THE POWER OF LINEAR PROGRAMMING FOR VALUED CSPS

JOHAN THAPPER AND STANISLAV ŽIVNÝ

Abstract. A class of *valued constraint satisfaction problems* (VCSPs) is characterised by a *valued constraint language*, a fixed set of cost functions on a finite domain. An instance of the problem is specified by a sum of cost functions from the language with the goal to minimise the sum. This framework includes and generalises well-studied constraint satisfaction problems (CSPs) and maximum constraint satisfaction problems (Max-CSPs).

Our main result is a precise algebraic characterisation of valued constraint languages whose instances can be solved *exactly* by the *basic linear programming relaxation*. Using this result, we obtain tractability of several novel and previously widely-open classes of VCSPs, including problems over valued constraint languages that are: (1) submodular on *arbitrary lattices*; (2) bisubmodular (also known as k-submodular) on *arbitrary finite domains*; (3) weakly (and hence strongly) tree-submodular on *arbitrary trees*.

Keywords: valued constraint satisfaction, fractional polymorphisms, fractional homomorphisms, submodularity, bisubmodularity, linear programming

Laboratoire d'Informatique (LIX), CNRS UMR 7161, École Polytechnique, 91128 Palaiseau, France
Department of Computer Science, University of Oxford, Oxford, UK
E-mail addresses: `thapper@lix.polytechnique.fr, standa.zivny@cs.ox.ac.uk`.

1. Introduction

The constraint satisfaction problem (CSP) provides a common framework for many theoretical and practical problems in computer science. An instance can be vaguely described as a set of variables to be assigned values from the domains of the variables so that all constraints are satisfied [44]. The CSP is NP-complete in general and thus we are interested in restrictions which give rise to tractable classes of problems. Following Feder & Vardi [21], we restrict the constraint language; that is, all constraint relations in a given instance must belong to a fixed, finite set of relations on the domain. The most successful approach to classifying language-restricted CSPs is the so-called algebraic approach [7, 29, 30], which has led to several complexity classifications [1, 4, 6, 8] and algorithmic characterisations [2, 27] going beyond the seminal work of Schaefer [46].

Motivated by reasons both theoretical (optimisation problems are different from decision problems) and practical (many problems are over-constrained and hence have no solution, or under-constrained and hence have many solutions), we study valued constraint satisfaction problems (VCSPs) [5, 47]. A valued constraint language is a finite set of cost functions on the domain, and a VCSP instance is given by a weighted sum of cost functions from the language with the goal to *minimise* the sum. (CSPs correspond to the case when the range of all cost functions is $\{0, \infty\}$, and Max-CSPs correspond to the case when the range of all cost functions is $\{0, 1\}$.[1]) The VCSP framework is very robust and has also been studied under different names such as Min-Sum problems, Gibbs energy minimisation, Markov Random Fields (MRF), Conditional Random Fields (CRF) and others in several different contexts in computer science [15, 40, 49].

Given the generality of the VCSP, it is not surprising that only few complexity classifications are known. In particular, only Boolean (on a 2-element domain) languages [11, 16] and conservative (containing all $\{0, 1\}$-valued unary cost functions) languages [36] have been completely classified with respect to exact solvability. On the algorithmic side, most known tractable languages are somewhat related to submodular functions on distributive lattices [10, 11, 32, 36].

An alternative approach for solving VCSPs is using linear programming (LP) and semidefinite programming (SDP); these have been used mostly for approximation [3, 18, 39, 45].

Contribution We study the power of the *basic linear programming relaxation* (BLP). Our main result (Theorem 4.1) is a precise characterisation of valued constraint languages for which BLP is a decision procedure. In more detail, we characterise valued constraint languages over which VCSP instances can be solved exactly by a certain basic linear program. Equivalently, we show precisely when a particular integer programming formulation of a VCSP has zero integrality gap. The characterisation is algebraic in terms of *fractional polymorphisms* [9].

Our work is the first link between solving VCSPs exactly using LP and the algebraic machinery for VCSPs introduced by Cohen et al. in [9, 12]. Part of the proof is inspired by the characterisation of width-1 CSPs [19, 21]. One of the main technical contributions is a construction of totally symmetric fractional polymorphisms of all arities (Theorem 4.4).

