An Algebraic Characterisation of Complexity for Valued Constraints

David A. Cohen[1], Martin C. Cooper[2], and Peter G. Jeavons[3]

[1] Department of Computer Science, Royal Holloway, University of London, UK
d.cohen@rhul.ac.uk
[2] IRIT, University of Toulouse III, France
cooper@irit.fr
[3] Computing Laboratory, University of Oxford, UK
peter.jeavons@comlab.ox.ac.uk

Abstract. Classical constraint satisfaction is concerned with the feasibility of satisfying a collection of constraints. The extension of this framework to include optimisation is now also being investigated and a theory of so-called soft constraints is being developed. In this extended framework, tuples of values allowed by constraints are given desirability weightings, or costs, and the goal is to find the most desirable (or least cost) assignment.
The complexity of any optimisation problem depends critically on the type of function which has to be minimized. For soft constraint problems this function is a sum of cost functions chosen from some fixed set of available cost functions, known as a valued constraint language. We show in this paper that when the costs are rational numbers or infinite the complexity of a soft constraint problem is determined by certain algebraic properties of the valued constraint language, which we call feasibility polymorphisms and fractional polymorphisms.
As an immediate application of these results, we show that the existence of a non-trivial fractional polymorphism is a necessary condition for the tractability of a valued constraint language with rational or infinite costs over any finite domain (assuming P $\neq$ NP).

1 Introduction

Classical constraint satisfaction is concerned with the feasibility of satisfying a collection of constraints. The extension of this framework to include optimisation is now also being investigated and a theory of so-called *soft* constraints is being developed.

Several alternative mathematical frameworks for soft constraints have been proposed in the literature, including the very general frameworks of 'semi-ring based constraints' and 'valued constraints' [2]. For simplicity, we shall adopt the valued constraint framework here. In this framework, every tuple of values allowed by a constraint has an associated *cost*, and the goal is to find an assignment with minimal total cost.

The general constraint satisfaction problem (CSP) is NP-hard, and so is unlikely to have a polynomial-time algorithm. However, there has been much success in finding tractable fragments of the CSP by restricting the types of relation allowed in the constraints. A set of allowed relations has been called a *constraint language*. For some constraint languages the associated constraint satisfaction problems with constraints chosen from that language are solvable in polynomial-time, whilst for other constraint languages this class of problems is NP-hard [21]; these are referred to as *tractable languages* and *NP-hard languages*, respectively. Dichotomy theorems, which classify each possible constraint language as either tractable or NP-hard, have been established for constraint languages over 2-element domains [29], and 3-element domains [3]. Considerable progress has also been made in the search for a complete characterisation of the complexity of constraint languages over finite domains of arbitrary size [4, 14, 19, 21].

The general valued constraint problem (VCSP) is also NP-hard, but again we can try to identify tractable fragments by restricting the types of allowed *cost functions* that can be used to define the valued constraints. A set of allowed cost functions has been called a *valued constraint language*. Much less is known about the complexity of the optimisation problems associated with different valued constraint languages, although some results have been obtained for certain special cases. In particular, a complete characterisation of complexity has been obtained for valued constraint languages over a 2-element domain with real-valued or infinite costs [6]. This result generalizes a number of earlier results for particular optimisation problems such as Max-Sat [11, 12].

One class of cost functions has been extensively studied: the so-called *submodular* functions. The problem of minimizing a real-valued submodular objective function occurs in many diverse application areas, including statistical physics [1], the design of electrical networks [25], and operations research [5, 33, 27]. One of the first problems to be recognized as a case of submodular function minimisation was the Max-Flow/Min-Cut problem [13]. Another class of examples arises in pure mathematics: the rank function of a matroid is always a submodular function [15]. Recently, several polynomial-time algorithms have been proposed for submodular function minimisation [17, 18, 30], although the complexity of the best of these general algorithms is $O(n^5 \min\{\log nM, n^2 \log n\})$ where M is an upper bound on the values taken by the function to be minimized [18]. More practical cubic time algorithms have been developed for many special cases [9, 24, 26], including the Max-Flow/Min-Cut problem [13], the minimization of a symmetric submodular function [28], the minimization of a $\{0, 1\}$-valued submodular function over a 2-element domain [12] and the minimization of any sum of binary submodular functions over an arbitrary finite domain [8].

The results of [12] show that submodularity is essentially the *only* property giving rise to tractable $\{0, 1\}$-valued constraint languages over a 2-element domain (see [6]). Jonsson et al. [23] recently generalized this result to 3-element domains. However, for languages allowing more general costs, or defined over larger finite domains, very little is known about the possible tractable cases.

In the classical CSP framework it has been shown that the complexity of any constraint language over any finite domain is determined by certain algebraic properties known as *polymorphisms* [19, 21]. This result has reduced the problem of the identification of tractable constraint languages to that of the identification of suitable sets of polymorphisms. In other words, it has been shown to be enough to study just those constraint languages which are characterised by having a given set of polymorphisms. This both reduces the number of different languages to be studied and allows the application of results from universal algebra to the study of the complexity of constraint languages.

To analyse the complexity of valued constraint languages in the VCSP framework we recently introduced a more general algebraic property known as a *multimorphism* [6, 7, 10]. Using this algebraic property we have shown that there are precisely eight maximal tractable valued constraint languages over a 2-element domain with real-valued or infinite costs, and each of these is characterised by having a particular form of multimorphism [6]. Furthermore, we have shown that many known maximal tractable valued constraint languages over larger finite domains are precisely characterised by a single multimorphism and that key NP-hard examples have (essentially) no multimorphisms [7].

In this paper we slightly generalise the notion of a multimorphism to that of a *fractional polymorphism*. We are then able to show that fractional polymorphisms, together with the polymorphisms of the underlying feasibility relations (which we call *feasibility polymorphims*), characterise the complexity of any valued constraint language, Γ, with non-negative rational or infinite costs over any finite domain. Specifically we show that:

- the class of all cost functions having the same fractional polymorphisms and feasibility polymorphisms as Γ corresponds precisely to the closure of Γ by three natural extension operators (one of which is *expressibility*);
- the extended class containing Γ together with all cost functions obtained using these three extension operators has the same complexity as Γ (up to polynomial-time reduction).

