1}^{s_1 \cdots s_k}$. It can be checked that mapping $\mathbf{g} = \mathbf{h} \circ \hat{\mathbf{g}}$ has the desired properties.

Part (c) By assumption, $x = \mathbf{g}^*(y)$ for some $\mathbf{g}^* \in \mathbb{H}^* \in \mathsf{Sinks}(\mathbb{G}, E)$ and $y \in [D^n]^m$. By part (b) there exists $\mathbf{g} \in \mathbb{H}$ satisfying $\mathbf{g} \circ \mathbf{g}^* = \mathbf{g}^*$. We get that $\mathbf{g}(x) = \mathbf{g}(\mathbf{g}^*(y)) = (\mathbf{g} \circ \mathbf{g}^*)(y) = \mathbf{g}^*(y) = x$.

Proof of Proposition 2.4.2

By assumption, we have $\hat{x} = \mathbf{g}^*(y)$ for some $\mathbf{g}^* \in \mathbb{G}^*$, $y \in D^m$ and $x = \mathbf{h}(\hat{x})$ for some $\mathbf{h} \in \mathbb{G}$.

Part (a) We have $x = (\mathbf{h} \circ \mathbf{g}^*)(y)$ with $\mathbf{h} \circ \mathbf{g}^* \in \mathbb{G}^*$; this establishes the claim.

Part (b) Let $\mathbb{H} \in \mathsf{Sinks}(\mathbb{G}, E)$ be the strongly connected component to which $\mathbf{g}^*$ belongs. There exists a path in $(\mathbb{H}, E[\mathbb{H}])$ from $\mathbf{h} \circ \mathbf{g}^* \in \mathbb{H}$ to $\mathbf{g}^* \in \mathbb{H}$, i.e. $\mathbf{g}^* = \mathbb{1}^{s_1 \cdots s_k} \circ \mathbf{h} \circ \mathbf{g}^*$ for some $s_1, \ldots, s_k \in \mathrm{supp}(\omega) = \mathbb{G}$. Define $\mathbf{g} = \mathbb{1}^{s_1 \cdots s_k} \in \mathbb{G}$, then $\mathbf{g}^* = \mathbf{g} \circ \mathbf{h} \circ \mathbf{g}^*$. We have $\hat{x} = \mathbf{g}^*(y) = (\mathbf{g} \circ \mathbf{h} \circ \mathbf{g}^*)(y) = (\mathbf{g} \circ \mathbf{h})(\hat{x}) = \mathbf{g}(x)$, as claimed.

Proof of Theorem 2.4.3

First, we make the following observation.

Proposition A.1.1. *Suppose vector ρ is a fractional polymorphism of Γ of arity $m \to m$ and $\mathbf{g} \in \mathrm{supp}(\rho)$. Then the following vector is also a fractional polymorphism of Γ of arity $m \to m$:*

$$\rho[\mathbf{g}] \;=\; \rho + \frac{\rho(\mathbf{g})}{2}\left[-\chi_{\mathbf{g}} + \sum_{s \in \omega} \omega(s)\chi_{\mathbf{g}^s}\right] \tag{A.1}$$

Proof. Denote the vector in the square brackets as δ. Consider function $f \in \Gamma$ and labeling $x \in [\mathrm{dom}\, f]^m$. Since ρ is a fractional polymorphism of Γ, we have $\mathbf{g}(x) \in [\mathrm{dom}\, f]^m$. We can write

$$\sum_{\mathbf{h} \in \mathrm{supp}(\rho[\mathbf{g}])} \delta(\mathbf{h}) f^m(\mathbf{h}(x)) = -f^m(\mathbf{g}(x)) + \sum_{s \in \mathrm{supp}(\omega)} \omega(s) f^m(\mathbf{g}^s(x)) \le 0$$

where the last inequality follows from condition (2.7a) applied to labelings $\mathbf{g}(x)$. Thus, adding the extra term to ρ in (A.1) will not violate the fractional polymorphism inequality for any $x \in [\mathrm{dom}\, f]^m$. □

Note that $\mathrm{supp}(\rho[\mathbf{g}]) = \mathrm{supp}(\rho) \cup \{\mathbf{g}^s \mid s \in \mathrm{supp}(\omega)\}$ for $\mathbf{g} \in \mathrm{supp}(\rho)$.

We claim that Γ admits a fractional polymorphism $\widehat{\rho}$ with $\mathrm{supp}(\widehat{\rho}) = \mathbb{G}$. Indeed, we can start with vector $\rho = \chi_{\mathbb{1}}$ and then repeatedly modify it as $\rho \leftarrow \rho[\mathbf{g}]$ for mappings $\mathbf{g} \in \mathrm{supp}(\rho)$ that haven't appeared before; after $|\mathbb{G}| - 1$ steps we get a vector $\widehat{\rho}$ with the claimed property.