This result allows us to demonstrate that several valued constraint languages are covered by our characterisation and thus are tractable; that is, VCSP instances over these languages can be solved exactly using BLP. New tractable languages include: (1) submodular languages on *arbitrary lattices*; (2) bisubmodular (also known as *k*-submodular) languages on *arbitrary finite domains*; (3) weakly (and hence strongly) tree-submodular languages on *arbitrary trees*. The complexity of (subclasses of) these languages has been mentioned explicitly as open problems in [20, 26, 35, 37]. More generally, we show that any valued constraint language with a binary multimorphism in which at least one operation is a semi-lattice operation is tractable (cf. Section 6). Our results cover *all* known tractable finite-valued constraint languages.

---

[1]With respect to exact solvability, Max-CSPs ("maximising the number of satisfied constraints") are polynomial-time equivalent to Min-CSPs ("minimising the number of unsatisfied constraints"). Therefore, with respect to exact solvability, Max-CSPs are polynomial-time equivalent to $\{0, 1\}$-valued VCSPs.

Related work Apart from identifying tractable classes of CSPs and VCSPs with respect to exact solvability, the approximability of Max-CSPs has attracted a lot of attention [17, 31, 33]. Under the assumption of the *unique games conjecture* [34], Raghavendra showed how to approximate all Max-CSPs and finite-valued VCSPs optimally [45].[2] For VCSPs that are tractable, Raghavendra's algorithms provide a PTAS, but it seems notoriously difficult to determine the approximation ratios of these algorithms. Very recently, Max-CSPs that are *robustly approximable* have been characterised as those having *bounded width* [3, 18, 39]. Specifically, Kun et al. studies the question of which Weighted Max-CSPs[3] can be robustly approximated using BLP [39]. Their result is related but incomparable to ours as it applies to robust approximability and not to exact solvability, except for the special case of width-1 CSPs. In particular, "solving" ("deciding") for us means finding an optimum solution to a VCSP instance, which is an optimisation problem, whereas "solving" in [39] means (ignoring their results on robust approximability, which do not apply here) the basic LP formulation of a CSP instance finds a solution if one exists.[4]

We remark that our tractability results apply to the minimisation problem of VCSP instances (i.e., the objective function is given by a sum of "local" cost functions) but not to objective functions given by an oracle. In particular, submodular functions given by an oracle can be minimised on distributive lattices [28, 48], diamonds [38], and several constructions on lattices preserving tractability have been identified [37], but it is widely open what happens on non-distributive lattices. Similarly, bisubmodular functions given by an oracle can be minimised in polynomial-time on domains of size 3 [24], but the complexity is open on domains of larger size [26]. It is known that strongly tree-submodular functions given by an oracle can be minimised in polynomial time on binary trees [35], but the complexity is open on general (non-binary) trees. Similarly, it is known that weakly tree-submodular functions given by an oracle can be minimised in polynomial time on chains and forks [35], but the complexity on (even binary) trees is open.

Extending the notion of (generalised) arc consistency for CSPs [23, 41] and several previously studied notions of arc consistencies for VCSPs [14], Cooper et al. introduced *optimal soft arc consistency* (OSAC) [13], which is a linear program relaxation of a given VCSP instance. Since OSAC is is a tighter relaxation than BLP (cf. Appendix C), all tractable classes identified in this paper are solved by OSAC as well. Similarly, since the basic SDP relaxation from [45] is tighter than BLP, all tractable cases identified in this paper are sovled by it as well.

2. PRELIMINARIES

The set of non-negative rational numbers is denoted by $\mathbb{Q}_{\geq 0}$. A *signature* τ is a set of *function symbols* f, each with an associated positive *arity*, $ar(f)$. A *valued τ-structure* A (also known as a *valued constraint language*, or just a *language*) consists of a *domain* $D = D(A)$, together with a function $f^A : D^{ar(f)} \to \mathbb{Q}_{\geq 0}$, for each function symbol $f \in \tau$. (To be precise, these are *finite-valued* structures. In Section 5, we will extend $\mathbb{Q}_{\geq 0}$ with infinity.)

Let A be a valued τ-structure. An instance of VCSP(A) is given by a valued τ-structure I. A solution to I is a function $h : D(I) \to D(A)$, its measure given by

$$\sum_{f \in \tau, \bar{x} \in D(I)^{ar(f)}} f_i^I(\bar{x}) f_i^A(h(\bar{x})).$$

The goal is to find a solution of minimum measure. This measure will be denoted by $\mathsf{Opt}_A(I)$.