This very general result has the immediate corollary that a finite-valued rational cost function is expressible over a valued constraint language if and only if it has all the fractional polymorphisms of that language.

The applications of these results to the search for tractable valued constraint languages are very similar to the applications of polymorphisms to the search for tractable constraint languages in the classical CSP. First, we need only consider valued constraint languages defined by these algebraic properties. This will greatly simplify the search for a characterisation of all tractable valued constraint languages. Secondly, by showing that there exists an NP-hard valued constraint language with only finite rational costs we show that any tractable valued constraint language with finite rational or infinite costs must have a non-trivial fractional polymorphism.

Hence the results of this paper provide a powerful new set of tools in the search for a polynomial-time/NP-hard dichotomy for finite-domain optimisation problems defined by valued constraints.

2 Valued Constraint Problems

In the valued constraint framework each constraint has an associated function which assigns a *cost* to each possible assignment of values and these costs are chosen from some *valuation structure*, satisfying the following definition.

Definition 1. *A **valuation structure**, Ω, is a totally ordered set, with a minimum and a maximum element (denoted 0 and ∞), together with a commutative, associative binary **aggregation operator** (denoted $\oplus$), such that for all $\alpha, \beta, \gamma \in \Omega$, $\alpha \oplus 0 = \alpha$ and $\alpha \oplus \gamma \geq \beta \oplus \gamma$ whenever $\alpha \geq \beta$.*

Definition 2. *An instance of the **valued constraint satisfaction problem, VCSP**, is a 4-tuple $\mathcal{P} = \langle V, D, C, \Omega \rangle$ where:*

- *V is a finite set of **variables**;*
- *D is a finite set of possible **values**;*
- *Ω is a valuation structure representing possible **costs**;*
- *C is a set of **constraints**. Each element of C is a pair $c = \langle \sigma, \phi \rangle$ where σ is a tuple of variables called the **scope** of c, and ϕ is a mapping from $D^{|\sigma|}$ to Ω, called the **cost function** of c.*

Definition 3. *For any VCSP instance $\mathcal{P} = \langle V, D, C, \Omega \rangle$, an **assignment** for $\mathcal{P}$ is a mapping $s : V \to D$. The **cost** of an assignment s, denoted $Cost_{\mathcal{P}}(s)$, is given by the aggregation of the costs for the restrictions of s onto each constraint scope, that is,*

$$Cost_{\mathcal{P}}(s) \stackrel{\text{def}}{=} \bigoplus_{\langle \langle v_1, v_2, \ldots, v_m \rangle, \phi \rangle \in C} \phi(s(v_1), s(v_2), \ldots, s(v_m)).$$

*A **solution** to $\mathcal{P}$ is an assignment with minimal cost, and the question is to find a solution.*

Definition 4. *A **valued constraint language** is any set Γ of cost functions from some fixed finite domain D to some fixed valuation structure Ω.*

We define VCSP(Γ) *to be the set of all VCSP instances in which all cost functions belong to Γ.*

The *complexity* of a valued constraint language Γ will be identified with the complexity of the associated VCSP(Γ).

Definition 5. *A valued constraint language Γ is called **tractable** if, for every finite subset $\Gamma_f \subseteq \Gamma$, there exists an algorithm solving any instance $\mathcal{P} \in$ VCSP(Γ_f) in time at most $p(|\mathcal{P}|)$, for some polynomial p.*

*Conversely, Γ is called **NP-hard** if there is some finite subset $\Gamma_f \subseteq \Gamma$ for which* VCSP(Γ_f) *is NP-hard.*

For the remainder of this paper we will focus for concreteness on the valuation structure containing the non-negative rational numbers ($\mathbb{Q}_+$) together with infinity (∞), with the standard addition operation, $+$ (extended so that $a + \infty = \infty$, for all a). This valuation structure will be denoted $\overline{\mathbb{Q}}_+$. It is sufficiently general to encode many standard optimisation problems (for examples, see [7, 10]).

3 Extending a Valued Constraint Language

We now define three ways to extend a valued constraint language. In Section 5 we will show that applying these three extensions does not alter the complexity of a valued constraint language.

What is more, in Section 6 we will give a simple algebraic characterisation of the languages obtained by applying all three extensions.

The first extension makes use of a natural equivalence relation on cost functions.

Definition 6. *Two cost functions ϕ and ϕ' are said to be **cost-equivalent** if one is obtained from the other by adding a constant cost and scaling by some constant factor. In other words, ϕ and ϕ' are cost-equivalent if they have the same arity r and there exist positive integers, a, b, and a finite constant c, such that*

$$\forall x \in D^r, \quad a\,\phi'(x) \;=\; b\,\phi(x) + c.$$

For any valued constraint language Γ the language consisting of all cost functions which are cost-equivalent to some member of Γ will be denoted $\Gamma_{\equiv}$.

The next extension adds in those cost functions which are *expressible* over Γ.

Definition 7. *For any VCSP instance $\mathcal{P} = \langle V, D, C, \Omega\rangle$, and any list $L = \langle v_1, \ldots, v_r\rangle$ of variables of $\mathcal{P}$, the **projection** of $\mathcal{P}$ onto L, denoted $\pi_L(\mathcal{P})$, is the r-ary cost function defined as follows:*

$$\pi_L(\mathcal{P})(x_1, \ldots, x_r) \stackrel{\text{def}}{=} \min_{\{s:V\to D \;\mid\; \langle s(v_1),\ldots,s(v_r)\rangle = \langle x_1,\ldots,x_r\rangle\}} Cost_{\mathcal{P}}(s)$$

*We say that a cost function ϕ is **expressible** over a constraint language Γ if there exists a VCSP instance $\mathcal{P} \in \mathrm{VCSP}(\Gamma)$ and a list L of variables of $\mathcal{P}$ such that $\pi_L(\mathcal{P}) = \phi$. We call the pair $\langle \mathcal{P}, L\rangle$ a **gadget** for expressing ϕ over Γ.*

For any valued constraint language Γ the language consisting of all cost functions expressible over Γ will be denoted $\exp(\Gamma)$.