Let Ω be the set of fractional polymorphisms ρ of Γ with $\mathrm{supp}(\rho) \subseteq \mathbb{G}$ that satisfy $\rho(\mathbf{g}) \ge \widehat{\rho}(\mathbf{g})$ for all $\mathbf{g} \in \widehat{\mathbb{G}}$. Set Ω is non-empty since it contains $\widehat{\rho}$. Let ρ be a vector in

Ω that maximizes $\rho(\widehat{\mathbb{G}}) = \sum_{\mathbf{g}\in\widehat{\mathbb{G}}} \rho(\mathbf{g})$. (This maximum is attained since Ω is a compact subset of $\mathbb{R}^{|\mathbb{G}|}$). We claim that $\mathrm{supp}(\rho) = \widehat{\mathbb{G}}$. Indeed, the inclusion $\widehat{\mathbb{G}} \subseteq \mathrm{supp}(\rho)$ is by construction. Suppose there exists $\mathbf{g} \in \mathrm{supp}(\rho) - \widehat{\mathbb{G}}$. By the condition of Theorem 2.4.3 there exists a path $\mathbf{g}_0, \ldots, \mathbf{g}_k$ in $(\mathbb{G}, E)$ from $\mathbf{g}_0 = \mathbf{g}$ such that $\mathbf{g}_0, \ldots, \mathbf{g}_{k-1} \in \mathbb{G} - \widehat{\mathbb{G}}$ and $\mathbf{g}_k \in \widehat{\mathbb{G}}$. It can be checked that vector $\rho' = \rho[\mathbf{g}_0] \ldots [\mathbf{g}_{k-1}]$ satisfies $\rho' \in \Omega$, $\rho'(\mathbf{g}) \geq \rho(\mathbf{g})$ for $\mathbf{g} \in \widehat{\mathbb{G}}$, and $\rho'(\mathbf{g}_k) > \rho(\mathbf{g}_k)$. This contradicts the choice of ρ.

Proof of Theorem 2.4.4(a)

Consider component $\mathbb{H} \in \mathsf{Sinks}(\mathbb{G}, E)$, and denote $\mathbb{H}^* = \arg\min\{f^m(\mathbf{g}(x)) \mid \mathbf{g} \in \mathbb{H}\}$. We claim that $\mathbb{H}^* = \mathbb{H}$. Indeed, consider $\mathbf{g} \in \mathbb{H}^*$. Applying inequality (2.7a) to labelings $\mathbf{g}(x) \in [\mathrm{dom}\, f]^m$ gives

$$\sum_{s\in\mathrm{supp}(\omega)} \omega(s) f^m(\mathbf{g}^s(x)) \;\leq\; f^m(\mathbf{g}(x)) \qquad \forall x \in [\mathrm{dom}\, f]^m \tag{A.2}$$

For each $s \in \mathrm{supp}(\omega)$ we have $\mathbf{g}^s \in \mathbb{H}$ and thus $f^m(\mathbf{g}^s(x)) \geq f^m(\mathbf{g}(x))$. This means that $f^m(\mathbf{g}^s(x)) = f^m(\mathbf{g}(x))$. We showed that if $\mathbf{g} \in \mathbb{H}^*$ and $(\mathbf{g}, \mathbf{h}) \in E$ then $\mathbf{h} \in \mathbb{H}^*$. Since $\mathbb{H}$ is a strongly connected component of $(\mathbb{G}, E)$, we conclude that $\mathbb{H} = \mathbb{H}^*$.

We showed that $f^m(\mathbf{g}(x))$ is the same for all $\mathbf{g} \in \mathbb{H}$. By Proposition 2.4.1(c) there exists $\mathbf{h} \in \mathbb{H}$ with $\mathbf{h}(x) = x$, and therefore $f^m(\mathbf{g}(x)) = f^m(\mathbf{h}(x)) = f^m(x)$ for all $\mathbf{g} \in \mathbb{H}$. Since this holds for any $\mathbb{H} \in \mathsf{Sinks}(\mathbb{G}, E)$, the claim follows.

Proof of Theorem 2.4.4(b)

We mainly follow an argument from [67] (although without using the language of Markov chains, relying on the Farkas lemma instead, as in [51]).

Let $(\mathbb{G}^*, E')$ be the subgraph of $(\mathbb{G}, E)$ induced by $\mathbb{G}^*$. For an edge $(\mathbf{g}, \mathbf{h}) \in E'$, define positive weight $w(\mathbf{g}, \mathbf{h}) = \sum_{s\in\mathrm{supp}(\omega):\mathbf{g}^s=\mathbf{h}} \omega(s)$. Note that we have $\sum_{\mathbf{h}:(\mathbf{g},\mathbf{h})\in E'} w(\mathbf{g}, \mathbf{h}) = 1$ for all $\mathbf{g} \in \mathbb{G}^*$.