---

[2]Note that Max-CSPs (={0,1}-valued VCSPs) and finite-valued VCSPs, respectively, are called CSPs and Generalised CSPs (GCSPs), respectively, in [45].

[3]In Weighted Max-CSPs, every constraint f is $\{0, c_f\}$-valued, where c_f is a positive constant. Weighted Max-CPSs are a special case of VCSPs.

[4]Note that CSPs are defined as {0,1}-valued in [39] and not as {0,∞}-valued, as in this paper. This is needed for the LP formulation and the measure of approximability. After all, [39] deals with Max-CSPs.

For an m-tuple $\bar{t}$, we denote by $\{\bar{t}\}$ the set of elements in $\bar{t}$. Furthermore, we denote by $[\bar{t}]$ the multiset of elements in $\bar{t}$.

2.1. Fractional Homomorphisms.

Let A and B be valued structures over the same signature τ. Let B^A denote the set of all functions from $D(A)$ to $D(B)$. A *fractional homomorphism* from A to B is a function $\omega : B^A \to \mathbb{Q}_{\geq 0}$, with $\sum_{g \in B^A} \omega(g) = 1$, such that for every function symbol $f \in \tau$ and tuple $\bar{a} \in D(A)^{ar(f)}$, it holds that

$$\sum_{g \in B^A} \omega(g) f^B(g(\bar{a})) \leq f^A(\bar{a}),$$

where the functions g are applied component-wise.

We write $A \to_f B$ to indicate the existence of a fractional homomorphism.

Proposition 2.1. *Assume that $A \to_f B$. Then $\mathsf{Opt}_A(I) \geq \mathsf{Opt}_B(I)$, for every instance I.*

Proof. Let ω be a fractional homomorphism from A to B, let $X = D(I)$ and let $h : X \to A$ be an arbitrary solution. Then,

$$\sum_{f,\bar{x}} f^I(\bar{x}) f^A(h(\bar{x})) \geq \sum_{f,\bar{x}} f^I(\bar{x}) \sum_{g \in B^A} \omega(g) f^B(g(h(\bar{x}))) = \sum_{g \in B^A} \omega(g) \sum_{f,\bar{x}} f^I(\bar{x}) f^B(g(h(\bar{x}))),$$

where the sums are over $f \in \tau$ and $\bar{x} \in X^{ar(f)}$. Hence, there exists a $g \in B^A$ such that the measure of the solution $g \circ h$ to I as an instance of $\mathrm{VCSP}(B)$ is no greater than the measure of the solution h to I as an instance of $\mathrm{VCSP}(A)$. $\square$

2.2. Fractional Polymorphisms.

Let A be a valued τ-structure, and let $D = D(A)$. An m-ary *operation* on D is a function $g : D^m \to D$. Let $\mathcal{O}_D^{(m)}$ denote the set of all m-ary operations on D. An m-ary *fractional operation* is a function $\omega : \mathcal{O}_D^{(m)} \to \mathbb{Q}_{\geq 0}$. Define $\|\omega\|_1 := \sum_g \omega(g)$. An m-ary fractional operation ω is called an m-ary *fractional polymorphism* [9] if $\|\omega\|_1 = 1$ and for every function symbol $f \in \tau$ and tuples $\bar{a}_1, \ldots, \bar{a}_m \in D^{ar(f)}$, it holds that

$$\sum_{g \in \mathcal{O}_D^{(m)}} \omega(g) f^A(g(\bar{a}_1, \ldots, \bar{a}_m)) \leq \frac{1}{m} \sum_{i=1}^{m} f^A(\bar{a}_i).$$

The set $\{g \mid \omega(g) > 0\}$ of operations is called the *support* of ω and is denoted by $\mathrm{supp}(\omega)$. Let S_m be the symmetric group on $\{1, \ldots, m\}$. An m-ary operation g is *symmetric*[5] if for every permutation $\pi \in S_m$, we have $g(x_1, \ldots, x_m) = g(x_{\pi(1)}, \ldots, x_{\pi(m)})$.

Definition 1. *A* totally symmetric fractional polymorphism ω *is a fractional polymorphism such that if $g \in \mathrm{supp}(\omega)$, then g is* symmetric.

