Complexity Classifications for the Valued Constraint Satisfaction Problem

Robert D. Powell

A Thesis presented for the degree of
Doctor of Philosophy

Engineering and Computing Sciences
Durham University
England

November 2015

Dedicated to

Arnold Powell

17th May 1933 - 4th September 2013

Complexity Classifications for the Valued Constraint Satisfaction Problem

Robert D. Powell

Submitted for the degree of Doctor of Philosophy
November 2015

Abstract

In a valued constraint satisfaction problem (VCSP), the goal is to find an assignment of values to variables that minimizes a given sum of functions. Each function in the sum depends on a subset of variables, takes values which are rational numbers or infinity, and is chosen from a fixed finite set of functions called a constraint language. We study how the computational complexity of this problem depends on the constraint language. We often consider the case where infinite values are disallowed, and refer to such constraint languages as being finite-valued.

If we consider such finite-valued constraint languages, the case where we allow variables to take two values was classified by Cohen et al., who show that submodular functions essentially give rise to the only tractable case. Non-submodular functions can be used to express the **NP**-hard Max Cut problem. We consider the case where the variables can take three values, and identify a new infinite set of functions called skew bisubmodular functions which imply tractability. We prove that submodularity with respect to some total order and skew bisubmodularity give rise to the only tractable cases, and in all other cases Max Cut can be expressed. We also show that our characterisation of tractable cases is tight, that is, none of the conditions can be omitted. Thus, our results provide a new dichotomy theorem in constraint satisfaction research. We also negatively answer the question of whether multimorphisms can capture all necessary tractable constraint languages.

We then study the VCSP as a homomorphism problem on digraphs. By adapting a proof designed for CSPs we show that each VCSP with a fixed finite constraint language is equivalent to one where the constraint language consists of one $\{0, \infty\}$-valued binary function (i.e. a digraph) and one finite-valued unary function. This latter problem is known as the Minimum Cost Homomorphism Problem for digraphs. We also show that our reduction preserves a number of useful algebraic properties of the constraint language.

Finally, given a finite-valued constraint language, we consider the case where the variables of our VCSP are allowed to take four values. We prove that 1-defect chain multimorphisms, which are required in the four element dichotomy of Min CSP, are a special case of more general fractional polymorphisms we call $\{\alpha, \beta\}$-1-defect fractional polymorphisms. We conclude with a conjecture for the four element case, and some interesting open problems which might lead to a tighter description of tractable finite-valued constraint languages on finite domains of any size.

Declaration

The work in this thesis is based on research carried out at the Department of Engineering and Computing Sciences, Durham University, England. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

Chapter 1 consists mainly of background material to introduce the problem being addressed, while chapter 2 introduces further background material and my own individual unpublished work. Chapters 3 and 4 contain my contributions to the joint publications [31] with Anna Huber and Prof. Andrei Krokhin, and [51] with Prof. Andrei Krokhin. Chapter 5 consists of my own individual unpublished work.

Acknowledgements

First of all I have to thank my supervisor, Prof. Andrei Krokhin, for introducing me to the study of valued constraint satisfaction which has eventually culminated in this thesis. I am indebted to him for his patient guidance, encouragement and helpful suggestions throughout my studies. This work would certainly not have been possible without his support.

My thanks go to the Engineering and Physical Sciences Research Council for the financial support that has allowed me to produce this thesis.

I also owe a great debt of gratitude to all of the staff and students of St Mary's College that I've had the pleasure to meet since 2006! The college community is truly something special, and my experience of studying in Durham would have been much the poorer without it. Special thanks have to go to St Mary's College bar, darts club and pool club, without which this thesis may have been finished somewhat sooner! Whilst it would be impossible to thank everyone individually here, there are a handful of people who deserve a special mention for their friendship and support over the years. First Mike Gillard, for vastly improving my darts, teaching me how to run a bar, and showing me how to complete a PhD (slowly). Next to Matt Watson Snr, Matt Watson Jnr, and Will Grummitt for the many enjoyable hours spent playing darts, pool and snooker. Last, but by no means least, to Ryan Hanley, for all the useful discussions and the regular reminders that I was actually in Durham to study!

Finally, I would like to thank all of my family and friends for their support during my time at Durham.

Contents

Abstract **1**

Declaration **3**

Acknowledgements **4**

1 Introduction **8**

1.1 Complexity Theory: A Brief Overview 8

1.2 Constraint Satisfaction Problems 10

1.3 Complexity of CSP 12

1.3.1 Constraint Languages 12

1.3.2 Polymorphisms 14

1.3.3 Known Dichotomy Theorems 17

1.4 Valued Constraint Satisfaction 19

1.5 Multimorphisms 21

1.6 Fractional Polymorphisms 23

1.7 Expressibility 24

1.8 Tractability and Complexity Classifications 25

2 Domain Reductions and Cores **30**

2.1 Introduction 30

2.2 Domain Reducing Multimorphisms 30

2.3 Cores 32

2.4 Multimorphisms as Digraphs 36

2.4.1 3-element domain 37

3 Skew Bisubmodularity and the Three Element Dichotomy **39**

3.1 Introduction 39

3.2 Fractional Polymorphisms 40

3.3 A Characterisation of α-bisubmodularity 42
3.4 A Dichotomy Theorem 45
3.5 Multimorphisms are not enough 54

4 Reducing VCSP to Min-Cost-Hom 58
4.1 Introduction 58
4.2 Homomorphisms 59
4.3 Proving Poly-Time Equivalence 61
4.3.1 Constructing $(\mathbb{D}, u)$ 62
4.3.2 Reduction from VCSP($w\mathcal{A}$) to MinCostHom($\mathbb{D}, u$) 64
4.3.3 Reduction from MinCostHom($\mathbb{D}, u$) to VCSP($w\mathcal{A}$) 65
4.4 Preservation of Algebraic Properties 73
4.5 Dealing with Short Components 78
4.6 A Hard Case 80

5 Fractional Polymorphisms as Digraphs 82
5.1 Introduction 82
5.2 Two Element Domains 84
5.3 Three Element Domains 85
5.4 Moving to Four Element Domains 89
5.5 1-defect chains 91
5.6 Generalising 1-defect chains 92
5.6.1 An example 94
5.7 Conjectures and Open Problems 96

List of Figures

1.1 Complexity classes. 10

2.1 The four possible connected digraphs on three vertices. 37

4.1 The digraph $\mathbb{D}$ built from the weighted structure $w\mathcal{A}$. 64
4.2 The target digraph $\mathbb{D}$ of Example 21. 81
4.3 The source digraph $\mathbb{G}$ of Example 21. 81

5.1 The three connected digraphs on three vertices with no 3-cycles. . . . 85
5.2 An example digraph exhibiting generalised submodularity. 90
5.3 The two possible 1-defect chains on the four element domain. 91
5.4 Seven digraphs describing fractional polymorphisms on four element domains. 97

Chapter 1

Introduction

1.1 Complexity Theory: A Brief Overview

A number of theoretical and practical problems that arise in computer science require the user to assign certain values to certain variables given a set of conditions they must adhere to. One of the simplest examples of these problems, worked on by children of all ages (and computer scientists!), is the map colouring problem. Given a map of some randomly shaped countries can you colour the map so that no two countries that share a border have the same colour?

Given two colours, say red and blue, there is a simple algorithm to solve the problem. First choose a country at random and colour it blue, then colour all of its neighbours in red. For each of those countries now coloured red, colour each of their neighbours in blue. Proceeding in this way will either yield a valid colouring of all the countries, or a point will be reached where the algorithm forces two neighbouring countries to have the same colour. The number of steps required to complete this algorithm is bounded by the number of countries (multiplied by some independent constant), therefore we say this algorithm runs in polynomial time in the number of countries.

Now consider you are given three colours, and you have the same condition that no two countries that share a border may have the same colour. If we start by colouring a country at random, then we still have a choice of colours for each of its neighbours. This additional choice makes the problem significantly more difficult. In fact, the best known algorithm for deciding if there is a valid 3-colouring of a particular map has a running time of $c \cdot 1.3289^n$ as shown in [5]. Therefore unlike

the 2-colouring example, the problem of 3-colouring cannot currently be solved in polynomial time, and it is not known if a polynomial time algorithm can even exist. However, if we were to be given a 3-colouring of a map, it is very easy to verify if it is a valid colouring or not.

Problems such as 2-colourability, which can be solved in polynomial time, are contained in the complexity class **P**. On the other hand 3-colourability falls into the complexity class **NP**. **NP** contains all problems for a which a valid solution can be easily verified, despite the fact it may be very difficult to find such a valid solution. It is easy to see that $\mathbf{P} \subseteq \mathbf{NP}$ as if we can efficiently find a valid solution we can easily verify a valid solution by simply finding it. However, a major unsolved problem in complexity theory asks the question: Does $\mathbf{P} = \mathbf{NP}$? A large number of proofs have been offered, claiming both equality and disequality, but so far none have been accepted as correct by the academic community.

A problem is said to be **NP**-hard if it is at least as hard as any problem in **NP**. More formally a problem is **NP**-hard if every problem in **NP** can be reduced to it in polynomial time. Problems that are both in **NP** and **NP**-hard are called **NP**-complete problems. It may be useful to consider the diagram of these complexity classes shown in Figure 1.1.

Assuming $\mathbf{P} \neq \mathbf{NP}$ the set of problems that are in **NP**, but neither in **P** or **NP**-Complete are often referred to as **NP**-Intermediate problems. A result of Ladner [47] states that the class of **NP**-Intermediate problems is non-empty and a number of problems have been suggested to occupy this complexity class. Well known examples of which include integer factoring, the discrete logarithm problem and graph isomorphism. For these types of problems there is no known polynomial time algorithm to solve them, but they have not been proved to be **NP**-complete. The discrete logarithm problem is of particular importance as several cryptographic algorithms base their security on the fact that solving a discrete logarithm problem is computationally difficult. Also in various prime number searches, the discrete logarithm problem is used in sieving algorithms to eliminate obvious non-prime candidates with small factors. If a polynomial time algorithm were to be found for the discrete

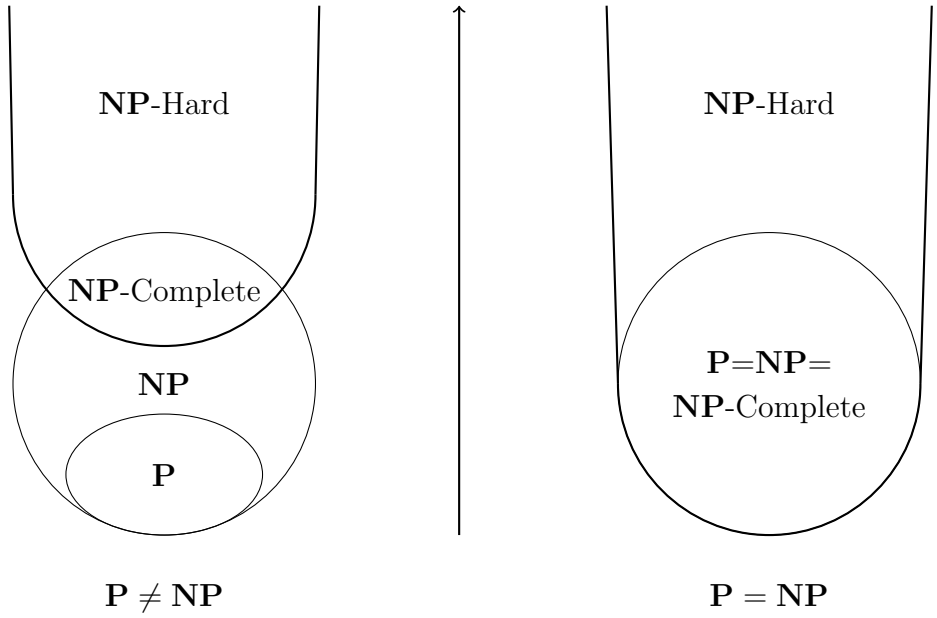


Figure 1.1: Complexity classes.

logarithm problem this would have profound effects on these fields.

Even with this brief introduction to the field, it is clear that the study of computational complexity is of huge importance, not just theoretically but also for practical applications such as computer security. Throughout this thesis we assume that $\mathbf{P} \neq \mathbf{NP}$, and investigate the complexity of a particular problem known as the valued constraint satisfaction problem, which we introduce later in this chapter, along with some known complexity results.

1.2 Constraint Satisfaction Problems

The problem of assigning some values to variables given a set of conditions, that we have introduced informally above, is formally known as a constraint satisfaction problem (or CSP for short). In order to formally define the CSP we first need to introduce some notation. Let D be a finite set we call the domain, and let D^n be all the possible n-tuples that can be constructed from elements of D. Any subset of D^n is called an n-ary relation.

Definition 1. *Let D be a set of labels, V be a set of variables, and C be a set of constraints. Each constraint $c \in C$ is a pair $(\boldsymbol{x}, R)$ where $\boldsymbol{x}$ is a list of m variables,*

and R is an m-ary relation over D. The constraint satisfaction problem is the decision problem that asks is there a function $f : V \to D$ that satisfies all of the constraints. That is, for every constraint, the assignment of values to the list of variables $\boldsymbol{x}$ appears in R.

If there exists such a function f, then we say that the CSP is satisfiable, otherwise it is said to be unsatisfiable.

A number of well known problems in computer science can be captured by this definition. Let us revisit the map colouring problem we discussed earlier, and express this explicitly as a CSP. If we are given k colours with which to colour the map this is known as the k-colouring problem.

Example 1. *Given a positive integer k and a graph $G = (V, E)$ the k-colouring problem asks us to find a valid colouring of the vertices V. To interpret this as a CSP we let the set of vertices V be the set of variables, and the domain is the set of colours $\{1, 2, ..., k\}$. For every edge $(a, b) \in E$ we have a constraint whose list of variables is $[a, b]$, and whose binary relation R contains all tuples where $a \neq b$. Finding a valid k-colouring of G is now the problem of finding a mapping $f : V \to D$ that satisfies all of the constraints.*

As another example consider the well studied n-queens problem, see [6] for a wide survey of known results and open problems. The basic n-queens problem is that of placing n queens on a $n \times n$ chessboard such that no two queens threaten each other, that is they do not appear on the same row, column or diagonal. We show that not only is it possible to describe this as a CSP problem, but it can be described in a number of ways. Here we present two examples of CSPs encoding the n-queens problem.

Example 2. *Let the variables, V be the individual squares of the chessboard $x_{i,j}$ with rows $i = 1, 2, ..., n$ and columns $j = 1, 2, ..., n$. Let $D = \{0, 1\}$ where 0 represents no queen and 1 represents a queen. The set of constraints come in three classes. Firstly there is a set of constraints for each row where $\sum_{j=1}^{n} x_{i,j} = 1$. Secondly there is set of constraints for each column where $\sum_{i=1}^{n} x_{i,j} = 1$. Finally, there are $4n - 2$ constraints introduced by the diagonals such that the sum on each diagonal is at most*

1. *A mapping $f : V \to D$ that satisfies all of those constraints is a valid solution to the n-queens problem.*

Example 3. *Let the variables, V, be the rows of the chessboard $x_1, x_2, ..., x_n$. Let $D = \{1, 2, ..., n\}$ which will represent the column in which a queen appears. The set of constraints this time only come from two sources. First there is a constraint for each column - for each $i, j \in D$, $x_i \neq x_j$. Secondly there is a constraint for each diagonal - for each $i, j \in D$, $x_i - x_j \neq i - j$ and $x_i - x_j \neq j - i$. A mapping $f : V \to D$ that satisfies all of those constraints is a valid solution to the n-queens problem.*

1.3 Complexity of CSP

We have seen that it is possible to describe some well known problems as CSPs but why are they interesting to theoreticians? It is simple to see that general CSP problems are hard to solve and we have explicitly seen an example of this with the 3-colourability problem. However the problem of 2-colourability shows us that some CSPs can be solved in polynomial time, and this is what is interesting from a complexity theory perspective. If we impose restrictions on the CSPs we consider we hope to reduce the complexity of the problem, making it computationally easier.

1.3.1 Constraint Languages

One of the most studied restrictions is known as a constraint language restriction, see [3, 4, 7, 9, 10, 23, 53]. This restricts the types of relations that can appear in the constraints of a CSP, thus potentially allowing us to find a polynomial time algorithm for this smaller set of problems.

Definition 2. *Let R_D be the set of all relations over a finite domain D. A constraint language over D is a finite set $\Gamma \subseteq R_D$.*

Definition 3. *Given a constraint language Γ we define CSP(Γ) to be the class of all instances of CSP where each constraint relation is in Γ.*

To demonstrate how constraint language restrictions affect the complexity of CSP consider the following two examples. First we consider the problem known as 2-

satisfiability (2-SAT) as a CSP, a problem that is polynomial time solvable with several known algorithms [2, 22, 46], the best of which solves the problem in linear time.

Example 4. *In the 2-SAT problem we are given a formula in conjunctive normal form where all clauses contain two variables. As an example we could have the following instance consisting of three clauses*

$$(x_0 \vee x_1) \wedge (\neg x_1 \vee x_2) \wedge (x_0 \vee x_2).$$

The equivalent problem CSP(Γ) *has domain* $D = \{0, 1\}$*, variables* $V = \{x_0, x_1, x_2\}$*, and a constraint for each of the three clauses. Consider the first clause, its scope would be* $[x_0, x_1]$ *and its relation is the set* $R_{00} = D^2 \setminus (0, 0)$*. Building a constraint for each clause our constraint language becomes* $\Gamma = \{R_{00}, R_{10}\}$*. Note that both the first and third clause have the same constraint relation but they have different scopes. It is simple to see that any 2-SAT problem can be represented as a CSP in this way with a constraint language consisting of relations that can be expressed as a disjunction with two literals. In reverse, any instance of CSP*(Γ) *where* Γ *consists solely of these relations is equivalent to a 2-SAT problem and therefore can be solved in polynomial time.*

Now consider the 3-satisfiability problem (3-SAT), which is one of Karp's 21 **NP**-complete problems [39].

Example 5. *In the 3-SAT problem we are given a formula in conjunctive normal form where all clauses contain three variables. As an example we could have the following instance consisting of three clauses*

$$(x_0 \vee x_1 \vee x_2) \wedge (\neg x_0 \vee x_1 \vee \neg x_2) \wedge (x_0 \vee \neg x_1 \vee x_2).$$

As in the previous example we can construct an equivalent instance of CSP(Γ) *with domain* $D = \{0, 1\}$*, variables* $V = \{x_0, x_1, x_2\}$*, and a constraint for each of the three clauses. However this time our constraint language would be* $\Gamma = \{R_{000}, R_{101}, R_{010}\}$*. Again any 3-SAT problem can be represented as a CSP in this way, and any instance of CSP*(Γ)*, where* Γ *consists of relations that are expressed as a disjunction of three literals, is equivalent to a 3-SAT problem, and thus is* **NP***-complete.*

In Example 4 we have a problem that can be solved in polynomial time, whereas in Example 5 we have a problem that is **NP**-complete, but the two problems have the same domain and variables. The only difference between the problems is the relations allowed by the constraint language, thus showing that the complexity of solving CSP(Γ) depends on Γ.

This leads to a major open problem in complexity theory. Whilst we have shown that some for some choice of Γ the problem CSP(Γ) is in **P**, and for a different choice of Γ the problem is instead **NP**-Complete, is it true that CSP(Γ) will always fall into one of these complexity classes? This dichotomy question was formally stated as a conjecture by Feder and Vardi [23].

Conjecture 1 ([23])**.** *For any constraint language* Γ*, the decision problem CSP*(Γ) *is either in* **P** *or it is* **NP***-complete.*

Without a definitive proof of this dichotomy we cannot rule out the possibility that there is a constraint language Γ such that CSP(Γ) is an **NP**-Intermediate problem, see [47], i.e. it is neither in **P** or **NP**-Complete. A proof of the dichotomy would not only close a long standing problem but also aid those who are practically trying to solve CSPs, by offering a fixed set of reasons for being polynomial time solvable (with appropriate algorithms), or conversely explaining why their CSP is **NP**-Complete. All results to date support the dichotomy hypothesis and we now introduce the algebra needed to present those results.

1.3.2 Polymorphisms

In order to describe the dichotomy results on two and three element domains we first introduce the algebraic notion of a polymorphism.

Definition 4. *Let* Γ *be a constraint language and let* R *be a constraint relation in* Γ *of arity* n*. A function* $f : D^k \to D$ *is called a polymorphism of* R *if for all combination of tuples:*

$$(x_{11}, ..., x_{1n}), ..., (x_{k1}, ..., x_{kn}) \in R \Rightarrow (f(x_{11}, ..., x_{k1}), ..., f(x_{1n}, ..., x_{kn}) \in R.$$

The function f is applied to the tuples componentwise as shown below:

$$\begin{array}{llll} (x_{11}, & x_{12}, & \cdots, & x_{1n}) \in R \\ (x_{21}, & x_{22}, & \cdots, & x_{2n}) \in R \\ \vdots & \vdots & & \vdots \\ (x_{k1}, & x_{k2}, & \cdots, & x_{kn}) \in R \\ \downarrow_f & \downarrow_f & & \downarrow_f \\ (y_1, & y_2, & \cdots, & y_n) \in R \end{array}$$

where $y_i = f(x_{1i}, x_{2i}, ..., x_{ki})$.

If f is a polymorphism of all constraint relations in Γ then we say that f is a polymorphism of Γ. The set of all polymorphisms of a constraint language Γ is often written as Pol(Γ).

Using this definition we can explicitly define the ternary polymorphisms majority and minority.

Definition 5. *A function $f : D^3 \to D$ is called a* majority *function if for all $x, y \in D$:*

$$f(y, x, x) = f(x, y, x) = f(x, x, y) = x.$$

Similarly a function $f : D^3 \to D$ is called a minority *function if for all $x, y \in D$:*

$$f(y, x, x) = f(x, y, x) = f(x, x, y) = y.$$

Note that on the two element domain there is precisely one majority function which we denote as Mjrty, and also precisely one minority function which we denote as Mnrty.

Example 6. *Consider a constraint language Γ with a single constraint relation $R = \{(0, 0, 0), (0, 0, 1), (1, 0, 0)\}$. It is easy to see that the Mjrty function defined above is a polymorphism of R (and therefore Γ). However, this constraint relation*

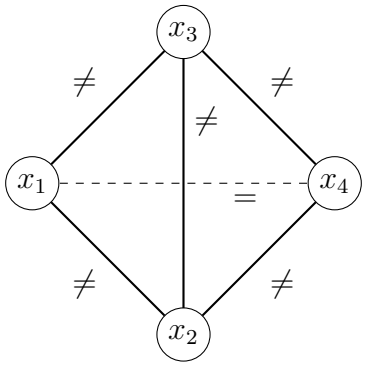
does not exhibit a Mnrty polymorphism as we now show:

$$\begin{array}{lll} (0, & 0, & 0) \in R \\ (0, & 0, & 1) \in R \\ (1, & 0, & 0) \in R \\ \downarrow & \downarrow & \downarrow \\ (1, & 0, & 1) \notin R \end{array}$$

Applying the Mnrty function componentwise we obtained the tuple $(1, 0, 1)$ *which is not a member of our constraint relation* R*. Therefore the Mnrty function is not a polymorphism of* R *(and therefore* Γ*).*

Before trying to determine the complexity of a constraint language it is useful to consider if we can add additional relations to the constraint language without changing its complexity. This is often referred to as expressibility, or the expressive power of the constraint language.

Example 7 ([64])**.** *Consider the CSP instance that consists of four variables* $V = \{x_0, x_1, x_2, x_3\}$*, with domain* $D = \{0, 1, 2\}$*, and constraints as depicted in the figure below, where* $\neq$ *is the binary disequality relation. It is easy to see that the variables* x_1 *and* x_4 *must always be assigned the same value, and hence they are implicitly constrained by the equality relation. Therefore we say* $=$ *is expressible by* $\neq$*.*



Definition 6 ([10])**.** *The expressive power of a constraint language* Γ*, denoted* $\langle\Gamma\rangle$*, is the set of relations that can be expressed using*

1. *relations from* $\Gamma \cup \{=\}$ *(where* $\{=\}$ *is the equality relation),*
2. *conjunction,*
3. *existential quantification.*

Theorem 1 ([10,35]). *Let Γ be a finite constraint language, and $\Delta \subseteq \langle\Gamma\rangle$. There is a polynomial time reduction from CSP(Δ) to CSP(Γ).*

This theorem can be rewritten in terms of the polymorphisms of the constraint languages.

Theorem 2 ([7, 35]). *Let Γ and Δ be finite constraint languages. If $Pol(\Delta) \subseteq Pol(\Gamma)$, then there is a polynomial time reduction from CSP(Γ) to CSP(Δ).*

This theorem proves that the complexity of CSP(Γ) is totally determined by the polymorphisms of Γ. Therefore the aim is now to identify which polymorphisms are responsible for describing polynomial time solvable CSPs.

1.3.3 Known Dichotomy Theorems

The following theorem describes the polynomial time solvable cases of CSP over a two element domain, originally proven by Schaefer [53]. Here however we choose to present the more modern algebraic view of the dichotomy as presented by Chen [13].

Theorem 3 ([13, 53]). *Let Γ be a constraint language and let $D = \{0, 1\}$. CSP(Γ) can be solved in polynomial time if it has one of the following six polymorphisms:*

1. *Constant unary operation 0;*
2. *Constant unary operation 1;*
3. *Binary AND operation $\wedge$;*
4. *Binary OR operation $\vee$;*
5. *Ternary Mjrty operation;*
6. *Ternary Mnrty operation.*

Otherwise CSP(Γ) is **NP***-complete.*

The complexity of CSP is also fully characterised on a three element domain.

Theorem 4 ([7]). *Let Γ be a constraint language and let $D = |3|$. CSP(Γ) can be solved in polynomial time if it has one of 10 algebraic properties. Otherwise CSP(Γ) is* **NP***-complete.*

We direct the reader to [7] for the details and proof of Theorem 4 .

Both of these results support the Feder-Vardi dichotomy conjecture, which is further supported by the following result and conjecture of Bulatov, Jeavons and

Krokhin [10] often referred to as the Algebraic CSP dichotomy conjecture. First we introduce a type of polymorphism called a Taylor polymorphism.

Definition 7 ([27])**.** *A polymorphism f is said to be inclusive in position i if it satisfies an identity involving two variables with different entries in position i. Let $a_j, b_j \in \{u, v\}$ for $j = 1, 2, ..., k$ with $a_i \neq b_i$, such that the identity*

$$f(a_1, a_2, ..., a_k) = f(b_1, b_2, ..., b_k)$$

holds for all variables $u, v \in V$. A polymorphism inclusive in every position is called a Taylor polymorphism.

Probably the most well known special case of a Taylor polymorphism is a near-unanimity polymorphism, which satisfies the identity

$$f(u, u, ..., u, v) = f(u, u, ..., v, u) = ... = f(v, u, ..., u, u) = u.$$

Maróti and McKenzie [49] showed that Taylor polymorphisms are equivalent, with respect to tractability, to a generalisation of near-unanimity polymorphisms called weak near-unanimity (WNU) polymorphisms which are idempotent, $f(u, u, ..., u) = u$, and satisfy the slightly weaker identity

$$f(u, u, ..., u, v) = f(u, u, ..., v, u) = ... = f(v, u, ..., u, u).$$

We also briefly introduce the notion of a core constraint language. Essentially if a constraint language is not core there is some domain element that is not required, which we can remove and therefore we only have to solve the problem over this smaller domain. Technically we define a core constraint language as follows:

Definition 8. *A constraint language Γ is called a core constraint language if every unary polymorphism of Γ is injective. An injective function is a function f such that if $a \neq b$ then $f(a) \neq f(b)$.*

Theorem 5 ([10])**.** *For any core constraint language Γ, if Γ does not exhibit a Taylor polymorphism (or equivalently a weak near-unanimity polymorphism) then CSP(Γ) is* **NP***-complete.*

Conjecture 2 ([10])**.** *For any core constraint language Γ, if Γ admits a Taylor polymorphism (or equivalently a weak near-unanimity polymorphism) then CSP(Γ) is in* **P***.*

Not only does the algebraic CSP dichotomy conjecture state that all problems CSP(Γ) are in either **P** or **NP**-complete, but it also suggests the exact boundary separating them. The known dichotomy results on two and three element domains presented previously both satisfy the conditions of the algebraic dichotomy conjecture.

1.4 Valued Constraint Satisfaction

The problem with standard constraint satisfaction problems is they are very strict, you can either satisfy all of the constraints or you cannot. Many problems require a more flexible framework, i.e. a framework that offers some amount of optimisation.

A simple framework that offers this flexibility is Max CSP. Here we take a CSP and we attempt to maximise the number of constraints that are satisfied. In this case all assignments of values to variables give a feasible solution, but the optimal solution is that which satisifes the most constraints. It may be possible that there is an optimal solution that satisfies all of the constraints.

Another problem that introduces optimisation is that of Min-Ones. Consider a CSP over a two element domain $D = \{0, 1\}$. The aim of the Min-Ones problem is to satisfy all of the constraints of the CSP, whilst minimising the number of variables that are assigned value 1. This problem essentially says that some solutions are better, or more desirable, than other solutions, and we want the most desirable solution.

However, in this thesis we primarily study the complexity of the valued constraint satisfaction problem (referred to as VCSP). It is the most general of the constraint satisfaction frameworks and can be used to express a number of well known problems, including CSP as a special case. In the VCSP framework every assignment of domain values to variables is a solution, and has an associated cost. The aim is to find a solution with minimal cost, referred to as an optimal solution. The VCSP is formally introduced below.

Let D be a fixed finite set. Let $\overline{\mathbb{Q}}_+$ be the non-negative rational numbers with infinity.

Let $\Phi_D^{(m)}$ be the set of all functions from D^m to $\overline{\mathbb{Q}}_+$, and then $\Phi_D = \bigcup_{m=1}^{\infty} \Phi_D^{(m)}$. The functions in Φ_D are often referred to as cost functions.

Definition 9. *Let $V = \{x_1, \ldots, x_n\}$ be a set of variables. A valued constraint over V is an m-ary expression, $\phi(\mathbf{x})$, where $\mathbf{x} \in V^m$ and $\phi \in \Phi_D^{(m)}$. An instance $\mathcal{I}$ of VCSP can be written as a single function $\phi_{\mathcal{I}}(x_1, \ldots, x_n) = \sum_{i=1}^{q} w_i \cdot \phi_i(\mathbf{x}_i)$ where the ϕ_i's are valued constraints and the w_i's are non-negative rational weights. An optimal solution to $\mathcal{I}$ is a mapping $h : V \to D$ that minimises $\phi_{\mathcal{I}}$.*

Definition 10. *A valued constraint language, Γ, is a finite subset of Φ_D. Given a valued constraint language Γ we define $VCSP(\Gamma)$ to be the class of all instances of VCSP where the cost functions of all the valued constraints are in Γ.*

Note that it is also possible to define both valued constraint languages over infinite domains, and infinite valued constraint languages. For the purposes of this thesis we will use the term constraint language to mean a finite valued constraint language (not to be confused with a finite-valued constraint language where cost functions return values in $\mathbb{Q}_+$).