Example 1. In this example we show that Γ is always a subset of $\exp(\Gamma)$.

Let Γ be a valued constraint language over a domain D with costs in Ω. Choose any $\gamma \in \Gamma$. Let the arity of γ be r and define $V = \{x_1, \ldots, x_r\}$. The pair $\langle\langle V, D, \{\langle\langle x_1, \ldots, x_r\rangle, \gamma\rangle\}, \Omega\rangle, \langle x_1, \ldots, x_r\rangle\rangle$ is a gadget for expressing γ over Γ.

The final extension adds cost functions obtained using a *feasibility* operator.

Definition 8. *For any cost function ϕ, with arity r, we denote by* $\mathrm{Feas}(\phi)$ *the r-ary cost function defined by*

$$\mathrm{Feas}(\phi)(x_1, x_2, \ldots, x_r) \stackrel{\text{def}}{=} \begin{cases} \infty & \text{if} \quad \phi(x_1, x_2, \ldots, x_r) = \infty \\ 0 & \text{if} \quad \phi(x_1, x_2, \ldots, x_r) < \infty. \end{cases}$$

For any valued constraint language Γ the language $\{\mathrm{Feas}(\phi) \mid \phi \in \Gamma\}$ *will be denoted* $\mathrm{Feas}(\Gamma)$.

4 Polymorphisms and Fractional Polymorphisms

For classical constraint satisfaction problems it has been shown that the complexity of a constraint language is characterised by certain algebraic properties of the relations in that language, known as *polymorphisms* [4, 19, 21]. This result was obtained by showing that the expressive power of a constraint language is determined by the polymorphisms of that language.

Polymorphisms are defined for *relations* rather than cost functions, but we will now define an analogous notion which can be applied to arbitrary valued constraint languages.

Definition 9. *For any r-ary cost function ϕ, we say that $f : D^k \to D$ is a k-ary **feasibility polymorphism** of ϕ if, for all $x_1, \ldots, x_r \in D^k$,*

$$\mathrm{Feas}(\phi)(f(x_1), \ldots, f(x_r)) \;\leq\; \sum_{i=1}^{k} \mathrm{Feas}(\phi)(x_1[i], \ldots, x_r[i]).$$

For any valued constraint language Γ, we will say that f is a feasibility polymorphism of Γ if f is a feasibility polymorphism of every cost function in Γ. The set of all feasibility polymorphisms of Γ will be denoted $\mathrm{Pol}(\Gamma)$*, and the finite subset containing all k-ary feasibility polymorphisms will be denoted* $\mathrm{Pol}_k(\Gamma)$*.*

Theorem 1. *For any valued constraint language Γ, and any cost function ϕ,* $\mathrm{Feas}(\phi) \in \exp(\mathrm{Feas}(\Gamma))$ *if and only if* $\mathrm{Pol}(\Gamma) \subseteq \mathrm{Pol}(\{\phi\})$

Proof. Follows immediately from the corresponding result for classical constraint languages (see, for example, Corollary 1 of [22]).

To obtain a more precise result about the expressive power of arbitrary valued constraint languages we need to generalize the definition of a polymorphism.

Definition 10. *A k-ary **weighted function** F on a set D is a set of the form $\{\langle w_1, f_1\rangle, \ldots, \langle w_n, f_n\rangle\}$ where each w_i is a positive integer, and each f_i is a distinct function from D^k to D.*

*For any r-ary cost function ϕ, we say that a k-ary weighted function F is a k-ary **fractional polymorphism** of ϕ if, for all $x_1, \ldots, x_r \in D^k$,*

$$k \sum_{i=1}^{n} w_i \phi(f_i(x_1), \ldots, f_i(x_r)) \;\leq\; \left(\sum_{i=1}^{n} w_i\right) \cdot \left(\sum_{i=1}^{k} \phi(x_1[i], \ldots, x_r[i])\right).$$

For any valued constraint language Γ, we will say that f is a fractional polymorphism of Γ if f is a fractional polymorphism of every cost function in Γ. The set of all fractional polymorphisms of Γ will be denoted $\mathrm{fPol}(\Gamma)$*.*

In earlier papers we introduced the notion of a *multimorphism* [6, 7, 10]. A multimorphism is precisely a k-ary fractional polymorphism where the sum of the weights w_i is exactly k.

Example 2. Consider the unary fractional polymorphism given by $\{\langle 1, c\rangle\}$ for some constant $c \in D$ (seen as a unary function). An r-ary cost function ϕ has this unary fractional polymorphism when, for all $x_1, \ldots, x_r \in D$,

$$\phi(c, \ldots, c) \;\leq\; \phi(x_1, \ldots, x_r)$$

Hence, if all cost functions $\phi \in \Gamma$ have this fractional polymorphism then we can solve any instance of VCSP(Γ) trivially, by assigning the value c to each variable. In fact, it was shown in [7, 10] that the class of all cost functions with this simple fractional polymorphism is a maximal tractable class.

Example 3. An r-ary cost function ϕ on an ordered set D has the binary fractional polymorphism $\{\langle 1, \min\rangle, \langle 1, \max\rangle\}$ when, for all $x_1, \ldots, x_r \in D^2$,

$$\begin{aligned}\phi(&\min(x_1[1], x_1[2]), \ldots, \min(x_r[1], x_r[2])) \\ &+\; \phi(\max(x_1[1], x_1[2]), \ldots, \max(x_r[1], x_r[2])) \\ &\leq\; \phi(x_1[1], \ldots, x_r[1]) \;+\; \phi(x_1[2], \ldots, x_r[2])\;.\end{aligned}$$

Hence, the fractional polymorphism $\{\langle 1, \min\rangle, \langle 1, \max\rangle\}$ exactly captures the notion of *submodularity* [15] which we can thus define as follows.