The *superposition* of an n-ary operation h with n m-ary operations $g_1, \ldots, g_n$ is the m-ary operation defined by $h[g_1, \ldots, g_n](x_1, \ldots, x_m) = h(g_1(x_1, \ldots, x_m), \ldots, g_n(x_1, \ldots, x_m))$. A set of operations is called a *clone* if it contains all projections and is closed under superposition. The smallest clone that contains a set of operations $\mathcal{F}$ is called the clone *generated* by $\mathcal{F}$. We say that an operation f is *generated* by $\mathcal{F}$ if it is contained in the clone generated by $\mathcal{F}$.

Definition 2. *The* superposition, $\omega[g_1, \ldots, g_n]$, *of an n-ary fractional polymorphism ω with n m-ary operations $g_1, \ldots, g_n$ is the m-ary fractional operation ω', where*

$$\omega'(h') = \sum_{h : h' = h[g_1, \ldots, g_n]} \omega(h).$$

---

[5]Symmetric operations are called totally symmetric in [39].

Note that in general ω' is not a fractional polymorphism, but it does satisfy the following inequality:

$$\sum_{h' \in \mathcal{O}_D^{(m)}} \omega'(h') f^A(h'(\bar{a}_1, \ldots, \bar{a}_m)) \quad = \quad \sum_{h \in \mathcal{O}_D^{(n)}} \omega(h) f^A(h[g_1, \ldots, g_n](\bar{a}_1, \ldots, \bar{a}_m))$$

$$\leq \quad \frac{1}{n} \sum_{i=1}^{n} f^A(g_i(\bar{a}_1, \ldots, \bar{a}_m)),$$

for every $f \in \tau$ and $\bar{a}_1, \ldots \bar{a}_m \in D^{ar(f)}$.

2.3. The Multiset-Structure $P^m(A)$. Let A be a valued τ-structure, $D = D(A)$, and let $m \geq 1$. We define the *multiset-structure*[6] $P^m(A)$ as the valued structure with domain $\left(\!\!\binom{D}{m}\!\!\right)$, where $\left(\!\!\binom{D}{m}\!\!\right)$ denotes the multisets of elements from D of size m, and for every k-ary function symbol $f \in \tau$, and $\alpha_1, \ldots, \alpha_k \in \left(\!\!\binom{D}{m}\!\!\right)$,

$$f^{P^m(A)}(\alpha_1, \ldots, \alpha_k) = \frac{1}{m} \min_{\bar{t}_i \in D^m : [\bar{t}_i] = \alpha_i} \sum_{i=1}^{m} f^A(\bar{t}_1[i], \ldots, \bar{t}_k[i]).$$

The following lemma follows from the definitions (the proof is in the appendix).

Lemma 2.2. *Let A be a valued structure and $m > 1$. Then $P^m(A) \to_f A$ if and only if A has an m-ary totally symmetric fractional polymorphism.*

3. BASIC LINEAR PROGRAMMING RELAXATION

Let I and A be valued structures over a common finite signature τ. Let $X = D(I)$ and $D = D(A)$. The *basic LP relaxation* (BLP) (sometimes also called the *standard*, or *canonical LP relaxation*) has variables $\lambda_{f,\bar{x},\sigma}$ for $f \in \tau$, $\bar{x} \in X^{ar(f)}$, $\sigma : \{\bar{x}\} \to D$; and variables $\mu_x(a)$ for $x \in X, a \in D$.

$$(1) \quad \begin{aligned} \min \quad & \sum_{f,\bar{x}} \sum_{\sigma : \{\bar{x}\} \to D} f^I(\bar{x}) f^A(\sigma(\bar{x})) \lambda_{f,\bar{x},\sigma} \\ \text{s.t.} \quad & \sum_{\sigma : \sigma(x) = a} \lambda_{f,\bar{x},\sigma} = \mu_x(a) \qquad \forall f \in \tau, \bar{x} \in X^{ar(f)}, x \in \{\bar{x}\}, a \in D \\ & \sum_{a \in D} \mu_x(a) = 1 \qquad \forall x \in X \\ & 0 \leq \lambda, \mu \leq 1 \end{aligned}$$

For any fixed A, BLP is polynomial in the size of a given VCSP(A) instance. Let IP be the program obtained from (1) together with the constraints that all variables take values in the range $\{0, 1\}$ rather than $[0, 1]$. This is an integer programming formulation of the original VCSP instance. The interpretation of the variables in IP is as follows: $\mu_x(a) = 1$ iff variable x is assigned value a; $\lambda_{f,\bar{x},\sigma} = 1$ iff constraint f on scope $\bar{x}$ is assigned tuple $\sigma(\bar{x})$. LP (1) is now a relaxation of IP and the question of whether (1) solves a given VCSP instance I is the question of whether IP has a non-zero integrality gap.