The VCSP framework is exceptionally broad, and as such it captures a large number of well studied problems. Here we give some examples of the problems that can be represented as VCSPs.

Example 8. *The standard CSP can be expressed as a VCSP, where all cost function are $\{0, \infty\}$-valued, representing tuples allowed and disallowed respectively. Valued constraints consisting of such $\{0, \infty\}$-valued cost functions are often referred to as crisp constraints.*

Example 9. *Consider the MAX CSP problem we referred to earlier. When seeking an optimal solution to an instance of MAX CSP, maximising the number of satisfied constraints is the same problem as minimising the number of unsatisfied constraints. Given an instance Φ of MAX CSP we can construct an equivalent instance of VCSP. Each constraint in Φ has a corresponding valued constraint which assigns cost 0 if the tuple is allowed, and cost 1 if the tuple is disallowed. Therefore any instance of MAX CSP is equivalent to an instance of VCSP consisting of only $\{0, 1\}$-valued cost functions.*

Example 10. *Consider the well known* **NP***-hard problem MAX CUT, where given an edge-weighted graph the aim is to partition the vertices into two sets and maximise the total weight of the edges with endpoints in different sets. It is easy to see that this problem can be expressed as a VCSP. Let* $f_{MC} : \{0,1\}^2 \to \mathbb{Q}_+$ *be such that* $f_{MC}(0,1) = f_{MC}(1,0) < f_{MC}(0,0) = f_{MC}(1,1)$*. Let* $\Gamma_{MC} = \{f_{MC}\}$*, then* $VCSP(\Gamma_{MC})$ *is equaivalent to MAX CUT, thus* $VCSP(\Gamma_{MC})$ *is also* **NP***-hard.*

1.5 Multimorphisms

As we have seen earlier, polymorphisms have proven to be important algebraic tools for characterising the complexity of constraint languages [4, 7, 10]. We can define polymorphisms of valued constraints in a similar fashion to that for relations seen in Definition 4.

Definition 11. *Let* $\phi : D^m \to \overline{\mathbb{Q}}_+$ *be a cost function and* $Feas(\phi) = \{\boldsymbol{x} \in D^m | \phi(\boldsymbol{x})$ *is finite*$\}$ *be the feasibility relation of* ϕ*. An operation* $f : D^k \to D$ *is a polymorphism of* ϕ *if for any* $x_1, x_2, \ldots, x_k \in Feas(\phi)$ *we have* $f(x_1, x_2, \ldots, x_k) \in Feas(\phi)$*.*

For any valued constraint language Γ we will say that f is a feasibility polymorphism of Γ if f is a feasibility polymorphism of all cost functions $\phi \in \Gamma$. The set of all feasibility polymorphisms of Γ will be denoted $\mathrm{Pol}(\Gamma)$.

In order to study the complexity of valued constraint languages however we require a more general notion, referred to as a multimorphism. These are essentially collections of polymorphisms, with each assigned the same weight. The definition does however allow the same polymorphism to appear more than once. For the purpose of this thesis we will not need the full generality of the definition of multimorphisms, and thus we only provide the definitions of the specific cases of unary, binary and ternary multimorphisms.

Definition 12. *Let* Γ *be a valued constraint language over a finite set* D*.* Γ *has the unary multimorphism* $\langle f \rangle$ *if every cost function* $\phi \in \Gamma$*, of arity* n*, satisfies the inequality:*

$$\phi(f(\boldsymbol{a})) \leq \phi(\boldsymbol{a})$$

for all $\boldsymbol{a} \in D^n$*, where* f *is applied component wise.*

Definition 13. *Let Γ be a valued constraint language over a finite set D. Γ has the binary multimorphism $\langle f_1, f_2\rangle$ if every cost function $\phi \in \Gamma$, of arity n, satisfies the inequality:*

$$\phi(f_1(\boldsymbol{a}, \boldsymbol{b})) + \phi(f_2(\boldsymbol{a}, \boldsymbol{b})) \leq \phi(\boldsymbol{a}) + \phi(\boldsymbol{b})$$

for all $\boldsymbol{a}, \boldsymbol{b} \in D^n$, where f_1, f_2 are applied component wise.

Definition 14. *Let Γ be a valued constraint language over a finite set D. Γ has the ternary multimorphism $\langle f_1, f_2, f_3\rangle$ if every cost function $\phi \in \Gamma$, of arity n, satisfies the inequality:*

$$\phi(f_1(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})) + \phi(f_2(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})) + \phi(f_3(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})) \leq \phi(\boldsymbol{a}) + \phi(\boldsymbol{b}) + \phi(\boldsymbol{c})$$

for all $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in D^n$, where f_1, f_2, f_3 are applied component wise.

Example 11 (Submodularity [24, 48, 54])**.** *Let $(D, \vee, \wedge)$ be an arbitrary lattice. A function $\phi : D^n \to \overline{\mathbb{Q}}_+$ is called submodular if it satisfies the inequality*

$$\phi(\boldsymbol{a} \vee \boldsymbol{b}) + \phi(\boldsymbol{a} \wedge \boldsymbol{b}) \leq \phi(\boldsymbol{a}) + \phi(\boldsymbol{b})$$

for all $\boldsymbol{a}, \boldsymbol{b} \in D^n$. If ϕ satisfies the inequality we say that ϕ admits or exhibits the multimorphism $\langle \vee, \wedge\rangle$.

Example 12 (Bisubmodularity [1, 24, 50, 52])**.** *Let $D = \{-1, 0, 1\}$, and fix the order $-1 > 0 < 1$ on D. Define binary operations $\vee_0$ and $\wedge_0$ on D, as follows:*

$$x \vee_0 y = \begin{cases} Max(x, y) & \text{if } \{x, y\} \neq \{-1, 1\} \\ 0 & \text{otherwise} \end{cases}$$

$$x \wedge_0 y = \begin{cases} Min(x, y) & \text{if } \{x, y\} \neq \{-1, 1\} \\ 0 & \text{otherwise} \end{cases}$$

where the operations Max and Min are taken with respect to the above order on D.

A function $\phi : D^n \to \overline{\mathbb{Q}}_+$ is called bisubmodular *if it satisfies the inequality:*

$$\phi(\mathbf{a} \vee_0 \mathbf{b}) + \phi(\mathbf{a} \wedge_0 \mathbf{b}) \leq \phi(\mathbf{a}) + \phi(\mathbf{b})$$

for all $\boldsymbol{a}, \boldsymbol{b} \in D^n$. If ϕ satisfies the inequality we say that ϕ admits or exhibits the multimorphism $\langle \vee_0, \wedge_0\rangle$.

Example 13. *Let Γ be a vauled constraint language over the domain $D = \{-1, 0, 1\}$ that consists of the single unary function ϕ, where $\phi(-1) = 0$, $\phi(0) = 1$ and $\phi(1) = 0$. It is easy to check that ϕ is submodular with respect to the order $1 > 0 > -1$. Let $x = 1$, $y = 0$, then we get:*

$$\phi(1 \vee 0) + \phi(1 \wedge 0) \leq \phi(1) + \phi(0) \Rightarrow \phi(1) + \phi(0) \leq \phi(1) + \phi(0).$$

Now let $x = -1$, $y = 0$, and we get:

$$\phi(-1 \vee 0) + \phi(-1 \wedge 0) \leq \phi(-1) + \phi(0) \Rightarrow \phi(0) + \phi(-1) \leq \phi(-1) + \phi(0).$$

Finally let $x = -1$, $y = 1$, and we get:

$$\phi(-1 \vee 1) + \phi(-1 \wedge 1) \leq \phi(-1) + \phi(1) \Rightarrow \phi(1) + \phi(-1) \leq \phi(-1) + \phi(1).$$

Given that the functions $\vee$ and $\wedge$ are symmetric we don't need to reverse the inputs. The case when $x = y$ holds with equality due to idempotency, and thus we have checked all possible pairs of inputs, and the submodularity inequality holds for them all. Therefore ϕ is a submodular function. However, it is also easy to check that ϕ is not bisubmodular. Let $x = -1$, $y = 1$, and we get:

$$\phi(-1 \vee_0 1) + \phi(-1 \wedge_0 1) \leq \phi(-1) + \phi(1) \Rightarrow 2\phi(0) \leq \phi(-1) + \phi(1).$$

which does not hold as $2 \cdot (1) \not\leq 0 + 0$. Therefore the function ϕ is not bisubmodular.

1.6 Fractional Polymorphisms

Since the introduction of multimorphisms in [17] an even more general notion of a fractional polymorphism was introduced in [15], as defined below.

Definition 15 ([15])**.** *Let $\phi : D^m \to \overline{\mathbb{Q}}_+$ be a cost function and let $C \subseteq Pol^{(k)}(\phi)$ be k-ary polymorphisms. A function $\omega : C \to [0, 1]$ is a k-ary fractional polymorphism of ϕ if it satisfies the following conditions:*

- $\sum_{f \in C} \omega(f) = 1$;
- *for all $x_1, x_2, ...x_k \in Feas(\phi)$*

$$\sum_{f \in C} \omega(f)\phi(f(x_1, x_2, ..., x_k)) \leq \frac{1}{k}(\phi(x_1) + \phi(x_2) + ... + \phi(x_k))$$

The set of polymorphisms $\{f|\omega(f) > 0\}$ are often referred to as the support of the fractional polymorphism ω, written as supp(ω). If ω is a fractional polymorphism of all the functions ϕ, of a valued constraint language Γ, we say that Γ admits or exhibits ω. The set of all fractional polymorphisms of a valued constraint language is written fPol(Γ).

Example 14. *Consider the general definition of a binary multimorphism from Definition 13. We can represent the binary multimorphism as a fractional polymorphism with $k = 2$, $w(f_1) = \frac{1}{2}$ and $w(f_2) = \frac{1}{2}$.*

Example 15. *We can rewrite the submodularity condition from Example 11 as a fractional polymorphism by letting the functions f_1 and f_2 in the example above be the polymorphisms Min and Max.*

Throughout the course of this thesis we will refer to certain properties of fractional polymorphisms which we describe here for reference.

Definition 16. *Let $f : D^k \to D$ be a polymorphism in the positive support of a fractional polymorphism ω. f is called idempotent if $f(x, x, ..., x) = x$ for all $x \in D$. An idempotent operation is called cyclic if $f(x_1, x_2, ..., x_k) = f(x_2, ..., x_k, x_1)$ for all $x_1, x_2, ..., x_k \in D$. Similarly an idempotent operation is called symmetric if $f(x_1, x_2, ..., x_k) = f(x_{\pi(1)}, x_{\pi(2)}, ..., x_{\pi(k)})$ for all $x_1, x_2, ..., x_k \in D$ and all permutations π. We say that the factional polymorphism ω is idempotent, cyclic or symmetric if all of the polymorphisms in supp(ω) have the appropriate property.*

1.7 Expressibility

Definition 17. *For a constraint language Γ, let $\langle\Gamma\rangle$ denote the set of all functions $\phi(x_1, \ldots, x_m)$ such that, for some instance $\mathcal{I}$ of* VCSP(Γ) *with objective function $\phi_{\mathcal{I}}(x_1, \ldots, x_m, x_{m+1}, \ldots, x_n)$, we have*

$$\phi(x_1, \ldots, x_m) = \min_{x_{m+1}, \ldots, x_n} \phi_{\mathcal{I}}(x_1, \ldots, x_m, x_{m+1}, \ldots, x_n).$$

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

Definition 18. *If functions $\phi, \phi' \in \Phi_D$ are such that ϕ can be obtained from ϕ' by* scaling *and* translating, *i.e. $\phi = a \cdot \phi' + b$ for some constants $a \in \mathbb{Q}_+$ and $b \in \mathbb{Q}$, then we write $\phi \equiv \phi'$. For $\Gamma \subseteq \Phi_D$, let $\Gamma_{\equiv} = \{\phi \mid \phi \equiv \phi' \text{ for some } \phi' \in \Gamma\}$.*

Theorem 6 ([15, 17])**.** *Let* Γ *and* Γ' *be constraint languages on* D *such that* $\Gamma' \subseteq \langle\Gamma\rangle_{\equiv}$.

- *If VCSP*(Γ) *is tractable then VCSP*(Γ') *is tractable.*
- *If VCSP*(Γ') *is* **NP***-hard then VCSP*(Γ) *is* **NP***-hard.*

Lemma 1. *We say that a finite-valued constraint language* Γ *can express MAX CUT (see Example 10) if there exists distinct* $a, b \in D$ *such that* Γ *can express a unary function* u *with* $argmin(u) = \{a, b\}$ *and a binary function* h *where* $h(a, b) = h(b, a) < h(a, a) = h(b, b)$*. If* Γ *can express MAX CUT then VCSP*(Γ) *is* **NP***-hard and all constraint languages* Γ *where VCSP*(Γ) *is known to be* **NP***-hard satisfy this condition [31, 59].*

We have already seen that polymorphisms capture the complexity of ordinary CSPs, and there exists an equivalent result for VCSP.

Theorem 7 ([14])**.** *For any* $\Gamma \subseteq \Phi_D$ *and any* $\phi \in \Phi_D$*, we have* $\phi \in \langle\Gamma\rangle_{\equiv}$ *if and only if Pol*$(\Gamma) \subseteq$ *Pol*(ϕ) *and fPol*$(\Gamma) \subseteq$ *fPol*(ϕ)*.*

Combined with Theorem 6 this implies that the tractability or **NP**-hardness of a valued constraint languages is characterised by its fractional polymorphisms and feasibility polymorphisms. When considering only finite-valued constraint languages, the complexity is determined by the fractional polymorphisms alone. It follows that it is sufficient to study fractional polymorphisms in attempting to identify all tractable finite-valued constraint languages.

1.8 Tractability and Complexity Classifications

Using the concept of multimorphisms a number of complexity classifications have been obtained. First is the dichotomy for valued constraint languages over a two element domain established in [17]. You will need to recall the definition of majority and minority operations from Definition 5.

Theorem 8 ([17])**.** *Let* Γ *be a valued constraint language on the two element domain* $D = \{0, 1\}$ *with costs in* $\overline{\mathbb{Q}}_+$*. If* Γ *has one of the following multimorphisms then* $VCSP(\Gamma)$ *is tractable:*

1. $\langle 0 \rangle$;
2. $\langle 1 \rangle$;
3. $\langle Max, Max \rangle$;
4. $\langle Min, Min \rangle$;
5. $\langle Min, Max \rangle$;
6. $\langle Mjrty, Mjrty, Mjrty \rangle$;
7. $\langle Mnrty, Mnrty, Mnrty \rangle$;
8. $\langle Mjrty, Mjrty, Mnrty \rangle$.

In all other cases $VCSP(\Gamma)$ *is* **NP***-hard.*

Example 16. *We can rewrite the ternary multimorphism* $\langle Mjrty, Mjrty, Mjrty \rangle$ *required in the Boolean VCSP dichotomy as a fractional polymorphism. Let* $k = 3$ *and the functions* f_1 *and* f_2 *be the polymorphisms Mjrty and Mnrty respectively, with* $\omega(f_1) = \frac{2}{3}$ *and* $\omega(f_2) = \frac{1}{3}$*. It is easy to check that all the conditions of being a fractional polymorphism hold.*

If we take the previous theorem and restrict the valuation structure by excluding infinite costs, some of the possible cases collapse into one another, giving the much simpler theorem below.

Theorem 9 ([17])**.** *Let* Γ *be a valued constraint language on the two element domain* $D = \{0, 1\}$ *with costs in* $\mathbb{Q}_+$*. If* Γ *has one of the following multimorphisms then* $VCSP(\Gamma)$ *is tractable:*

1. $\langle 0 \rangle$;
2. $\langle 1 \rangle$;
3. $\langle Min, Max \rangle$.

In all other cases $VCSP(\Gamma)$ *is* **NP***-hard.*

There is also a known dichotomy for conservative valued constraint languages, which we define below. In order to present the dichotomy we also need the definition of tournament operations and (symmetric) tournament pair multimorphisms.

Definition 19 ([44])**.** *A valued constraint language* Γ *is called conservative if* Γ *contains all* $\{0, 1\}$*-valued unary cost functions.*

Definition 20. *A binary operation* $f : D^2 \to D$ *is a tournament operation if* f *is commutative,* $f(x, y) = f(y, x)$*, and* f *is conservative,* $f(x, y) \in \{x, y\}$*, for all*

$x, y \in D$. The dual, g, of a tournament operation, f, is the tournament operation satisfying $f(x, y) \neq g(x, y)$ when $x \neq y$.

A multimorphism $\langle f, g \rangle$ is a tournament pair multimorphism if both f and g are tournament operations. If g is the dual of f then this is known as a symmetric tournament pair multimorphism (STP).

Theorem 10 ([44]). *Let Γ be a conservative valued constraint language over a finite domain D. VCSP(Γ) is tractable if:*

- *Γ admits a conservative binary multimorphism $\langle f_1, f_2 \rangle$, and*
- *Γ admits a conservative ternary multimorphism $\langle g_1, g_2, g_3 \rangle$,*

and there is a set M of two-element subsets of D such that:

- *for every pair $\{a, b\} \in M$, $\langle f_1, f_2 \rangle$ restricted to $\{a, b\}$ is an STP multimorphism, and*
- *for every pair $\{a, b\} \notin M$, $\langle g_1, g_2, g_3 \rangle$ restricted to $\{a, b\}$ is a $\langle Mjrty, Mjrty, Mnrty \rangle$ multimorphism.*

Otherwise VCSP(Γ) is **NP***-hard.*

This characterisation has recently been simplified to the following:

Theorem 11 ([60]). *Let Γ be a conservative valued constraint language. If Pol(Γ) contains a majority polymorphism then VCSP(Γ) is tractable. Otherwise VCSP(Γ) is* **NP***-hard.*

Again if we restrict the cost functions to finite values then the classification can be significantly simplified.

Theorem 12 ([44]). *Let Γ be a conservative finite-valued constraint language. If Γ admits an STP multimorphism then VCSP(Γ) is tractable, otherwise VCSP(Γ) is* **NP***-hard.*

This result can be further simplified as it can be shown that a finite-valued constraint language that admits an STP multimorphism is also submodular on some total ordering. The proof of this is given implicitly in [16] and explicitly in [43].

Consider the conditions of submodularity and bisubmodularity shown previously in Examples 11 and 12 respectively. It has long been known that if a function

exhibits either of these conditions it can be minimised in polynomial time, but the algorithms required are complex and unique. For example there are numerous different polynomial time algorithms known to solve submodular problems, see [24, 32–34, 54].

However, thanks to recent work of Thapper and Živný [57], there is now a much simpler algorithm that implies tractability for both submodular and bisubmodular functions as well as numerous other functions. This algorithm is called the basic linear programming relaxation, known from hereon as BLP. Given an instance Φ of VCSP, defined by $\Phi(\mathbf{x}) = \sum_{i=1}^{q} w_i \cdot \phi_i(\mathbf{x}_i)$, with variables V we define BLP(Φ) as follows:

$$\begin{aligned} \mathrm{BLP}(\Phi) = \ & \min \sum_{i=1}^{q} \sum_{\mathbf{s}_i \in D^{\mathbf{x}_i}} w_i \cdot \phi_i(\mathbf{s}_i) \lambda_{i,\mathbf{s}_i} \\ \text{s.t.} \sum_{\mathbf{s}_i \in D^{\mathbf{x}_i} | \mathbf{s}_i(x)=a} & \lambda_{i,\mathbf{s}_i} = \ \mu_x(a), \quad 1 \leq i \leq q, \ x \in \mathbf{x}_i, \ a \in D \\ \sum_{a \in D} & \mu_x(a) = \ 1, \quad x \in V \\ & \lambda_{i,\mathbf{s}_i} = \ 0, \quad 1 \leq i \leq q, \ \phi_i(\mathbf{s}_i) = \infty \end{aligned}$$

We minimise over the variables $\mu_x(a)$, where $x \in V$ and $a \in D$, and $\lambda_{i,\mathbf{s}_i}$, where $1 \leq i \leq q$ and $\mathbf{s}_i \in D^{\mathbf{x}_i}$. These variables take real values on the interval $[0, 1]$, and can be seen as probability distributions on D and $D^{\mathbf{x}_i}$ respectively.

Given an instance Φ of VCSP, we say that BLP solves Φ if the optimal value of BLP(Φ) equals the optimal value of Φ. BLP solves a valued constraint language Γ if BLP solves every instance $\Phi \in \mathrm{VCSP}(\Gamma)$. Finally it is shown in [43, 57] that if BLP solves Γ, then Γ is tractable. We will use the main theorems of [57] throughout this thesis, and therefore we present them here.

Theorem 13 ([42, 57])**.** *Let Γ be a valued constraint language with cost functions that only take finite values. The following are all equivalent:*

1. *BLP solves VCSP(Γ).*
2. *For every $k > 1$, Γ has a k-ary symmetric fractional polymorphism.*
3. *Γ has a binary symmetric fractional polymorphism.*

4. *For every $n > 1$, Γ has a fractional polymorphism ω such that supp(ω) generates a symmetric n-ary operation.*

A semilattice operation is any operation $f : D^2 \to D$ where f is associative, commutative and idempotent. In other words $f(x, f(y, z)) = f(f(x, y), z)$, $f(x, y) = f(y, x)$ and $f(x, x) = x$ respectively. Every semilattice operation trivially generates symmetric operations of all arities - hence we have the following corollary:

Corollary 1 ([43])**.** *If Γ has a fractional polymorphism with a semilattice operation in its support, then BLP solves VCSP(Γ).*

Consider the submodularity and bisubmodularity multimorphisms from Examples 11 and 12 respectively. If a constraint language Γ consists of all submodular cost functions then VCSP(Γ) is tractable by BLP as the submodularity multimorphism has the semilattice operations $\vee$ and $\wedge$ in its support. If Γ consists of all bisubmodular cost functions then VCSP(Γ) is tractable by BLP as the bisubmodularity multimorphism has the semilattice operation $\wedge_0$ in its support.

We will repeatedly make use of the fact that a single semilattice operation in the support of a fractional polymorphism implies tractability of the underlying constraint language throughout this thesis.

Chapter 2

Domain Reductions and Cores

2.1 Introduction

Following the successful complexity classification of VCSPs on 2-element domains [17], the obvious extension is to consider VCSPs on 3-element domains. Discovering a similar dichotomy result on the 3-element domain could offer useful insights into the types of fractional polymorphisms that give rise to tractable instances of VCSP on finite domains of any size.

Let us consider the VCSP over the domain $D = \{-1, 0, 1\}$ with costs in $\mathbb{Q}_+$. We have already seen that the fractional polymorphisms (multimorphisms in this case) $\langle Min, Max\rangle$ and $\langle min_0, max_0\rangle$ define tractable constraint languages, as will $\langle -1\rangle$, $\langle 0\rangle$ and $\langle 1\rangle$. Where do we start looking for other possible fractional polymorphisms that could describe tractable languages? It seems reasonable to conduct a case analysis of the multimorphisms we can derive from the polymorphisms such as Min,Max,min_0 and max_0 as these simple polymorphisms imply tractability by Theorem 13.

2.2 Domain Reducing Multimorphisms

For the purposes of this section we need the following definition:

Definition 21. *Let $D = \{-1, 0, 1\}$ and define the binary operations max_{-1} and min_{-1} as follows:*

$$max_{-1}(x, y) = \begin{cases} Max(x, y) & \text{if } \{x, y\} \neq \{0, 1\} \\ -1 & \text{otherwise} \end{cases}$$

$$min_{-1}(x,y) = \begin{cases} Min(x,y) & \text{if } \{x,y\} \neq \{0,1\} \\ -1 & \text{otherwise} \end{cases}$$

where the operations Max and Min are taken with respect to some given order on the elements of D. The operations max_0, min_0, max_1 and min_1 can be defined likewise.

Now consider the multimorphisms $\langle f, g \rangle$ where $f, g \in \{Min, Max, min_x, max_x\}$ and $x \in D$. It can be quickly checked that most of these multimorphisms imply the presence of one of those five tractable multimorphisms identified previously. However there are six cases that do not, and the presence of any one of these six multimorphisms imply that one of the elements of the domain is unnecessary, and can simply be thrown away. Hence we will now refer to them as domain reducing multimorphisms. This allows us to solve the problem using the results for two-element domains [17] instead.

Theorem 14. *Let Γ be a valued constraint language over domain $D = \{-1, 0, 1\}$ with cost functions $\phi : D^n \to \mathbb{Q}_+$. The elements of the domain satisfy the order $1 > 0 > -1$. If Γ has one of the following six multimorphisms then $VCSP(\Gamma)$ is tractable by submodular minimisation over a domain of reduced size:*

- $\langle Min, max_0 \rangle$
- $\langle Min, max_1 \rangle$
- $\langle Min, max_{-1} \rangle$
- $\langle Max, min_0 \rangle$
- $\langle Max, min_1 \rangle$
- $\langle Max, min_{-1} \rangle$

Proof. For simplicity we shall prove just the first case, but a very similar argument follows for all cases. Assume Γ has the multimorphism $\langle Min, max_0 \rangle$, then all cost functions $\phi \in \Gamma$ (of arity n) must satisfy the inequality:

$$\phi(Min(\mathbf{a}, \mathbf{b})) + \phi(max_0(\mathbf{a}, \mathbf{b})) \leq \phi(\mathbf{a}) + \phi(\mathbf{b}) \text{ for all } \mathbf{a}, \mathbf{b} \in D^n.$$

If we consider the case where $\{\mathbf{a}, \mathbf{b}\} = \{\mathbf{1}, \mathbf{\text{-}1}\}$ then this inequality becomes:

$$\phi(\mathbf{\text{-}1}) + \phi(\mathbf{0}) \leq \phi(\mathbf{1}) + \phi(\mathbf{\text{-}1}) \Rightarrow \phi(\mathbf{0}) \leq \phi(\mathbf{1}).$$

Therefore any solution that assigns value 1 to a variable could instead assign the value 0 to the variable and the overall cost cannot be worse. This gives us the benefit of being able to simply ignore the domain element 1 and solve this problem over the domain $D = \{-1, 0\}$. Over this reduced domain the function max_0 is exactly the function Max, so our multimorphism simplifies to $\langle Min, Max\rangle$ and the constraint language Γ is tractable by known results. □

2.3 Cores

In order to simplify the complexity classifications of VCSP over larger domains we would prefer not having to consider domain reducing multimorphisms. It is useful therefore to introduce the notion of core valued constraint languages. Intuitively a valued constraint language is not core if there is an element of its domain, $a \in D$, such that any instance has an optimal solution that does not use a. We can simply remove the element a from D reducing the problem to a smaller domain. This allows us to only have to consider core constraint languages when attempting to prove complexity classifications.

A number of different definitions of core constraint languages have appeared in literature. The technical definition of a core that we use in [31] is the following.

Definition 22 ([31])**.** *A valued constraint language Γ on D is a core if, for each $a \in D$, there is an instance $\mathcal{I}_a$ of VCSP(Γ) such that a appears in every optimal solution to $\mathcal{I}$.*

The following proposition further reduces the set of interesting core valued constraint languages. Given a valued constraint language Γ, let Γ_c be the language containing the functions of Γ and all the functions obtained by fixing variables in functions of Γ, e.g. $g(x, y) = f(x, a, y) \in \Gamma_c$ if $f \in \Gamma$ and $a \in D$.

Proposition 1 ([31])**.** *Let Γ be a core valued constraint language on an arbitrary finite domain D. Then*

1. *$\langle\Gamma_c\rangle$ contains a set of unary functions $\{u_a|a \in D\}$ such that argmin(u_a)=$\{a\}$,*
2. *VCSP(Γ) is tractable if and only if VCSP($\Gamma_c \cup \{u_a|a \in D\}$) is tractable,*

3. *VCSP*(Γ) *is* **NP***-hard if and only if VCSP*($\Gamma_c \cup \{u_a | a \in D\}$) *is* **NP***-hard.*

It is therefore sufficient to consider only core valued constraint languages Γ which are closed under fixing values for a subset of variables and which contain a unary function u_a with $\operatorname{argmin}(u_a) = \{a\}$ for each $a \in D$. The last condition also implies that $\mathrm{fPol}(\Gamma_c)$ consists of the idempotent members of $\mathrm{fPol}(\Gamma)$, a condition which proved important in the algebraic approach to the CSP [3, 10].

Before proving Proposition 1 we need an auxiliary lemma. For a mapping $\varphi : \{x_a | a \in D\} \to D$, let s_φ be the unary operation on D such that $s_\varphi(a) = \varphi(x_a)$.

Lemma 2. *Assume that* Γ *is a core. There exists an instance* $\mathcal{I}_p$ *of* $\mathrm{VCSP}(\Gamma)$ *with variables* $V = \{x_a | a \in D\}$ *such that, for each optimal solution* $\varphi \in Opt(\mathcal{I}_p)$*, the following holds:*

1. *the operation* s_φ *is injective (i.e. a permutation),*
2. *for every instance* $\mathcal{I}'$ *of* $\mathrm{VCSP}(\Gamma)$ *and every* $\varphi' \in Opt(\mathcal{I}')$*, the mapping* $s_\varphi \circ \varphi'$ *is also in* $Opt(\mathcal{I}')$*.*

Proof. Since Γ is a core, for every element $a \in D$, there exists an instance $\mathcal{I}_a$ of $\mathrm{VCSP}(\Gamma)$ such that a is in the image of all optimal solutions to $\mathcal{I}_a$. Assume without loss of generality that the sets of variables in $V_{\mathcal{I}_a}$ in these instances are pairwise disjoint. Let f_a be the objective function in $\mathcal{I}_a$, and consider the instance $\mathcal{I}_1$ of $\mathrm{VCSP}(\Gamma)$ whose objective function is $\sum_{a \in D} f_a$. The image of every optimal solution to $\mathcal{I}_1$ must be equal to D. Arbitrarily choose one optimal solution to $\mathcal{I}_1$ and, for each $a \in D$, identify with x_a all variables in $V_{\mathcal{I}_1}$ that are mapped to a in the chosen solution. We get a new instance $\mathcal{I}_2$ of $\mathrm{VCSP}(\Gamma)$ with variables $V = \{x_a | a \in D\}$. Notice that the image of each optimal solution φ to $\mathcal{I}_2$ is still all of D because any optimal solution to $\mathcal{I}_2$ gives rise to an optimal solution to $\mathcal{I}_1$ with the same image. Hence, $\mathcal{I}_2$ satisfies condition (1) of the lemma, and the mapping φ_{id} defined by $\varphi_{id}(x_a) = a$ is an optimal solution to $\mathcal{I}_2$. Let f_2 denote the objective function of $\mathcal{I}_2$.