We claim that there exists vector $\lambda \in \mathbb{R}^{\mathbb{G}^*}_{\geq 0}$ that satisfies

$$\sum_{\mathbf{g}:(\mathbf{g},\mathbf{h})\in E'} w(\mathbf{g},\mathbf{h})\lambda_{\mathbf{g}} - \lambda_{\mathbf{h}} = 0 \qquad \forall \mathbf{h} \in \mathbb{G}^* \tag{A.3a}$$

$$\sum_{\mathbf{g}\in\mathbb{G}^*} \lambda_{\mathbf{g}} = 1 \tag{A.3b}$$

Indeed, suppose system (A.3) does not have a solution. By Farkas Lemma (see Lemma 2.5.4), there exists a vector $y \in \mathbb{R}^{\mathbb{G}^*}$ and a scalar $z \in \mathbb{R}$ such that

$$z - y_{\mathbf{g}} + \sum_{\mathbf{h}:(\mathbf{g},\mathbf{h})\in E'} w(\mathbf{g},\mathbf{h}) y_{\mathbf{h}} \geq 0 \qquad \forall \mathbf{g} \in \mathbb{G}^* \tag{A.4a}$$

$$z < 0 \tag{A.4b}$$

Consider $\mathbf{g} \in \mathbb{G}^*$ with the maximum value of $y_{\mathbf{g}}$. We have

$$0 \leq z - y_{\mathbf{g}} + \sum_{\mathbf{h}:(\mathbf{g},\mathbf{h})\in E'} w(\mathbf{g},\mathbf{h}) y_{\mathbf{h}} \leq z - y_{\mathbf{g}} + \sum_{\mathbf{h}:(\mathbf{g},\mathbf{h})\in E'} w(\mathbf{g},\mathbf{h}) y_{\mathbf{g}} = z - y_{\mathbf{g}} + y_{\mathbf{g}} = z$$

This contradicts (A.4b), and thus proves that vector $\lambda \geq 0$ satisfying (A.3) exists. Next, we will show that this vector satisfies the property of Theorem 2.4.4(b).

Let us rewrite condition (2.7b) as follows:

$$\sum_{s\in \mathrm{supp}(\omega)} \omega(s) f(x^{\mathbf{g}^s i}) \leq \frac{1}{m-1} \sum_{j\in[m]-\{i\}} f(x^{\mathbf{g}j}) \qquad \forall \mathbf{g} \in \mathbb{G}^*, i \in [m] \tag{A.5}$$

Multiplying this inequality by $\lambda_{\mathbf{g}}$ and summing over $\mathbf{g} \in \mathbb{G}^*$ (for a fixed $i \in [m]$) gives

$$\sum_{\mathbf{g}\in\mathbb{G}^*} \sum_{\mathbf{h}:(\mathbf{g},\mathbf{h})\in E'} w(\mathbf{g},\mathbf{h})\lambda_{\mathbf{g}} f(x^{\mathbf{h}i}) \leq \frac{1}{m-1} \sum_{\mathbf{g}\in\mathbb{G}^*} \lambda_{\mathbf{g}} \sum_{j\in[m]-\{i\}} f(x^{\mathbf{g}j}) \qquad \forall i \in [m] \tag{A.6}$$

Rearranging terms gives

$$\sum_{\mathbf{h}\in\mathbb{G}^*} \left[\sum_{\mathbf{g}:(\mathbf{g},\mathbf{h})\in E'} w(\mathbf{g},\mathbf{h})\lambda_{\mathbf{g}} \right] f(x^{\mathbf{h}i}) \leq \frac{1}{m-1} \sum_{j\in[m]-\{i\}} \sum_{\mathbf{g}\in\mathbb{G}^*} \lambda_{\mathbf{g}} f(x^{\mathbf{g}j}) \qquad \forall i \in [m] \tag{A.7}$$

By (A.3a) the expression in the square brackets equals $\lambda_{\mathbf{h}}$, and therefore (A.7) can be

rewritten as

$$f_i^\lambda(x) \leq \frac{1}{m-1} \sum_{j \in [m]-\{i\}} f_j^\lambda(x) \qquad \forall i \in [m] \tag{A.8}$$

Consider index $i \in [m]$ with the maximum value of $f_i^\lambda(x)$. We have $f_i^\lambda(x) \geq f_j^\lambda(x)$ for all $j \in [m] - \{i\}$, which together with (A.8) gives $f_i^\lambda(x) = f_j^\lambda(x)$ for all $j \in [m] - \{i\}$, as claimed.

A.2 Expressing special unary functions

In this part of the appendix we present a partial result that came from attempting to adapt certain techniques from the finite-valued case [67]. In particular, how a binary symmetric fractional polymorphism is constructed using special unary functions. We were not able to finish this line of argument but for the sake of completeness we include a partial result. Perhaps, it is of independent interest.

We aimed to prove that if (certain special) Γ admits a cyclic fractional polymorphism, then it admits a binary symmetric fractional polymorphism (a fact that we, in the end, proved by other means – and with slightly milder assumptions – as a part of Theorem 2.3.4).