4. CHARACTERISATION

Definition 3. *Let $BLP(I, A)$ denote the optimum of (1). We say that BLP solves VCSP(A) if $BLP(I, A) = \mathsf{Opt}_A(I)$ for every instance I of VCSP(A).*

---

[6]A similar structure for $\{0, \infty\}$-valued languages was introduced in [39].

Solving the BLP provides an optimum value of the VCSP. To obtain an assignment achieving this value, we apply self-reduction: Successively try each possible value for a variable and solve the altered LP. Once the new optimum matches the original one, proceed with the next variable.

Theorem 4.1 (Main). *Let A be a valued structure over a finite signature. TFAE:*

(i) BLP solves VCSP(A).
(ii) For every $m > 1$, $P^m(A) \to_f A$.
(iii) For every $m > 1$, A has an m-ary totally symmetric fractional polymorphism.
(iv) For every $n > 1$, A has a fractional polymorphism ω_n such that $\mathrm{supp}(\omega_n)$ generates an n-ary symmetric operation.

The rest of this section is devoted to proving Theorem 4.1. We start with proving $(ii) \Rightarrow (i)$.

Theorem 4.2. *Assume that $P^m(A) \to_f A$ for every $m > 1$. Then BLP solves VCSP(A).*

Proof. Let λ^*, μ^* be an optimal solution to (1). Let M be a positive integer such that $M \cdot \lambda^*$ and $M \cdot \mu^*$ are both integral.

Let $\nu : X \to \left(\binom{D}{M}\right)$ be defined by mapping x to the multiset in which the elements are distributed according to μ_x^*, i.e., the number of occurrences of a in $\nu(x)$ is equal to $M \cdot \mu_x^*(a)$ for each $a \in D$.

Let f be a k-ary function symbol in τ, and let

$$\sum_{\bar{x}, \sigma : \{\bar{x}\} \to D} f^I(\bar{x}) f^A(\sigma(\bar{x})) \lambda_{f,\bar{x},\sigma}^* = \sum_{\bar{x}} f^I(\bar{x}) \Big(\sum_{\sigma : \{\bar{x}\} \to D} \lambda_{f,\bar{x},\sigma}^* f^A(\sigma(\bar{x})) \Big)$$

be the sum of all terms of the objective function in which f occurs; Now, write

$$M \cdot \sum_{\sigma : \{\bar{x}\} \to D} \lambda_{f,\bar{x},\sigma}^* f^A(\sigma(\bar{x})) = f^A(\bar{a}_1) + \cdots + f^A(\bar{a}_M),$$

where the $\bar{a}_i \in D^k$ are such that a $\lambda_{f,\bar{x},\sigma}^*$-fraction are equal to $\sigma(\bar{x})$.

Let $\bar{a}_i' = (\bar{a}_1[i], \ldots, \bar{a}_M[i])$ for $i = 1, \ldots, k$.

$$\begin{aligned}
\sum_{\sigma : \{\bar{x}\} \to D} \lambda_{f,\bar{x},\sigma}^* f^A(\sigma(\bar{x})) &= \frac{1}{M} \sum_{i=1}^{M} f^A(\bar{a}_i) = \frac{1}{M} \sum_{i=1}^{M} f^A(\bar{a}_1'[i], \ldots, \bar{a}_k'[i]) \\
&\geq \frac{1}{M} \min_{\bar{t}_i \in D^M : [\bar{t}_i] = [\bar{a}_i']} \sum_{i=1}^{M} f^A(\bar{t}_1[i], \ldots, \bar{t}_k[i]) = f^{P^M(A)}(\nu(\bar{x})),
\end{aligned}$$

where the last equality follows as the number of a's in $\bar{a}_i'$ is $M \cdot \sum_{\sigma : \sigma(\bar{x}[i]) = a} \lambda_{f,\bar{x},\sigma}^* = M \cdot \mu_{\bar{x}[i]}^*(a)$.