Definition 11. *A cost function (over an ordered domain) is* ***submodular*** *if and only if it has the fractional polymorphism* $\{\langle 1, \min\rangle, \langle 1, \max\rangle\}$.

Recall from the Introduction that submodular function minimization is a central problem in discrete optimization. The notion of a fractional polymorphism allows us to capture the property of submodularity by using a particular weighted function. It also allows us to generalize to many other properties by considering different weighted functions.

5 Extensions Preserve Tractability

In this section we will show that the three extensions defined in Section 3 all preserve the tractability or NP-hardness of a valued constraint language.

Theorem 2. *For any valued constraint language* Γ*, we have:*

1. $\Gamma_{\equiv}$ *is tractable if and only if* Γ *is tractable, and* $\Gamma_{\equiv}$ *is NP-hard if and only if* Γ *is NP-hard.*
2. $\exp(\Gamma)$ *is tractable if and only if* Γ *is tractable, and* $\exp(\Gamma)$ *is NP-hard if and only if* Γ *is NP-hard.*
3. $\Gamma \cup \mathrm{Feas}(\Gamma)$ *is tractable if and only if* Γ *is tractable, and* $\Gamma \cup \mathrm{Feas}(\Gamma)$ *is NP-hard if and only if* Γ *is NP-hard.*

Proof. For part (1), by Definition 5, it is sufficient to show that for any finite subset Γ' of $\Gamma_{\equiv}$ there exists a polynomial-time reduction from VCSP(Γ') to VCSP(Γ''), where Γ'' is a finite subset of Γ.

Let Γ' be a finite subset of $\Gamma_{\equiv}$ and let $\mathcal{P}'$ be any instance of VCSP(Γ'). By Definition 6, any cost function $\phi' \in \Gamma_{\equiv}$ is cost-equivalent to some cost function $\phi \in \Gamma$. Hence we can replace each of the constraints $\langle \sigma, \phi' \rangle$ in $\mathcal{P}'$ with a new constraint $\langle \sigma, \phi \rangle$, where $\phi \in \Gamma$ and $a\phi' = b\phi + c$ for some positive integers a, b and some (positive or negative) constant c, to obtain an instance $\mathcal{P}$ of VCSP(Γ). The constant c is added to the cost of all assignments and so does not affect the choice of solution. The effect of the scale factors a and b can be simulated by taking b copies of the new constraint in $\mathcal{P}$ and a copies of all other constraints in $\mathcal{P}$. The values of a, b are constants determined by the finite set Γ', so this construction can be carried out in polynomial time in the size of $\mathcal{P}'$.

For part (2), by Definition 5, it is sufficient to show that for any finite subset Γ' of $\exp(\Gamma)$ there exists a polynomial-time reduction from VCSP(Γ') to VCSP(Γ''), where Γ'' is a finite subset of Γ.

Let Γ' be a finite subset of $\exp(\Gamma)$ and let $\mathcal{P}'$ be any instance of VCSP(Γ'). By Definition 7, any cost function $\phi' \in \exp(\Gamma)$ can be constructed by using some gadget $\langle \mathcal{P}_{\phi'}, L \rangle$ where $\mathcal{P}_{\phi'}$ is an instance of VCSP(Γ). Hence we can simply replace each constraint in $\mathcal{P}'$ which has a cost function ϕ' not already in Γ with the corresponding gadget to obtain an instance $\mathcal{P}$ of VCSP(Γ) which is equivalent to $\mathcal{P}'$. The maximum size of any of the gadgets used is a constant determined by the finite set Γ', so this construction can be carried out in polynomial time in the size of $\mathcal{P}'$.

For part (3), by Definition 5, it is sufficient to show that for any finite subset Γ' of $\Gamma \cup \mathrm{Feas}(\Gamma)$ there exists a polynomial-time reduction from VCSP(Γ') to VCSP(Γ''), where Γ'' is a finite subset of Γ.

Let Γ' be a finite subset of $\Gamma \cup \mathrm{Feas}(\Gamma)$ and let $\mathcal{P}'$ be any instance of VCSP(Γ'). By Definition 8, any cost function $\phi' \in \mathrm{Feas}(\Gamma)$ is obtained from some cost function $\phi \in \Gamma$ by setting all finite values to 0. Now assume that $\mathcal{P}'$ has k constraints with cost functions in $\mathrm{Feas}(\Gamma)$. If we replace each of these constraints $\langle \sigma, \phi' \rangle$ with a new constraint $\langle \sigma, \phi \rangle$, where $\phi \in \Gamma$ and $\mathrm{Feas}(\phi) = \phi'$, then we obtain an instance $\mathcal{P}$ of VCSP(Γ).

Let M be the maximum finite value taken by any cost function in the finite set Γ', and let m be the minimum difference between any two distinct finite values taken on by cost functions in Γ'. The cost of any assignment for $\mathcal{P}$ differs by at most kM from the cost of the same assignment for $\mathcal{P}'$. Hence if we also replace all the remaining constraints $\langle \sigma, \phi \rangle$ of $\mathcal{P}'$ with $\lceil \frac{Mk}{m} + 1 \rceil$ copies of $\langle \sigma, \phi \rangle$, then we obtain an instance of VCSP(Γ) with the same solutions as $\mathcal{P}'$. Since M and m are constants determined by the finite set Γ', this construction can be carried out in polynomial time in the size of $\mathcal{P}'$.

We can now combine all three extensions to obtain the following result.

Definition 12. *For any valued constraint language, Γ, we define the **closure** of Γ, denoted $\widehat{\Gamma}$, as follows:*

$$\widehat{\Gamma} \stackrel{\text{def}}{=} (\exp(\Gamma \cup \mathrm{Feas}(\Gamma)))_{\equiv}.$$

Corollary 1. *A valued constraint language Γ is tractable if and only if $\widehat{\Gamma}$ is tractable; similarly, Γ is NP-hard if and only if $\widehat{\Gamma}$ is NP-hard.*

6 Characterising $\widehat{\Gamma}$

The main result of this paper is the following theorem, which characterises the extended language $\widehat{\Gamma}$ in terms of the feasibility polymorphisms and fractional polymorphisms of Γ.