Let $\varphi \in Opt(\mathcal{I}_2)$ be such that s_φ falsifies condition (2) of the lemma. That is, there is an instance $\mathcal{I}_3$ of $\mathrm{VCSP}(\Gamma)$ and $\varphi_3 \in Opt(\mathcal{I}_3)$ such that $s_\varphi \circ \varphi_3$ is not optimal for

$\mathcal{I}_3$. Clearly, $\varphi \neq \varphi_{id}$. For each $a \in D$, identify with x_a all variables x in $V_{\mathcal{I}_3}$ with $\varphi_3(x) = a$. The obtained instance $\mathcal{I}_4$ has the following properties: $V_{\mathcal{I}_4} \subseteq \{x_a | a \in D\}$, the mapping φ_4 defined as the restriction of φ_{id} to $V_{\mathcal{I}_4}$ is an optimal solution to $\mathcal{I}_4$, while $s_\varphi \circ \varphi_4$ is not. Let f_4 denote the objective function of $\mathcal{I}_4$, and consider the instance $\mathcal{I}_5$ with variables $V_{\mathcal{I}_5} = \{x_a | a \in D\}$ and objective function $W \cdot f_2 + f_4$ where W is large enough to ensure that each optimal solution to $\mathcal{I}_5$ must be an optimal solution to $\mathcal{I}_2$. Furthermore, notice that φ_{id} is an optimal solution of $\mathcal{I}_5$, while φ is not. Thus, we will replace $\mathcal{I}_2$ with $\mathcal{I}_5$ and repeat this procedure until we remove from the set of optimal solutions all mappings φ such that s_φ falsifies condition (2) of the lemma. Since there are finitely many such mappings, we eventually obtain the desired instance $\mathcal{I}_p$. □

Proof. (of Proposition 1). To prove item (1), take the instance $\mathcal{I}_p$ from Lemma 2. Let f_p be the objective function of $\mathcal{I}_p$. For any $a \in D$, consider the unary function u_a obtained from f_p by fixing each x_b with $b \neq a$ to b. Since φ_{id} is an optimal solution to $\mathcal{I}_p$, $a \in argmin(u_a)$. On the other hand, by Lemma 2(1), $u_a(a) < u_a(b)$ for each $b \neq a$. It is easy to see that $u_a \in \langle \Gamma_c \rangle$.

By Theorem 6, the problems VCSP(Γ_c) and VCSP($\Gamma_c \cup \{u_a | a \in D\}$) have the same complexity. Therefore, to prove items (2) and (3), it suffices to show that problems VCSP(Γ) and VCSP(Γ_c) have the same complexity. Take an arbitrary instance $\mathcal{I}$ of VCSP(Γ_c) and assume without loss of generality that the sets of variables of $\mathcal{I}$ and $\mathcal{I}_p$ are disjoint. Replace each f_c in $\mathcal{I}$ by the corresponding f with constants, and then replace each constant a by the variable x_a. The new instance $\mathcal{I}_1$ is an instance of VCSP(Γ), denote its objective function by f_1. Consider the instance $\mathcal{I}_2$ of VCSP(Γ) with objective function $f_1 + W \cdot f_p$ where W is a large enough number to ensure that each optimal solution to $\mathcal{I}_2$, when restricted to $V_{\mathcal{I}_p} = \{x_a | a \in D\}$, is an optimal solution to $\mathcal{I}_p$. Since φ_{id} is an optimal solution to $\mathcal{I}_p$, the optimal solutions to $\mathcal{I}$ are precisely restrictions to $V_{\mathcal{I}}$ of those optimal solutions to $\mathcal{I}_2$ whose restriction on $V_{\mathcal{I}_p}$ is φ_{id}.

Each optimal solution φ to $\mathcal{I}$ gives rise to an optimal solution to $\mathcal{I}_2$ which coincides with φ on $V_{\mathcal{I}}$ and with φ_{id} on $V_{\mathcal{I}_p}$. In the other direction, let φ_2 be an optimal solution

to $\mathcal{I}_2$. Its restriction to $V_{\mathcal{I}_p}$ is an optimal solution $\varphi_2^{'}$ to $\mathcal{I}_p$. By Lemma 2(1), the operation $s_{\varphi_2^{'}}$ is a permutation on D. By applying Lemma 2 to $\mathcal{I}' = \mathcal{I}_p$, it follows that each mapping ψ_t such that $s_{\psi_t} = s^t_{\varphi_2^{'}}$, $t \geq 1$, is an optimal solution to $\mathcal{I}_p$. Choose t so that $s^t_{\varphi_2^{'}} = s^{-1}_{\varphi_2^{'}}$. Now apply Lemma 2(2) to ψ_t and $\varphi_2 \in Opt(\mathcal{I}_2)$. It follows that $\varphi_2^{''} = s^{-1}_{\varphi_2^{'}} \circ \varphi_2$ is an optimal solution to $\mathcal{I}_2$, and by construction $\varphi_2^{''}(x_a) = a$ for each $a \in D$. Hence, the restriction of $\varphi_2^{''}$ to $V_{\mathcal{I}}$ is an optimal solution to $\mathcal{I}$. □

Thapper and Živný [59] present a different definition of a core but then go on to show their definition coincides with that of Definition 22. First we need the following definition.

Definition 23 ([59])**.** *Let $S \subseteq D$. The sub-language $\Gamma[S]$ of Γ induced by S is the valued constraint language defined on domain S and containing the restriction of every function $f \in \Gamma$ onto S.*

Definition 24 ([59])**.** *A valued constraint language Γ is a core if for every unary fractional polymorphism ω of Γ, supp(ω) contains only injective operations. A valued constraint language Γ' is a core of Γ if Γ' is a core and $\Gamma' = \Gamma[h(D)]$ for some $h \in supp(\omega)$ with ω a fractional polymorphism of Γ.*

The proof that the definitions of cores given in Definition 22 and Definition 24 are equivalent is given in Lemma 4.2 of [59].

Finally we consider the most recent definition of a core valued constraint language as given by Kozik and Ochremiak [45].

Definition 25 ([45])**.** *A valued constraint language Γ is a core if for every unary fractional polymorphism ω of Γ, supp(ω) contains only bijective operations. Γ is called a rigid core valued constraint language if it exhibits exactly one unary polymorphism, which is the identity.*

The definition of core constraint languages given by Thapper and Živný (Definition 24) and Kozik and Ochremiak (Definition 25) are identical, however it is shown in [45] that we can restrict ourselves to rigid cores when searching for tractable valued constraint languages, thus simplifying the problem.

Now we remark that the complexity classification for VCSP over two element do-

mains with only finite costs (Theorem 9) can be simplified using the notion of core constraint languages. The first two cases, $\langle 0 \rangle$ and $\langle 1 \rangle$, clearly imply that the constraint language Γ is not core and thus we get the following simple theorem:

Theorem 15. *Let Γ be a core valued Boolean constraint language with costs in $\mathbb{Q}_+$. If Γ has the multimorphism $\langle Min, Max \rangle$ then $VCSP(\Gamma)$ is tractable, otherwise it is* **NP***-hard.*

2.4 Multimorphisms as Digraphs

All known tractable finite-valued constraint languages we have examined so far exhibit symmetric binary multimorphisms, that is, for all pairs of domain elements x and y, if $f(x, y) = x$ then $g(x, y) = y$, or if $f(x, y) = z$ then $g(x, y) = z$. Examples of symmetric binary multimorphisms include Submodularity [24, 48, 54], Bisubmodularity [1, 24, 50, 52] and Tournament Pairs [16, 44]. The simplicity of symmetric binary multimorphisms allows for them to be easily represented as digraphs. First let each element of the domain of our problem be a vertex in the digraph, and then represent the functions f and g as arcs. If $f(x, y) = x$ then we draw a directed arc from vertex y to vertex x, and the function g is that opposing the direction of the arcs, so $g(x, y) = y$. If f and g are non-conservative i.e. $f(x, y) = z$ then $g(x, y) = z$, then we draw no arc between the vertices x and y.

As we have previously seen, we only want to consider core constraint languages when proving complexity classifications. However we know from Theorem 14 that certain multimorphisms imply the underlying constraint language is not core, hence the digraphs representing those multimorphisms should be excluded.

The following lemma of Thapper and Živný [59] shows that we only need to consider connected digraphs, else the underlying constraint language is not core.

Take a digraph describing a fractional polymorphism of a core valued constraint language Γ over D and replace the directed edges with undirected edges, call this graph S. Let T be a graph that records the definable two element subsets of the domain D. That is T is the undirected graph $T = (V(T), E(T))$ with:

- $V(T) = D$

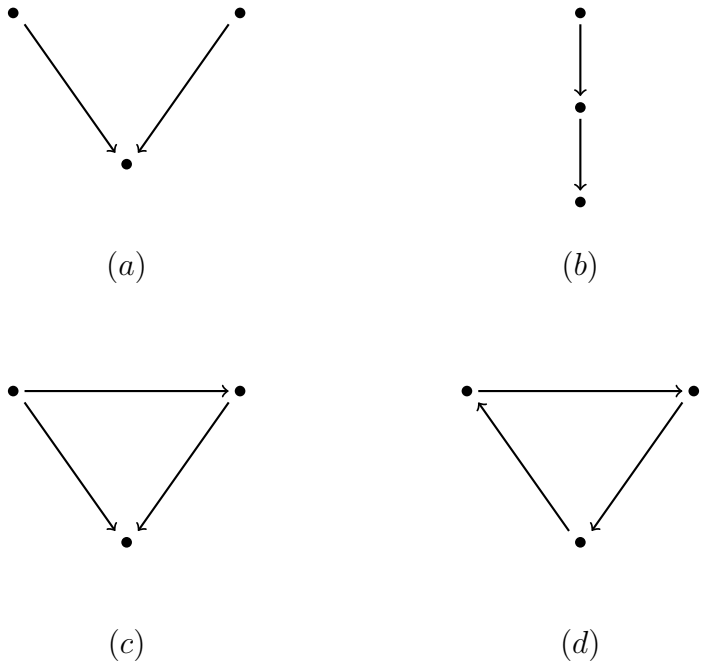


Figure 2.1: The four possible connected digraphs on three vertices.

- $E(T) = \{\{a, b\} |$ there exists a unary function u expressible from Γ such that argmin $u = \{a, b\}\}$

Lemma 3. *$E(T) \subseteq E(S)$ and T is connected.*

Over small domains where it is possible to easily draw all possible connected digraphs this could help identify novel multimorphisms that imply tractability, and perhaps build dichotomy theorems from. It is trivial to check that the only connected digraph on the two element domain describes submodularity, and we know this is the only multimorphism responsible for tractability of finite-valued constraint languages over two elements (see Theorem 15).

2.4.1 3-element domain

Drawing and examining all the possible binary multimorphisms on three elements may lead to novel tractable cases as yet unidentified, and help build a dichotomy theorem. Consider all the possible connected digraphs on three vertices, they all fall into one of the four classes seen in Figure 2.1.

Firstly lets consider the digraphs (c) and (d) of Figure 2.1. The multimorphisms described by these digraphs are known as symmetric tournament pair multimor-

phisms [16], which we defined in Definition 20.

The tractability of valued constraint languages that admit a symmetric tournament pair multimorphism was proven in [16] by a reduction to the problem of minimising submodular functions. It is also shown that any valued constraint language with a tournament pair multimorphism is tractable by reduction to a symmetric tournament pair [16]. Note that by construction of our digraphs we can only describe symmetric tournament pairs as the function g is always the dual of f. It has more recently been shown that any valued constraint language with finite costs that admits a symmetric tournament pair multimorphism is also submodular with respect to some total ordering of its domain elements [43]. Due to this we do not need to consider these multimorphisms in a complexity classification on the three element domain as we know it must include submodularity already.

Now consider the digraphs (a) and (b). Both of these digraphs describe well known multimorphisms already known to be tractable. Digraph (a) describes bisubmodularity, often seen written as $\langle min_0, max_0 \rangle$, which in this case would imply the vertex with the two incoming arcs is labelled 0. Digraph (b) describes submodularity, with the total order of the domain elements in the direction of the arcs. Therefore these two digraphs offer nothing new in the search for more tractable fragments.

This confirms that the only symmetric binary multimorphisms that identify tractable core valued constraint languages on three element domains are $\langle Min, Max \rangle$ and $\langle min_0, max_0 \rangle$. This leads to a number of questions as to why no dichotomy has yet been proven. Firstly, is the three element domain the first case that requires the more general notion of fractional polymorphisms to define its tractable cases? Possibly there is some binary multimorphism which is non-conservative and non-symmetric that has not been considered, or it might even be required to consider ternary multimorphisms? Finally it may be that there is some alternative reason for hardness on three element domains that is yet to be identified. In Chapter 3 we will consider an example of a core valued constraint language which will help to ultimately answer these questions.

Chapter 3

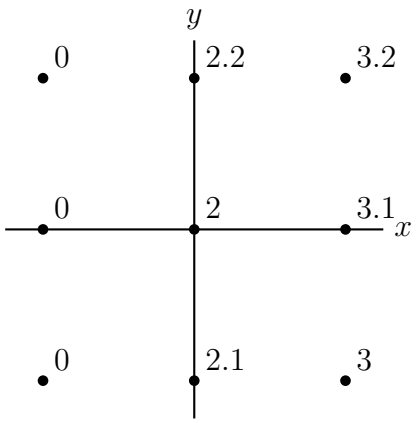
Skew Bisubmodularity and the Three Element Dichotomy

3.1 Introduction

As the approach of drawing multimorphisms as digraphs had yielded no clues in the hunt for a dichotomy for finite-valued constraint languages on the three element domain it is clear a different tactic is required. The idea this time is to construct an example of a core finite-valued constraint language which is neither submodular or bisubmodular, yet also not obviously **NP**-hard. The hope is that such an example will lead to a new tractable class of constraint languages or a new reason for hardness that is unrequired on the two element domain. We thank Vladimir Kolmogorov for the following example.

Note that throughout this chapter we often represent a unary function $f : D \to \mathbb{Q}_+$ where $D = \{-1, 0, 1\}$ as a vector of values $[f(-1), f(0), f(1)]$ for simplicity.

Example 17. *Let Γ be the valued constraint language over $D = \{-1, 0, 1\}$ which consists of the three unary functions $u_1 = [5, 1, 0]$, $u_2 = [1, 0, 0]$ and $u_3 = [0, 0, 1]$ and the binary function h defined below:*



Note that we can also express the unary function $u_4(x) = min_y h(x, y) = [0, 2, 3]$.

The unary cost functions $u_1, u_2, u_3, u_4 \in \Gamma$ prohibit all unary polymorphisms except for the trivial identity polymorphism, hence Γ is a rigid core valued constraint language. Γ is not submodular under any labelling of domain elements, in this particular labelling $h(1,-1)+h(-1,1) \not\geq h(-1,-1)+h(1,1)$. Γ also does not exhibit a bisubmodular multimorphism. The unary function u_2 excludes bisubmodularity with root -1, u_3 excludes root 1 and u_4 excludes root 0.

3.2 Fractional Polymorphisms

Let us continue to consider the constraint language Γ from Example 17. Using a quick computational search it is easy to identify number of fractional polymorphisms expressed by Γ. If we fix the domain ordering to $-1 > 0 < 1$, one of those fractional polymorphisms identified is the following:

$$\phi(x) + \phi(y) \geq \frac{1}{3}\phi(x \wedge_0 y) + \frac{1}{3}\phi(x \vee_0 y) + \frac{2}{3}\phi(p_{ca}(x, y)) + \frac{2}{3}\phi(q_{ac}(x, y))$$

for all $x, y \in D$, where

$$p_{ca}(x,y) = \begin{cases} 0 & \text{if (x,y)=(1,-1)} \\ x & \text{otherwise} \end{cases} \quad \text{and } q_{ac}(x,y) = \begin{cases} 0 & \text{if (x,y)=(-1,1)} \\ y & \text{otherwise} \end{cases}$$

This fractional polymorphism has a bisubmodular type term and two functions p_{ca} and q_{ac} that are nearly first and second projections respectively. We now know that VCSP(Γ) is tractable by BLP [57] because this fractional polymorphism has the semilattice operation $\wedge_0$. The problem with this fractional polymorphism is that it

is very specific to the example constraint language, and the operations p_{ca} and q_{ac} are not symmetric. It would be ideal if we could replace p_{ca} and q_{ac} with alternative operations, and to see if this is possible we apply the fractional polymorphism to itself.

Substitute x with $p_{ca}(x, y)$ and y with $q_{ac}(x, y)$ and apply the fractional polymorphism to obtain the inequality:

$$\begin{aligned}\phi(p_{ca}(x,y)) + \phi(q_{ac}(x,y)) \geq \frac{1}{3}\phi((p_{ca}(x,y)) \wedge_0 (q_{ac}(x,y))) + \frac{1}{3}\phi((p_{ca}(x,y)) \vee_0 (q_{ac}(x,y))) \\ + \frac{2}{3}\phi(p_{ca}((p_{ca}(x,y)),(q_{ac}(x,y)))) + \frac{2}{3}\phi(q_{ac}((p_{ca}(x,y)),(q_{ac}(x,y))))\end{aligned}$$

If we test all possible values for x and y we can simplify the terms on the RHS to obtain the inequality:

$$\phi(p_{ca}(x,y)) + \phi(q_{ac}(x,y)) \geq \frac{1}{3}\phi(x \wedge_0 y) + \frac{1}{3}\phi(x \vee_{-1} y) + \frac{2}{3}\phi(p_{ca}(x,y)) + \frac{2}{3}\phi(q_{ac}(x,y))$$

Rearrange the inequality and scale to get:

$$\phi(p_{ca}(x,y)) + \phi(q_{ac}(x,y)) \geq \phi(x \wedge_0 y) + \phi(x \vee_{-1} y)$$

Substituting back into our original fractional polymorphism we obtain:

$$\phi(x) + \phi(y) \geq \phi(x \wedge_0 y) + \frac{1}{3}\phi(x \vee_0 y) + \frac{2}{3}\phi(x \vee_{-1} y)$$

The resulting fractional polymorphism now looks very similar to the bisubmodularity multimorphism, but the weight of the operation $\vee_0$ has been reduced to allow a positive weight to be associated with the operation $\vee_{-1}$. This leads us to the following definition.

Definition 26. *Let $\alpha \in (0, 1]$. We say that a function $f : \{-1, 0, 1\}^n \to \mathbb{Q}_+$ is* α-bisubmodular (towards 1) *if it has the fractional polymorphism μ such that $\mu(\wedge_0) = 1/2$, $\mu(\vee_0) = \alpha/2$, and $\mu(\vee_1) = (1 - \alpha)/2$.*

I.e. A function f, of arity n, is α-bisubmodular (towards 1) if, for all tuples $\mathbf{a}, \mathbf{b} \in \{-1, 0, 1\}^n$,

$$f(\mathbf{a} \wedge_0 \mathbf{b}) + \alpha \cdot f(\mathbf{a} \vee_0 \mathbf{b}) + (1 - \alpha) \cdot f(\mathbf{a} \vee_1 \mathbf{b}) \leq f(\mathbf{a}) + f(\mathbf{b}). \tag{3.2.1}$$

A unary function f is α-bisubmodular if and only if $(1+\alpha)\cdot f(0) \leq f(-1) + \alpha \cdot f(1)$.

Note that α-bisubmodular functions towards -1 can be defined by using $\vee_{-1}$ instead of $\vee_1$. For simplicity we assume that α-bisubmodular functions are skew towards 1, unless explicitly stated otherwise. Notice also that the 1-bisubmodular functions (towards 1 or -1) are the ordinary bisubmodular functions from Example 12.

Example 18. *Let us reconsider the functions in Example 17. It is simple to check that all of those functions are $\frac{1}{3}$-bisubmodular (towards -1). As a concrete example we show how easy it is to check the unary function $u_4(x) = [0, 2, 3]$. A function is $\frac{1}{3}$-bisubmodular (towards -1) if it satisfies the inequality*

$$\phi(x) + \phi(y) \geq \phi(x \wedge_0 y) + \frac{1}{3}\phi(x \vee_0 y) + \frac{2}{3}\phi(x \vee_{-1} y)$$

The only interesting case is $\{x, y\} = \{1, -1\}$ as otherwise this inequality checks for submodularity which is trivially true for unary functions. Let $x = 1$ and $y = -1$, we get:

$$\begin{aligned}\phi(1) + \phi(-1) &\geq \phi(0) + \frac{1}{3}\phi(0) + \frac{2}{3}\phi(-1)\\ \Rightarrow 3 &\geq 2 + \frac{2}{3}.\end{aligned}$$

Thus we satisfy the inequality and this unary function is $\frac{1}{3}$-bisubmodular (towards -1).

In fact checking if functions are α-bisubmodular is very simple, as we prove in the following section.

3.3 A Characterisation of α-bisubmodularity

Let $\leq$ denote the partial order on $\{-1, 0, 1\}$ such that $0 \leq t$ for all $t \in \{-1, 0, 1\}$ and -1 and 1 are incomparable. Also let $\leq$ denote the componentwise partial order on $\{-1, 0, 1\}^n$. For every $\mathbf{c} \in \{-1, 1\}^n$, let $\mathbf{c}^{\downarrow} = \{\mathbf{x} \in \{-1, 0, 1\}^n | \mathbf{x} \leq \mathbf{c}\}$. This set is called the orthant of $\mathbf{c}$.

Definition 27. *For every $\boldsymbol{c} \in \{-1, 1\}^n$, a function $f : \{-1, 0, 1\}^n \to \mathbb{Q}_+$ is submodular in the orthant of $\boldsymbol{c}$ if the α-bisubmodularity inequality holds for all $\boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{c}^{\downarrow}$.*

Note that in any given orthant, in each coordinate only one of -1 or 1 can appear. Therefore the condition of satisfying α-bisubmodularity simply becomes submodularity with the elements ordered as $0 < 1$ and $0 < -1$.

Proposition 2. *A function $f : \{-1, 0, 1\}^n \to \mathbb{Q}_+$ is α-bisubmodular for some $\alpha \in (0, 1]$ if and only if the following conditions hold:*

1. *f is submodular in every orthant*
2. *every unary function obtained from f by fixing values for all but one variable is α-bisubmodular*

In order to prove this proposition we first need the following lemma.

Lemma 4. *If $f : \{-1, 0, 1\}^n \to \mathbb{Q}_+$ is submodular in every orthant and every unary function obtained from f by fixing all but one variable is α-bisubmodular then every unary function obtained from f by fixing and identifying variables is α-bisubmodular.*

Proof. Let f be a function that satisfies the conditions of the lemma. Let $g : \{-1, 0, 1\}^m \to \mathbb{Q}_+$, with $1 \leq m \leq n$, be the function obtained from f by fixing values for some variables. Let $h : \{-1, 0, 1\} \to \mathbb{Q}_+$ be obtained from g by identifying all the remaining variables i.e. $h(x) = g(x, ..., x)$. We have to show that $(1 + \alpha)h(0) \leq h(-1) + \alpha h(1)$.

We will use induction on m. The case $m = 1$ holds by the assumption of the lemma, now assume the result holds for $m - 1$.

Induction hypothesis applied to g(x,...,x,1) gives:

$$g(-1, ..., -1, 1) + \alpha g(1, ..., 1, 1) \geq (1 + \alpha)g(0, ..., 0, 1)$$

Submodularity in the orthant of (-1,...,-1,1) gives:

$$g(-1, ..., -1, 0) + g(0, ..., 0, 1) \geq g(0, ..., 0, 0) + g(-1, ..., -1, 1)$$

The assumption on unary functions applied to g(-1,...,-1,x) gives:

$$g(-1, ..., -1, -1) + \alpha g(-1, ..., -1, 1) \geq (1 + \alpha)g(-1, ..., -1, 0)$$

Multiplying the second inequality by $(1 + \alpha)$ and summing the three inequalities gives the required $h(-1) + \alpha h(1) \geq (1 + \alpha)h(0)$. □

Using this lemma we can now prove Proposition 2.

Proof. (of Proposition 2). The only if direction follows from the definitions, so it remains to prove the other direction. Let f satisfy conditions (1) and (2) of the proposition. By Lemma 4 every unary function obtained from f by fixing and identifying variables is α-bisubmodular.

Let $\mathbf{x}, \mathbf{y} \in \{-1, 0, 1\}^n$. For any $x, y \in D$, we have $x \wedge_0 y \leq x \leq (x \vee_0 y) \vee_0 x$ and $(x \vee_0 y) \wedge_0 y \leq (x \vee_0 y) \vee_0 x$, and thus $\mathbf{x} \wedge_0 \mathbf{y}, \mathbf{x}, (\mathbf{x} \vee_0 \mathbf{y}) \wedge_0 \mathbf{y}$ and $(\mathbf{x} \wedge_0 \mathbf{y}) \wedge_0 \mathbf{x}$ are all in the orthant of $\mathbf{x} \vee_0 \mathbf{y}) \vee_0 \mathbf{x}$. This gives

$$f(\mathbf{x}) + f((\mathbf{x} \vee_0 \mathbf{y}) \wedge_0 \mathbf{y}) \geq f(\mathbf{x} \wedge_0 \mathbf{y}) + f((\mathbf{x} \vee_0 \mathbf{y}) \vee_0 \mathbf{x}) \tag{3.3.2}$$

For any $x, y \in D$, we have $(x \vee_0 y) \wedge_0 y \leq y \leq (x \vee_0 y) \vee_0 y$ and $x \vee_0 y \leq (x \vee_0 y) \vee_0 y$ and thus $(\mathbf{x} \vee_0 \mathbf{y}) \wedge_0 \mathbf{y}, \mathbf{y}, \mathbf{x} \vee_0 \mathbf{y}$ and $(\mathbf{x} \vee_0 \mathbf{y}) \vee_0 \mathbf{y}$ are all in the orthant of $(\mathbf{x} \vee_0 \mathbf{y}) \vee_0 \mathbf{y}$. This gives

$$f(\mathbf{y}) + f(\mathbf{x} \vee_0 \mathbf{y}) \geq f((\mathbf{x} \vee_0 \mathbf{y}) \wedge_0 \mathbf{y}) + f((\mathbf{x} \vee_0 \mathbf{y}) \vee_0 \mathbf{y}) \tag{3.3.3}$$

For $a, b \in \{-1, 0, 1\}$, define the binary operations $\vee_{a,b}$ as follows:

$$x \vee_{a,b} y = \begin{cases} x \vee_0 y & \text{if } \{x, y\} \neq \{-1, 1\} \\ a & \text{if } x = -1 \text{ and } y = 1 \\ b & \text{if } x = 1 \text{ and } y = -1 \end{cases}$$

For any $x, y \in D$, we have $x \vee_0 y \leq x \vee_{0,1} y \leq x \vee_1 y$ and $x \vee_{1,0} y \leq x \vee_1 y$, and thus $\mathbf{x} \vee_0 \mathbf{y}, \mathbf{x} \vee_{0,1} \mathbf{y}, \mathbf{x} \vee_{1,0} \mathbf{y}$ and $\mathbf{x} \vee_1 \mathbf{y}$ are all in the orthant of $\mathbf{x} \vee_1 \mathbf{y}$. This gives

$$f(\mathbf{x} \vee_{0,1} \mathbf{y}) + f(\mathbf{x} \vee_{1,0} \mathbf{y}) \geq f(\mathbf{x} \vee_0 \mathbf{y}) + f(\mathbf{x} \vee_1 \mathbf{y}) \tag{3.3.4}$$

Applying α-bisubmodularity of the unary function obtained from f by identifying all those variables with indices i such that $\mathbf{x}_i = -1$ and $\mathbf{y}_i = 1$ and fixing all other variables gives

$$f((\mathbf{x} \vee_0 \mathbf{y}) \vee_0 \mathbf{x}) + \alpha f(\mathbf{x} \vee_1 \mathbf{y}) \geq (1 + \alpha) f(\mathbf{x} \vee_{0,1} \mathbf{y}) \tag{3.3.5}$$

The same arguement with identifying all those variables with indices i such that $\mathbf{x}_i = 1$ and $\mathbf{y}_i = -1$ and fixing all other variables gives

$$f((\mathbf{x} \vee_0 \mathbf{y}) \vee_0 \mathbf{y}) + \alpha f(\mathbf{x} \vee_1 \mathbf{y}) \geq (1 + \alpha) f(\mathbf{x} \vee_{1,0} \mathbf{y}) \tag{3.3.6}$$

We can then create the following chain of inequalities

$$\begin{aligned}
&f(\mathbf{x}) + f(\mathbf{y}) + f(\mathbf{x} \vee_0 \mathbf{y}) + 2\alpha f(\mathbf{x} \vee_1 \mathbf{y}) \\
&\geq f(\mathbf{x}) + f((\mathbf{x} \vee_0 \mathbf{y}) \wedge_0 \mathbf{y}) + f((\mathbf{x} \vee_0 \mathbf{y}) \vee_0 \mathbf{y}) + 2\alpha f(\mathbf{x} \vee_1 \mathbf{y}) \\
&\geq f(\mathbf{x} \wedge_0 \mathbf{y}) + f((\mathbf{x} \vee_0 \mathbf{y}) \vee_0 \mathbf{x}) + f((\mathbf{x} \vee_0 \mathbf{y}) \vee_0 \mathbf{y}) + 2\alpha f(\mathbf{x} \vee_1 \mathbf{y}) \\
&\geq f(\mathbf{x} \wedge_0 \mathbf{y}) + (1+\alpha) f(\mathbf{x} \vee_{0,1} \mathbf{y}) + (1+\alpha) f(\mathbf{x} \vee_{1,0} \mathbf{y}) \\
&\geq f(\mathbf{x} \wedge_0 \mathbf{y}) + (1+\alpha) f(\mathbf{x} \vee_0 \mathbf{y}) + (1+\alpha) f(\mathbf{x} \vee_1 \mathbf{y})
\end{aligned}$$

where the first inequality follows from (3.3.3), the second from (3.3.2), the third from (3.3.5) and (3.3.6), and the final one from (3.3.4). By comparing and rearranging the first and final expressions in the chain of inequalities above we get the required

$$f(\mathbf{x}) + f(\mathbf{y}) \geq f(\mathbf{x} \wedge_0 \mathbf{y}) + \alpha f(\mathbf{x} \vee_0 \mathbf{y}) + (1-\alpha) f(\mathbf{x} \vee_1 \mathbf{y}).$$

□

3.4 A Dichotomy Theorem

In this section we will generalise the following two theorems, the classification for the Boolean case [17] and the complexity classification for the case of a three-element domain and 0-1-valued functions [37], to a general classification for the case of a three-element domain.

Theorem 16 ([17])**.** *Let* Γ *be a core constraint language on* $D = \{0, 1\}$*. Either* Γ *consists of submodular functions and VCSP*(Γ) *is tractable, or* Γ *can express MAX CUT and VCSP*(Γ) *is* **NP***-hard.*

Theorem 17 ([37])**.** *Let* $|D| = 3$ *and let* Γ *be a core constraint language on* D *consisting of 0-1-valued functions. If the elements of* D *can be labelled* $-1, 0, 1$ *such that each function in* Γ *is submodular on the chain* $-1 < 0 < 1$*, then VCSP*(Γ) *is tractable. Otherwise,* Γ *can express MAX CUT and VCSP*(Γ) *is* **NP***-hard.*

Now we generalise these two results as we present a dichotomy theorem for the three element domain with core finite-valued constraint languages, which is the main result of [31].