We now have

$$\begin{aligned}
BLP(I, A) &= \sum_{f, \bar{x}} \sum_{\sigma : \{\bar{x}\} \to D} f^I(\bar{x}) f^A(\sigma(\bar{x})) \lambda_{f,\bar{x},\sigma}^* \\
&= \sum_{f \in \tau, \bar{x}} f^I(\bar{x}) \Big(\sum_{\sigma : \{\bar{x}\} \to D} \lambda_{f,\bar{x},\sigma}^* f^A(\sigma(\bar{x})) \Big) \\
&\geq \sum_{f \in \tau, \bar{x}} f^I(\bar{x}) f^{P^M(A)}(\nu(\bar{x})) \\
&= \mathrm{Opt}_{P^M(A)}(I)
\end{aligned}$$

It follows that $\mathrm{Opt}_A(I) \geq BLP(I, A) \geq \mathrm{Opt}_{P^M(A)}(I)$. Since $P^M(A) \to_f A$, the result then follows from Proposition 2.1. $\square$

To prove $(i) \Rightarrow (ii)$, we express the existence of a fractional homomorphism $P^m(A) \to_f A$ as a system of linear inequalities. We then apply a variant of Farkas' Lemma to show that if for some

$m > 1$ there is no such fractional homomorphism, then there exists an instance I of $\mathrm{VCSP}(A)$ with a strictly greater optimum than $\mathrm{BLP}(I, A)$ (the proof is in the appendix).

Theorem 4.3. *Let A be a valued structure and assume that BLP solves $\mathrm{VCSP}(A)$. Then $P^m(A) \to_f A$ for every $m > 1$.*

Lemma 2.2 proves $(ii) \Leftrightarrow (iii)$.

Since $(iii) \Rightarrow (iv)$ follows trivially, it remains to show that $(iv) \Rightarrow (iii)$.

Theorem 4.4. *Let A be a valued structure and assume that for every $n > 1$, A has a fractional polymorphism ω_n that generates an n-ary symmetric operation. Then, for every $m > 1$, A has an m-ary totally symmetric fractional polymorphism.*

Proof. For an m-ary operation g, let $\tilde{g}$ denote the equivalence class of g under the relation:

$$g \sim g' \Leftrightarrow g(x_1, \ldots, x_m) = g'(x_{\pi(1)}, \ldots, x_{\pi(m)}) \text{ for some } \pi \in S_m.$$

Note that we have $|\tilde{g}| = 1$ if and only if g is symmetric.

We say that a fractional operation ω is *weight-symmetric* if

$$\omega(g) = \omega(g') \text{ whenever } g \sim g'.$$

We construct an m-ary totally symmetric fractional polymorphism by building a rooted tree in a number of stages. At each stage of the construction, every node u of the tree contains an m-ary weight-symmetric fractional operation with support on a single equivalence class of $\sim$. For a node u, we will also denote this fractional operation by u. Since u is weight-symmetric, it follows that $u(g) = u(g')$ for all $g, g' \in \mathrm{supp}(u)$. This common weight for the operations in the support of u will be denoted by $w(u)$. A node u with $|\mathrm{supp}(u)| = 1$ will be called *final*.

The following invariants are maintained throughout the construction.

(a) Every non-leaf node has at least one final child.

(b) For every node u, we have $w(u) > 0$.

(c) For every non-leaf node v,

$$\sum_{u \text{ is a child of } v} \|u\|_1 = \|v\|_1.$$

(d) For every non-leaf node v, every $f \in \tau$, and all tuples $\bar{a}_1, \ldots, \bar{a}_m \in D^{ar(f)}$,

$$\sum_g v(g) f^A(g(\bar{a}_1, \ldots, \bar{a}_m)) \geq \sum_{u \text{ is a child of } v} \sum_g u(g) f^A(g(\bar{a}_1, \ldots, \bar{a}_m)).$$

We say that a leaf u *is covered (by v)*, and that v is *a covering node (of u)* if $\mathrm{supp}(u) = \mathrm{supp}(v)$ and u is a (proper) descendant of v. We say that v is a *minimal* covering node of u if no descendant of v is a covering node of u.