Theorem 3. *For any valued constraint language Γ with costs in $\overline{\mathbb{Q}}_+$, and any cost function ϕ taking values in $\overline{\mathbb{Q}}_+$, $\phi \in \widehat{\Gamma}$ if and only if* $\mathrm{Pol}(\Gamma) \subseteq \mathrm{Pol}(\{\phi\})$ *and* $\mathrm{fPol}(\Gamma) \subseteq \mathrm{fPol}(\{\phi\})$.

The following result is an immediate consequence of Corollary 1 and Theorem 3.

Corollary 2. *The tractability or NP-hardness of a valued constraint language Γ with costs in $\overline{\mathbb{Q}}_+$ is determined by its feasibility polymorphisms and fractional polymorphisms.*

We also observe that when the cost functions in Γ take finite rational values only, the tractability or NP-hardness is determined by the fractional polymorphisms alone. Conversely, when $\Gamma = \mathrm{Feas}(\Gamma)$ the tractability or NP-hardness is determined by the feasibility polymorphisms alone.

We will prove Theorem 3 in two halves. First we show, in Proposition 1, that the feasibility polymorphisms and fractional polymorphisms of Γ are preserved by all members of $\widehat{\Gamma}$. Then we show, in Theorem 4, that every cost function with all the feasibility polymorphisms and fractional polymorphisms of Γ is in fact a member of $\widehat{\Gamma}$.

Proposition 1. *If Γ is any valued constraint language then* $\mathrm{Pol}(\Gamma) = \mathrm{Pol}(\widehat{\Gamma})$ *and* $\mathrm{fPol}(\Gamma) = \mathrm{fPol}(\widehat{\Gamma})$.

Proof. This follows immediately from the fact that feasibility polymorphisms and fractional polymorphisms are preserved by aggregating cost functions, projecting onto subsets of variables, adding constants, scaling by a natural number, and applying the feasibility operator.

To see this note that if ϕ_1 and ϕ_2 both satisfy the inequality in Definition 9, then so does any extension of ϕ_1 and ϕ_2 obtained by adding dummy arguments. So also does $\phi_1 \oplus \phi_2$ (by the monotonicity of the $\oplus$ operation). So also does the projection of each ϕ_i onto any list of arguments. Hence if Γ has the feasibility polymorphism f, then so does any cost function expressible over Γ. Furthermore adding a constant to ϕ preserves this inequality, and scaling by a natural number also preserves the inequality. Finally replacing ϕ_i with $\mathrm{Feas}(\phi_i)$ also preserves this inequality.

Similar remarks apply to the inequality in Definition 10.

To establish Theorem 4 below we will use the following result, which is a variant of the well-known Farkas' Lemma used in linear programming [27, 31].

Lemma 1 (Farkas 1894). *Let S and T be finite sets of indices, where T is the disjoint union of two subsets, $T_\geq$ and $T_=$. For all $i \in S$, and all $j \in T$, let $a_{i,j}$ and b_j be rational numbers. Exactly one of the following holds:*

– *Either there exists a set of non-negative rational numbers* $\{x_i \mid i \in S\}$ *and a rational number* C *such that*

$$\text{for each } j \in T_{\geq}, \quad \sum_{i \in S} a_{i,j}\, x_i \;\geq\; b_j + C, \qquad \text{and,}$$
$$\text{for each } j \in T_{=}, \quad \sum_{i \in S} a_{i,j}\, x_i \;=\; b_j + C.$$

– *Or else there exists a set of integers* $\{y_j \mid j \in T\}$ *such that* $\sum_{j \in T} y_j = 0$ *and:*

$$\text{for each } j \in T_{\geq}, \quad y_j \;\geq\; 0,$$
$$\text{for each } i \in S, \quad \sum_{j \in T} y_j\, a_{i,j} \;\leq\; 0, \qquad \text{and}$$
$$\sum_{j \in T} y_j\, b_j \;>\; 0.$$

Such a set is called a ***certificate of unsolvability****.*

The proof of Theorem 4 also uses a number of constructions related to the notion of an *indicator problem*, as introduced in [20, 22].

Definition 13. *A* k***-matching*** *of a valued constraint language* Γ *is defined to be a pair* $\langle M, \gamma \rangle$ *where*

– γ *is a cost function in* Γ *with arity* r*, and*
– M *is a* $k \times r$ *matrix of elements of* D *such that* γ *has a finite value when applied to any of the* k *rows.*

Definition 14. *A* k***-weighting****,* X*, of a valued constraint language* Γ *is defined to be a mapping from the set of all* k*-matchings of* Γ *to the non-negative integers.*

Note that a k-weighting of Γ can be seen as associating a multiplicity (possibly zero) with each k-matching.

Definition 15. *Given a* finite *valued constraint language* Γ *over a finite set* D *and a* k*-weighting,* X*, of* Γ*, we define the* X***-weighted indicator problem*** *over* Γ*, denoted* $\mathcal{IP}(\Gamma, X)$*, as follows:*

– *The set of variables of* $\mathcal{IP}(\Gamma, X)$ *is the set* D^k *consisting of all* k*-tuples of elements from* D*.*
– *The domain of* $\mathcal{IP}(\Gamma, X)$ *is the domain* D *of* Γ*.*
– *The constraints of* $\mathcal{IP}(\Gamma, X)$ *are defined as follows. Note that any list of* r *variables,* $v_1, \ldots, v_r$*, can be seen as (the columns of) a* $k \times r$ *matrix. For every list* S *of variables, if* $\langle S, \gamma \rangle$ *is a* k*-matching of* Γ*, then* $\mathcal{IP}(\Gamma, X)$ *has the constraint* $\langle S, \mathrm{Feas}(\gamma) \rangle$ *and* $X(\langle S, \gamma \rangle)$ *copies of the constraint* $\langle S, \gamma \rangle$*.*