The strategy used in [67] to construct such polymorphism given that Γ cannot express certain NP-Hard function (instead of operating with a cyclic polymorphism) could be vaguely described as follows:

(i) Express certain special unary functions (with a bit of additional expressive power, namely fixing variables).

(ii) Show that a candidate binary fractional polymorphism “acts submodular” on many pairs of domain elements.

(iii) Use the submodular behaviour to imply symmetry of the fractional polymorphism, thus concluding the argument.

We were not able to follow this strategy entirely; the inability to guarantee finite costs was particularly painful in (iii). However, we managed to obtain results similar to (i) using alternative techniques (and a cyclic fractional polymorphism as a starting point).

We will need a weaker notion of a core than the rigid core introduced in Section 2.1.

Definition A.2.1. Language Γ is called a *core* if for every unary fractional polymorphism φ of Γ, $\operatorname{supp}(\varphi)$ contains only bijections.

Restriction to core languages is a recurring feature in many VCSP and CSP proofs. Also for our purposes this restriction is without loss of generality, although we omit the proof. For a more thorough treatment of core languages we refer to [54, 39]. For the rest of this section we will focus on a finite core and block-finite language Γ on domain D with partitions $\{D_v \mid v \in V\}$.

Before formally stating the main result of this section, let us define a certain graph, whose connectivity will be in the spotlight.

For an arbitrary language Γ on a domain D consider an undirected graph $T(\Gamma)$ on vertex set D with $a, b \in D$, connected by an edge if there is a finite-valued $u_{ab} \in \langle\Gamma\rangle$ such that $\arg\min u_{ab} = \{a, b\}$.

The result on connectivity of T from [67] operates (without loss of generality) with a language Γ_c that is obtained from functions in Γ by fixing values for some variables, e.g., $g(x, y) = f(x, a, b, y) \in \Gamma_c$ if $f \in \Gamma$ and $a, b \in D$.

Theorem A.2.2 ([67])**.** *Let Γ be a core finite-valued language. Then $T(\Gamma_c)$ is connected.*

We do not have the language Γ_c at hand since with this reduction we would lose (otherwise essential) block-finiteness. However, additionally using a cyclic fractional polymorphism saves the day and we can formulate our result as follows.

Theorem A.2.3. *Let Γ be a core valued language that is block-finite with domain D partitioned into $\{D_v \mid v \in V\}$. Assume that Γ admits a cyclic fractional polymorphism ω. Then $T(\Gamma)$ is connected within each D_v.*

We will need a classical result about cores and small technical lemma about block-finite languages.

Proposition A.2.4 ([39]). *Let Γ be a language on domain D that is a core. Then $\langle\Gamma\rangle$ contains a function f of arity $|D|$ such that every solution $x \in \arg\min f$ contains all labels in D.*

Lemma A.2.5. *Let Γ be a block-finite language on domain D partitioned into $\{D_v \mid v \in V\}$. Let $f \in \langle\Gamma\rangle$ be an r-ary function $f(x_1, \ldots, x_r)$. Then at least one the following three statements is true:*

(a) f has a dummy variable (i.e. the values of f depend only on a subset of coordinates)

(b) $f(x) = \infty$ whenever elements of x are pairwise distinct.

(c) $\operatorname{dom} f \subseteq D_{v_1} \times \ldots \times D_{v_r}$ for some $v_1, \ldots, v_r \in V$.

Proof. For functions f in Γ this follows immediately from Definiton 2.3.3. For $f \in \langle\Gamma\rangle$, that is for a Γ-instance $\mathcal{I}$, we proceed with a little more care. Let $\mathcal{I}'$ be the instance before minimizing out. If a variable of $\mathcal{I}$ is unconstrained in $\mathcal{I}'$, then it is a dummy variable of $\mathcal{I}$, thus (a) holds. If every variable of $\mathcal{I}$ is constrained by a function from $\Gamma \setminus \{=_D\}$ in $\mathcal{I}'$, then Definiton 2.3.3(c) applies and we are in case (c). The only remaining objects of interest are equivalence classes of variables constrained only by $=_D$ in $\mathcal{I}'$. If such equivalence class contains at least two variables of $\mathcal{I}$, we are in case (b). If it meets $\mathcal{I}$ in just one variable, this variable becomes dummy after minimizing out and we are in (a) again. □

Since we are restricted to a core language all unary $g \in \mathrm{fPol}^+(\Gamma)$ are permutations of D. We will show that in fact only the identity permutation can live in $\mathrm{fPol}^+(\Gamma)$. Let us recall one last lemma before we get to it.