At the beginning of the construction, the tree consists of a single root r with $\mathrm{supp}(r)$ being the set of m-ary projections and $w(r) = \frac{1}{m}$. We then apply the following two steps:

- *Expansion:* A leaf u that is not final and not covered is chosen to be expanded. This amounts to adding a finite non-empty set of children to u while maintaining the invariants. The expansion step is repeated until no longer applicable.

- *Pruning:* A leaf that is not final and covered is removed together with a number of internal nodes while maintaining the invariants. The pruning step is repeated until no longer applicable.

Since there is a finite number of m-ary operations, and hence a finite number of equivalence classes of $\sim$, it follows that, eventually, every leaf in the tree that is not final must be covered. Hence, the expansion step is only applicable a finite number of times.

Each round of pruning shrinks the tree by at least one node, but no final leaf is ever removed. Therefore we eventually obtain a tree containing only final leaves, at which time the pruning step is

no longer applicable. Let $\mathcal{L}$ be the set of leaves in the final tree. By repeated application of invariant (d), starting from the root, $\sum_{u \in \mathcal{L}} u$ is then an m-ary totally symmetric fractional polymorphism.

Expansion. We expand a leaf u with $|\mathrm{supp}(u)| = n$ as follows: Let ω be a k-ary fractional polymorphism of A such that $\mathrm{supp}(\omega)$ generates an n-ary symmetric operation t.

We will define a sequence of m-ary weight-symmetric fractional operations ν_i, each with $\|\nu_i\|_1 = \|u\|_1$. Let $\nu_0 = u$. Assume that ν_{i-1} has been defined for some $i \geq 1$. Let $l_{i-1} = \min\{\nu_{i-1}(g) \mid g \in \mathrm{supp}(\nu_{i-1})\}$ be the minimum weight of an operation in the support of ν_{i-1}. The fractional operation ν_i is obtained by subtracting from ν_{i-1} an equal amount of weight from each operation in $\mathrm{supp}(\nu_{i-1})$ and adding this weight as superpositions of ω by all possible choices of operations in ν_{i-1}. The amount subtracted from each operation is $\frac{1}{2}l_{i-1}$ so that every operation in $\mathrm{supp}(\nu_{i-1})$ is also in $\mathrm{supp}(\nu_i)$. Formally ν_i is defined as follows:

$$\nu_i = \nu_{i-1} - \frac{1}{2}l_{i-1}\chi_{i-1} + \sum_{(g_1,\ldots,g_k)\in\mathrm{supp}(\nu_{i-1})^k} \frac{1}{2}l_{i-1}\frac{1}{K}\omega[g_1,\ldots,g_k],$$

where $K = |\mathrm{supp}(\nu_{i-1})|^k$ and χ_{i-1} is the indicator function of $\mathrm{supp}(\nu_{i-1})$.

By definition $\|\nu_i\|_1 = \|\nu_{i-1}\|_1 = \|u\|_1$. To verify that ν_i is weight-symmetric, it suffices to verify that the sum

$$(2) \qquad \sum_{(g_1,\ldots,g_k)\in\mathrm{supp}(\nu_{i-1})^k} \omega[g_1,\ldots,g_k]$$

is weight-symmetric. Let $g \sim g'$, let $\pi \in S_m$ be such that $g(x_1,\ldots,x_m) = g'(x_{\pi(1)},\ldots,x_{\pi(m)})$ and let $g_j'(x_1,\ldots,x_m) = g_j(x_{\pi(1)},\ldots,x_{\pi(m)})$ for $1 \leq j \leq m$. Since ν_{i-1} is weight-symmetric, it follows that $g_i \in \mathrm{supp}(\nu_{i-1})$ if and only if $g_i' \in \mathrm{supp}(\nu_{i-1})$. Therefore the terms $\omega(h)h[g_1,\ldots,g_k]$ in (2) such that $g = h[g_1,\ldots,g_k]$ are in bijection with the terms $\omega(h)h[g_1',\ldots,g_k']$ such that $g' = h[g_1',\ldots,g_k']$. So the fractional operation in (2) assigns the same weight to g and g'.