Theorem 18. *Let* $|D| = 3$ *and let* Γ *be a core constraint language on* D*. If the elements of* D *can be renamed* $-1, 0, 1$ *in such a way that:*

- *each function in* Γ *is submodular on the chain* $-1 < 0 < 1$*, or*
- *there is some* $\alpha \in (0, 1]$ *such that each function in* Γ *is* α*-bisubmodular*

then VCSP(Γ) *is tractable. Otherwise,* Γ *can express MAX CUT and VCSP*(Γ) *is* **NP***-hard.*

The tractability aspect follows directly from Theorem 3.1 of [57], which proves VCSP(Γ) can be solved in polynomial time by BLP (as discussed in Section 1.8). For the hardness part, by Lemma 1 and Proposition 1, it suffices to show that Γ can express MAX CUT. In order to prove this and complete the proof of Theorem 18 we need the following four lemmas.

By Proposition 1 we know that for a core valued constraint language Γ, the problem VCSP(Γ) has the same complexity as VCSP($\Gamma_c \cup \{u_a | a \in D\}$), hence let $\Gamma = \Gamma_c \cup \{u_a | a \in D\}$. Note that changing Γ in this fashion does not affect the properties identified in Theorem 18.

Lemma 5. *Let* Γ *be a core valued constraint language with* $|D| = 3$*. Assume for each* $a \in D$*,* Γ *contains a unary function* u_a *with argmin*$(u_a) = \{a\}$*. For at least two of the distinct two-element subsets* $X \subseteq D$*,* $\langle\Gamma\rangle$ *contains the unary function* u_X *with argmin*$(u_X) = X$*.*

Proof. Let $D = \{a, b, c\}$. For convenience we write a unary function f as a vector $[f(a), f(b), f(c)]$. By translating and scaling we assume $u_a = [0, 1, \beta]$, $u_b = [\gamma, 0, 1]$ and $u_c = [1, \delta, 0]$ where $\beta, \gamma, \delta > 0$.
Consider the following:

$$[1, \delta, 0] + (1 - \delta)[0, 1, \beta] = [1, 1, (1 - \delta)\beta]$$
$$(\beta - 1)[1, \delta, 0] + \delta[0, 1, \beta] = [\beta - 1, \beta\delta, \beta\delta]$$

If $(1 - \delta)\beta > 1$ then $1 - \delta > 0$ and $\beta - 1 > \beta\delta$, then the unary functions above are $u_{\{a,b\}}$ and $u_{\{b,c\}}$.
Now consider:

$$\gamma[1, \delta, 0] + (\delta - 1)[\gamma, 0, 1] = [\gamma\delta, \gamma\delta, \delta - 1]$$
$$(1 - \gamma)[1, \delta, 0] + [\gamma, 0, 1] = [1, (1 - \gamma)\delta, 1]$$

If $(1-\gamma)\delta > 1$ then $1-\gamma > 0$ and $\delta - 1 > \gamma\delta$, then the unary functions above are $u_{\{a,b\}}$ and $u_{\{a,c\}}$.

Finally consider:

$$[0,1,\beta] + (1-\beta)[\gamma,0,1] = [(1-\beta)\gamma,1,1]$$
$$(\gamma-1)[0,1,\beta] + \beta[\gamma,0,1] = [\beta\gamma,\gamma-1,\beta\gamma]$$

If $(1-\beta)\gamma > 1$ then $1-\beta > 0$ and $\gamma - 1 > \beta\gamma$, then the unary functions above are $u_{\{b,c\}}$ and $u_{\{a,c\}}$.

Thus assume that $(1-\delta)\beta \leq 1$, $(1-\gamma)\delta \leq 1$ and $(1-\beta)\gamma \leq 1$. Note that it is impossible for more than one of these inequalities to be equality. For example, if the first two inequalities are equalities then $1-\delta > 0$, so $\delta < 1$, which in turn implies that $1-\gamma > 1$, i.e $\gamma < 0$, which is not permitted. Therefore at least two of these inequalities must be strict.

Let $(1-\delta)\beta < 1$, $(1-\gamma)\delta < 1$ and $(1-\beta)\gamma = 1$, the unary functions $u_{\{a,c\}}$ and $u_{\{b,c\}}$ are generated as follows:

$$\beta[1,\delta,0] + [0,1,\beta] = [\beta,1+\beta\delta,\beta]$$
$$\delta[\gamma,0,1] + [1,\delta,0] = [1+\gamma\delta,\delta,\delta]$$

Let $(1-\delta)\beta < 1$, $(1-\beta)\gamma < 1$, and $(1-\gamma)\delta = 1$, the unary functions $u_{\{a,b\}}$ and $u_{\{a,c\}}$ are generated as follows:

$$\gamma[0,1,\beta] + [\gamma,0,1] = [\gamma,\gamma,\beta\gamma+1]$$
$$\beta[1,\delta,0] + [0,1,\beta] = [\beta,1+\beta\delta,\beta]$$

Let $(1-\beta)\gamma < 1$, $(1-\gamma)\delta < 1$, and $(1-\delta)\beta = 1$, the unary functions $u_{\{a,b\}}$ and $u_{\{b,c\}}$ are generated as follows:

$$\gamma[0,1,\beta] + [\gamma,0,1] = [\gamma,\gamma,\beta\gamma+1]$$
$$\delta[\gamma,0,1] + [1,\delta,0] = [1+\gamma\delta,\delta,\delta]$$

Finally we have the case where all three inequalities are strict, $(1-\beta)\gamma < 1$, $(1-\gamma)\delta < 1$, and $(1-\delta)\beta < 1$, and all three unary functions $u_{\{a,b\}}$, $u_{\{a,c\}}$ and $u_{\{b,c\}}$ are generated

as follows:

$$\gamma[0,1,\beta] + [\gamma,0,1] = [\gamma,\gamma,\beta\gamma+1]$$
$$\beta[1,\delta,0] + [0,1,\beta] = [\beta,1+\beta\delta,\beta]$$
$$\delta[\gamma,0,1] + [1,\delta,0] = [1+\gamma\delta,\delta,\delta]$$

Thus we can always express at least two of the three unary functions $u_{\{a,b\}}$, $u_{\{a,c\}}$ or $u_{\{b,c\}}$. □

By Theorem 6, the problems VCSP(Γ) and VCSP($\Gamma \cup \{u_X | u_X \in \langle\Gamma\rangle\}$) have the same complexity, hence we can now let $\Gamma = \Gamma \cup \{u_X | u_X \in \langle\Gamma\rangle\}$.

We can always rename the elements of D into $-1, 0, 1$ such that the two 2-element subsets guarenteed by Lemma 5 are $\{-1, 0\}$ and $\{0, 1\}$. From now on we assume that $D = \{-1, 0, 1\}$ and that Γ contains the unary functions $u_{\{-1,0\}}$ and $u_{\{0,1\}}$, as well as the functions u_{-1},u_0 and u_1 as Γ is core. The unary function $u_{\{-1,1\}}$ may or may not be present in Γ. By translating and scaling we can assume $u_{\{-1,0\}} = [0, 0, 1]$ and $u_{\{0,1\}} = [1, 0, 0]$.

Lemma 6. *One of the following holds.*

1. $\langle\Gamma\rangle$ *contains a function* $u_{\{-1,1\}}$ *such that* $\operatorname{argmin}(u_{\{-1,1\}}) = \{-1, 1\}$,
2. *for some* $\alpha \in (0, 1]$, *every unary function in* $\langle\Gamma\rangle$ *is* α*-bisubmodular towards 1,*
3. *for some* $\alpha \in (0, 1]$, *every unary function in* $\langle\Gamma\rangle$ *is* α*-bisubmodular towards* -1.

Proof. Assume $\langle\Gamma\rangle$ contains a unary function h with $h(-1) \leq h(0) \geq h(1)$. If only one of the inequalities is strict we can add h to one of u_{-1} or u_1 with a suitable coefficient to obtain a function where both inequalities are strict. Once both inequalities are strict we can add to h either $u_{\{-1,0\}}$ and $u_{\{0,1\}}$ with a suitable coefficient to obtain a function u' with $\operatorname{argmin}(u') = \{-1, 1\}$, which can be translated and scaled into $u_{\{-1,1\}}$. If condition (1) of the lemma does not hold then we can make the following

assumption:

$$\text{No unary function } h \in \langle\Gamma\rangle \text{ satisfies } h(-1) \leq h(0) \geq h(1), \text{ unless } h(-1) = h(0) = h(1). \tag{*}$$

Now we need to show that every unary function in $\langle\Gamma\rangle$ is α-bisubmodular towards 1 or towards -1. Let Λ be the set of all unary functions from D to $\mathbb{Q}$ obtained by translating and scaling each function in $\langle\Gamma\rangle_{\equiv}$ so that each function $g \in \Lambda$ satisfies $g(0) = 0$ and $g(-1) \in \{-1, 0, 1\}$. It suffices to show that there is an $\alpha \in (0, 1]$ such that all $g \in \Lambda$ satisfy $0 \leq \alpha \cdot g(1) + g(-1)$, hence α-bisubmodular towards 1, or all $g \in \Lambda$ satisfy $0 \leq \alpha \cdot g(-1) + g(1)$ hence α-bisubmodular towards -1. Note that when we scale the functions u_{-1} and u_1 to be in Λ they have the following properties: $u_{-1}(-1) = -1$, $u_{-1}(1) > 0$, $u_1(-1) \in \{0, 1\}$ and $u_1(1) < 0$.

We prove the case that shows all functions in Λ must be α-bisubmodular towards 1, else we get a contradiction with (*). It is easy to see that if all unary functions in Λ satisfy $f(-1) + f(1) \geq 0$, then they are all 1-bisubmodular. Therefore consider the case of a unary function $f \in \Lambda$ where $f(-1) + f(1) < 0$. We perform a case analysis on the three possible values for $f(-1)$.

1. $f(-1) = 0$. Then $f(1) < 0$, but this contradicts (*).

2. $f(-1) = 1$. Let Z_1 be the set of all unary functions g in Λ with $g(-1)+g(1) < 0$ and $g(-1) = 1$. Then $g(1) < -1$ for all $g \in Z_1$. Let $\alpha = \inf_{g \in Z_1}(-\frac{1}{g(1)})$, then $0 \leq \alpha < 1$. If $\alpha = 0$ there is a $g' \in Z_1$ with $g'(1) < -u_{-1}(1)$, but the function $g' + u_{-1}$ contradicts (*). If $\alpha > 0$ all unary functions $g \in \Lambda$ with $g(-1) = 1$ are α-bisubmodular towards 1. Now either all unary functions in Λ are α-bisubmodular towards 1, or there is a unary function $h \in \Lambda$ with $h(-1) = \{0, -1\}$ and $\alpha \cdot h(1) + h(-1) < 0$. If $h(-1) = 0$, then $h(1) < 0$, and we contradict (*). Let $h(-1) = -1$, then $h(1) > 0$ else we contradict (*), and the inequality $\alpha \cdot h(1) + h(-1) < 0$ forces $\alpha < \frac{1}{h(1)}$. By definition of α there is a unary function $g' \in Z_1$ with $-\frac{1}{g(1)} < \frac{1}{h(1)}$. The function $g' + h$ contradicts (*).

3. $f(-1) = -1$. Let Z_{-1} be the set of all unary functions $g \in \Lambda$ with $g(-1) = -1$. Now $g(1) > 0$ for all $g \in Z_{-1}$ by (*). Let $\alpha = \inf_{g \in Z_{-1}} g(1)$. Since $f \in Z_{-1}$ and $f(1) < 1$, we have $0 \leq \alpha < 1$. If $\alpha = 0$ there is a unary function $g \in Z_{-1}$

with $g(1) < -u_1(1)$, and the function $g + u_1$ contradicts (*).

A similar case analysis shows that all functions in Λ must be α-bisubmodular towards -1, or else one gets a contradiction with Assumption (*). □

If $u_{\{-1,1\}} \in \langle\Gamma\rangle$, by Theorem 6, the problems VCSP(Γ) and VCSP($\Gamma \cup u_{\{-1,1\}}$) have the same complexity, hence we can let $\Gamma = \Gamma \cup u_{\{-1,1\}}$.

Let us first consider the case when $u_{\{-1,1\}} \in \Gamma$, where $u_{\{-1,1\}} = [0, 1, 0]$. Clearly, any function from $\Phi_D^{(1)}$ can be obtained as a positive linear combination of $u_{\{-1,0\}}$, $u_{\{0,1\}}$, and $u_{\{-1,1\}}$. This means that Γ contains all unary functions. Such constraint language are called *conservative*, and their complexity is classified in [44]. This classification can be stated as follows: either Γ has an STP multimorphism or else Γ can express MAX CUT. It is now known that an STP multimorphism can be simply reduced to submodularity, thus in this case Theorem 18 holds.

Let us assume for the rest of this section that $u_{\{-1,1\}} \notin \Gamma$. By Lemma 6, we have that, for some $\alpha \in (0, 1]$, the unary functions in Γ are α-bisubmodular, all towards 1 or all towards -1. Let us assume they are all α-bisubmodular towards 1, the other case is symmetric. If every function in Γ is α-bisubmodular then we are done. Otherwise, by Proposition 2, Γ contains a function which is not submodular in some orthant. The following lemma is well known, see [61]. The notion of submodularity from Example 11 can be naturally extended to the direct product of lattices, by defining the operations component-wise.

Lemma 7. *Let $D_1, \ldots, D_n$ be finite chains. If a function $f : D_1 \times \ldots \times D_n \to \mathbb{Q}_+$ is not submodular then some binary function obtained from f by fixing all but two coordinates is not submodular.*

By Lemma 7 we can assume that Γ contains a *binary* function which is not submodular in some orthant. If Γ contains a binary function which is not submodular in the orthant of (1,1) or (-1,-1) then, by Lemma 7.8 of [17], Γ expresses MAX CUT with $u_{\{-1,0\}}$ or $u_{\{0,1\}}$ respectively and we are done. So assume that Γ contains a binary function f that is not submodular in the orthant of (-1,1).

If every function in Γ is submodular on the chain $-1 < 0 < 1$ then we are done.

Otherwise, by Lemma 7, Γ contains a binary function, g, which is not submodular on this chain. We can assume that the function g is submodular both in the orthant of (1,1) and in the orthant of (-1,-1), otherwise we are done. If g satisfies both $g(1,0)+g(0,-1) \leq g(0,0)+g(1,-1)$ and $g(0,1)+g(-1,0) \leq g(0,0)+g(-1,1)$ then it can be easily checked that g is submodular on $-1 < 0 < 1$. Since this is not the case, at least one of the inequalities fails. We can assume, permuting the variables of g if necessary, that $g(1,0) + g(0,-1) > g(0,0) + g(1,-1)$. The following lemma finishes the proof of Theorem 18.

Lemma 8. *If Γ contains binary functions f and g such as above then $\langle\Gamma\rangle$ contains a binary function which is not submodular in the orthant of $(-1,-1)$.*

Proof. By translating, we can assume that $f(0,0) = 0 = g(0,0)$, so we have $f(0,1)+f(-1,0) < f(-1,1)$ and $g(1,-1) < g(1,0)+g(0,-1)$. We define the binary function f' as follows:

$$f'(x,y) = \begin{cases} f(x,y) + (f(-1,0) - f(0,1))u_{\{-1,0\}}(y) & \text{if } f(0,1) < f(-1,0), \\ f(x,y) + (f(0,1) - f(-1,0))u_{\{0,1\}}(x) & \text{if } f(0,1) > f(-1,0), \\ f(x,y) & \text{if } f(0,1) = f(-1,0). \end{cases}$$

We have $f' \in \langle\Gamma\rangle$, $f'(0,1) = f'(-1,0)$, and $f'(0,1) + f'(-1,0) < f'(-1,1)$. Now we can obtain, by scaling f', a binary function f'' in $\langle\Gamma\rangle$ with $f''(0,0) = 0$, $f''(0,1) = f''(-1,0) = 1$, and $f''(-1,1) > 2$. We can assume that $f = f''$ from the beginning.

We define the binary function g' as follows:

$$g'(x,y) = \begin{cases} g(x,y) + (g(0,-1) - g(1,0))u_{\{-1,0\}}(x) & \text{if } g(1,0) < g(0,-1), \\ g(x,y) + (g(1,0) - g(0,-1))u_{\{0,1\}}(y) & \text{if } g(1,0) > g(0,-1), \\ g(x,y) & \text{if } g(1,0) = g(0,-1). \end{cases}$$

We have $g' \in \langle\Gamma\rangle$, $g'(1,0) = g'(0,-1)$ and $g'(1,-1) < g'(1,0) + g'(0,-1)$. If $g'(1,-1) \leq 0$, let $C > -g(1,-1)$ and define the binary function g'' as

$$g''(x,y) = g'(x,y) + C(u_{\{-1,0\}}(x) + u_{\{0,1\}}(y))$$

for all $x, y \in D$. Otherwise we let $g'' = g'$. We have $g'' \in \langle\Gamma\rangle$, $g''(1,0) = g''(0,-1)$, and $0 < g''(1,-1) < g''(1,0)+g''(0,-1)$. Now we can obtain, by scaling g'', a binary

function g''' in $\langle\Gamma\rangle$ with $g'''(0,0) = 0$, $g'''(0,-1) = g'''(1,0) = 1$, and $0 < g'''(1,-1) < 2$. We can assume that $g = g'''$ from the beginning.

Note that $f(-1,1) - 2 > 0$ and $g(1,-1) > 0$. By scaling the function u_1, we can obtain a unary function u_1' in $\langle\Gamma\rangle$ with $g(1,-1) < u_1'(0) < \min\{2, f(-1,1) + g(1,-1) - 2\}$ and $u_1'(1) = 0$. By adding $u_{\{0,1\}}$ with a large enough coefficient, we can obtain a unary function v in $\langle\Gamma\rangle$ which still fulfils

$$g(1,-1) < v(0) < \min\{2, f(-1,1) + g(1,-1) - 2\} \text{ and } v(1) = 0,$$

but where the value $v(-1)$ is as large as we want. For our purposes

$$\begin{aligned} v(-1) \geq \max\{&2 - f(0,-1) - g(-1,0), 3 - f(-1,-1) - g(-1,0), \\ &3 - f(0,-1) - g(-1,-1), 4 - f(-1,-1) - g(-1,-1)\} \end{aligned}$$

is large enough. Now $\langle\Gamma\rangle$ also contains the binary function s defined by

$$s(x,z) := \min_{y\in D}\{f(x,y) + v(y) + g(y,z)\}$$

for all $x, z \in D$. We have

$$\begin{aligned} s(-1,0) &= \min\{f(-1,0) + v(0) + g(0,0), f(-1,1) + v(1) + g(1,0)\} \\ &= \min\{1 + v(0), 1 + f(-1,1)\} \\ &= 1 + v(0), \end{aligned}$$

$$\begin{aligned} s(0,-1) &= \min\{f(0,0) + v(0) + g(0,-1), f(0,1) + v(1) + g(1,-1)\} \\ &= \min\{1 + v(0), 1 + g(1,-1)\} \\ &= 1 + g(1,-1), \end{aligned}$$

$$\begin{aligned} s(0,0) &= \min\{f(0,0) + v(0) + g(0,0), f(0,1) + v(1) + g(1,0)\} \\ &= \min\{v(0), 2\} \\ &= v(0), \end{aligned}$$

$$\begin{aligned} s(-1,-1) &= \min\{f(-1,0) + v(0) + g(0,-1), f(-1,1) + v(1) + g(1,-1)\} \\ &= \min\{2 + v(0), f(-1,1) + g(1,-1)\} \\ &= 2 + v(0). \end{aligned}$$

Since $g(1,-1) < v(0)$, it is easy to see that s is not submodular in the orthant of $(-1,-1)$. □

As s is not submodular in the orthant of $(-1,-1)$, we can express MAX CUT with $u_{\{0,1\}}$. Therefore VCSP($\Gamma \cup s$) is **NP**-hard, and as the problems VCSP(Γ) and VCSP($\Gamma \cup s$) have the same complexity by Theorem 6, we know that VCSP(Γ) is **NP**-hard.

This is the first dichotomy result for VCSP where there are infinitely many necessary conditions for tractability as proven in the following proposition.

Proposition 3. *For every rational $\alpha \in (0,1]$, there is a core constraint language Γ_α on $\{-1,0,1\}$ satisfying all of the following conditions:*

1. *Γ_α is α-bisubmodular, but not α'-bisubmodular for any $\alpha' \neq \alpha$.*
2. *For any permutation of the names of $-1, 0, 1$ and any $\alpha' \in (0,1]$, Γ_α is not α'-bisubmodular under that renaming, with the only exception when $\alpha = \alpha' = 1$ and the renaming swaps 1 and -1.*
3. *Γ_α is not submodular on any chain on D,*

Proof. Let $\alpha = p/q$ where $0 < p \leq q$ are positive integers. Consider the following functions:

- unary $e = [1,0,1]$, $u_\alpha = [p+q, q, 0]$, and $v_\alpha = [0, p, p+q]$
- binary f_α such that $f_\alpha(1,-1) = f_\alpha(-1,1) = 1$, $f_\alpha(0,-1) = f_\alpha(-1,0) = 1+q$, $f_\alpha(-1,-1) = 1+p+q$, and $f(x,y) = 0$ on the remaining pairs (x,y).

Let $\Gamma_\alpha = \{e, u_\alpha, v_\alpha, f_\alpha\}$. It can be directly checked that all functions in Γ_α are α-bisubmodular (Proposition 2 can also be used for checking f_α) and that the unary functions in Γ_α make it a core.

Notice that f_α is not submodular when restricted to $\{-1,1\}$. Therefore Γ_α is not submodular on any chain on $\{-1,0,1\}$. It is easy to check that u_α is not α'-bisubmodular for any $\alpha' > \alpha$, and v_α is not α'-bisubmodular for any $\alpha' < \alpha$. We demonstrate the first case here.

For the unary function u_α to be α'-bisubmodular it must satisfy the inequality:

$$(1+\alpha')\cdot u_\alpha(0) \le u_\alpha(-1) + \alpha'\cdot u_\alpha(1)$$
$$(1+\alpha')\cdot q \le p + q + \alpha'\cdot 0$$
$$\alpha' q \le p$$
$$\alpha' \le p/q$$
$$\alpha' \le \alpha$$

Therefore u_α is not α'-bisubmodular for any $\alpha' > \alpha$. A similar argument proves v_α is not α'-bisubmodular for any $\alpha' < \alpha$.

It is also easy to check that the unary operations guarantee that any permutation of the names of elements $-1, 0, 1$ cannot make Γ_α α'-bisubmodular for any α', except swapping -1 and 1 when $\alpha = \alpha' = 1$. □

3.5 Multimorphisms are not enough

As mentioned before all known tractable core valued constraint languages were characterised by multimorphisms. It has been shown that the complexity of valued constraint languages can be characterised by fractional polymorphisms [15], but it was an open question, asked by Thapper and Živný [58], as to whether all VCSPs solvable by BLP could be characterised by just multimorphisms. The discovery of α-bisubmodularity now answers that question as the set of 1/2-bisubmodular functions cannot be defined by multimorphisms. Clearly not every unary function is 1/2-bisubmodular, so by showing that each multimorphism of a set of 1/2-bisubmodular functions captures unary functions that are not 1/2-bisubmodular we prove that tractable VCSPs cannot be characterised by multimorphisms. This result complements a previous observation that multimorphisms are also not enough for capturing expressibility (Appendix B of [19]).

It suffices to prove the following proposition:

Proposition 4. *There is a finite set Γ of 1/2-bisubmodular functions such that each multimorphism of Γ is a multimorphism of every unary function on $\{-1, 0, 1\}$.*

Proof. Let μ be a multimorphism of a function f. Then there are operations $F_1, \ldots, F_k \in O_D^{(k)}$ such that

$$\sum_{i=1}^{k} f(F_i(\mathbf{x}_1, \ldots, \mathbf{x}_k)) \leq \sum_{i=1}^{k} f(\mathbf{x}_i) \tag{3.5.7}$$

for all $\mathbf{x}_1, \ldots, \mathbf{x}_k \in D^n$, where n is the arity of f. We define the function $\mathbf{F} : D^k \to D^k$ by $\mathbf{F} = (F_1, \ldots, F_k)$ and identify μ with $\mathbf{F}$. For the proof of Proposition 4 we will use the following 1/2-bisubmodular functions:

- unary functions $u_{\{-1,0\}} = [0, 0, 1]$ and $u_{\{0,1\}} = [1, 0, 0]$,
- unary functions $v_1 = [-1, 0, 2]$ and $v_{-1} = [1, 0, -2]$,
- binary commutative function b such that $b(-1,-1) = 4$, $b(-1,0) = 2$, $b(-1,1) = -1$, $b(0,0) = 0$, $b(0,1) = -2$ and $b(1,1) = -4$.

It is easy to check that these functions are indeed 1/2-bisubmodular. Let $\Gamma = \{u_{\{-1,0\}}, u_{\{0,1\}}, v_1, v_{-1}, b\}$. If $\mathbf{F}$ preserves each tuple in D^k as a multiset, we say that $\mathbf{F}$ preserves multisets. It is easy to see that in this case the inequality (3.5.7) holds with equality for every unary function. So the following lemma finishes the proof of Proposition 4. □

Lemma 9. *If $\boldsymbol{F}$ is a multimorphism of Γ then $\boldsymbol{F}$ preserves multisets.*

Proof. We first show, using the unary functions from Γ, that if a multiset is not preserved by $\mathbf{F}$, then $\mathbf{F}$ modifies it in the following way: the number of 1s is reduced by some number $x \in \mathbb{N}$, and the number of -1s is reduced by 2x.

Let $(d_1, \ldots, d_k) \in D^k$, ℓ be the number of 1s in $(d_1, \ldots, d_k)$ and m the number of -1s. Let ℓ' be the number of 1s in $\mathbf{F}(d_1, \ldots, d_k)$ and m' the number of -1s. Applying (3.5.7) with $u_{\{-1,0\}}$ and $u_{\{0,1\}}$ gives $\ell' \leq \ell$ and $m' \leq m$, and applying (3.5.7) with v_1 and v_{-1} gives

$$2\ell' - m' = 2\ell - m.$$

Now we will show, using the binary function $b \in \Gamma$, that any multiset has to be preserved by $\mathbf{F}$. Let $\mathbf{d} = (d_1, \ldots, d_k) \in D^k$, ℓ the number of 1s in $\mathbf{d}$ and m the number of -1s. Let ℓ' be the number of 1s in $\mathbf{F}(\mathbf{d})$ and m' the number of -1s.

Without loss of generality let $d_1 = \cdots = d_\ell = 1$ and $F_1(\mathbf{d}) = \cdots = F_{\ell'}(\mathbf{d}) = 1$. We have $\ell' = \ell - x$ and $m' = m - 2x$ for some $x \in \mathbb{Z}_{\geq 0}$. If $x = 0$ we are done.

Let $[k]$ be the set of indices of $\mathbf{d}$, and $[\ell]$ the set of indices where $\mathbf{d}$ has a 1. As an example - if $\mathbf{d} = (1, 1, 1, -1, 0, 0)$ then $[k] = \{1, 2, 3, 4, 5, 6\}$ and $[\ell] = \{1, 2, 3\}$.

For every $p \in [\ell]$ let $\mathbf{x}^{(p)} \in (D^2)^k$ be defined as follows: $\mathbf{x}_p^{(p)} := (d_p, -1) = (1, -1)$ and $\mathbf{x}_i^{(p)} := (d_i, 1)$ for every $i \in [k] \backslash \{p\}$. Also for every $p \in [\ell]$ let $\mathbf{e}^{(p)} \in D^k$ be defined as follows: $\mathbf{e}_p^{(p)} := -1$ and $\mathbf{e}_i^{(p)} := 1$ for every $i \in [k] \backslash \{p\}$.

We have:

$$\begin{aligned}\sum_{i=1}^{k} b(\mathbf{x}_i^{(p)}) &= -1 - 4(\ell - 1) - m - 2(k - \ell - m) \\ &= -2k - 2\ell + m + 3.\end{aligned}$$

The multiset $(-1, 1, \ldots, 1)$ has to be preserved by $\mathbf{F}$, so for every $p \in [\ell]$ there is exactly one $j^{(p)} \in \{1, \ldots, k\}$ with $F_{j^{(p)}}(\mathbf{e}^{(p)}) = -1$. If $F_{j^{(p)}}(\mathbf{d}) = -1$ we have:

$$\begin{aligned}\sum_{i=1}^{k} b(F_i(\mathbf{x}_1^{(p)}, \ldots, \mathbf{x}_k^{(p)})) &= -4\ell' + 4 - (m' - 1) - 2(k - \ell' - m') \\ &= -2k - 2\ell' + m + 5 \\ &= -2k - 2(\ell - x) + (m - 2x) + 5 \\ &= -2k - 2\ell + m + 5\end{aligned}$$

which is a contradiction to (3.5.7).

If $F_{j^{(p)}}(\mathbf{d}) = 0$ we have:

$$\begin{aligned}\sum_{i=1}^{k} b(F_i(\mathbf{x}_1^{(p)}, \ldots, \mathbf{x}_k^{(p)})) &= -4\ell' - m' + 2 - 2(k - \ell' - m' - 1) \\ &= -2k - 2\ell' + m' + 4 \\ &= -2k - 2(\ell - x) + (m - 2x) + 4 \\ &= -2k - 2\ell + m + 4\end{aligned}$$

which also is a contradiction to (3.5.7).

So for every $p \in [\ell]$ we have $F_{j^{(p)}}(\mathbf{d}) = 1$. This yields $\{j^{(p)} | p \in [\ell]\} = [\ell']$, and if $\ell' < \ell$, there must be two different indices $p, q \in [\ell]$ such that $j^{(p)} = j^{(q)}$.

Let $\mathbf{x} \in (D^2)^k$ be defined as follows: $\mathbf{x}_p := (1, -1)$, $\mathbf{x}_q := (-1, 1)$ and $\mathbf{x}_i := (1, 1)$ for every $i \in [k] \backslash \{p, q\}$. Then we have:

$$\sum_{i=1}^{k} b(\mathbf{x}_i) = -1 - 1 - 4(k-2) = -4k + 6 \text{ and}$$

$$\sum_{i=1}^{k} b(F_i(\mathbf{x}_1^{(p)}, \ldots, \mathbf{x}_k^{(p)})) = 4 - 4(k-1) = -4k + 8$$

which is a contradiction to (3.5.7).

So we cannot have $\ell' < \ell$, and thus we have $\ell' = \ell$ and $m' = m$. □

Chapter 4

Reducing VCSP to Min-Cost-Hom

4.1 Introduction

It is known that any CSP with a fixed constraint language is polynomial time equivalent to one where the constraint language consists of a single binary relation (i.e. a digraph) [11, 12, 23]. A recent proof of this by Bulín *et al.* [11, 12] gives a reduction that preserves a number of algebraic properties of the constraint language that are known to charaterise the complexity of the corresponding CSP. In this chapter we adapt that proof to the VCSP framework and show that each VCSP, with a fixed valued constraint language of finite size, is polynomial-time equivalent to one where the constraint language consists of a single $\{0, \infty\}$-valued binary function (i.e. a digraph), and a single finite-valued unary function. Problems of this type have already been studied as the Minimum Cost Homomorphism Problem (referred to as MinCostHom), which makes this result somewhat surprising as it was believed that MinCostHom was essentially a more restricted optimisation problem than VCSP. We also prove that this reduction preserves some important algebraic properties of the valued constraint language.