Lemma A.2.6 ([54]). *For any n-ary $f \in \langle\Gamma\rangle$, a tuple $x \in D^n$, and unary $g \in \mathrm{fPol}^+(\Gamma)$ we have $f(g(x)) = f(x)$.*

Proposition A.2.7. *Let Γ be a core valued language that is block-finite with domain D partitioned into $\{D_v \mid v \in V\}$. Assume that Γ admits a cyclic fractional polymorphism ω. Then the only unary operation in $\mathrm{fPol}^+(\Gamma)$ is identity.*

Proof. First, since Γ is a core by Proposition A.2.4 $\langle\Gamma\rangle$ contains a function $\hat{f}$ of arity $n = |D|$ such that every tuple $x \in \arg\min \hat{f}$ contains each label from D exactly once. Let us fix one such tuple z. Note that neither (a) nor (b) from Lemma A.2.5 apply to $\hat{f}$ so we have the case (c) and each coordinate restricted to a particular D_v.

Assume for contradiction that there is a $g \in \mathrm{fPol}^+(\Gamma)$ with $g(a) = b$ with $a \neq b$ from D. Reorder variables of $\hat{f}$ so that $z_1 = a$ and $z_2 = b$.

If we had $a \in D_u$, $b \in D_w$ for $u \neq w$, we could apply g to z (g is a polymorphism of $\mathrm{Feas}(\Gamma)$ after all) and obtain a tuple in $\mathrm{dom}\,\hat{f}$ with b in the first coordinate. But then the first coordinate can take values from both D_u and D_w, a contradiction.

So from now on let a, $b \in D_v$ for some $v \in V$. Set $g^0(a) = a$, and recursively $g^{i+1}(a) = g(g^i(a))$ (so in particular $g^1(a) = b$) until $g^k(a) = g_0(a) = a$ (such k exists since g is a permutation). Note that $g^i(a) \in D_v$ for every i (the argument from the previous paragraph applies).

Instead of $\hat{f}$ we restrict our attention to binary $f(z_1, z_2) = \min_{z_3,\ldots,z_n} \hat{f}(z_1, \ldots, z_n\}$. Note that $f \in \langle\Gamma\rangle$ and that by construction we have $(a, b) \in \arg\min f$ and $f(a, b) < f(x, x)$ for any $x \in D$. Due to Lemma A.2.6 also $(g^i(a), g^{i+1}(a)) \in \arg\min f$ for every i. Moreover, for $x \in D_v$ the costs $f(x, x)$ are finite since Γ is block-finite and $f \in \langle\Gamma\rangle$. In particular, $f(a, a) < \infty$. The if we let $X = \min f$, $Y = f(a, a)$, and $Z = \min_{x\in D} f(x, x)$, we have $X < Z \leq Y < \infty$.

Now is the time to apply the cyclic fractional polymorphism ω. Due to Lemma 2.1.18 we may assume the arity p of ω is any prime number. We will choose p such that $p \equiv 1 \pmod{k}$ and $p > (Y - X)/(Z - X)$. Such choice is possible due to (a weak version of) Dirichlet's Theorem on arithmetic progressions.

We will apply ω to p tuples $(x_0, \ldots, x_{p-1})$. For $i = 0, \ldots p - 2$ these are $x_i = (g^{i \bmod k}(a), g^{i+1 \bmod k}(a))$ and finally $x_{p-1} = (g^0(a), g^0(a)) = (a, a)$. It is easy to verify that the vector of first coordinates is a cyclic shift of the vector of second coordinates. We obtain

$$Z = \sum_{h\in\mathrm{supp}(\omega)} \omega(h)Z \leq \sum_{h\in\mathrm{supp}(\omega)} \omega(h)f(h(\ldots), h(\ldots)) \leq \frac{p-1}{p}X + \frac{1}{p}Y.$$

Our choice of p was such that this is a contradiction. □

As the final step, we aim to express certain useful unary functions from Γ. The following result will come to help.

Proposition A.2.8 ([54]). *Let Γ be a core finite constraint language. For every $m \geq 1$ there exists $f \in \langle\Gamma\rangle$ of arity $|D|^m$ and a rational number P, such that for every operation $g\colon D^m \to D$ the following conditions are satisfied:*

1. $f(g) \geq P$,
2. $f(g) < \infty$ *if and only if* $g \in \mathrm{Pol}(\Gamma)$,
3. $f(g) = P$ *if and only if* $g \in \mathrm{fPol}^+(\Gamma)$,

where the notation $f(g)$ should be read as $f(g(x_1), \ldots, g(x_n))$, where $x_1, \ldots, x_n$ is some fixed ordering of D^m (and thus $n = |D|^m$).

Proposition A.2.9. *Let Γ be a core block-finite language on domain D with partitions $\{D_v \mid v \in V\}$. Then for each $a \in D$ there is a unary function $u_a \in \langle\Gamma\rangle$ such that $\arg\min u_a = \{a\}$ and u_a is finite-valued on the D_v containing a.*

Proof. Knowing from Proposition A.2.7 that the only unary function in $\mathrm{fPol}^+(\Gamma)$ is the identity, one can use Proposition A.2.8 (with $m = 1$) to construct a function $f \in \langle\Gamma\rangle$ of arity $|D|$ with $\arg\min f = \{(1, \ldots, |D|)\}$ (assuming $D = \{1, \ldots, |D|\}$). Then given $a \in D$ one can easily construct the desired u_a by minimizing out all other coordinates than the a-th one. Finite values of u_a on the corresponding D_v are guaranteed since Γ is block-finite. □

These special unary functions are already sufficient for proving the connectivity of $T(\Gamma)$.