Theorem 4. *Let* Γ *be a* finite *valued constraint language over a finite set* D *with costs in* $\overline{\mathbb{Q}}_+$*, and let* $\phi : D^r \to \overline{\mathbb{Q}}_+$ *be any cost function such that* $\mathrm{Pol}(\Gamma) \subseteq \mathrm{Pol}(\{\phi\})$*.*

Either $\phi \in \widehat{\Gamma}$*, or else there is some fractional polymorphism of* Γ *which is not a fractional polymorphism of* ϕ*.*

Proof. The idea of the proof is as follows. We will attempt to construct a weighted indicator problem to express a cost function ϕ' which is cost-equivalent to ϕ. If this succeeds, then we have shown that $\phi \in \widehat{\Gamma}$, since every weighted indicator problem for Γ is an instance of VCSP($\exp(\Gamma \cup \text{Feas}(\Gamma))$).

On the other hand, if this fails then we will show that we must have an unsatisfiable collection of equations and inequations. We will then use Lemma 1, to get a certificate of insolvability. This certificate will give us the required fractional polymorphism of Γ that is not a fractional polymorphism of ϕ.

We now give the details of the proof. Let k be the number of r-tuples for which the value of ϕ is finite and fix an arbitrary order, $\langle x_1, \ldots, x_k \rangle$, for these tuples. This list of tuples can be viewed as (the rows of) a matrix with k rows and r columns, which we will call S_ϕ.

Note that (the columns of) S_ϕ can be viewed as a list $\langle s_1, \ldots, s_r \rangle$ of k-tuples, and hence as a list of variables of an indicator problem. We will now try to find some k-weighting X of Γ so that the X-weighted indicator problem $\mathcal{IP}(\Gamma, X)$ can be used to express a cost function ϕ' which is cost-equivalent to ϕ. More precisely, we will seek to find a k-weighting X such that $\langle \mathcal{IP}(\Gamma, X), S_\phi \rangle$ is a gadget for expressing such a ϕ'.

The variables of $\mathcal{IP}(\Gamma, X)$ are the possible k-tuples over D, so each assignment to these variables can be viewed as a function $f : D^k \to D$. Whatever k-weighting X we choose, the set of assignments for $\mathcal{IP}(\Gamma, X)$ which have infinite cost is the same as for the classical indicator problem for Feas(Γ) of order k, as defined in [20, 22]. Hence, by Theorem 1 of [22], every assignment for $\mathcal{IP}(\Gamma, X)$ which has a finite cost corresponds to a feasibility polymorphism of Γ. Since we are assuming that ϕ has all of the feasibility polymorphisms of Γ, it follows that $\langle \mathcal{IP}(\Gamma, X), S_\phi \rangle$ is a gadget for some r-ary cost function which is finite-valued exactly when ϕ is finite-valued.

Now consider any $f : D^k \to D \in \text{Pol}_k(\Gamma)$. By Definition 9, we know that, for any k-matching $\langle S, \gamma \rangle$ of Γ, we must have $\gamma(f(S)) < \infty$, where $f(S)$ denotes the tuple of values obtained by applying f to each column of S. Since ϕ has all the feasibility polymorphisms of Γ, we also have that $\phi(f(S_\phi)) < \infty$.

We now define a finite system of inequalities and equations, with finite coefficients, which together specify the required properties for an unknown constant C and k-weighting X, to ensure that $\langle \mathcal{IP}(\Gamma, X), S_\phi \rangle$ is a gadget for $\phi + C$.

For each $f \in \text{Pol}_k(\Gamma)$,

$$\sum_{\gamma \in \Gamma} \sum_{\{\text{all } k\text{-matchings } \langle S, \gamma \rangle\}} X(\langle S, \gamma \rangle)\ \gamma(f(S)) \quad \geq \quad \phi(f(S_\phi)) + C. \quad (1)$$

For each projection $e \in \text{Pol}_k(\Gamma)$,

$$\sum_{\gamma \in \Gamma} \sum_{\{\text{all } k\text{-matchings } \langle S, \gamma \rangle\}} X(\langle S, \gamma \rangle)\ \gamma(e(S)) \quad = \quad \phi(e(S_\phi)) + C. \quad (2)$$

We claim that if a non-negative rational solution X to this system of inequalities and equations exists, then $\langle \mathcal{IP}(\Gamma, X), S_\phi \rangle$ is a gadget for the cost function $\phi + C$.

To see this, note that the set of inequalities imply that the value of the projection of $\mathcal{IP}(\Gamma, X)$ onto the list of variables S_ϕ must be at least as great as the value of $\phi + C$, for all possible assignments. Furthermore, from the set of equations, and the choice of S_ϕ, it follows that whenever $\phi + C$ is finite, the value of the projection of $\mathcal{IP}(\Gamma, X)$ onto the list of variables S_ϕ must be the same as $\phi + C$.

By applying a suitable scale factor to the solution obtained, we can choose an *integer*-valued X such that $\langle \mathcal{IP}(\Gamma, X), S_\phi \rangle$ is a gadget for some cost function which is cost-equivalent to ϕ.

On the other hand, if this system of equations and inequalities has no solution, then we appeal to Lemma 1, to get a certificate of insolvability. That is, in this case we know that there exists a set of integers $\{y_f \mid f \in \mathrm{Pol}_k(\Gamma)\}$, such that $\sum_{f \in \mathrm{Pol}_k(\Gamma)} y_f = 0$, $y_f \geq 0$ when f is not a projection, and:

$$\text{for each } k\text{-matching } \langle S, \gamma \rangle \text{ of } \Gamma, \quad \sum_{f \in \mathrm{Pol}_k(\Gamma)} y_f \ \gamma(f(S)) \ \leq \ 0, \quad \text{and} \tag{3}$$

$$\sum_{f \in \mathrm{Pol}_k(\Gamma)} y_f \ \phi(f(S_\phi)) \ > \ 0 \tag{4}$$

Let $m = \min\{y_f \mid f \text{ is a projection }\}$. Since $\sum_{f \in \mathrm{Pol}_k(\Gamma)} y_f = 0$, we know that $m < 0$.