It should be noted that a complementary result has been obtained in [18]. There it is shown (with a much simpler construction) that any VCSP can be reduced to a VCSP consisting of a single unary weighted relation and binary relations (but not a single binary relation as we achieve).

4.2 Homomorphisms

Firstly we introduce homomorphisms and graph homomorphisms, and then show how to cast CSP and VCSP as homomorphism problems.

Definition 28. *Let τ be a relational signature, that is a set of relational symbols R each with an associated arity $ar(R)$. A relational τ-structure $\mathcal{A}$ consists of a finite domain D together with a relation $R^{\mathcal{A}}$ on D of arity $ar(R)$ for each $R \in \tau$. Let $\mathcal{X}$ and $\mathcal{A}$ be τ-structures with domains X and D respectively. A homomorphism from $\mathcal{X}$ to $\mathcal{A}$ is a function $h : X \to D$, such that for each $R \in \tau$ and each tuple $\boldsymbol{x} \in R^{\mathcal{X}}$, $h(\boldsymbol{x}) \in R^{\mathcal{A}}$.*

Definition 29. *Let $\mathcal{A}$ be a finite relational τ-structure. Then $CSP(\mathcal{A})$ is the following decision problem: Given a finite τ-structure $\mathcal{X}$ is there a homomorphism from $\mathcal{X}$ to $\mathcal{A}$?*

For the purposes of this chapter we define a digraph as a structure $\mathbb{G} = (V^{\mathbb{G}}, E^{\mathbb{G}})$ with vertices $v \in V^{\mathbb{G}}$ and directed edges $e \in E^{\mathbb{G}}$.

Example 19. *The digraph homomorphism problem asks whether an input digraph $\mathbb{G}$ admits a homomorphism to a fixed digraph $\mathbb{H}$. I.e. Is there a mapping $h : V^{\mathbb{G}} \to V^{\mathbb{H}}$ such that if $(u, v) \in E^{\mathbb{G}}$ then $(h(u), h(v)) \in E^{\mathbb{H}}$. This is also known as the $\mathbb{H}$-colouring problem [26], and if $\mathbb{H}$ is a complete graph on k vertices then it is the k-colouring problem.*

It is known that restricting CSP from general structures to digraphs does not reduce the difficulty of classifying the complexity.

Theorem 19 ([11, 12, 23])**.** *For every structure $\mathcal{A}$, there is a digraph $\mathbb{H}$ such that* CSP($\mathcal{A}$) *and* CSP($\mathbb{H}$) *are polynomial-time equivalent.*

Now we introduce the Minimum Cost Homomorphism Problem.

Definition 30. *In the* MinCostHom($\mathcal{A}$) *problem, one is given an input τ-structure $\mathcal{X}$ and, in addition, for each $x \in X$, a unary cost function $u_x : D \to \mathbb{Q}_+$ specifying the cost of mapping x to each individual element in D. The goal is to decide whether there is a homomorphism h from $\mathcal{X}$ to $\mathcal{A}$ and if so find one of minimal total cost $\sum_{x \in X} u_x(h(x))$. For a set Δ of unary cost functions, let* MinCostHom($\mathcal{A}, \Delta$)

denote the subproblem of MinCostHom($\mathcal{A}$) *where all unary functions in instances are of the form* $w \cdot u$ *where* $w \in \mathbb{Q}_+$ *and* $u \in \Delta$. *If* $\Delta = \{u\}$, *we write simply* MinCostHom($\mathcal{A}, u$).

The problem MinCostHom($\mathcal{A}$) was studied in a series of papers, and complete complexity classifications were given in [25] for undirected graphs, in [28] for digraphs, and in [55] for general structures. Partial complexity classifications for the problem MinCostHom($\mathcal{A}, \Delta$) were obtained in [56, 62, 63]. One can see that MinCostHom is an intermediate problem between CSP and VCSP, as there is an optimisation aspect, but it is limited in the sense that it is controlled by separate unary functions, without explicit interactions of variables.

We will now define VCSP as a homomorphism problem, following [57] (see also [14]). This will allow us to easily reuse many results from [11, 12].

Definition 31. *A* weighted relation ρ *of arity* k *on a set is a function from some* k*-ary relation* R *on this set to* $\mathbb{Q}_+$. *A* weighted τ-structure $w\mathcal{A}$ *is* τ*-structure such that each relation* $\rho^{w\mathcal{A}}$ *in* $w\mathcal{A}$ *is weighted, i.e.* $\rho^{w\mathcal{A}} : R^{\mathcal{A}} \to \mathbb{Q}_+$. *By ignoring the weight functions, one can turn a weighted* τ*-structure* $w\mathcal{A}$ *into an ordinary, unweighted,* τ*-structure* $\mathcal{A}$.

An instance of VCSP($w\mathcal{A}$) *is given by a weighted* τ*-structure* $w\mathcal{X}$. *A feasible solution to this instance is a homomorphism* h *from* $\mathcal{X}$ *to* $\mathcal{A}$, *and its cost is given by*

$$cost(h) = \sum_{R \in \tau, \mathbf{x} \in R^{\mathcal{X}}} \rho^{w\mathcal{X}}(\boldsymbol{x}) \cdot \rho^{w\mathcal{A}}(h(\boldsymbol{x})).$$

The goal is to decide if such a homomorphism exists, and if so find one with minimal cost.

Definition 32. *For an* m*-ary cost function* $\phi : D^m \to \overline{\mathbb{Q}}_+$, *we define the feasibility relation,* $Feas(\phi)$, *of* ϕ *as follows:* $(x_1, x_2, ..., x_m) \in Feas(\phi) \Leftrightarrow \phi(x_1, x_2, ..., x_m) < \infty$.

There is an obvious correspondence between valued constraint languages and weighted structures: any constraint language Γ can be turned into a weighted structure $w\mathcal{A}$ as follows. Turn each function $\phi \in \Gamma$ into a weighted relation $\rho : Feas(\phi) \to \mathbb{Q}_+$, simply by ignoring the tuples with infinite cost, then introduce a signature τ con-

taining a symbol R_ϕ of arity k for each function $\phi \in \Gamma$ of arity k. Clearly, one obtains a weighted τ-structure. One can also reverse this procedure to convert a weighted structure into a valued constraint language.

The correspondence between VCSP($w\mathcal{A}$) and VCSP(Γ) can be seen as follows. If $w\mathcal{X}$ is an instance of VCSP($w\mathcal{A}$), then one can view the domain of $w\mathcal{X}$ as the set of variables in an instance $\mathcal{I}$ of VCSP(Γ), and each tuple $\mathbf{x} \in R^{\mathcal{X}}$ gives rise to a valued constraint $\phi(\mathbf{x})$ with weight $\rho^{w\mathcal{X}}(\mathbf{x})$ where $\phi \in \Phi_D$ is the function obtained by extending $\rho^{w\mathcal{A}}$ with infinite values. Then the homomorphisms from $\mathcal{X}$ to $\mathcal{A}$ are precisely the solutions to $\mathcal{I}$ of finite cost, and the correspondence preserves the costs. Thus, VCSP($w\mathcal{A}$) and VCSP(Γ) are effectively the same problem.

Note that if all functions in Γ are $\{0, \infty\}$-valued, i.e. VCSP(Γ) is in fact a CSP, then the weighted structure $w\mathcal{A}$ obtained from Γ as described above will be 0-weighted, i.e. effectively unweighted, and VCSP($w\mathcal{A}$) is the same problem as CSP($\mathcal{A}$). It is also clear that if all functions in Γ are $\{0, \infty\}$-valued or unary finite-valued, then VCSP(Γ) is MinCostHom($\mathcal{A}, \Delta$) for the obvious choice of $\mathcal{A}$ and Δ.

4.3 Proving Poly-Time Equivalence

Firstly we introduce some simple definitions relating to digraphs that we will need in the construction of the main theorem of this paper. As already seen we define a digraph as a structure $\mathbb{G} = (V^{\mathbb{G}}, E^{\mathbb{G}})$ with vertices $v \in V^{\mathbb{G}}$ and directed edges $e \in E^{\mathbb{G}}$. We write the directed edge $(a, b) \in E^{\mathbb{G}}$ as $a \to b$ for simplicity.

Definition 33. *A digraph $\mathbb{P}$ is an oriented path if it consists of a sequence of vertices $v_0, v_1, ...v_k$ such that precisely one of $(v_{i-1}, v_i), (v_i, v_{i-1})$ is an edge, for each $i = 1, ..., k$. We denote the initial vertex v_0 by $\iota\mathbb{P}$.*

Definition 34. *Given a digraph $\mathbb{G}$, any two vertices a and b in $\mathbb{G}$ are connected if there is an oriented path between them.*

A digraph $\mathbb{G}$ is said to be connected if it has only a single connected component. The length of an oriented cycle is defined as being the absolute value of the difference between edges oriented in one direction and edges oriented in the opposite direction. A connected digraph is then balanced if all of its cycles have zero length [23]. The

vertices of a balanced digraph can be organised into levels, that is for every directed edge $(a, b) \in E^{\mathbb{G}}$, $lvl(b) = lvl(a) + 1$. The minimum level of $\mathbb{G}$ is 0, and the top level is the height of $\mathbb{G}$. We will often refer to vertices on level 0 as base vertices.

Here we present the main theorem of this chapter, and the remainder of this section details the proof.

Theorem 20. *Let $w\mathcal{A}$ be a weighted structure that is a rigid core. There is a balanced digraph $\mathbb{D}$ which is a rigid core and a finite-valued function u such that problems* VCSP($w\mathcal{A}$) *and* MinCostHom($\mathbb{D}, u$) *are polynomial-time equivalent.*

Proof. We can assume without loss of generality that $w\mathcal{A}$ contains only one weighted relation, say of arity n. If it contains more, say, $\rho_1, \ldots, \rho_t$ where, for $1 \leq j \leq t$, the arity of ρ_j is n_j, than the standard trick is to take the direct product ρ of these relations. Specifically, if $\rho_j : R_j \to \mathbb{Q}_+$ then $\rho : R_1 \times \ldots \times R_t \to \mathbb{Q}_+$ is such that $\rho(\mathbf{a}_1, \ldots, \mathbf{a}_t) = \rho_1(\mathbf{a}_1) + \ldots + \rho_t(\mathbf{a}_t)$. It is well known and not hard to see that replacing $\rho_1, \ldots, \rho_t$ with ρ does not change the complexity of VCSP($w\mathcal{A}$).

The reduction from VCSP($w\mathcal{A}$) to MinCostHom($\mathbb{D}, u$) follows from Lemmas 11 and 12, and the reduction in the opposite direction is shown in Lemma 13. □

The proofs of these results rely heavily on the proofs developed for ordinary CSPs by Bulín et al. [11,12], with $\mathbb{D}$ being exactly the same digraph constructed for CSPs, and u constructed using similar ideas. In particular if the constraint language $\mathcal{A}$ is crisp, then $u \equiv 0$, and our VCSP is essentially a CSP, and our proof follows the proof of [11, 12].

4.3.1 Constructing $(\mathbb{D}, u)$

We will take a rigid core valued constraint language of finite size, Γ, and construct a balanced digraph $\mathbb{D}$ and a unary function u, with $\mathcal{D}(\Gamma) = (\mathbb{D}, u)$, such that $VCSP(\Gamma)$ and $VCSP(\mathcal{D}(\Gamma))$ are polynomial-time equivalent.

We use the same construction as [11,12] for zigzags and single edges, that is a zigzag is the oriented path $\bullet \to \bullet \leftarrow \bullet \to \bullet$ and a single edge is the path $\bullet \to \bullet$.

Recall that n is the arity of the single weighted relation ρ in $w\mathcal{A}$. For $S \subseteq \{1, 2, \ldots, n\}$ define $\mathbb{Q}_{S,l}$ to be a single edge if $l \in S$, and a zigzag if $l \in \{1, 2, \ldots, n\} \setminus S$.

As in [11, 12] we define the oriented path $\mathbb{Q}_S$ (of height n+2) by

$$\mathbb{Q}_S = \bullet \to \bullet \dot{+} \mathbb{Q}_{S,1} \dot{+} \mathbb{Q}_{S,2} \dot{+} \cdots \dot{+} \mathbb{Q}_{S,n} \dot{+} \bullet \to \bullet$$

where $\dot{+}$ denotes the concatenation of paths.

Now take R, i.e. the underlying relation of ρ, and the domain D of $w\mathcal{A}$, and as a starting point consider the digraph with vertices $D \cup R$ and edges $D \times R$. Then replace every edge $(d, \mathbf{a}) \in D \times R$ with the path $\mathbb{Q}_{\{i:d=a_i\}}$, and this is the digraph $\mathbb{D}$. To complete the construction, we define a unary function u. Let $V^{\mathbb{D}}$ be the vertices of $\mathbb{D}$, and note that $R \subseteq V^{\mathbb{D}}$. Let u to be a unary function from $V^{\mathbb{D}}$ to $\mathbb{Q}_+$ such that

$$u(v) = \begin{cases} \rho(v) & \text{if } v \in R \\ 0 & \text{otherwise.} \end{cases}$$

The digraph $\mathbb{D}$ is identical to the digraph defined in [11,12] and due to our definition of R the number of vertices in $\mathbb{D}$ remains as $(3n+1)|R||D| + (1-2n)|R| + |D|$ and the number of edges as $(3n+2)|R||D| - 2n|R|$ as proven in [11, 12]. Also as noted in [11, 12] this construction can be performed in polynomial time.

Example 20. *Consider the weighted structure $w\mathcal{A}$ over the domain $D = \{0, 1\}$ with the single weighted relation*

$$\rho(x, y) = \begin{cases} 2 & \text{if } (x, y) = (0, 1) \\ 1 & \text{if } (x, y) = (1, 0) \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The digraph $\mathbb{D}$ constructed from ρ is shown in Figure 4.1. The unary function built from ρ is

$$u(v) = \begin{cases} 2 & \text{if } v = (0, 1) \\ 1 & \text{if } v = (1, 0) \\ 0 & \text{otherwise} \end{cases}$$

for every vertex $v \in V^{\mathbb{D}}$.

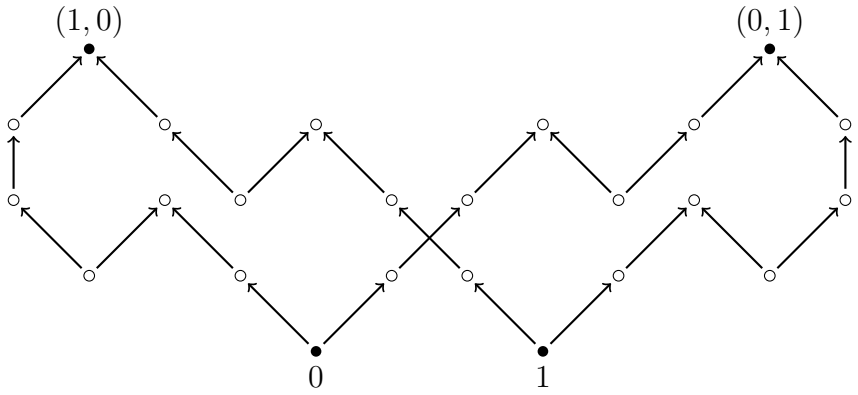


Figure 4.1: The digraph $\mathbb{D}$ built from the weighted structure $w\mathcal{A}$.

Lemma 4.1 of [12] states that the unary polymorphisms of relation R and of digraph $\mathbb{D}$ are in one-to-one correspondence. Hence, we immediately get the following.

Lemma 10. *$w\mathcal{A}$ is a rigid core if and only $\mathbb{D}$ is a rigid core.*

4.3.2 Reduction from VCSP($w\mathcal{A}$) to MinCostHom($\mathbb{D}, u$)

Given an instance of VCSP($w\mathcal{A}_0$), where $w\mathcal{A}_0$ is some weighted structure, the variables in any constraint in that instance are explicitly constrained. It is also possible that any subset of variables in that instance are implicitly constrained due to combinations of constraints. The weighted relation describing this implicit constraint may not belong to $w\mathcal{A}_0$, but is said to be expressible by $w\mathcal{A}_0$.

Definition 35. *Given an instance $w\mathcal{X}$ of* VCSP($w\mathcal{A}_0$) *with domain $X = \{x_1, ..., x_n\}$, and a tuple of distinct elements $W = (x_1, \ldots, x_t)$, where $t \leq n$, define weighted relation $\rho^W_{w\mathcal{X}}$ as follows:*

$$\rho^W_{w\mathcal{X}}(a_1, \ldots, a_t) = \min_{h:\mathcal{X}\to\mathcal{A},\ h(x_1,\ldots,x_t)=(a_1,\ldots,a_t)} cost(h). \tag{4.3.1}$$

Note that if, for some $(a_1, \ldots, a_t)$, there is no homomorphism $h : \mathcal{X} \to \mathcal{A}$ such that $h(x_1, \ldots, x_t) = (a_1, \ldots, a_t)$ then $\rho^W_{w\mathcal{X}}(a_1, \ldots, a_t)$ is undefined. A weighted relation ρ is expressible *by $w\mathcal{A}_0$ if there is an instance $w\mathcal{X}$ of* VCSP($w\mathcal{A}_0$) *with domain X, and $W \subseteq X$, such that $\rho = \rho^W_{w\mathcal{X}}$.*

Lemma 11 ([17])**.** *Let $w\mathcal{A}$ and $w\mathcal{A}_0$ be weighted structures. If every weighted relation ρ in $w\mathcal{A}$ is expressible by $w\mathcal{A}_0$, then* VCSP($w\mathcal{A}$) *is polynomial-time reducible to* VCSP($w\mathcal{A}_0$).

Let $w\mathcal{A}_0$ be the weighted relational structure containing only weighted relations $E^{\mathbb{D}}$ and u, where $E^{\mathbb{D}}$ simply assigns 0 to every edge of $\mathbb{D}$.

Lemma 12. *$w\mathcal{A}$ is expressible by $w\mathcal{A}_0$.*

Proof. All we need to see is that the domain D and weighted relation ρ of $w\mathcal{A}$ can be expressed by $w\mathcal{A}_0$, in the sense of Equation (4.3.1). Consider the weighted structure $w\mathcal{X}$ that contains the binary 0-weighted relation corresponding to the oriented path $\mathbb{Q}_\emptyset$ (defined above) and the empty unary relation. Let x be the initial vertex of $\mathbb{Q}_\emptyset$ and let $W = \{x\}$. Then D, as a unary 0-weighted relation, can be defined as $\rho^W_{w\mathcal{X}}$. Indeed, the homomorphisms from $\mathcal{X}$ to $\mathcal{A}$ are simply homomorphisms from $\mathbb{Q}_\emptyset$ to $\mathbb{D}$, and it is clear that the images of x under such homomorphisms are exactly the base elements of $\mathbb{D}$, i.e., precisely the elements of D.

To express ρ, consider the weighted structure $w\mathcal{X}$ whose binary 0-weighted relation corresponds to the digraph obtained by identifying the terminal vertices of n directed paths $\mathbb{Q}_{\{1\}}, \ldots, \mathbb{Q}_{\{n\}}$, and whose unary relation contains a single element y, the common vertex of the paths $\mathbb{Q}_{\{1\}}, \ldots, \mathbb{Q}_{\{n\}}$, with weight 1. Let $W = \{x_1, \ldots, x_n\}$ where x_i is the initial element of $\mathbb{Q}_{\{i\}}$. Again, it is not hard to see from the definitions of $\mathbb{D}$ and u that this $\rho^W_{w\mathcal{X}}$ is precisely the required weighted relation ρ. □

4.3.3 Reduction from MinCostHom$(\mathbb{D}, u)$ to VCSP$(w\mathcal{A})$

It is well known that fixed template CSPs can be modelled as homomorphism problems [23], and we will use this in the feasiblity aspect of our main theorem as it is in [11, 12]. In order to complete the proof of the following lemma, and therefore the proof of the main theorem of this chapter, we first require a technical detail.

Let $w\mathcal{A}'$ be the structure obtained from $w\mathcal{A}$ by adding the weighted relation ρ_0, which is obtained from ρ by making every weight 0. It is shown in [14] that VCSP$(w\mathcal{A})$ and VCSP$(w\mathcal{A}')$ are polynomial-time equivalent.

Lemma 13. MinCostHom$(\mathbb{D}, u)$ *reduces to* VCSP$(w\mathcal{A}')$ *in polynomial time.*

Proof. Let $(\mathbb{G}, W)$ be an instance of MinCostHom$(\mathbb{D}, u)$, where $\mathbb{G}$ is a digraph and W is a unary weighted relation over $V^{\mathbb{G}}$ responsible for the optimisation aspect of

the problem. Formally (to match Definition 30), we can assume that the vertices of $\mathbb{G}$ outside W have function $0 \cdot u$ applied to them.

Our reduction is a modification of a reduction in [11, 12]. Specifically, Stages 1 and 3a of the reduction are exactly the same, but Stages 2 and 3b are modified.

Stage 1: Verify $\mathbb{G}$ is balanced and test height.

This initial check is to verify that that $\mathbb{G}$ is balanced and that $\mathbb{G}$ has height not greater than m, the height of $\mathbb{D}$. If either of these conditions fail there can be no homomorphism from $\mathbb{G}$ to $\mathbb{D}$ and we return the fixed NO instance of VCSP($w\mathcal{A}'$). It is easy to see that these checks can be performed in polynomial time.

It is clear that if $\mathbb{G}$ is balanced and has the right height then, under any homomorphism from $\mathbb{G}$ to $\mathbb{D}$, only vertices of top level in $\mathbb{G}$ can be mapped to the vertices of top level in $\mathbb{D}$. Therefore, any vertex in W that is not from the top level cannot possibly affect the cost of a homomorphism, and so can be safely removed from W. From now we will assume that W, if non-empty, contains only top level vertices of $\mathbb{G}$.

Stage 2: Elimination of short components.

If $\mathbb{G}$ contains a connected component $\mathbb{H}$ of height less than m (a short component) then we can find an optimal solution for this component directly in polynomial time as in [11,12], the proof of which is in Section 4.5. We repeat this procedure for every short component and if there is any component $\mathbb{H}$ for which there is no solution then we return the fixed NO instance of VCSP($w\mathcal{A}'$). Provided there is an optimal solution to all short components we can ignore these components for the remainder of the reduction. If $\mathbb{G}$ itself is of height less than m then finding an optimal solution to every connected component completes the reduction and we output some fixed YES instance of VCSP($w\mathcal{A}'$).

Stage 3a: Construction of $\mathbb{B}'$.

This stage is identical to Stage 3A of [11,12] and is included here only for complete-

ness. In this stage we build the "object" $\mathbb{B}'$, which consists of a list of tuples, some of them with subscripts, and a list of equalities. These tuples contain sets of vertices of $\mathbb{D}$ and new vertices created during the algorithm. We will also need the following technical detail from [11, 12].

Say that a digraph $\mathbb{H}$ is *satisfiable in a digraph* $\mathbb{H}'$ if there is a homomorphism from $\mathbb{H}$ to $\mathbb{H}'$. It is shown in [11, 12] that, for any connected balanced digraph $\mathbb{H}$, there is a smallest set $S \subseteq \{1, ..., n\}$ such that $\mathbb{H}$ is satisfiable in $\mathbb{Q}_S$, and this set can be efficiently found. We denote this set by $\Gamma(\mathbb{H})$[1].

Any new vertices created in the algorithm to construct $\mathbb{B}'$ should be unique, and we remind the reader that the height of $\mathbb{G}$ is m and the arity of the weighted relation ρ in $w\mathcal{A}$ is n. Say that the *internal components* of $\mathbb{G}$ are the connected components of the induced subgraph of $\mathbb{G}$ obtained by removing all vertices of height 0 and m. The algorithm is as follows:

Firstly we create tuples from the top level vertices of $\mathbb{G}$. For each such vertex e, there will be one tuple with subscript e.
For every vertex e in $\mathbb{G}$ of height m and for $i = 1$ to n, do the following:

1. Identify all internal components C of $\mathbb{G}$ such that $i \in \Gamma(C)$ and C has an edge to e.

2. If there are no such components, add a new vertex x to the ith component (vertex set) of the output tuple for e.

3. Else, for each such internal component C, do

   (a) If C has edges to vertices of level 0 in $\mathbb{G}$, say $b_1, \ldots, b_j$, then add them to the ith vertex set of the output tuple for e.

   (b) Else, add a new vertex x to the ith vertex set of the output tuple for e.

This completes the first part of the algorithm for constructing $\mathbb{B}'$ and now we create tuples (that will not have subscripts) using the base vertices of $\mathbb{G}$.

[1]Notation $\Gamma(\mathbb{H})$ is from [11, 12], not related to valued constraint languages Γ.

For every level 0 vertex b and for $i = 1$ to n do the following:

1. Identify all internal components C of $\mathbb{G}$ such that C has an edge to b, but no edge to any vertex of height m.
2. For each such internal component, if they exist, do
   (a) if $i \in \Gamma(C)$ then add b to the ith vertex set of the output tuple for b.
   (b) if $i \notin \Gamma(C)$ then add a new vertex x in the ith vertex set of the output tuple for b.

The algorithm is completed by creating a list of equalities, L, showing (some of the) vertices that will have to be mapped to the same vertex by any homomorphism from $\mathbb{G}$ to $\mathbb{D}$. The rules for creating L are as follows.

i. If there is an internal component with edges to distinct vertices e and f of height m in $\mathbb{G}$, then we write $e = f$.

ii. If there is an internal component with edges to distinct vertices b and c of height 0 in $\mathbb{G}$, then we write $b = c$.

Stage 3b: Construction of $w\mathcal{X}'$.

In a modification to the construction of structure $\mathbb{B}$ in the proof of [11, 12], we construct weighted structure $w\mathcal{X}'$ containing two weighted relations ρ'_w and ρ'_0. The relation ρ'_0 will be the 0-weighted relation identical to the (unique) relation in structure $\mathbb{B}$ from [11,12], whilst ρ'_w is built using W, and controls the optimisation aspect of the reduction.

We start with building the equality graph. Its vertices are the base vertices of $\mathbb{G}$ and the new vertices created in stage 3a of the algorithm. The rules to create edges in the equality graph are taken directly from [11, 12].

The three rules for creating an edge in the equality graph are:

i. Add an edge from vertex a to vertex b if a and b lie in the same vertex set in $\mathbb{B}'$.

ii. Add an edge if $a = b$ is an equality in L.

iii. Add an edge if a and b appear in the ith vertex set of two tuples with subscripts e and f where $e = f$ is an equality in L.

Each element in the domain X' of $w\mathcal{X}'$ will be the set of vertices of a connected component of the equality graph. It follows from the construction that all vertices from the same connected component have to be mapped to the same element under any homomorphism from $\mathbb{G}$ to $\mathbb{D}$.

To obtain the tuples of ρ'_0 we replace the vertex set in every coordinate of every tuple of $\mathbb{B}'$ with the set of vertices of the connected component containing that vertex set in the equality graph, and remove all of the book-keeping subscripts. Each tuple in the weighted relation ρ'_0 is assigned weight 0.

The weighted relation ρ'_w will be defined on (some of the) tuples on which ρ'_0 is defined. Assume that $\rho'_0(A_1, \ldots, A_n)$ is defined. Then $\rho'_w(A_1, \ldots, A_n)$ is defined if and only if there is $e \in W$ such that the tuple $(B_1, \ldots, B_n) \in \mathbb{B}'$ with subscript e satisfies $B_i \subseteq A_i$ for all $1 \leq i \leq n$. In this case, the weight of $\rho'_w(A_1, \ldots, A_n)$ is the sum of weights $w(e)$ over all such e.

It is stated in [11,12] that homomorphisms from $\mathbb{G}$ to $\mathbb{D}$ are in one-to-one correspondence with homomorphisms from $\mathcal{X}'$ to $\mathcal{A}'$ which we elaborate here. Let $h : \mathbb{G} \to \mathbb{D}$, we want to show there is a corresponding homomorphism $h' : \mathcal{X}' \to \mathcal{A}'$. By construction $\mathcal{X}'$ consists of tuples whose elements are vertex sets, where each vertex set can contain bottom vertices of $\mathbb{G}$ and any new vertices created in stage 3a of the algorithm above. Consider all vertex sets of all tuples of $\mathcal{X}'$. No two of these vertex sets have any vertices in common, unless the vertex sets are identical, as they would have been identified and grouped together by the equality graph in stage 3b. If a vertex set consists of only new vertices then that vertex set appears only once in one tuple of $\mathcal{X}'$ (note that we ignore repeated tuples in $\mathcal{X}'$). It is noted in [11, 12] that any tuple in $\mathcal{X}'$ that consists of only new vertices can be ignored. We note that such a tuple can be mapped anywhere by a homomorphism h, and therefore the homomorphisms h and h' are only in one-to-one correspondence if we ignore such tuples.

The homomorphism h maps bottom vertices of $\mathbb{G}$ to bottom vertices of $\mathbb{D}$, and by definition the bottom vertices of $\mathbb{D}$ are the elements of $\mathcal{A}'$. If we consider a tuple in $\mathcal{X}'$ where each of its elements is a vertex set containing at least one bottom vertex of $\mathbb{G}$, then we know which element this maps to in $\mathcal{A}'$ as we know where h mapped that vertex to in $\mathbb{D}$. Therefore the homomorphism h fully determines how h' acts on vertex sets containing at least one bottom vertex of $\mathbb{G}$.

Now consider the case of a tuple in $\mathcal{X}'$ which has at least one element which is a vertex set containing only new vertices. We will assume this element is in the ith position of the tuple. These new vertices were created in stage 3a, either in step 2 or step 3(b) by a specific top vertex of $\mathbb{G}$, or in step 2(b) by a specific base vertex of $\mathbb{G}$, and will therefore fall into exactly one of the following two cases.

First consider the case where a new vertex was created from a specific top vertex e of $\mathbb{G}$. The new vertex was created as there is either no internal component C with an edge to e such that $i \in \Gamma(C)$ (Step 2), or all the internal components C with edges to e and $i \in \Gamma(C)$ have no edges to any base vertices (Step 3(b)). Given a homomorphism $h : \mathbb{G} \to \mathbb{D}$ we can identify which top vertex t in $\mathbb{D}$ the vertex e is mapped to. Therefore we can identify the unique internal component C_D in $\mathbb{D}$ with $i \in \Gamma(C_D)$ that has t as its top vertex, and the base vertex of $\mathbb{D}$ to which it is connected. Thus we have identified the base vertex in $\mathbb{D}$ (i.e. the element in $\mathcal{A}'$) which is mapped to by our vertex set containing only new vertices, and therefore determined how h' acts on this vertex set.