Proposition A.2.10 ([67]). *Let Γ be a constraint language on domain D such that for each $a \in D$ there is finite-valued $u_a \in \langle\Gamma\rangle$ such that $\arg\min u_a = \{a\}$. Then $T(\Gamma)$ is connected.*

Now proof of the Theorem A.2.3 follows easily. Fix a subdomain D_v, restrict functions of $\langle\Gamma\rangle$ (in particular those from Proposition A.2.9) to that subdomain, and apply Proposition A.2.10.

A.3 Non matching realizable even $\triangle$-matroid

Here we prove Proposition 3.3.3 which says that not every even Δ-matroid of arity six is matching realizable. We do it by first showing that matching realizable even Δ-matroids satisfy certain decomposition property and then we exhibit an even Δ-matroid of arity six which does not possess this property and thus is not matching realizable.

Lemma A.3.1. *Let M be a matching realizable even Δ-matroid and let $f, g \in M$. Then $f \,\Delta\, g$ can be partitioned into pairs of variables $P_1, \dots P_k$ such that $f \oplus P_i \in M$ and $g \oplus P_i \in M$ for every $i = 1 \dots k$.*

Proof. Fix a graph $G = (N, E)$ that realizes M and let $V = \{v_1, \dots, v_n\} \subseteq N$ be the nodes corresponding to variables of M. Let E_f and E_g be the edge sets from matchings that correspond to tuples f and g. Now consider the graph $G' = (N, E_f \,\Delta\, E_g)$ (symmetric difference of matchings). Since both E_f and E_g cover each node of $N \setminus V$, the degree of all such nodes in G' will be zero or two. Similarly, the degrees of nodes in $(V \setminus (f \,\Delta\, g))$ are either zero or two leaving $f \,\Delta\, g$ as the set of nodes of odd degree, namely of degree one. Thus G' is a union of induced cycles and paths, where the paths pair up the nodes in $f \,\Delta\, g$. Let us use this pairing as $P_1, \dots, P_k$.

Each such path is a subset of E and induces an alternating path with respect to both E_f and E_g. After altering the matchings accordingly, we obtain new matchings that witness $f \oplus P_i \in M$ and $g \oplus P_i \in M$ for every i. □

Lemma A.3.2. *There is an even Δ-matroid of arity 6 which does not have the property from Lemma A.3.1.*

Proof. Let us consider the set M with the following tuples:

000000	100100	011011	111111
	011000	100111	
	001100	110011	
	001010	110101	
	000101	111010	
	001001	001111	
	010001	101101	
	100010	101011	
		111100	

With enough patience or with computer aid one can verify that this is indeed an even Δ-matroid. However, there is no pairing satisfying the conclusion of Lemma A.3.1 for tuples $f = (000000)$, and $g = (111111)$. In fact the set of pairs P for which both $f \oplus P \in M$ and $g \oplus P \in M$ is $\{v_1, v_4\}$, $\{v_2, v_3\}$, $\{v_3, v_4\}$, $\{v_3, v_5\}$, $\{v_4, v_6\}$ (see the first five lines in the middle of the table above) but no three of these form a partition on $\{v_1, \ldots, v_6\}$. □

A.4 Non coverable Δ-matroid

In Section 3.6, we claimed that not every Δ-matroid is coverable. The next lemma gives a counterexample.

Lemma A.4.1. *There is a Δ-matroid that is not coverable*

Proof. Let us consider the set M with the following tuples:

0000	0001
1100	1101
1010	1011
0110	0111
1110	

First, note that M is symmetrical in its first three coordinates. Verifying that M is a Δ-matroid is straightforward (or not resistant to computer power). However, if we choose $\alpha = (0000)$ we claim there is no suitable even Δ-matroid M_α.

Assume there is one. Note that all even tuples (1100), (1010), (0110), (1110), (1111) are reachable from α so they must be in M_α. However, by flipping the last coordinate in $\alpha = (0000)$, we find that one of the three tuples $(1001), (0101), (0011)$ must be in M_α (simply by the definition of an even Δ-matroid). By symmetry of M it suffices to rule out (1001). Since $(1001) \in M_\alpha \setminus M$, there should be both $(0001) \in M$ and $(1000) \in M$. The latter is not true and we have a contradiction. □

A.5 Classes of Δ-matroids that are efficiently coverable

As we promised, here we will show that all classes of Δ-matroids that were previously known to be tractable are efficiently coverable.

Co-independent Δ-matroids

Definition A.5.1. A Δ-matroid M is *co-independent* if whenever $\alpha \notin M$, then $\alpha \oplus u \in M$ for every u in the scope of M.