Define a set of integers $\{z_f \mid f \in \mathrm{Pol}_k(\Gamma)\}$ as follows:

$$z_f = \begin{cases} y_f - m & \text{if } f \text{ is a projection} \\ y_f & \text{otherwise} \end{cases}$$

Now we have that each $z_f \geq 0$, and that $\sum_{f \in \mathrm{Pol}_k(\Gamma)} z_f = k|m|$.

Let E be the set of all k-ary projections. It follows (rewriting Equation 3) that for any k-matching $\langle S, \gamma \rangle$ of Γ

$$|m| \sum_{e \in E} \gamma(e(S)) \quad \geq \sum_{f \in \mathrm{Pol}_k(\Gamma)} z_f \ \gamma(f(S))$$

Moreover, if S is any $k \times r$ matrix for which $\langle S, \gamma \rangle$ is not a k-matching of Γ, then $\sum_{e \in E} \gamma(e(S)) = \infty$. Hence, for any set of k-tuples $x_1, \ldots, x_r$ we get that

$$\left(\sum_{f \in \mathrm{Pol}_k(\Gamma)} z_f \right) \cdot \left(\sum_{i=1}^{k} \gamma(x_1[i], \ldots, x_r[i]) \right) \geq \ k \sum_{f \in \mathrm{Pol}_k(\Gamma)} z_f \ \gamma(f(x_1), \ldots, f(x_r))$$

which precisely states that the k-ary weighted function $\{\langle z_f, f \rangle \mid f \in \mathrm{Pol}_k(\Gamma)\}$ is a fractional polymorphism of Γ.

On the other hand, rewriting Equation 4 in the same way gives:

$$\left(\sum_{f \in \mathrm{Pol}_k(\Gamma)} z_f \right) \cdot \left(\sum_{i=1}^{k} \phi(s_1[i], \ldots, s_r[i]) \right) < \ k \sum_{f \in \mathrm{Pol}_k(\Gamma)} z_f \ \phi(f(s_1), \ldots, f(s_r))$$

where $s_1, \ldots, s_r$ are the columns of S_ϕ. This provides a witness that the weighted function $\{\langle z_f, f \rangle \mid f \in \mathrm{Pol}_k(\Gamma)\}$ is *not* a fractional polymorphism of ϕ.

7 A Necessary Condition For Tractability

In this section, we exhibit a well-known (finite-valued) intractable cost function $\phi_{\neq}$ and use it to establish a necessary condition for a valued constraint language to be tractable.

Definition 16. *Define the binary cost function $\phi_{\neq} : D^2 \to \mathbb{Q}_+$ as follows:*

$$\phi_{\neq}(x, y) \stackrel{\text{def}}{=} \begin{cases} 1 \text{ if } x = y \\ 0 \text{ otherwise} \end{cases}$$

The cost function $\phi_{\neq}(x, y)$ penalizes the assignment of the same value to its two arguments.

Lemma 2. *For any set D with $|D| \geq 2$, $\text{VCSP}(\{\phi_{\neq}\})$ is NP-hard.*

Proof. For $|D| = 2$, this follows immediately from the fact that the version of Max-Sat consisting of only XOR constraints is NP-hard [11].

For $|D| \geq 3$, a polynomial-time algorithm to solve $\text{VCSP}(\{\phi_{\neq}\})$ would immediately provide a polynomial-time algorithm to determine whether a graph has a $|D|$-colouring, which is a well-known NP-complete problem [16].

Definition 17. *A k-ary **fractional projection** is a k-ary weighted function $\{\langle n, \pi_1\rangle, \ldots, \langle n, \pi_k\rangle\}$ where n is a constant and each π_i is the projection that returns its ith argument.*

Lemma 3. *For any cost function ϕ, $\text{fPol}(\{\phi\})$ contains all fractional projections.*

Proof. Consider the inequality in Definition 10 defining a fractional polymorphism. All fractional projections satisfy this inequality (with equality), so they are all fractional polymorphisms of any cost function ϕ.

Theorem 5. *If all fractional polymorphisms of a valued constraint language Γ are fractional projections, then Γ is NP-hard.*

Proof. Suppose that every fractional polymorphism of Γ is a fractional projection. By Lemma 3, we have that $\text{fPol}(\Gamma) \subseteq \text{fPol}(\{\phi_{\neq}\})$.

Since $\text{Feas}(\phi_{\neq})$ is the cost function whose costs are all zero, it follows that $\phi_{\neq}$ has all possible feasibility polymorphisms. Hence $\text{Pol}(\Gamma) \subseteq \text{Pol}(\{\phi_{\neq}\})$.

By Theorem 3, we have $\phi_{\neq} \in \widehat{\Gamma}$, so by Lemma 2, $\widehat{\Gamma}$ is NP-hard. Hence, by Corollary 1, it follows that Γ is also NP-hard.

Hence, assuming that P $\neq$ NP, we have that any tractable valued constraint language must have some fractional polymorphism which is not a fractional projection.

8 Conclusion

We have shown that the complexity of any valued constraint language with rational-valued or infinite costs is determined by certain algebraic properties of the cost functions, which we have identified as feasibility polymorphisms and fractional polymorphisms.

When the cost functions can only take on the values zero or infinity, the optimisation problem VCSP collapses to the classical constraint satisfaction problem, CSP. In previous papers we have shown that the existence of a non-trivial polymorphism is a necessary condition for tractability in this case [19, 21]. This result sparked considerable activity in the search for and characterization of tractable constraint languages [3, 4, 21]. We hope that the results in this paper will provide a similar impetus for the characterization of tractable valued constraint languages using algebraic methods.

Of course there are still many open questions concerning the complexity of valued constraint satisfaction problems. In particular, it will be interesting to see how far these results can be extended to other valuation structures, and to more general frameworks, such as the semiring-based framework.