Now consider the case where the new vertex was created by a base vertex b of $\mathbb{G}$ (Step 2(b)). Note that we only create a new vertex if there is at least one internal component of $\mathbb{G}$ with an edge to b that has no edge to any top vertex. Furthermore we note that if there is such an internal component and it has $\Gamma(C) = \emptyset$ the tuple we would obtain in $\mathcal{X}'$ would consist of only new vertices and should be ignored. Let C be an internal component with an edge to b and $\Gamma(C) \neq \emptyset$. Under a homomorphism $h : \mathbb{G} \to \mathbb{D}$ we can identify where the base vertex b is mapped to in $\mathbb{D}$, and also where the vertices of C are mapped to. Thus we can identify the specific top vertex t of $\mathbb{D}$ that is at the end of the oriented path that C mapped to. Now we can use the

same method as before to identify the base vertex of $\mathbb{D}$ that our new vertex must map to. That is we identify the unique internal component C_D in $\mathbb{D}$ with $i \in \Gamma(C_D)$ that has t as its top vertex, and then identify the base vertex of $\mathbb{D}$ to which it is connected. The homomorphism h maps our new vertex to that base vertex of $\mathbb{D}$, and therefore we have determined how h' acts on this vertex set.

The above cases cover all possible vertex sets that can appear in tuples of $\mathcal{X}'$, and hence we have fully defined h' given h. It remains to argue that h' is a homomorphism from $\mathcal{X}'$ to $\mathcal{A}'$, i.e. for each tuple $\mathbf{x} \in \mathcal{X}'$, $h'(\mathbf{x}) \in \mathcal{A}'$. By construction the tuple $\mathbf{x} \in \mathcal{X}'$ corresponds to a top vertex e of $\mathbb{G}$, which is mapped, by the homomorphism h, to a top vertex $t = h(e)$ of $\mathbb{D}$. The vertex t has a corresponding tuple in $\mathcal{A}'$, and this is the tuple $h'(\mathbf{x}) \in \mathcal{A}'$. This can be seen by considering an element of the tuple $\mathbf{x}$, each of which is a vertex set - a set of base vertices of $\mathbb{G}$. These vertices all map to the same base vertex in $\mathbb{D}$, which then corresponds with the appropriate element in the tuple $h'(\mathbf{x}) \in \mathcal{A}'$. Therefore for every tuple $\mathbf{x} \in \mathcal{X}'$ we have $h'(\mathbf{x}) \in \mathcal{A}'$, and hence $h' : \mathcal{X}' \to \mathcal{A}'$ is a homomorphism.

The argument in the reverse direction is much simpler. Given an arbitrary homomorphism $h' : \mathcal{X}' \to \mathcal{A}'$, we can easily recover the homomorphism $h : \mathbb{G} \to \mathbb{D}$. By construction the tuples in $\mathcal{X}'$ correspond to top vertices of $\mathbb{G}$, while their elements are sets of bottom vertices (and new vertices). Likewise the tuples of $\mathcal{A}'$ correspond to top vertices of $\mathbb{D}$, and their elements are bottom vertices. Therefore given the homomorphism h' we know which top vertices of $\mathbb{G}$ map to which top vertices of $\mathbb{D}$, and likewise for the bottom vertices. Each internal component of $\mathbb{G}$ is then forced into mapping onto the only satisfiable path available in $\mathbb{D}$, and thus we have recovered the homomorphism $h : \mathbb{G} \to \mathbb{D}$.

Given that we can fully determine h' given h, and vice versa, we have successfully proven that h and h' are in one-to-one correspondence.

It also follows from our construction of ρ'_w that the corresponding homomorphisms will have the same cost as we now show.

Consider a homomorphism $h : \mathbb{G} \to \mathbb{D}$, it will have the following cost:

$$cost(h) = \sum_{e \in W} w_e \cdot u(h(e))$$

The corresponding homomorphism $h' : \mathcal{X}' \to \mathcal{A}'$ will have the following cost:

$$cost(h') = \sum_{\substack{(A_1,\ldots,A_n) \text{ s.t.} \\ \rho'_w \text{ is defined}}} \rho'_w \cdot \rho(h'(A_1), \ldots, h'(A_n))$$

Let e be a top vertex in $\mathbb{G}$, then $h(e)$ is a top level vertex in $\mathbb{D}$. Also let $(A_1, \ldots, A_n)$ be a tuple in $\mathcal{X}'$, then $(h'(A_1), \ldots, h'(A_n))$ is a tuple in $\mathcal{A}'$. By construction each of the top vertices of $\mathbb{G}$ corresponds with a tuple in $\mathcal{X}'$, and each of the top level vertices in $\mathbb{D}$ corresponds with a tuple in $\mathcal{A}'$. Given the one-to-one correspondence between the homomorphisms h and h', if the vertex e in $\mathbb{G}$ corresponds with the tuple $(A_1, \ldots, A_n)$ in $\mathcal{X}'$, then the vertex $h(e)$ in $\mathbb{D}$ corresponds with the tuple $(h'(A_1), \ldots, h'(A_n))$ in $\mathcal{A}'$. The cost of applying the homomorphism h to the vertex e is $u(h(e))$. Similarly the cost of applying the homomorphism h' to the corresponding tuple $(A_1, \ldots, A_n)$ is $\rho(h'(A_1), \ldots, h'(A_n))$. It follows from the definitions of u and ρ that we have $u(h(e)) = \rho(h'(A_1), \ldots, h'(A_n))$.

Finally we argue that costs of h and h' are equal. First consider the case where all of the vertices $e \in W$ map to unique top vertices in $\mathbb{D}$, that is if $e_1 \neq e_2 \Rightarrow h(e_1) \neq h(e_2)$. Then $\rho'_w = w_e$ by definition, and therefore $cost(h) = cost(h')$. Now consider the case where h maps at least two vertices in W, say e_1 and e_2, to the same top vertex in $\mathbb{D}$. This implies that both $u(h(e_1)) = \rho(h'(A_1), \ldots, h'(A_n))$ and $u(h(e_2)) = \rho(h'(A_1), \ldots, h'(A_n))$. Therefore the weight ρ'_w must be the sum of the weights w_{e_1} and w_{e_2}, in order for $cost(h) = cost(h')$, and this holds by the definition of ρ'_w.

If the set W is empty then the cost of h is undefined. In turn there would be no tuples $(A_1, \ldots, A_n)$ such that ρ'_w is defined, and the cost of h' would also be undefined. This reduces the problem to the feasibility problem as in [11, 12].

□

4.4 Preservation of Algebraic Properties

The study of algebraic properties (e.g. polymorphisms) of constraint languages has been very useful in classifying the computational complexity of CSPs. It has been the basis of a number of important results such as the CSP dichotomy proof on 3-element domains [7] and the work of Barto and Kozik [4] and Bulatov [8] describing constraint languages that are solvable by local consistency methods (problems of bounded width). The algebraic CSP dichotomy conjecture [10] predicts, in terms of polymorphisms, where the split between polynomial time and **NP**-complete problems occurs.

The important properties of polymorphisms are usually given by *identities*, i.e. equalities of terms that hold for all choices of the variables involved in them. Recall some of the important types of operations we have seen previously:

- An operation f is *idempotent* if it satisfies the identity $f(x, \dots, x) = x$.
- A k-ary ($k \geq 2$) operation f is *weak near unanimity (WNU)* if it is idempotent and satisfies the identities $f(y, x, \dots, x, x) = f(x, y, \dots, x, x) = \cdots = f(x, x, \dots, x, y)$.
- A k-ary ($k \geq 2$) operation f is *cyclic* if $f(x_1, x_2, \dots, x_k) = f(x_2, \dots, x_k, x_1)$.
- A k-ary ($k \geq 2$) operation f is *symmetric* if $f(x_1, \dots, x_k) = f(x_{\pi(1)}, \dots, x_{\pi(k)})$ for each permutation π on $\{1, \dots, k\}$.

For example, the algebraic dichotomy conjecture can be re-stated as follows [10, 49]: for a core structure $\mathcal{A}$, CSP($\mathcal{A}$) is tractable if $\mathcal{A}$ has a WNU polymorphism of some arity, and **NP**-complete otherwise. For a core $\mathcal{A}$, the problems CSP($\mathcal{A}$) has bounded width if and only if $\mathcal{A}$ has WNU polymorphisms of almost all arities [4]. For a core weighted structure $w\mathcal{A}$ such that each weighted relation in $w\mathcal{A}$ is defined on all tuples of the corresponding arity, VCSP($w\mathcal{A}$) is solvable in polynomial time if $w\mathcal{A}$ has symmetric fractional polymorphisms of all arities [59], and it is **NP**-hard otherwise.

It is well known and easy to see that a (weighted or unweighted) structure is a rigid

core if and only if all its polymorphisms are idempotent.

An operational signature is a set of operation symbols with arities assigned to them. An identity is an expression $t_1 = t_2$ where t_1 and t_2 are terms in this signature. An identity $t_1 = t_2$ is said to be *linear* if both t_1 and t_2 involve at most one occurrence of an operation symbol, and *balanced* if the variables occuring in t_1 and t_2 are the same (e.g. $f(x, x, y) = g(y, x, x)$). (This notion is not related to balanced digraphs). A set Σ of identities is linear if it only contains linear identities, idempotent if for each operation symbol, f, the identity $f(x, x, ..., x) = x$ is in Σ and balanced if all of the identities in Σ are balanced. Note the identities defining WNU, symmetric and cyclic operations above are linear and balanced.

Recall the structure $w\mathcal{A}_0$ from Lemma 12. Since $w\mathcal{A}$ is expressible in $w\mathcal{A}_0$, every fractional polymorphism of $w\mathcal{A}_0$, when restricted to D, is a fractional polymorphism of $w\mathcal{A}$ (see [14]). Hence the presence of a fractional polymorphism $\omega : C \to [0, 1]$ such that the operations in C satisfy some set of identities carries over from $w\mathcal{A}_0$ to $w\mathcal{A}$. We show that, for linear balanced sets of identities, the converse is also true.

First we must introduce some facts about connected components of powers of $\mathbb{D}$. Let $\mathbb{D}^k$ be the direct kth power of the digraph $\mathbb{D}$, i.e. its vertices are the k-tuples of elements of $\mathbb{D}$, and $(\mathbf{c}, \mathbf{d})$ is an edge in $\mathbb{D}^k$ if and only if, for all $1 \leq i \leq k$, (c_i, d_i) is an edge in $\mathbb{D}$. Consider the diagonal of $\mathbb{D}$, i.e. the set $\{(c, \ldots, c) \mid c \in V^{\mathbb{D}}\}$. Clearly, the diagonal is contained in one connected component of $\mathbb{D}^k$, denote it by Δ_k. We will need some properties of $\mathbb{D}^k$ proven in [12], and remind the reader of the fact that the vertices of a balanced digraph can be organised into levels, such that for every directed edge (a, b), $lvl(b) = lvl(a) + 1$.

Lemma 14 ([12])**.** *We have both $D^k \subseteq \Delta_k$ and $R^k \subseteq \Delta_k$.*

Lemma 15 ([12])**.** *Assume that a connected component Δ' of $\mathbb{D}^k$ contains a tuple $\mathbf{c} = (c_1, \ldots, c_k)$ such that $lvl(c_1) = \ldots = lvl(c_k)$. Then every $\mathbf{d} = (d_1, \ldots, d_k)$ satisfies $lvl(d_1) = \ldots = lvl(d_k)$ and also either $\Delta' = \Delta_k$ or Δ' is one-element.*

As in [11, 12] we define a linear ordering on the vertices of the digraph $\mathbb{D}$. For every $e = (a, \mathbf{r}) \in D \times R$, denote the path $\mathbb{Q}_{\{i:a=r_i\}}$ in $\mathbb{D}$ by $\mathbb{P}_e$. Also, write $\mathbb{P}_{e,l}$ to mean

$\mathbb{Q}_{S,l}$ where $\mathbb{P}_e = \mathbb{Q}_S$. First, fix a linear ordering $\preceq_1$ on D and extend it to any linear ordering $\preceq$ of $E = D \times R$ such that if $(d_1, \mathbf{t}_1) \preceq (d_2, \mathbf{t}_2) \preceq (d_1, \mathbf{t}_3)$ then $d_1 = d_2$. Now define the mapping $\epsilon : V^{\mathbb{D}} \to E$ by setting $\epsilon(x)$ to be the $\preceq$-minimal $e \in E$ such that $x \in \mathbb{P}_e$. Finally we define the linear order $\sqsubseteq$ on the vertices of the digraph $\mathbb{D}$, where $x \sqsubset y$ if either:

$lvl(x) < lvl(y)$, or

$lvl(x) = lvl(y)$ and $\epsilon(x) \prec \epsilon(y)$, or

$lvl(x) = lvl(y)$, $\epsilon(x) = \epsilon(y)$, and x is closer to $\iota\mathbb{P}_{\epsilon(x)}$ than y.

Lemma 16 ([12])**.** *Let K and L be subsets of $V^{\mathbb{D}}$ such that $L \not\subseteq R$ and*

- *for every $x \in K$ there is $y' \in L$ such that $x \to y'$ is an edge in $\mathbb{D}$, and*
- *for every $y \in L$ there is $x' \in K$ such that $x' \to y$ is an edge in $\mathbb{D}$.*

If c and d are the $\sqsubset$-minimal elements of K and L, respectively, then $c \to d$ is an edge in $\mathbb{D}$.

Now we introduce the main theorem of this section.

Theorem 21. *Let $w\mathcal{A}$ and $(\mathbb{D}, u)$ be as in Theorem 20. If $w\mathcal{A}$ has a k-ary fractional polymorphism $\omega : C \to [0,1]$ such that operations in C satisfy a linear balanced set Σ of identities then $(\mathbb{D}, u)$ also has a k-ary fractional polymorphism $\omega_0 : C_0 \to [0,1]$ such that there is a bijection between C and C_0, and the operations in C_0 satisfy Σ. In particular, if ω is such that some operation in C (or all non-projection operations in C) is WNU then the same holds for ω_0. Similarly, if ω is such that some operation in C (or all non-projection operations in C) is cyclic or symmetric then the same holds for ω_0.*

Proof. It is shown in [11, 12] how polymorphisms of ρ can be transformed (in fact, extended) to polymorphisms of $\mathbb{D}$ in such a way that any set of linear balanced identities carries over. The transformation there is designed to preserve not only balanced identities, and, for our purposes, we can use a simplified version of it. Let C_0 be obtained from C by applying this (simplified) transformation to all non-projection operations in C and extending projection operations in C so that they stay projection operations. Obtain ω_0 from ω by using this bijection between C_0 and

C, i.e. keep the weights of operations the same. Then ω_0 will be a fractional polymorphism of $\mathbb{D}$. Indeed, since $\mathbb{D}$ is 0-weighted, the last condition in the definition of a fractional polymorphism will be trivially satisfied, while the other conditions trivially carry over. Hence, it only remains to ensure that ω_0 is a fractional polymorphism of u. For this, we extend the operations f from C to operations f_0 on $V^{\mathbb{D}}$ in such a way that, for any $v_1, \ldots, v_k \in V^{\mathbb{D}}$, we have $f_0(v_1, \ldots, v_k) \in R$ only if $v_1, \ldots, v_k \in R$. With this condition, the fact that ω_0 is a fractional polymorphism of u follows from the fact that ω is a fractional polymorphism of ρ, as we show in the rest of the proof.

Let Σ be a set of linear balanced identities in operations symbols $\{f_\lambda \mid \lambda \in \Lambda\}$ such that, interpreting each f_λ as a specific operation $f_\lambda^{\mathcal{A}} \in C$, the operations $\{f_\lambda^{\mathcal{A}} \mid \lambda \in \Lambda\}$ satisfy Σ. We can without loss of generality assume that $\{f_\lambda^{\mathcal{A}} \mid \lambda \in \Lambda\}$ is the set of all non-projection operations in C.

We will extend each projection operation in C to the corresponding projection on $V^{\mathbb{D}}$ and each non-projection operation $f_\lambda^{\mathcal{A}} \in C$ to a polymorphism $f_\lambda^{\mathbb{D}}$ of $\mathbb{D}$ in such a way that $\{f_\lambda^{\mathbb{D}} \mid \lambda \in \Lambda\}$ will also satisfy Σ. The construction will also ensure that ω_0 obtained from ω as described above is indeed a fractional polymorphism of u.

As in [11, 12], let the digraph $\mathbb{Z}$ be the zigzag with vertices labelled 00, 01, 10 and 11, such that we describe the oriented path $00 \to 01 \leftarrow 10 \to 11$. Given a vertex pair $\{x, y\}$ in the zigzag, define the operation $\wedge$ such that $x \wedge y$ is the vertex closer to 00. For each $\lambda \in \Lambda$, let $f_\lambda^{\mathbb{Z}}(x_1, ..., x_k) = \bigwedge_{i=1}^{k} x_i$ where k is the arity of f_λ. It is clear that the set $\{f_\lambda^{\mathbb{Z}} \mid \lambda \in \Lambda\}$ satisfies any balanced set of identities.

Now we define polymorphisms $\{f_\lambda^{\mathbb{D}} \mid \lambda \in \Lambda\}$. Fix $\lambda \in \Lambda$, assume that f_λ is a k-ary non-projection operation and let $\mathbf{c} \in (V^{\mathbb{D}})^k$. If $\mathbf{c} \in R^k$ then $(f_\lambda^{\mathcal{A}})^{(k)}(\mathbf{c})$ will denote the element of R obtained by applying $f_\lambda^{\mathcal{A}}$ to the tuples $c_1, \ldots, c_k \in R$ component-wise. Note that $(f_\lambda^{\mathcal{A}})^{(k)}(\mathbf{c}) \in R$ because $f_\lambda^{\mathcal{A}}$ is a polymorphism of R. Similarly, we can apply $f_\lambda^{\mathcal{A}}$ to elements $e_1, \ldots, e_k \in D \times R$ and obtain an element $(f_\lambda^{\mathcal{A}})^{k+1}(e_1, ..., e_k)$ from $D \times R$.

Construct $f^{\mathbb{D}}_{\lambda}$ as follows:

Case 1. $\mathbf{c} \in D^k \cup R^k$.

1a. If $\mathbf{c} \in D^k$, we define $f^{\mathbb{D}}_{\lambda}(\mathbf{c}) = f^{\mathcal{A}}_{\lambda}(\mathbf{c})$.

1b. If $\mathbf{c} \in R^k$, we define $f^{\mathbb{D}}_{\lambda}(\mathbf{c}) = (f^{\mathcal{A}}_{\lambda})^{(k)}(\mathbf{c})$.

Case 2. $\mathbf{c} \in \Delta_k \backslash (D^k \cup R^k)$.

Let $e_i = \epsilon(c_i)$ for $1 \leq i \leq k$ and $e = (f^{\mathcal{A}}_{\lambda})^{k+1}(e_1, ..., e_k)$. Let $1 \leq l \leq k$ be minimal such that $c_i \in \mathbb{P}_{e_i,l}$ for all $1 \leq i \leq k$.

2a. If $\mathbb{P}_{e,l}$ is a single edge, then we define $f^{\mathbb{D}}_{\lambda}(\mathbf{c})$ to be the vertex from $\mathbb{P}_{e,l}$ having the same level as all the c_i's.

If $\mathbb{P}_{e,l}$ is a zigzag then at least one of the $\mathbb{P}_{e_i,l}$'s is a zigzag as well. For every $1 \leq i \leq k$ such that $\mathbb{P}_{e_i,l}$ is a zigzag let $\Phi_i : \mathbb{P}_{e_i,l} \to \mathbb{Z}$ be the unique isomorphism. Let Φ denote the isomorphism from $\mathbb{P}_{e,l}$ to $\mathbb{Z}$.

2b. If all of the $\mathbb{P}_{e_i,l}$'s are zigzags, then $f^{\mathbb{D}}_{\lambda}(\mathbf{c}) = \Phi^{-1}(f^{\mathbb{Z}}_{\lambda}(\Phi_1(c_1), ..., \Phi_m(c_k)))$.

2c. Else, we define $f^{\mathbb{D}}_{\lambda}(\mathbf{c})$ to be the $\sqsubseteq$-minimal element from the set $\{\Phi^{-1}(\Phi_i(c_i)) | \mathbb{P}_{e_i,l}$ is a zigzag$\}$

Case 3. $\mathbf{c} \notin \Delta_k$.

Define $f^{\mathbb{D}}_{\lambda}(\mathbf{c})$ to be the $\sqsubseteq$-minimal element from the set $\{c_1, ..., c_k\}$.

The definition of $f^{\mathbb{D}}_{\lambda}$ in [12] is similar, but Case 3 there is split into three subcases (3a)-(3c), which is unnecessary for our purposes, as we use their (3c) throughout our Case 3. The proof that $f^{\mathbb{D}}_{\lambda}$ is a polymorphism of $\mathbb{D}$ is a subset of the proof of Claim 5.7 in [12].

It remains to show that ω_0 is a fractional polymorphism of u, i.e. ω_0 and u satisfy the last condition in the definition of a fractional polymorphism. When applied to u, this condition says that, for any $x_1, \ldots, x_k \in V^{\mathbb{D}}$, we have $\sum_{f \in C_0} \omega_0(f) u(f(x_1, x_2, ..., x_k)) \leq \frac{1}{k}(u(x_1) + u(x_2) + ... + u(x_k))$. Recall that, by definition, $u(x) = 0$ for all $x \in V^{\mathbb{D}} \backslash R$. By inspecting our definition of $f^{\mathbb{D}}_{\lambda}$, it is clear that if $f^{\mathbb{D}}_{\lambda}(x_1, \ldots, x_k) \in R$ then $x_1, \ldots, x_k \in R$. Thus, if not all $x_1, \ldots, x_k$ are in R, the only possi-

ble non-zero terms appear in the RHS of the inequality, and hence it is trivially true. On the other hand, if all $x_1, \ldots, x_k$ are in R then $f(x_1, \ldots, x_k) \in R$ and so $u(f(x_1, \ldots, x_k)) = \rho(f(x_1, \ldots, x_k))$. In this case, the inequality holds because the inequality $\sum_{f \in C} \omega(f)\rho(f(x_1, x_2, ..., x_k)) \leq \frac{1}{k}(\rho(x_1) + \rho(x_2) + ... + \rho(x_k))$ holds for ω. □

4.5 Dealing with Short Components

In this section, we provide the argument justifying Stage 2 of the algorithm from the proof of Lemma 13.

Although we consider VCSPs, when eliminating short components of an input digraph $\mathbb{G}$, we first check that the components are satisfiable in $\mathbb{D}$ using the method for standard CSPs as in [11]. This involves testing that components are satisfiable in some fixed family of directed paths, and then we can identify their associated costs. Recall that a path $\mathbb{Q}_S$ has zigzags in every position i where $i \notin S$, and thus we have the following lemma:

Lemma 17 ([11])**.** *Consider the paths $\mathbb{Q}_S$ where $S \subseteq \{1, ..., k\}$.*

1. $\mathrm{CSP}(\mathbb{Q}_{[k]\setminus\{i\}})$ *is solvable in polynomial time for any $i \in \{1, \ldots, k\}$, even when singleton unary relations are added.*

2. *For any $S \subseteq \{1, \ldots, k\}$ the problem* $\mathrm{CSP}(\mathbb{Q}_S)$ *is solvable in polynomial time.*

Definition 36. *Let $\mathbb{Q}_{S_1}$, $\mathbb{Q}_{S_2}, \ldots, \mathbb{Q}_{S_l}$ be paths that all have the same initial (or terminal) vertex in $\mathbb{D}$. Define $\mathbb{F}$ to be the fan structure obtained when these paths are amalgamated at their shared vertex v, and $u|_{\mathbb{F}}$ be our unary function restricted to $\mathbb{F}$.*

Lemma 18. $\mathrm{MinCostHom}(\mathbb{F}, u|_{\mathbb{F}})$*, restricted to inputs of height less than m, is polynomial-time solvable.*

Proof. Consider an instance $H = (\mathbb{H}, W)$ of $\mathrm{MinCostHom}(\mathbb{F}, u|_{\mathbb{F}})$. We may assume $\mathbb{H}$ has a homomorphism to $\mathbb{F}$, has height strictly less than the height of $\mathbb{F}$, and is a single component. We call a homomorphism from $\mathbb{H}$ to $\mathbb{F}$ a solution, and $\mathbb{H}$ is satisfiable if it has a solution. We now consider the following cases:

1. First check if $\mathbb{H}$ has a solution that does not involve v.

   In this case $\mathbb{H}$ must be satisfiable within at least one of the paths $\mathbb{Q}_{S_i}$ of $\mathbb{F}$ and by applying (2) of Lemma 17 to every path $\mathbb{Q}_{S_i}$ we can identify all paths where $\mathbb{H}$ is feasible. It is possible that a vertex u of $\mathbb{H}$ can be interpreted at different heights in a path $\mathbb{Q}_{S_i}$, and we use the unary singletons to fix a particular height for u to test for a solution. If we do not find a solution fixing u at that height we successively try new heights for u. If we find no solutions for $\mathbb{H}$ at any height in any path $\mathbb{Q}_{S_i}$ then we continue to case 2. If there is a solution and a vertex of $\mathbb{H}$ maps to a top level vertex t in $\mathbb{F}$ and $t \in W$ then the cost of that mapping is $u|_{\mathbb{F}}(t)$. If $\mathbb{H}$ has multiple solutions with non-zero cost then we choose the solution with the minimal cost, and $\mathbb{H}$ reduces to a single valued tuple of the objective function of $\mathcal{A}$ determined by t, with cost $u|_{\mathbb{F}}(t)$. If no vertex of $\mathbb{H}$ maps to a vertex in W in any feasible solution then $\mathbb{H}$ has no influence on the optimisation problem and is ignored for the remainder of the reduction.

2. $\mathbb{H}$ has a solution involving v.

   If $v \notin W$ we follow the same procedure given in [11] as there is no optimisation to consider. That procedure is given here for completeness. First choose a vertex $h \in \mathbb{H}$, and check if h can be interpreted at height m in the same way as (1) of Lemma 17. Consider the components C_j of the induced subgraph obtained by removing the vertices of height m from $\mathbb{H}$. Test every component C_j for satisfaction in a path $\mathbb{Q}_{S_i}$, with the highest level vertices of C_j constrained to be at height $m-1$ in $\mathbb{Q}_{S_i}$. If every component C_j can be satisfied in some path $\mathbb{Q}_{S_i}$, then $\mathbb{H}$ is satisfiable in $\mathbb{F}$. Should $\mathbb{H}$ not be satisfiable in $\mathbb{F}$ for a particular choice of vertex h then select a new vertex h, and repeat until $\mathbb{H}$ is found to be satisfiable in $\mathbb{F}$ (as we assume $\mathbb{H}$ has a feasible solution).

   If $v \in W$ then $\mathbb{H}$ reduces to a single valued tuple in the objective function of $\mathcal{A}$ determined by v, with cost $u|_{\mathbb{F}}(v)$.

□

4.6 A Hard Case

An interesting problem to consider now is which digraphs can capture hard VCSPs. For the case of ordinary CSPs there is the following result.

Theorem 22 ([23])**.** *Every CSP is polynomial time equivalent to a balanced digraph homomorphism problem with only 5 levels (but not just 4 levels, which is polynomial time solvable).*

Note that 5 levels means the digraph has height 4, and 4 levels means the digraph has height 3.

Now we offer an example of a digraph and unary function which can capture MAX CUT, and therefore the standard definition of hardness we use for VCSP.

Example 21. *Let $D = \{0, 1\}$ and consider the digraph $\mathbb{D}$ shown Figure 4.2. Let the unary function $u(v)$ be*

$$u(v) = \begin{cases} 1 & \text{if } v = b \text{ or } v = c \\ 0 & \text{otherwise} \end{cases}$$

for every vertex $v \in V^{\mathbb{D}}$.

Now consider the instance $(\mathbb{G}, W)$ of MinCostHom$(\mathbb{D}, u)$*, where $\mathbb{G}$ is the source digraph shown in Figure 4.3 and W contains both of the top level vertices of $\mathbb{G}$. It is straightfoward to see that the homomorphism that maps $s \to x$ and $t \to y$ has cost 0, as does the homomorphism that maps $s \to y$ and $t \to x$. However the homomorphism that maps $s \to x$ and $t \to x$ has cost 2, and likewise for the homomorphism that maps $s \to y$ and $t \to y$. If we consider these homomorphisms the possible mappings of variables to domain values then we have a problem with $cost(0, 0) = cost(1, 1) > cost(0, 1) = cost(1, 0)$, and thus we capture MAX CUT.*

Note that the digraph $\mathbb{D}$ could have an oriented path from x to c and similarly from y to b that consist of a single edge followed by two zigzags and another single edge. However no path in $\mathbb{G}$ can possibly map onto these oriented paths, so they are omitted from $\mathbb{D}$ for clarity of the diagram.

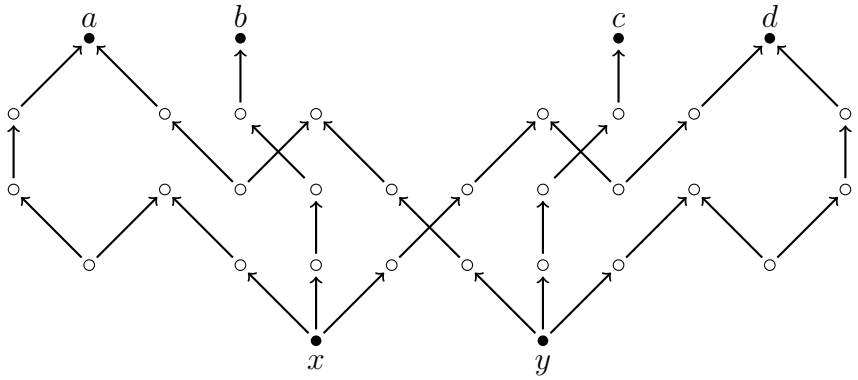


Figure 4.2: The target digraph $\mathbb{D}$ of Example 21.

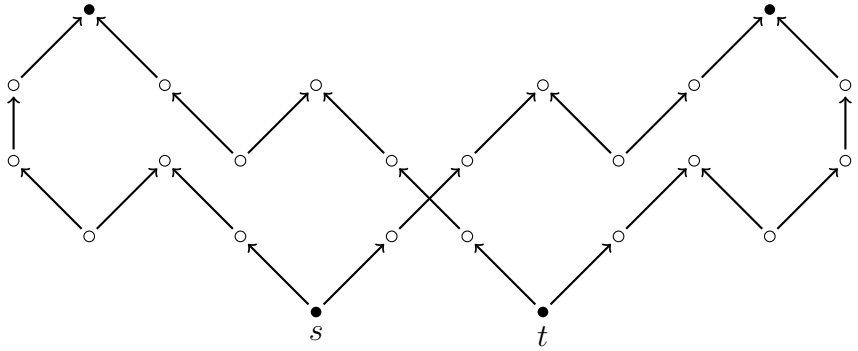


Figure 4.3: The source digraph $\mathbb{G}$ of Example 21.

Chapter 5

Fractional Polymorphisms as Digraphs

5.1 Introduction

As we have seen in Chapter 2 all tractable multimorphisms $\langle f, g \rangle$ are representable as digraphs, with each domain element represented as a vertex, and the functions f and g as arcs. However, based on the work in Chapter 3 it is now proven that multimorphisms cannot capture all tractable constraint languages, and as such we need the more general definition of fractional polymorphisms. An obvious problem to consider now is can we describe these necessary fractional polymorphisms as digraphs in a similar fashion?