Let V be the set of variables of M. In this case we choose M_α to be the Δ-matroid that contains all members of $\{0,1\}^V$ of the same parity as α. This trivially satisfies the first two conditions in the definition of a Δ-matroid. To see the third condition, observe that whenever $\gamma \in M_\alpha \setminus M$, the co-independence of M gives us that $\gamma \oplus u \in M$ for every $u \in V$, so we are done.

Moreover, each set M_α is roughly as large as M itself: A straightforward double counting argument gives us that $M \geq 2^{|V|-1}$, so listing M_α can be done in time linear in $|M|$.

Compact Δ-matroids

We present the definition of compact Δ-matroids in an alternative form compared to [41].

Definition A.5.2. Function $F\colon \{0,1\}^V \to \{0,\ldots,|V|\}$ is called a *generalized counting function* (gc-function) if

1. for each $\alpha \in \{0,1\}^V$ and $v \in V$ we have $F(\alpha \oplus v) = F(\alpha) \pm 1$ and;
2. if $F(\alpha) > F(\beta)$ for some $\alpha, \beta \in \{0,1\}^V$, then there exist $u, v \in \alpha \Delta \beta$ such that $F(\alpha \oplus u) = F(\alpha) - 1$ and $F(\beta \oplus v) = F(\beta) + 1$

An example of such function is the function which simply counts the number of ones in a tuple.

Definition A.5.3. We say that a $S \subseteq \{0, 1, \dots n\}$ is 2*-gap free* if whenever $x \notin S$ and $\min S < x < \max S$, then $x + 1, x - 1 \in S$. A set of tuples M is *compact-like* if $\alpha \in M$ if and only if $F(\alpha) \in S$ for some gc-function F and a 2-gap free subset S of $\{0, 1, \dots |V|\}$.

The difference to the presentation in [41] is that they give an explicit set of possible gc-functions (without using the term gc-function). However, we decided for more brevity and omit the description of the set.

Lemma A.5.4. *Each compact-like set of tuples M is a Δ-matroid.*

Proof. Let the gc-function F and the 2-gap free set S witness that M is compact-like. Take $\alpha, \beta \in M$ and $u \in \alpha \Delta \beta$. If $F(\alpha \oplus u) \in S$, then $\alpha \oplus u \in M$ and we are done. Thus we have $F(\alpha \oplus u) \neq F(\beta)$. We need to find a $v \in \alpha \Delta \beta$ such that $F(\alpha \oplus u \oplus v) \in S$.

Let us assume $F(\alpha \oplus u) > F(\beta)$. Since F is a gc-function we can find $v \in (\alpha \oplus u) \Delta \beta$ (note that $u \neq v$) such that $F(\alpha \oplus u \oplus v) = F(\alpha \oplus u) - 1$. Now we have either $F(\alpha) = F(\alpha \oplus u \oplus v) \in S$, or $F(\alpha) > F(\alpha \oplus u) > F(\alpha \oplus u \oplus v) \geq F(\beta)$, which again means $F(\alpha \oplus u \oplus v) \in S$ because S does not have 2-gaps.

The case when $F(\alpha \oplus u) < F(\beta)$ is handled analogously. □

It turns out that any practical class of compact-like Δ-matroids is efficiently coverable:

Lemma A.5.5. *Assume $\mathcal{M}$ is a class of compact-like Δ-matroids where the description of each $M \in \mathcal{M}$ includes a set S_M (given by a list of elements) and a function F_M witnessing that M is compact-like and there is a polynomial p such that the time to compute $F_M(\alpha)$ is at most $p(|M|)$. Then $\mathcal{M}$ is efficiently coverable.*

Proof. Given $M \in \mathcal{M}$ and $\alpha \in M$, we let M_α be the compact-like even Δ-matroid given by the function F_M and the set $U = [\min S_M, \max S_M] \cap \{F_M(\alpha) + 2k \colon k \in \mathbb{Z}\}$. It is an easy observation that $\alpha, \beta \in \{0,1\}^V$ have the same parity if and only if $F_M(\alpha)$ and $F_M(\beta)$

have the same parity, so all members of M_α have the same parity. In particular M_α contains all $\beta \in M$ of the same parity as α. Moreover, the set U is 2-gap free, so M_α is an even Δ-matroid.

Let now $\gamma \in M_\alpha \setminus M$. Then $F_M(\gamma) \notin S_M$. Since S_M is 2-gap free and $F_M(\gamma)$ is not equal to $\min S_M$, nor $\max S_M$, it follows that both $F_M(\gamma)+1$ and $F_M(\gamma)-1$ lie in S_M. Therefore, $\gamma \oplus v \in M$ for any $v \in V$ by the first property of gc-functions.