References

1. Anglès d'Auriac, J-Ch., Igloi, F., Preismann, M. & Sebö, A. "Optimal cooperation and submodularity for computing Potts' partition functions with a large number of statistics", *J. Physics A: Math. Gen.* 35, (2002), pp. 6973–6983.
2. Bistarelli, S., Fargier, H., Montanari, U., Rossi, F., Schiex, T. & Verfaillie, G. "Semiring-based CSPs and valued CSPs: Frameworks, properties and comparison", *Constraints* 4, (1999), pp. 199–240.
3. Bulatov, A.A. "A dichotomy theorem for constraints on a three-element set", *Proc. 43rd IEEE Symposium on Foundations of Computer Science (FOCS'02)*, (2002), pp. 649–658.
4. Bulatov, A.A., Jeavons, P. & Krokhin, A. "Classifying the complexity of constraints using finite algebras", *SIAM Journal on Computing* 34, (2005), pp. 720–742.
5. Burkard, R., Klinz, B. & Rudolf, R. "Perspectives of Monge properties in optimization", *Discrete Applied Mathematics* 70, (1996), pp. 95–161.
6. Cohen, D., Cooper, M.C. & Jeavons, P. "A complete characterisation of complexity for Boolean constraint optimization problems", *Proc. 10th Int. Conf. on Principles and Practice of Constraint Programming (CP'04)*, *LNCS* 3258, (2004), pp. 212–226.
7. Cohen, D., Cooper, M.C., Jeavons, P. & Krokhin, A. " Soft constraints: complexity and multimorphisms" *Proceedings of CP'03*, *LNCS* 2833, (2003), pp. 244–258.
8. Cohen, D., Cooper, M.C., Jeavons, P. & Krokhin, A. "A maximal tractable class of soft constraints" *Journal of Artificial Intelligence Research* 22, (2004), pp. 1-22.
9. Cohen, D., Cooper, M.C., Jeavons, P. & Krokhin, A. "Supermodular functions and the complexity of Max CSP", *Discrete Applied Mathematics*, 149 (1-3),(2005), pp. 53–72.
10. Cohen, D., Cooper, M.C., Jeavons, P. & Krokhin, A. "The Complexity of Soft Constraint Satisfaction", *Artificial Intelligence*, to appear.

11. Creignou, N. "A dichotomy theorem for maximum generalised satisfiability problems", *Journal of Computer and Systems Sciences* 51(3), (1995), pp. 511–522.
12. Creignou, N., Khanna, S. & Sudan, M. *Complexity classification of Boolean constraint satisfaction problems*, volume 7 of *SIAM Monographs on Discrete Mathematics and Applications*, (2001).
13. Cunningham, W.H. "Minimum cuts, modular functions, and matroid polyhedra", *Networks* 15(2), (1985), pp. 205–215.
14. Feder, T. & Vardi, M.Y. "The computational structure of monotone monadic SNP and constraint satisfaction: a study through datalog and group theory", *SIAM Journal on Computing* 28(1), (1998), pp. 57–104.
15. Fujishige, S. *Submodular Functions and Optimization*, 2nd edn., Annals of Discrete Mathematics, Vol. 58, Elsevier, (2005).
16. Garey, M.R. & Johnson, D.S., *Computers and Intractability: A guide to the theory of NP-completeness*, W.H. Freeman, (1979).
17. Iwata, S. "A fully combinatorial algorithm for submodular function minimization", *Journal of Combinatorial Theory, Series B* 84(2), (2002), pp. 203–212.
18. Iwata, S., Fleischer, L. & Fujishige, S. "A combinatorial, strongly polynomial-time algorithm for minimizing submodular functions", *Journal of the ACM* 48(4), (2001), pp. 761–777.
19. Jeavons, P.G. "On the algebraic structure of combinatorial problems", *Theoretical Computer Science* 200 (1998), pp. 185–204.
20. Jeavons, P.G. "Constructing constraints", *Proceedings of CP'98)*, *Lecture Notes in Computer Science* 1520, (1998), pp. 2–17.
21. Jeavons, P.G., Cohen D.A. & Gyssens, M. "Closure properties of constraints", *Journal of the ACM* 44, (1997), pp. 527–548.
22. Jeavons, P.G., Cohen D.A. & Gyssens, M. "How to determine the expressive power of constraints", *Constraints*, 4, (1999), pp. 113–131.
23. Jonsson, P., Klasson, M. & Krokhin, A. "The approximability of three-valued Max CSP", *SIAM Journal of Computing*, 35, (2006), pp. 1329–1349.
24. Nagamochi, H. & Ibaraki, I. "A note on minimizing submodular functions", *Information Processing Letters* 67, (1998), pp. 239–244.
25. Narayanan, H., *Submodular Functions and Electrical Networks*, Annals of Discrete Mathematics 54, North Holland (London, New York, Amsterdam), (1997).
26. Narayanan, H. "A note on the minimisation of symmetric and general submodular functions" *Discrete Applied Mathematics* 131(2), (2003), pp. 513–522.
27. Nemhauser, G.L. & Wolsey, L.A. *Integer and Combinatorial Optimisation*, Wiley, (1999).
28. Queyranne, M. "Minimising symmetric submodular functions", *Mathematical Programming* 82(1-2), (1998), pp. 3–12.
29. Schaefer, T.J. "The complexity of satisfiability problems", *Proc. 10th ACM Symposium on Theory of Computing (STOC'78)*, (1978), pp. 216–226.
30. Schrijver, A. "A combinatorial algorithm minimizing submodular functions in strongly polynomial time", *J. Combinatorial Theory*, Series B, 80, (2000), pp. 346–355.
31. Schrijver, A. *Theory of Linear and Integer Programming*, Wiley, (1986).
32. Szendrei, A., *Clones in Universal Algebra*, Seminaires de Mathematiques Superieures, University of Montreal, 99, (1986).
33. Topkis, D. *Supermodularity and Complementarity*, Princeton University Press, (1998).