Ideally we would like to define a set of digraphs (on a finite domain) that describe only the specific fractional polymorphisms necessary for tractability of finite-valued constraint languages on that domain. We already know that a single semilattice operation in the support of a fractional polymorphism is enough to ensure tractability (see Theorem 13) of a constraint language, however only on two and three element domains have we managed to narrow this down to specific necessary fractional polymorphisms. This question of whether tight (or at least tighter) conditions for tractability can be found is asked explicitly in [36]

In this chapter we present a definition for describing fractional polymorphisms as digraphs, and attempt to define a set of digraphs for each domain that describe only the necessary fractional polymorphisms for tractability on that domain. These sets of digraphs (and thus the fractional polymorphisms they describe) on domains of size two and three correlate exactly with the known tight dichotomy theorems for domains of size two (see Theorem 15) and three (see Theorem 18). The only

digraph that fits the definition on the two element domain describes submodularity, and the two digraphs on the three element domain describe submodularity and α-bisubmodularity. We also begin to investigate the set of digraphs defined on four element domains, and offer generalisations of some previously identified tractable fractional polymorphisms.

For the purposes of this chapter we consider fractional polymorphisms whose support contains only symmetric binary operations. Given a cost function ϕ, of arity k, any fractional polymorphism ω of ϕ that we consider can be written in the form:

$$\phi(\mathbf{x}) + \phi(\mathbf{y}) \geq \sum_{f \in supp(\omega)} w(f)\phi(f(\mathbf{x}, \mathbf{y})) \quad \text{and} \quad \sum_{f \in supp(\omega)} w(f) = 2$$

for all tuples $\mathbf{x}, \mathbf{y} \in D^k$.

Given the result of Thapper and Živný [59] that we presented earlier (see Lemma 3) we know that a core valued constraint language will generate a connected digraph. Therefore we offer the following definition of describing fractional polymorphisms as digraphs, the idea for which is based on the operations present in the description of skew bisubmodularity (see Definition 26).

Definition 37. *Let Γ be a rigid core finite-valued constraint language over a finite domain D. Let $\mathbb{G}$ be the connected directed acyclic graph whose vertices are the elements of D and whose arcs describe a partial order of the elements of D, e.g if a, b, c are elements of D that have the partial order $a > b > c$, then the digraph contains directed arcs from a to b and b to c.*

For any digraph $\mathbb{G}$ we say Γ exhibits the fractional polymorphism described by $\mathbb{G}$ if every cost function $\phi \in \Gamma$, of arity k, satisfies the inequality:

$$\phi(\boldsymbol{x}) + \phi(\boldsymbol{y}) \geq s(\boldsymbol{x}, \boldsymbol{y}) + t(\boldsymbol{x}, \boldsymbol{y})$$

for all tuples $\boldsymbol{x}, \boldsymbol{y} \in D^k$, where we define $s(\boldsymbol{x}, \boldsymbol{y})$ and $t(\boldsymbol{x}, \boldsymbol{y})$ as:

$$s(\boldsymbol{x}, \boldsymbol{y}) = \sum_{c \in D} w_c^s \phi(\boldsymbol{x} \wedge_c \boldsymbol{y}) \quad \textit{and} \quad \sum_{c \in D} w_c^s = 1,$$

$$t(\boldsymbol{x}, \boldsymbol{y}) = \sum_{c \in D} w_c^t \phi(\boldsymbol{x} \vee_c \boldsymbol{y}) \quad \textit{and} \quad \sum_{c \in D} w_c^t = 1.$$

The operations $\wedge_c$ and $\vee_c$ are applied componentwise, returning the greatest lower bound and least upper bound respectively, if they exist, and the domain element c if not.

We should note that it is often possible to simplify the resulting fractional polymorphisms by applying them to themselves. The condition of being a directed acyclic graph excludes directed cycles, but it is already known that directed cycles describe tournament pair multimorphisms which have been proven to be tractable by reduction to submodularity [16, 43]. Thus any directed cycle could be replaced with a chain, and this is a case that can be captured by Definition 37. Undirected 3-cycles also represent tournament pair multimorphisms (and hence are reducible to submodularity) but are automatically replaced by chains in Definition 37, due to the partial order condition.

5.2 Two Element Domains

On the two element domain there is only one digraph that fits Definition 37, that is the digraph on two vertices with a single edge between them.

Proposition 5. *The fractional polymorphism described by the digraph on two vertices with a single edge between them implies that all cost functions ϕ, of arity k, of a core valued constraint language are submodular.*

Proof. The digraph has a single pair of vertices connected by an edge and as such we obtain the following functions s and t:

$$s(\mathbf{x},\mathbf{y}) = w_x^s\phi(\mathbf{x}\wedge_x\mathbf{y}) + w_y^s\phi(\mathbf{x}\wedge_y\mathbf{y}) = w_x^s\phi(\mathbf{x}\wedge\mathbf{y}) + w_y^s\phi(\mathbf{x}\wedge\mathbf{y}) = \phi(\mathbf{x}\wedge\mathbf{y})$$

$$t(\mathbf{x},\mathbf{y}) = w_x^t\phi(\mathbf{x}\vee_x\mathbf{y}) + w_y^t\phi(\mathbf{x}\vee_y\mathbf{y}) = w_x^t\phi(\mathbf{x}\vee\mathbf{y}) + w_y^t\phi(\mathbf{x}\vee\mathbf{y}) = \phi(\mathbf{x}\vee\mathbf{y})$$

So our fractional polymorphism inequality is simply:

$$\phi(\mathbf{x}) + \phi(\mathbf{y}) \geq \phi(\mathbf{x}\wedge\mathbf{y}) + \phi(\mathbf{x}\vee\mathbf{y})$$

for all tuples $\mathbf{x},\mathbf{y}\in D^k$, which describes submodularity as defined in Example 11. □

The only other digraph on two vertices is that with no edges, but it does not fit Definition 37 as it is not connected. Lemma 3 imples that such a digraph describes

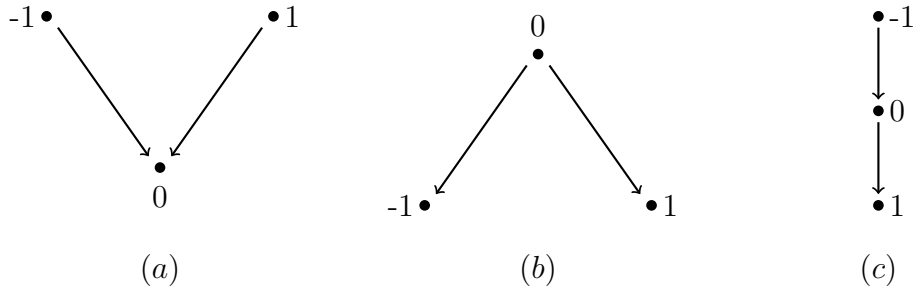


Figure 5.1: The three connected digraphs on three vertices with no 3-cycles.

a fractional polymorphism which only captures valued constraint languages that are not core.

Therefore Definition 37 is tight for the two element domain as it captures only submodularity, and this is known to be the only condition for tractability for core valued constraint languages on two elements.

5.3 Three Element Domains

As we have seen in Chapter 3, the three element domain is the first to exhibit a dichotomy where fractional polymorphisms are necessary, as multimorphisms cannot capture all tractable constraint languages. While it was possible to describe bisubmodularity as a digraph as shown in Chapter 2, it was the fact that we had no way to describe α-bisubmodularity as a digraph that led to us constructing Definition 37.

Through the following propositions we consider all digraphs on three vertices that satisfy Definition 37, and they can be seen in Figure 5.1. Remember that the cycles that can be drawn on three vertices that can be seen in Figure 2.1 do not fit the description of digraphs that we are considering. They are known to be tractable by a reduction to submodularity with respect to some totally ordered chain, so providing we can describe submodularity we still capture these cases.

Proposition 6. *Any valued constraint language that exhibits the fractional polymorphism described by digraph (c) of Figure 5.1 is submodular.*

Proof. By definition:

$$s(\mathbf{x}, \mathbf{y}) = w^s_{-1}\phi(\mathbf{x} \wedge_{-1} \mathbf{y}) + w^s_1\phi(\mathbf{x} \wedge_1 \mathbf{y}) + w^s_0\phi(\mathbf{x} \wedge_0 \mathbf{y})$$

$$t(\mathbf{x}, \mathbf{y}) = w^t_{-1}\phi(\mathbf{x} \vee_{-1} \mathbf{y}) + w^t_1\phi(\mathbf{x} \vee_1 \mathbf{y}) + w^t_0\phi(\mathbf{x} \vee_0 \mathbf{y})$$

As every pair of vertices has both a greatest lower bound and least upper bound we can simplify s and t as:

$$(\mathbf{x} \wedge_{-1} \mathbf{y}) = (\mathbf{x} \wedge_1 \mathbf{y}) = (\mathbf{x} \wedge_0 \mathbf{y}) = (\mathbf{x} \wedge \mathbf{y})$$

$$(\mathbf{x} \vee_{-1} \mathbf{y}) = (\mathbf{x} \vee_1 \mathbf{y}) = (\mathbf{x} \vee_0 \mathbf{y}) = (\mathbf{x} \vee \mathbf{y})$$

Thus our fractional polymorphism inequality simplifies down to:

$$\phi(\mathbf{x}) + \phi(\mathbf{y}) \geq \phi(\mathbf{x} \wedge \mathbf{y}) + \phi(\mathbf{x} \vee \mathbf{y})$$

for all tuples $\mathbf{x}, \mathbf{y} \in D^k$, where k is the arity of ϕ. This inequality again describes submodularity (submodularity on a chain in this case). □

Proposition 7. *Any valued constraint language that exhibits the fractional polymorphism described by digraph (a) or digraph (b) of Figure 5.1 is α-bisubmodular.*

Proof. First consider digraph (a). Every pair of domain elements has a greatest lower bound, so we can simplify s as we did in the previous proof, but the pair $\{-1, 1\}$ does not have a least upper bound. Therefore we obtain the following functions:

$$s(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x} \wedge \mathbf{y})$$

$$t(\mathbf{x}, \mathbf{y}) = w^t_{-1}\phi(\mathbf{x} \vee_{-1} \mathbf{y}) + w^t_1\phi(\mathbf{x} \vee_1 \mathbf{y}) + w^t_0\phi(\mathbf{x} \vee_0 \mathbf{y})$$

Note that the operation $\wedge$ defined here is identical to the operation $\wedge_0$ in the definition of α-bisubmodularity. Now we have the fractional polymorphism inequality:

$$\phi(\mathbf{x}) + \phi(\mathbf{y}) \geq \phi(\mathbf{x} \wedge \mathbf{y}) + w^t_{-1}\phi(\mathbf{x} \vee_{-1} \mathbf{y}) + w^t_1\phi(\mathbf{x} \vee_1 \mathbf{y}) + w^t_0\phi(\mathbf{x} \vee_0 \mathbf{y})$$

for all tuples $\mathbf{x}, \mathbf{y} \in D^k$, where k is the arity of ϕ.

If $w^t_{-1} = 0$ or $w^t_1 = 0$ then this fractional polymorphism is exactly the definition of α-bisubmodularity (see Definition 26), with $w^t_0 = \alpha$. If $w^t_{-1} = 0$ and $w^t_1 = 0$

then $w_0^t = 1$ and the fractional polymorphism describes 1-bisubmodularity, which is the original definition of bisubmodularity (as in Example 12). If both $w_{-1}^t > 0$ and $w_1^t > 0$ then we need to massage this fractional polymorphism to look like the definition of α-bisubmodularity. We do that as follows.

Substitute $\mathbf{x}$ with $\mathbf{x} \vee_{-1} \mathbf{y}$ and $\mathbf{y}$ with $\mathbf{x} \vee_1 \mathbf{y}$ and apply the fractional polymorphism to itself to obtain:

$$\begin{aligned}\phi(\mathbf{x} \vee_{-1} \mathbf{y}) + \phi(\mathbf{x} \vee_1 \mathbf{y}) \geq \phi((\mathbf{x} \vee_{-1} \mathbf{y}) \wedge (\mathbf{x} \vee_1 \mathbf{y})) + w_{-1}^t \phi((\mathbf{x} \vee_{-1} \mathbf{y}) \vee_{-1} (\mathbf{x} \vee_1 \mathbf{y})) \\ + w_1^t \phi((\mathbf{x} \vee_{-1} \mathbf{y}) \vee_1 (\mathbf{x} \vee_1 \mathbf{y})) + w_0^t \phi((\mathbf{x} \vee_{-1} \mathbf{y}) \vee_0 (\mathbf{x} \vee_1 \mathbf{y}))\end{aligned}$$

which simplifies to:

$$\phi(\mathbf{x} \vee_{-1} \mathbf{y}) + \phi(\mathbf{x} \vee_1 \mathbf{y}) \geq (1 + w_0^t)\phi(\mathbf{x} \vee_0 \mathbf{y}) + w_{-1}^t \phi(\mathbf{x} \vee_{-1} \mathbf{y}) + w_1^t \phi(\mathbf{x} \vee_1 \mathbf{y})$$

Without loss of generality assume $w_{-1}^t \geq w_1^t$, so $w_{-1}^t = w_1^t + \epsilon$ where $\epsilon = w_{-1}^t - w_1^t$.

Now we can rearrange our inequality:

$$(1 - w_1^t)\phi(\mathbf{x} \vee_{-1} \mathbf{y}) + (1 - w_1^t)\phi(\mathbf{x} \vee_1 \mathbf{y}) \geq (1 + w_0^t)\phi(\mathbf{x} \vee_0 \mathbf{y}) + \epsilon\phi(\mathbf{x} \vee_{-1} \mathbf{y})$$

Scaling this, substituting back into our fractional polymorphism inequality, and simplifying the coefficients we obtain:

$$\phi(\mathbf{x}) + \phi(\mathbf{y}) \geq \phi(\mathbf{x} \wedge \mathbf{y}) + \frac{w_{-1}^t - w_1^t}{1 - w_1^t}\phi(\mathbf{x} \vee_{-1} \mathbf{y}) + \frac{1 - w_{-1}^t}{1 - w_1^t}\phi(\mathbf{x} \vee_0 \mathbf{y})$$

for all tuples $\mathbf{x}, \mathbf{y} \in D^k$, where k is the arity of ϕ. Let $\frac{1-w_{-1}^t}{1-w_1^t} = \alpha$ then $\frac{w_{-1}^t - w_1^t}{1-w_1^t} = 1 - \alpha$ and again this is exactly the definition of α-bisubmodularity given in Definition 26.

Now consider digraph (b). In the case of regular bisubmodularity this digraph is simply the dual of digraph (a), but this is not quite the case when considering fractional polymorphisms instead of multimorphisms. Now every pair of elements has a least upper bound, but the pair $\{-1, 1\}$ does not have a greatest lower bound. Therefore we obtain the following functions:

$$s(\mathbf{x}, \mathbf{y}) = w_{-1}^s \phi(\mathbf{x} \wedge_{-1} \mathbf{y}) + w_1^s \phi(\mathbf{x} \wedge_1 \mathbf{y}) + w_0^s \phi(\mathbf{x} \wedge_0 \mathbf{y})$$

$$t(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x} \vee \mathbf{y})$$

Note that the operation $\vee$ defined here is identical to the operation $\vee_0$ in the definition of α-bisubmodularity. Now we have the fractional polymorphism inequality:

$$\phi(\mathbf{x}) + \phi(\mathbf{y}) \geq w^s_{-1}\phi(\mathbf{x} \wedge_{-1} \mathbf{y}) + w^s_1\phi(\mathbf{x} \wedge_1 \mathbf{y}) + w^s_0\phi(\mathbf{x} \wedge_0 \mathbf{y}) + \phi(\mathbf{x} \vee \mathbf{y})$$

for all tuples $\mathbf{x}, \mathbf{y} \in D^k$, where k is the arity of ϕ.

If $w^s_{-1} = 0$ and $w^s_1 = 0$ then $w^s_0 = 1$ and the fractional polymorphism describes 1-bisubmodularity, which is the original definition of bisubmodularity. If $w^s_{-1} = 0$ or $w^s_1 = 0$ then this fractional polymorphism returns exactly as α-bisubmodularity would return on the same inputs with the correct coefficients. In a similar way to the previous case if both $w^s_{-1} > 0$ and $w^s_1 > 0$ then we need to massage this fractional polymorphism to look more like the definition of α-bisubmodularity. We do that as follows.

Substitute $\mathbf{x}$ with $\mathbf{x} \wedge_{-1} \mathbf{y}$ and $\mathbf{y}$ with $\mathbf{x} \wedge_1 \mathbf{y}$ and apply the fractional polymorphism to itself to obtain:

$$\begin{aligned}\phi(\mathbf{x} \wedge_{-1} \mathbf{y}) + \phi(\mathbf{x} \wedge_1 \mathbf{y}) \geq \phi((\mathbf{x} \wedge_{-1} \mathbf{y}) \vee (\mathbf{x} \wedge_1 \mathbf{y})) + w^s_{-1}\phi((\mathbf{x} \wedge_{-1} \mathbf{y}) \wedge_{-1} (\mathbf{x} \wedge_1 \mathbf{y})) \\ + w^s_1\phi((\mathbf{x} \wedge_{-1} \mathbf{y}) \wedge_1 (\mathbf{x} \wedge_1 \mathbf{y})) + w^s_0\phi((\mathbf{x} \wedge_{-1} \mathbf{y}) \wedge_0 (\mathbf{x} \wedge_1 \mathbf{y}))\end{aligned}$$

which simplifies to:

$$\phi(\mathbf{x} \wedge_{-1} \mathbf{y}) + \phi(\mathbf{x} \wedge_1 \mathbf{y}) \geq (1 + w^s_0)\phi(\mathbf{x} \wedge_0 \mathbf{y}) + w^s_{-1}\phi(\mathbf{x} \wedge_{-1} \mathbf{y}) + w^s_1\phi(\mathbf{x} \wedge_1 \mathbf{y})$$

Without loss of generality assume $w^s_{-1} \geq w^s_1$, so $w^s_{-1} = w^s_1 + \epsilon$ where $\epsilon = w^s_{-1} - w^s_1$.

Now we can rearrange our inequality:

$$(1 - w^s_1)\phi(\mathbf{x} \wedge_{-1} \mathbf{y}) + (1 - w^s_1)\phi(\mathbf{x} \wedge_1 \mathbf{y}) \geq (1 + w^s_0)\phi(\mathbf{x} \wedge_0 \mathbf{y}) + \epsilon\phi(\mathbf{x} \wedge_{-1} \mathbf{y})$$

Scaling this, substituting back into our fractional polymorphism inequality, and simplifying the coefficients we obtain:

$$\phi(\mathbf{x}) + \phi(\mathbf{y}) \geq \phi(\mathbf{x} \vee \mathbf{y}) + \frac{w^s_{-1} - w^s_1}{1 - w^s_1}\phi(\mathbf{x} \wedge_{-1} \mathbf{y}) + \frac{1 - w^s_{-1}}{1 - w^s_1}\phi(\mathbf{x} \wedge_0 \mathbf{y})$$

Let $\frac{1-w^s_{-1}}{1-w^s_1} = \alpha$ then $\frac{w^s_{-1}-w^s_1}{1-w^s_1} = 1 - \alpha$ and we get the following inequality:

$$\phi(\mathbf{x}) + \phi(\mathbf{y}) \geq \phi(\mathbf{x} \vee \mathbf{y}) + (1 - \alpha)\phi(\mathbf{x} \wedge_{-1} \mathbf{y}) + \alpha\phi(\mathbf{x} \wedge_0 \mathbf{y})$$

for all tuples $\mathbf{x}, \mathbf{y} \in D^k$, where k is the arity of ϕ.

Given the same inputs this will always return the same output as the well known α-bisubmodularity fractional polymorphism, so this is simply a different characterisation of the constraint languages captured by α-bisubmodularity. If $\alpha = 1$ this is regular bisubmodularity again, and if $\alpha = 0$ this fractional polymorphism describes a valued constraint language that is not core. □

Following the analysis of the fractional polymorphisms generated by each of the digraphs in Figure 5.1 we see that the connected digraphs each describe a tractable fractional polymorphism, and that we get a tight description of the tractable cases on the three element domain. However we would prefer not to have two digraphs that capture the same tractable constraint languages, but this would require a change to definition 37. The possibility of altering this definition is considered later.

5.4 Moving to Four Element Domains

We have seen that all the fractional polymorphisms responsible for describing tractable constraint languages on two and three element domains can be represented as connected directed acyclic graphs. Now we move to four element domains, applying the same definition of representing fractional polymorphisms as digraphs, in an attempt to provide a tight set of conditions for tractability.

Let us begin by considering the few known results on four element domains. First we introduce the notion of generalised submodularity.

Definition 38. *Let D be a finite lattice - that is a partially ordered set where each pair of elements $\{a, b\} \in D$ has a least upper bound, $\vee$, and greatest upper bound, $\wedge$. Any valued constraint language that admits the multimorphism $\langle \vee, \wedge \rangle$ is said to be submodular over the lattice D.*

It is clear that constraint languages with the multimorphism $\langle \vee, \wedge \rangle$ are tractable by BLP.

Submodularity as we have introduced it previously is the case of generalised submodularity where D is a totally ordered lattice, i.e. a chain. Generalised submodularity

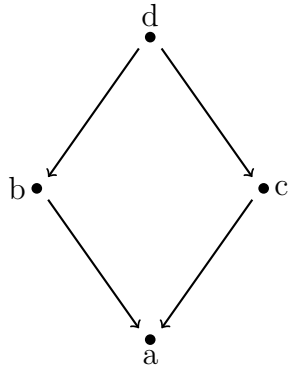


Figure 5.2: An example digraph exhibiting generalised submodularity.

only starts to be interesting on domains of size four or greater as on smaller domains submodularity can only occur on a chain.

A multimorphism of this form is easily described as a digraph. Consider the example over the four element domain shown in Figure 5.2. The constraint language is not submodular (on a chain) as the domain elements b and c are incomparabale, but it does exhibit the generalised submodularity multimorphism as every pair of elements have a least upper bound and greatest lower bound. This digraph is also considered in [38] as a special case of a 1-defect chain on four element domains, which we investigate in the following section.

Another fractional polymorphism known to describe tractable valued constraint languages on four element domains is k-submodularity as defined by Huber and Kolmogorov [30]. This is a generalisation of bisubmodularity (which is only defined on three element domains), which is the special case of k-submodularity where $k = 2$. When considering four element domains we want the specific case of 3-submodularity, and it is already known that valued constraint languages that are 3-submodular can be minimised in polynomial time by BLP [42, 57], see also Theorem 13. However, as in the case of bisubmodularity it is likely we will need to generalise 3-submodularity to allow for skewness.

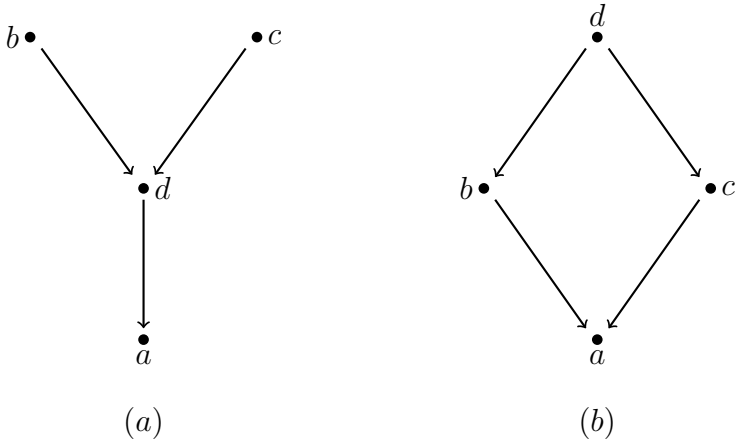


Figure 5.3: The two possible 1-defect chains on the four element domain.

5.5 1-defect chains

Finally we consider a multimorphism introduced by Jonsson, Kuivinen and Thapper [38] that was required in the four element dichotomy for Min CSP. They refer to it as a 1-defect chain multimorphism as it has a single pair of incomparible elements in its domain.

Definition 39 ([38])**.** *Let b and c be distinct elements on D. Let $(D;<)$ be a partial order that relates all pairs of domain elements except b and c. We call $\langle f, g\rangle$ a 1-defect multimorphism if $f, g : D^2 \to D$ are both commutative functions and satisfy the following conditions:*

- *If $\{x, y\} \neq \{b, c\}$, then $f(x, y) = Min(x, y)$ and $g(x, y) = Max(x, y)$*
- *If $\{x, y\} = \{b, c\}$, then $\{f(x, y), g(x, y)\} \cap \{x, y\} = \emptyset$, and $f(x, y) < g(x, y)$.*

The tractability of languages admitting a 1-defect chain multimorphism was proven in [38] in both the VCSP and Min CSP frameworks. Furthermore 1-defect chains have also been shown to be tractable by BLP (Example 6 of [57]).

The two examples of 1-defect chains on four element domains are shown in Figure 5.3. The diamond (digraph (b)) we have already seen in Figure 5.2 as an example of generalised submodularity. Digraph (a) on the other hand describes a new tracable class that was required to classify Min CSP over four elements. In [38] the authors

claim this digraph describes the multimorphism $\langle f, g \rangle$ where:

$$f: \begin{array}{cccc} a & a & a & a \\ a & b & a & d \\ a & a & c & d \\ a & d & d & d \end{array} \qquad g: \begin{array}{cccc} a & b & c & d \\ b & b & d & b \\ c & d & c & c \\ d & b & c & d \end{array}$$

Neither of these operations is familiar from previous work. The operation f is almost the greatest lower bound, except for $f(b, c) = a$, which would return d if f was the greatest lower bound operation. Finally a question - if we ignore the vertex a this digraph would describe α-bisubmodularity under the new definition of drawing fractional polymorphisms as digraphs. Therefore is it also possible to skew this multimorphism to obtain something more general in which this 1-defect chain occurs as a special case?

5.6 Generalising 1-defect chains

Let us apply our fractional polymorphism definition to digraph (a) of Figure 5.3. It fits the definition of being a connected directed acyclic graph based on a partial order of its domain elements ($b > d > a$ and $c > d > a$ using the example from Figure 5.3). Every pair of vertices has a greatest lower bound, but the pair (b, c) do not have a least upper bound, so we obtain the following functions:

$$s(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x} \wedge \mathbf{y})$$

$$t(\mathbf{x}, \mathbf{y}) = w_a^t \phi(\mathbf{x} \vee_a \mathbf{y}) + w_b^t \phi(\mathbf{x} \vee_b \mathbf{y}) + w_c^t \phi(\mathbf{x} \vee_c \mathbf{y}) + w_d^t \phi(\mathbf{x} \vee_d \mathbf{y})$$

Giving us the fractional polymorphism inequality:

$$\phi(\mathbf{x}) + \phi(\mathbf{y}) \geq \phi(\mathbf{x} \wedge \mathbf{y}) + w_a^t \phi(\mathbf{x} \vee_a \mathbf{y}) + w_b^t \phi(\mathbf{x} \vee_b \mathbf{y}) + w_c^t \phi(\mathbf{x} \vee_c \mathbf{y}) + w_d^t \phi(\mathbf{x} \vee_d \mathbf{y})$$

for all tuples $\mathbf{x}, \mathbf{y} \in D^k$, where k is the arity of ϕ.

Substitute $\mathbf{x}$ with $\mathbf{x} \vee_b \mathbf{y}$ and $\mathbf{y}$ with $\mathbf{x} \vee_c \mathbf{y}$ and apply the fractional polymorphism to itself. Simplifying the inequality gives:

$$\phi(\mathbf{x} \vee_b \mathbf{y}) + \phi(\mathbf{x} \vee_c \mathbf{y}) \geq w_a^t \phi(\mathbf{x} \vee_a \mathbf{y}) + w_b^t \phi(\mathbf{x} \vee_b \mathbf{y}) + w_c^t \phi(\mathbf{x} \vee_c \mathbf{y}) + (1 + w_d^t) \phi(\mathbf{x} \vee_d \mathbf{y})$$

Without loss of generality assume $w_b^t \geq w_c^t$, so $w_b^t = w_c^t + \epsilon$ where $\epsilon = w_b^t - w_c^t$. Now we can rearrange our inequality:

$$(1 - w_c^t)\phi(\mathbf{x} \vee_b \mathbf{y}) + (1 - w_c^t)\phi(\mathbf{x} \vee_c \mathbf{y}) \geq w_a^t \phi(\mathbf{x} \vee_a \mathbf{y}) + \epsilon\phi(\mathbf{x} \vee_b \mathbf{y}) + (1 + w_d^t)\phi(\mathbf{x} \vee_d \mathbf{y})$$

Scaling this, substituting back into our fractional polymorphism inequality, and simplifying the coefficients we obtain:

$$\phi(\mathbf{x}) + \phi(\mathbf{y}) \geq \phi(\mathbf{x} \wedge \mathbf{y}) + \frac{w_a^t}{1 - w_c^t}\phi(\mathbf{x} \vee_a \mathbf{y}) + \frac{w_b^t - w_c^t}{1 - w_c^t}\phi(\mathbf{x} \vee_b \mathbf{y}) + \frac{w_d^t + w_c^t}{1 - w_c^t}\phi(\mathbf{x} \vee_d \mathbf{y})$$

for all tuples $\mathbf{x}, \mathbf{y} \in D^k$, where k is the arity of ϕ. Let $\frac{w_a^t}{1-w_c^t} = \alpha$, and $\frac{w_d^t + w_c^t}{1-w_c^t} = \beta$, then $\frac{w_b^t - w_c^t}{1-w_c^t} = 1 - \alpha - \beta$. We call this class of fractional polymorphisms $\{\alpha, \beta\}$-1-defects.

Definition 40. *Let $D = \{a, b, c, d\}$ and let $(D;>)$ be a partial order that relates all pairs of domain elements except b and c such that $b > d > a$ and $c > d > a$. A valued constraint language Γ exhibits an $\{\alpha, \beta\}$-1-defect fractional polymorphism (towards b) if for all cost functions $\phi \in \Gamma$, of arity k, the following holds:*

$$\phi(\boldsymbol{x}) + \phi(\boldsymbol{y}) \geq \phi(\boldsymbol{x} \wedge \boldsymbol{y}) + \alpha\phi(\boldsymbol{x} \vee_a \boldsymbol{y}) + \beta\phi(\boldsymbol{x} \vee_d \boldsymbol{y}) + (1 - \alpha - \beta)\phi(\boldsymbol{x} \vee_b \boldsymbol{y})$$

for all tuples $\boldsymbol{x}, \boldsymbol{y} \in D^k$.

Note that α, β-1-defects can be be skewed towards c instead of b by replacing the operation $\vee_b$ with $\vee_c$. For simplicity we will assume that $\{\alpha, \beta\}$-1-defects are skewed towards b, unless explicitly stated otherwise.

Valued constraint languages that exhibit an $\{\alpha, \beta\}$-1-defect fractional polymorphism can be minimised in polynomial time by BLP. In the case of α-bisubmodularity taking the value $\alpha = 1$ instantly described the special case of bisubmodularity, but we cannot do that in this case to obtain the 1-defect chain identified in [38].