It remains to show how to construct M_α in polynomial time. We begin by adding to M_α all tuples of M of the same parity as α. Then we go through all tuples $\beta \in M$ of parity different from α and for each such β we calculate $F_M(\beta \oplus v)$ for all $v \in V$. If $\min S < F(\beta \oplus v) < \max S$, we add $\beta \oplus v$ to M_α. By the argument in the previous paragraph, this procedure will eventually find and add to M_α all tuples γ such that $F_M(\gamma) \in U \setminus S_M$. □

Local and binary Δ-matroids

We will avoid giving the definitions of local and binary Δ-matroids. Instead, we will rely on a result from [24] saying that both of these classes avoid a certain substructure. This will be enough to show that both binary and local matroids are efficiently coverable.

Definition A.5.6. Let M, N be two Δ-matroids where $M \subseteq \{0,1\}^V$. We say that M contains N as a *minor* if we can get N from M by a sequence of the following operations: Choose $c \in \{0,1\}$ and $v \in V$ and take the Δ-matroid we obtain by fixing the value at v to c and deleting v:

$$M_{v=c} = \{\beta \in \{0,1\}^{V \setminus \{v\}} \colon \exists \alpha \in M,\, \alpha(v) = c \wedge \forall u \neq v,\, \alpha(u) = \beta(u)\}.$$

Definition A.5.7. The interference Δ-matroid is the ternary Δ-matroid given by the tuples $\{(0,0,0),\ (1,1,0),\ (1,0,1),\ (0,1,1),\ (1,1,1)\}$. We say that a Δ-matroid M is *interference free* if it does not contain any minor isomorphic (via renaming variables or flipping the values 0 and 1 of some variables) to the interference Δ-matroid.

Lemma A.5.8. *If M is an interference-free Δ-matroid and $\alpha, \beta \in M$ are such that $|\alpha \,\Delta\, \beta|$ is odd, then we can find $v \in \alpha \,\Delta\, \beta$ so that $\alpha \oplus v \in M$.*

Proof. Let us take $\beta' \in M$ so that $\alpha \Delta \beta' \subseteq \alpha \Delta \beta$ and $|\alpha \Delta \beta'|$ is odd and minimal possible. If $|\alpha \Delta \beta'| = 1$, we are done. Assume thus that $|\alpha \Delta \beta'| = 2k + 3$ for some $k \in \mathbb{N}_0$. Applying the Δ-matroid property on α and β' (with α being the tuple changed) k many times, we get a set of $2k$ variables $U \subseteq \alpha \Delta \beta'$ such that $\alpha \oplus U \in M$ (since β' is at minimal odd distance from α, in each step we need to switch exactly two variables of α).

Let the three variables in $\alpha \Delta \beta' \setminus U$ be x, y, and z and consider the matroid P on x, y, z we get from M by fixing the values of all $v \notin \{x, y, z\}$ to those of $\alpha \oplus U$ and deleting these variables afterward. Moreover, we switch 0s and 1s so that the triple corresponding to $(\alpha(x), \alpha(y), \alpha(z))$ is $(0, 0, 0)$. We claim that P is the interference matroid: It contains the triple $(0, 0, 0)$ (because of $\alpha \oplus U$) and $(1, 1, 1)$ (as witnessed by β') and does not contain any of the triples $(1, 0, 0)$, $(0, 1, 0)$, or $(0, 0, 1)$ (for then β' would not be at minimal odd distance from α). Applying the Δ-matroid property on $(1, 1, 1)$ and $(0, 0, 0)$ in each of the three variables then necessarily gives us the tuples $(0, 1, 1)$, $(1, 0, 1)$, and $(1, 1, 0) \in P$. □

Corollary A.5.9. *Let M be an interference-free Δ-matroid. If M contains at least one even tuple then the set $\mathrm{Even}(M)$ of all even tuples of M forms a Δ-matroid. The same holds for $\mathrm{Odd}(M)$ the set of all odd tuples of M. In particular, M is efficiently coverable by the even Δ-matroids $\mathrm{Even}(M)$ and $\mathrm{Odd}(M)$.*

Proof. We show only that $\mathrm{Even}(M)$ is a Δ-matroid; the case of $\mathrm{Odd}(M)$ is analogous and the covering result immediately follows.

Take $\alpha, \beta \in \mathrm{Even}(M)$ and let v be a variable v such that $\alpha(v) \neq \beta(v)$. We want $u \neq v$ so that $\alpha(u) \neq \beta(u)$ and $\alpha \oplus u \oplus v \in M$. Apply the Δ-matroid property of M to α and β, changing the tuple α. If we get $\alpha \oplus v \oplus u \in M$ for some u, we are done, so let us assume that we get $\alpha \oplus v \in M$ instead. But then we recover as follows: The tuples $\alpha \oplus v$ and β have different parity, so by Lemma A.5.8 there exists a variable u so that $(\alpha \oplus v)(u) \neq \beta(u)$ (i.e. $u \in \alpha \Delta \beta \setminus \{v\}$) and $\alpha \oplus v \oplus u \in M$. □

It is mentioned in [24] (Section 4) that the interference Δ-matroid is among the forbidden minors for both local and binary (minors B1 and L2) Δ-matroids. Thus both of those classes are efficiently coverable.