Proposition 8. *The 1-defect chain multimorphism on the four element domain is the special case {1,0}-1-defect fractional polymorphism.*

Proof. Consider the {1,0}-1-defect fractional polymorphism. All cost functions that exhibit this fractional polymorphism satisfy the following inequality:

$$\phi(\mathbf{x}) + \phi(\mathbf{y}) \geq \phi(\mathbf{x} \wedge \mathbf{y}) + \phi(\mathbf{x} \vee_a \mathbf{y})$$

for all tuples $\mathbf{x}, \mathbf{y} \in D^k$, where k is the arity of ϕ, where:

$$\wedge : \begin{array}{cccc} a & a & a & a \\ a & b & d & d \\ a & d & c & d \\ a & d & d & d \end{array} \qquad \vee_a : \begin{array}{cccc} a & b & c & d \\ b & b & a & b \\ c & a & c & c \\ d & b & c & d \end{array}$$

Compare this to the original 1-defect chain multimorphism $\langle f, g \rangle$ given in [38], where:

$$f : \begin{array}{cccc} a & a & a & a \\ a & b & a & d \\ a & a & c & d \\ a & d & d & d \end{array} \qquad g : \begin{array}{cccc} a & b & c & d \\ b & b & d & b \\ c & d & c & c \\ d & b & c & d \end{array}$$

The only difference is what element the functions return with the input {b,c}. It is simple to prove however that applying the multimorphism $\langle \wedge, \vee_a \rangle$ twice returns the same as $\langle f, g \rangle$.

Substitute $\mathbf{x}$ with $\mathbf{x} \wedge \mathbf{y}$ and $\mathbf{y}$ with $\mathbf{x} \vee_a \mathbf{y}$ and apply the multimorphism $\langle \wedge, \vee_a \rangle$ to obtain:

$$\phi(\mathbf{x} \wedge \mathbf{y}) + \phi(\mathbf{x} \vee_a \mathbf{y}) \geq \phi((\mathbf{x} \wedge \mathbf{y}) \wedge (\mathbf{x} \vee_a \mathbf{y})) + \phi((\mathbf{x} \wedge \mathbf{y}) \vee_a (\mathbf{x} \vee_a \mathbf{y}))$$

It is a simple exercise to confirm that $\phi((\mathbf{x} \wedge \mathbf{y}) \wedge (\mathbf{x} \vee_a \mathbf{y})) = \phi(f(\mathbf{x}, \mathbf{y}))$ and likewise that $\phi((\mathbf{x} \wedge \mathbf{y}) \vee_a (\mathbf{x} \vee_a \mathbf{y})) = \phi(g(\mathbf{x}, \mathbf{y}))$. Thus substituting back into our original fractional polymorphism inequality we obtain:

$$\phi(\mathbf{x}) + \phi(\mathbf{y}) \geq \phi(f(\mathbf{x}, \mathbf{y})) + \phi(g(\mathbf{x}, \mathbf{y}))$$

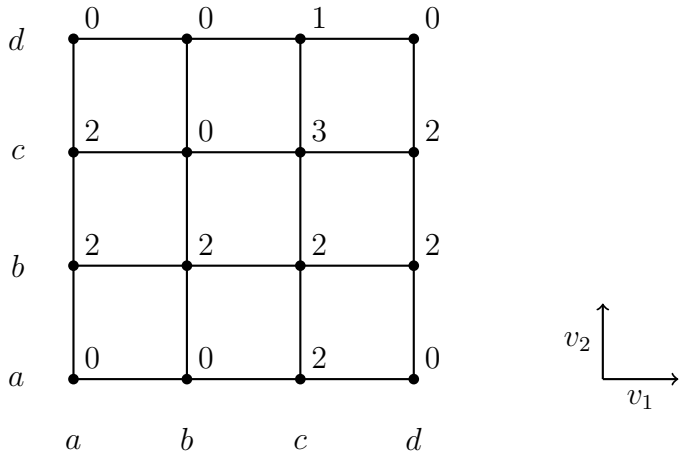
Therefore if a valued constraint language Γ exhibits a {1,0}-1-defect chain fractional polymorphism it also exhibits the 1-defect chain multimorphism $\langle f, g \rangle$, thus proving the 1-defect chain is a special case of $\{\alpha, \beta\}$-1-defect fractional polymorphism. □

5.6.1 An example

It is trivial to see that constraint languages with $\{\alpha, \beta\}$-1-defect fractional polymorphisms are tractable, but a more difficult question to answer is are they necessary in building a dichotomy in the four element case?

Here we present an example of a valued constraint language that satisfies the definition of having an $\{\alpha, \beta\}$-1-defect fractional polymorphism but does not exhibit generalised submodularity or the 1-defect chain $\langle f, g \rangle$.

Example 22. *Let Γ be the valued constraint language over $D = \{a, b, c, d\}$, that consists of the four unary functions $u_1 = [0, 2, 1, 1]$, $u_2 = [2, 0, 3, 1]$, $u_3 = [1, 3, 0, 1]$ and $u_4 = [1, 2, 1, 0]$ and the binary function $h(v_1, v_2)$ defined below:*



It is simple to check that Γ exhibits a $\{\frac{1}{2}, 0\}$-1-defect fractional polymorphism with defect $\{b, c\}$ such that $b > d > a$ and $c > d > a$.

The binary function $h(v_1, v_2)$ is not submodular under any domain ordering. With this ordering we see that:

$$h(b, c) + h(c, b) \not\geq h(b, b) + h(c, c)$$

The binary function also excludes a 1-defect chain multimorphism under any domain ordering. With this ordering we see that:

$$h(b, c) + h(c, c) \not\geq h(a, c) + h(d, c)$$

It is trivial to check this constraint language excludes submodularity and 1-defect chain multimorphisms under any domain ordering as the majority of orderings are forbidden by the unary functions.

Therefore this example demonstrates a tractable core valued constraint language that does not exhibit any other known fractional polymorphism over four elements, and

thus adds weight to the argument that this class of fractional polymorphisms may well be necessary in constructing a complexity dichotomy on the four element domain.

Note that the $\{\alpha, \beta\}$-1-defect fractional polymorphism exhibited by the constraint language in Example 22 is not unique. It also exhibits the $\{0, \frac{1}{2}\}$-1-defect fractional polymorphism, and appears to exhibit all such fractional polymorphisms where $\alpha + \beta = \frac{1}{2}$. However this is not the case in general, as demonstrated in the following example.

Example 23. *Let Γ be the valued constraint language over $D = \{a, b, c, d\}$, that consists of the four unary functions $u_1 = [0, 2, 1, 2]$, $u_2 = [2, 0, 3, 1]$, $u_3 = [1, 3, 0, 1]$ and $u_4 = [1, 2, 1, 0]$ and the binary function $h(v_1, v_2)$ as defined in the previous example.*

Γ exhibits a $\{\frac{1}{2}, 0\}$-1-defect fractional polymorphism but no other $\{\alpha, \beta\}$-1-defect fractional polymorphism. The weight of the operation $\vee_b$ is fixed at $\frac{1}{2}$ by the binary function h and the unary function u_3. The weight of the operation $\vee_a$ cannot be more than $\frac{1}{2}$ in order to balance the weights of the fractional polymorphism, and cannot be less than $\frac{1}{2}$ due to the unary function u_1.

5.7 Conjectures and Open Problems

As we know there is no complexity dichotomy on the four element domain that identifies the exact fractional polymorphisms necessary for tractability. In this section we offer a conjecture identifying the fractional polymorphisms we believe necessary to construct such a dichotomy.

In the two and three element case we identified that all the fractional polymorphisms necessary to construct the tractability side of the dichotomy could be described by connected directed acyclic graphs based on the partial order of the domain elements. Given this we can draw all the interesting digraphs on four vertices as shown in Figure 5.4 (ignoring duals).

Digraphs (a) and (b) describe submodularity on a chain and lattice respectively as we have seen numerous times. Digraph (c) represents the $\{\alpha, \beta\}$-1-defect fractional polymorphism developed earlier in this chapter. Digraph (e) describes a fractional

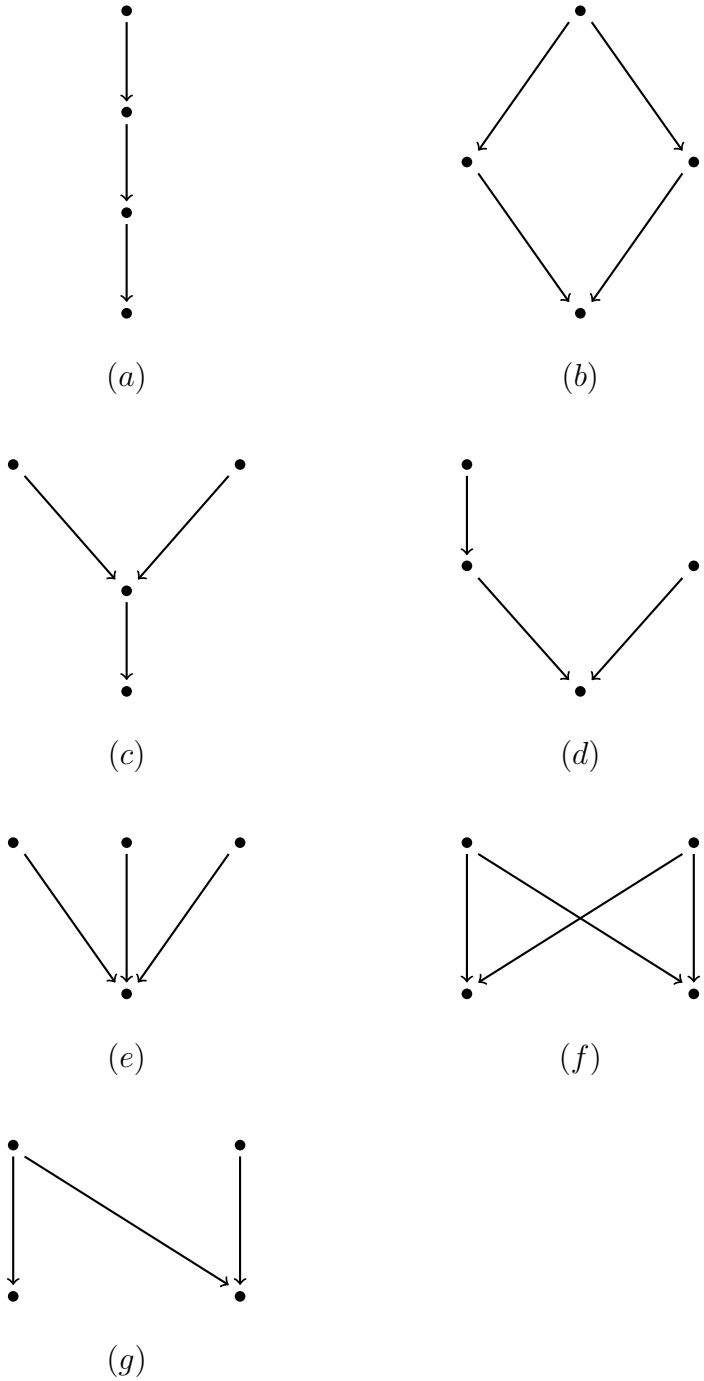


Figure 5.4: Seven digraphs describing fractional polymorphisms on four element domains.

polymorphism which has k-submodularity [30] (3-submodularity on the four element domain) as a special case. Future work would include generalising 3-submodularity to allow for skewness in the same way as bisubmodularity on the three element domain. It should also be noted that digraph (e) has 3 defects (pairs of incomparable domain elements), and this is the most possible given Lemma 3. The removal of any edge from (e) disconnects the digraph and would imply the underlying constraint language is not core.

Digraphs (d), (f) and (g) are all yet to be studied but appear to be interesting for different reasons. Digraph (d) could be described as a 2-defect chain, and can probably allow for skewness, although it is unclear which domain elements it would be possible to skew towards. Digraphs (f) and (g) on the other hand both have one pair of domain elements that have a greatest lower bound but no least upper bound, and one pair of domain elements that have a least upper bound but no greatest lower bound. Therefore both of these digraphs will describe a fractional polymorphism that includes all eight operations $\wedge_w, \wedge_x, \wedge_y, \wedge_z, \vee_w, \vee_x, \vee_y$ and $\vee_z$ in its support.

Despite significant computational searching we are yet to find an example of a core valued constraint language with a fractional polymorphism described by digraph (g) that has no other fractional polymorphism described by digraphs (a)-(e). However we cannot offer a full proof that the fractional polymorphisms described by digraphs (f) and (g) are unnecessary in building a dichotomy on the four element domain. With the lack of an obvious counter-example we offer the following conjecture.

Conjecture 3. *Let* Γ *be a core finite-valued constraint language over a four element domain. If* Γ *exhibits a fractional polymorphisms described by any of the digraphs (a)-(e) in Figure 5.4 then VCSP*(Γ) *is polynomial time solvable. Otherwise VCSP*(Γ) *is* **NP***-hard.*

If this conjecture is proven true it means the necessary fractional polymorphisms on domains of sizes two, three and four all display the property of having the greatest lower bound for all pairs of their domain elements. In fact, all known classification results of finite-valued constraint languages have a necessary condition for tractabil-

ity that requires all pairs of domain elements to have a greatest lower bound. These include the following (identified as tractable by BLP in [36]):

- Core {0,1}-valued languages over a two element domain [20, 40], a three element domain [37], or including all unary {0,1}-valued functions [21]. Submodularity on a chain is the necessary condition for tractability, hence we have the greatest lower bound for all pairs of domain elements.
- Core {0,1}-valued languages over a four element domain [38] are tractable if they are submodular on a lattice or exhibit a 1-defect chain. Submodularity on a lattice has a greatest lower bound for all pairs of domain elements, and we have shown the 1-defect chain multimorphism is a special case of the $\{\alpha, \beta\}$-1-defect fractional polymorphism, which has the greatest lower bound in its support.
- Core finite-valued languages over a two element domain [17] are tractable if they are submodular, and hence the domain elements have a greatest lower bound.
- Core finite-valued languages over a three element domain [31] are tractable if they are submodular on a chain, or skew bisubmodular. Both of these conditions have the greatest lower bound in their support.
- Finite-valued languages containing all {0,1}-valued unary cost functions [44] are tractable if they are submodular on a chain, and hence all pairs of domain elements have a greatest lower bound.

An interesting problem to study based on this would be the following:

Problem 1. *Let* Γ *be a core finite-valued constraint language over a finite domain. Is it true that VCSP*(Γ) *is polynomial time solvable only when the domain elements exhibit a partial order such that all pairs of domain elements have a greatest lower bound, and hence the fractional polymorphisms of* Γ *have the greatest lower bound in their support, and otherwise VCSP*(Γ) *is* **NP***-hard?*

If this problem can be proven true we will have found a tighter condition that captures all necessary tractable finite-valued constraint languages than that given

in [59], which states that all tractable finite-valued constraint languages exhibit a binary symmetric fractional polymorphism.

Furthermore we would be able to strengthen Definition 37.

Definition 41. *Let Γ be a rigid core finite-valued constraint language over a finite domain D. Let $\mathbb{G}$ be the connected directed acyclic graph whose vertices are the elements of D and whose arcs describe a partial order of the elements of D, such that every pair of domain elements has a greatest lower bound.*

For any digraph $\mathbb{G}$ we say Γ exhibits the fractional polymorphism described by $\mathbb{G}$ if every cost function $\phi \in \Gamma$, of arity k, satisfies the inequality:

$$\phi(\boldsymbol{x}) + \phi(\boldsymbol{y}) \geq \phi(\boldsymbol{x} \wedge \boldsymbol{y}) + t(\boldsymbol{x}, \boldsymbol{y})$$

for all tuples $\boldsymbol{x}, \boldsymbol{y} \in D^k$, where we define $t(\boldsymbol{x}, \boldsymbol{y})$ as:

$$t(\boldsymbol{x}, \boldsymbol{y}) = \sum_{c \in D} w_c^t \phi(\boldsymbol{x} \vee_c \boldsymbol{y}) \quad \textit{and} \quad \sum_{c \in D} w_c^t = 1.$$

This definition ensures that there is a single vertex at the base of the digraph (if you have two vertices at the base they will not have a greatest lower bound as in digraphs (f) and (g) of Figure 5.4). Hence this definition also captures the tree submodular functions defined by Kolmogorov [41]. It is shown in [41] that strong tree-submodularity implies weak tree-submodularity which is defined as follows.

Definition 42 ([41])**.** *A function f, of arity k, is weakly tree-submodular over a domain D if it satisfies:*

$$f(\boldsymbol{x}) + f(\boldsymbol{y}) \geq f(\boldsymbol{x} \wedge \boldsymbol{y}) + f(\boldsymbol{x} \vee \boldsymbol{y})$$

for all tuples $\boldsymbol{x}, \boldsymbol{y} \in D^k$, where $a \wedge b$ is the greatest lower bound of a and b and $a \vee b$ is the unique node on the path from a to b such that the distance from a to $a \vee b$ is the same as the distance from b to $a \wedge b$.

It is easy to see that the weakly tree-submodular fractional polymorphism is a special case of Definition 41.

It should also be noted that fractional polymorphisms of the form described in Definition 41 bear a striking resemblance to the notion of submodularity with fractional

joins introduced by Hirai [29]. It is stated in [29] that submodularity with fractional joins captures all the well known tractable cases we have seen throughout this thesis: submodularity, bisubmodularity, submodularity on trees, skew-bisubmodularity and k-submodularity. An interesting problem would be to check if $\{\alpha, \beta\}$-1-defect fractional polymorphisms are captured, and hence the 1-defect chains of [38]. It would also be interesting to determine how Definition 41 and submodularity with fractional joins are connected, and if they capture exactly the same fractional polymorphisms.

Bibliography

[1] K. Ando, S. Fujishige, and T. Naitoh, "A characterization of bisubmodular functions," *Discrete Mathematics*, vol. 148, no. 13, pp. 299 – 303, 1996.

[2] B. Aspvall, M. Plass, and R. Tarjan, "A linear-time algorithm for testing the truth of certain quantified boolean formulas," *Information Processing Letters*, vol. 8, no. 3, pp. 121–123, 1979.

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

[4] ——, "Constraint Satisfaction Problems Solvable by Local Consistency Methods," *Journal of the ACM*, vol. 61(1), 2014.

[5] R. Beigel and D. Eppstein, "3-coloring in time o(1.3289n)," *J. Algorithms*, vol. 54, no. 2, pp. 168–204, 2005.

[6] J. Bell and B. Stevens, "A survey of known results and research areas for n-queens," *Discrete Mathematics*, vol. 309, no. 1, pp. 1–31, 2009.

[7] A. Bulatov, "A Dichotomy Theorem for Constraint Satisfaction Problems on a 3-Element Set," *Journal of the ACM*, vol. 53(1), pp. 66–120, 2006.

[8] ——, "Bounded relational width," Unpublished manuscript, 2009.

[9] ——, "Complexity of conservative constraint satisfaction problems," *ACM Transactions on Computational Logic*, vol. 12(4), 2011.

[10] A. Bulatov, P. Jeavons, and A. Krokhin, "Classifying the Complexity of Constraints Using Finite Algebras," *SIAM Journal on Computing*, vol. 34(3), pp. 720–742, 2005.

[11] J. Bulín, D. Delic, M. Jackson, and T. Niven, "On the reduction of the CSP dichotomy conjecture to digraphs," in *CP'13*, vol. 8124 of LNCS, 2013, pp. 184–199.

[12] ——, "A finer reduction of constraint problems to digraphs," Technical report, arXiv:1406.6413, 2014.

[13] H. Chen, "A rendezvous of logic, complexity, and algebra," *ACM Computing Surveys*, vol. 42, no. 1, pp. 2:1–2:32, 2009.

[14] D. Cohen, M. Cooper, P. Creed, P. Jeavons, and S. Živný, "An Algebraic Theory of Complexity for Discrete Optimisation," *SIAM Journal on Computing*, vol. 42(5), pp. 1915–1939, 2013.

[15] D. Cohen, M. Cooper, and P. Jeavons, "An Algebraic Characterisation of Complexity for Valued Constraints," in *CP'06*, vol. 4204 of LNCS, 2006, pp. 107–121.

[16] ——, "Generalising submodularity and horn clauses: Tractable optimization problems defined by tournament pair multimorphisms," *Theoretical Computer Science*, vol. 401, no. 1-3, pp. 36–51, 2008.

[17] D. Cohen, M. Cooper, P. Jeavons, and A. Krokhin, "The complexity of soft constraint satisfaction," *Artificial Intelligence*, vol. 170(11), pp. 983–1016, 2006.

[18] D. Cohen, M. Cooper, P. Jeavons, and S. Živný, "Binarisation via dualisation for valued constraints," in *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*, 2015, pp. 3731–3737.

[19] D. Cohen, P. Creed, P. Jeavons, and S. Živný, "An algebraic theory of complexity for valued constraints: Establishing a Galois connection," Research Report CS-RR-10-16, Computing Laboratory, University of Oxford, 2010.

[20] N. Creignou, S. Khanna, and M. Sudan, "Complexity classification of boolean constraint satisfaction problems," *SIAM Monographs on Discrete Mathematics and Applications*, vol. 7, 2001.

[21] V. Deineko, P. Jonsson, M. Klasson, and A. Krokhin, "The approximability of Max CSP with fixed-value constraints," *Journal of the ACM*, vol. 55, no. 4, 2008.

[22] S. Even, A. Itai, and A. Shamir, "On the complexity of timetable and multicommodity flow problems," *SIAM Journal on Computing*, vol. 5, no. 4, pp. 691–703, 1976.

[23] 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*, vol. 28, pp. 57–104, 1998.

[24] S. Fujishige, "Submodular Functions and Optimization," volume 58 of Annals of Discrete Mathematics, Elsevier, 2nd edition, 2005.

[25] G. Gutin, P. Hell, A. Rafiey, and A. Yeo, "A dichotomy for minimum cost graph homomorphisms," *European Journal of Combinatorics*, vol. 29(4), pp. 900–911, 2008.

[26] P. Hell and J. Nešetřil, *Graphs and Homomorphisms*. Oxford University Press, 2004.

[27] ——, "Colouring, Constraint Satisfaction, and Complexity," *Computer Science Review*, vol. 2(3), pp. 143–163, 2008.

[28] P. Hell and A. Rafiey, "The Dichotomy of Minimum Cost Homomorphism Problems for Digraphs," *SIAM Journal on Discrete Mathematics*, vol. 26(4), pp. 1597–1608, 2012.

[29] H. Hirai, "Discrete convexity and polynomial solvability in minimum 0-extension problems," *SODA'13*, pp. 1770–1778, 2013.

[30] A. Huber and V. Kolmogorov, "Towards Minimising k-Submodular Functions," arXiv:1309.5469, 2013.

[31] A. Huber, A. Krokhin, and R. Powell, "Skew Bisubmodularity and Valued CSPs," *SIAM Journal on Computing*, vol. 43(3), pp. 1064–1084, 2014.

[32] S. Iwata, "Submodular function minimization," *Math. Program.*, vol. 112, no. 1, pp. 45–64, 2007.

[33] S. Iwata, L. Fleischer, and S. Fujishige, "A combinatorial strongly polynomial algorithm for minimizing submodular functions," *J. ACM*, vol. 48, no. 4, pp. 761–777, 2001.

[34] S. Iwata and J. B. Orlin, "A simple combinatorial algorithm for submodular function minimization," in *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '09, 2009, pp. 1230–1237.

[35] P. Jeavons, "On the algebraic structure of combinatorial problems," *Theoretical Computer Science*, vol. 200, no. 1-2, pp. 185–204, 1998.

[36] P. Jeavons, A. Krokhin, and S. Živný, "The Complexity of Valued Constraint Satisfaction," in *Bulletin of the BEATCS*, vol. 113, 2014, pp. 21–55.

[37] P. Jonsson, M. Klasson, and A. Krokhin, "The Approximability of Three-valued MAX CSP," *SIAM Journal on Computing*, vol. 35(6), pp. 1329–1349, 2006.

[38] P. Jonsson, F. Kuivinen, and J. Thapper, "Min CSP on Four Elements: Moving Beyond Submodularity," in *CP'11*, vol. 6876 of LNCS, 2011, pp. 438–453.

[39] R. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, ser. The IBM Research Symposia Series. Springer, 1972, pp. 85–103.

[40] S. Khanna, M. Sudan, L. Trevisan, and D. Williamson, "The approximability of constraint satisfaction problems," *SIAM Journal on Computing*, vol. 30, no. 6, pp. 1863–1920, 2001.

[41] K. Kolmogorov, "Submodularity on a tree: Unifying $L^{\#}$-convex and bisubmodular functions," *MFCS'11*, vol. 6907 of LNCS, pp. 400–411, 2011.

[42] V. Kolmogorov, "The Power of Linear Programming for Finite-valued CSPs: A Constructive Characterization," in *Proceedings of the 40th International Colloquium on Automata, Languages, and Programming (ICALP'13)*, 2013, pp. 625–636.

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

[44] V. Kolmogorov and S. Živný, "The complexity of conservative valued CSPs," *Journal of the ACM*, vol. 60, no. 2, pp. 10:1–10:38, 2013.

[45] M. Kozik and J. Ochremiak, "Algebraic Properties of Valued Constraint Satisfaction Problems," in *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP'15)*, vol. 9134 of LNCS, 2015, pp. 846–858.

[46] M. R. Krom, "The decision problem for a class of first-order formulas in which all disjunctions are binary," *Mathematical Logic Quarterly*, vol. 13, no. 1-2, pp. 15–20, 1967.

[47] R. Ladner, "On the structure of polynomial time reducibility," *Journal of the ACM*, vol. 22, no. 1, pp. 155–171, 1975.

[48] L. Lovász, "Submodular functions and convexity," in *Mathematical Programming The State of the Art*, A. Bachem, B. Korte, and M. Grötschel, Eds. Springer, 1983, pp. 235–257.

[49] M. Maróti and R. McKenzie, "Existence theorems for weakly symmetric operations," *Algebra Universalis*, vol. 59(3-4), pp. 463–489, 2008.

[50] S. McCormick and S. Fujishige, "Strongly Polynomial and Fully Combinatorial Algorithms for Bisubmodular Function Minimization," *Mathematical Programming*, vol. 122, pp. 87–120, 2009.

[51] R. Powell and A. Krokhin, "A Reduction from Valued CSP to Min Cost Homomorphism Problem for Digraphs," Technical report, arXiv:1507.01776, 2015.

[52] L. Qi, "Directed Submodularity, Ditroids and Directed Submodular Flows," *Mathematical Programming*, vol. 42(3), pp. 579–599, 1988.

[53] T. Schaefer, "The complexity of satisfiability problems," in *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing (STOC '78)*, 1978, pp. 216–226.

[54] A. Schrijver, "Combinatorial optimization: Polyhedra and Efficiency," volume 24 of Algorithms and Combinatorics, Springer, 2003.

[55] R. Takhanov, "A dichotomy theorem for the general minimum cost homomorphism problem," in *In Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS10)*, 2010, pp. 657–668.

[56] ——, "Extensions of the minimum cost homomorphism problem," in *Proceedings of the 16th Annual International Conference on Computing and Combinatorics (COCOON'10)*, 2010, pp. 328–337.

[57] J. Thapper and S. Živný, "The power of linear programming for valued CSPs," in *53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS'12)*, 2012, pp. 669–678.

[58] ——, "The power of linear programming for valued CSPs," arXiv:1204.1079, 2012.

[59] ——, "The complexity of finite-valued CSPs," in *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing (STOC'13)*, 2013, pp. 695–704.

[60] ——, "Sherali-Adams relaxations for valued CSPs," in *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP'15)*, vol. 9134 of LNCS, 2015, pp. 1058–1069.

[61] D. Topkis, "Minimizing a submodular function on a lattice," *Operations Research*, vol. 26, no. 2, pp. 305–321, 1978.

[62] 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)*, 2013, pp. 804–815.

[63] ——, "Computational complexity of the extended minimum cost homomorphism problem on three-element domains," in *Proceedings of the 31st International Symposium on Theoretical Aspects of Computer Science (STACS'14)*, 2014, pp. 651–662.

[64] S. Živný, "The Complexity and Expressive Power of Valued Constraints," PhD Thesis, University of Oxford, 2009.

Index

Γ, 20

$\Phi_D^{(m)}$, 20

Φ_D, 20

$\mathbb{Q}_S$, 63

α-Bisubmodularity, 41

$\langle\Gamma\rangle$, 24

$\langle\vee,\wedge\rangle$, 22

$\langle\vee_0,\wedge_0\rangle$, 22

$\vee$, 17

$\vee_0$, 22

$\wedge$, 17

$\wedge_0$, 22

$w\mathcal{A}$, 60

$w\mathcal{X}$, 60

$\{\alpha,\beta\}$-1-defect chains, 93

1/2-bisubmodular, 54

2-SAT, 13

3-SAT, 13

Algebraic CSP dichotomy conjecture, 73

Assignment, 11, 19

Bisubmodularity, 22, 38

BLP, 28, 46

Colouring, 8, 11, 59

Computational Complexity, 10

- **NP**, 10
- **P**, 10

Constraint, 10

- Crisp, 20
- Valued, 20

Cost function, 20

CSP, 10

- 2-element, 17
- 3-element, 17
- Bounded width, 73
- Complexity, 14
- Constraint language, 12
- Core, 18
- Dichotomy, 14, 17
- Expressibility, 16
- Graph homomorphism, 59

Cyclic, 24, 73

Digraph, 59, 61

- Balanced, 61
- Equality graph, 68
- Oriented path, 61
- Short component, 66, 78

Domain, 10

Dual, 27

Feas, 21, 60

Fractional polymorphism, 23

- fPol, 24
- supp, 24

Homomorphism, 59

- CSP, 59
- VCSP, 60

Idempotent, 24

Identity, 74
  Balanced, 74
  Idempotent, 74
  Linear, 74

k-submodularity, 90

Max CSP, 19
MAX CUT, 20, 25, 45, 80
Min CSP, 91
Min-Ones, 19
MinCostHom, 58
Multimorphism, 21
  1-defect chain, 91
  Symmetric binary, 36
  Binary, 22
  Digraph, 36
  Symmetric tournament pair (STP), 26, 38, 50
  Ternary, 22
  Tournament pair, 26
  Unary, 21

n-queens, 11

Optimisation, 19, 20, 28, 32, 33, 60, 65, 68
Orthant, 42
  Submodular, 42

Polymorphism, 14
  Majority, 15
  Minority, 15
  Near-unanimity, 18
  Taylor, 18
  Weak near-unanimity (WNU), 18, 73

Relation, 10
Rigid core, 35, 62

Semilattice, 29
Submodularity, 22, 28, 38
Symmetric, 73

Tractability, 25

Valued Constraint Language, 20
  Conservative, 26, 50
VCSP, 19
  0-1-valued, 45
  2-element, 25, 36, 45
  3-element, 45
  4-element, 89
  4-element dichotomy conjecture, 96
  Core, 32
  Dichotomy, 25, 36, 45
  Expressibility, 24
  Graph homomorphism, 60
  Instance, 20
  Rigid core, 35
  Valued constraint language, 20

Weakly tree-submodular, 100
Weighted τ-structure, 60
Weighted relation, 60