Let e be an expression for t consisting of superpositions of projections and operations from $\mathrm{supp}(\omega)$. We recursively define the *nested depth*, $d = d(e)$, of e as follows: $d(p) = 0$ for every projection p; and $d(h[g_1,\ldots,g_k]) = 1 + \max_{1 \leq i \leq k} d(g_i)$.

Let $\mathrm{supp}(u) = \{g_1,\ldots,g_n\}$. Using $\mathrm{supp}(\nu_0) = \mathrm{supp}(u)$ and the fact that $\mathrm{supp}(\nu_i)$ contains all superpositions of operations in $\mathrm{supp}(\nu_{i-1})$, it follows that $t[g_1,\ldots,g_n] \in \mathrm{supp}(\nu_d)$. Now, we add a child v to u for every equivalence class in the set $\{\tilde{g} \mid g \in \mathrm{supp}(\nu_d)\}$. For an added child v with $\mathrm{supp}(v) = \tilde{g}$, we let $w(v) = \nu_d(g)$.

Invariant (a) holds as $t[g_1,\ldots,g_n] \in \mathrm{supp}(\nu_d)$ is symmetric: for all $\pi \in S_m$ there is a $\pi' \in S_n$ such that $t[g_1,\ldots,g_n](x_{\pi(1)},\ldots,x_{\pi(m)}) = t[g_{\pi'(1)},\ldots,g_{\pi'(n)}](x_1,\ldots,x_m) = t[g_1,\ldots,g_n](x_1,\ldots,x_m)$. Invariants (b) and (c) hold by construction. For each $i \geq 1$, we have

$$\sum_g \nu_{i-1}(g)f^A(g(\bar{a}_1,\ldots,\bar{a}_m)) \geq \sum_g \nu_i(g)f^A(g(\bar{a}_1,\ldots,\bar{a}_m))$$

for all $f \in \tau$ and $\bar{a}_1,\ldots\bar{a}_m \in D^{ar(f)}$. Therefore invariant (d) also holds after expanding u.

Pruning. The pruning step maintains an additional invariant, namely that every leaf that is not final is covered. Pruning is accomplished as follows. Pick a minimal covering node v. Let $\nu = \nu_v + \nu_\perp$ be the fractional operation induced by the leaves in the subtree rooted at v, where ν_v is the part of ν with the same support as v and $\nu_\perp$ is the part of ν with support disjoint from v. Inductively, by invariant (d),

$$\sum_g v(g)f^A(g(\bar{a}_1,\ldots,\bar{a}_m)) \geq \sum_g \nu_v(g)f^A(g(\bar{a}_1,\ldots,\bar{a}_m)) + \sum_g \nu_\perp(g)f^A(g(\bar{a}_1,\ldots,\bar{a}_m)),$$

for all $f \in \tau$ and $\bar{a}_1, \ldots \bar{a}_m \in D^{ar(f)}$. We simplify this inequality as follows.

$$\sum_g v(g) f^A(g(\bar{a}_1, \ldots, \bar{a}_m)) \geq \sum_g \frac{1}{1-\kappa} \nu_\perp(g) f^A(g(\bar{a}_1, \ldots, \bar{a}_m)),$$

where $\kappa = \|\nu_v\|_1 / \|v\|_1$.

Remove all nodes below v and add a new child u to v for every equivalence class in the set $\{\tilde{g} \mid g \in \text{supp}(\nu_\perp)\}$. For an added child u with $\text{supp}(u) = \tilde{g}$, we let $w(u) = \frac{1}{1-\kappa} \nu_\perp(g)$.

By invariant (a) the node v is guaranteed to have at least one final child (leaf). Hence, by invariant (b), $\nu_\perp$ is not identically 0. By induction on (c), it follows that $\|v\|_1 > \|\nu_v\|_1$, so $\kappa < 1$ and the new weights are defined and positive. So invariant (b) holds. Furthermore,

$$\sum_{u \text{ is a child of } v} \|u\|_1 = \frac{1}{1-\kappa} \|\nu_\perp\|_1 = \frac{1}{1-\kappa} (\|\nu\|_1 - \|\nu_v\|_1) = \frac{1}{1-\kappa} (\|v\|_1 - \|\nu_v\|_1) = \|v\|_1,$$

so invariant (c) holds. Invariant (d) holds by construction. Finally, invariant (a) also holds since for any final child u of v, there is still a child of v with support on the same equivalence class as u.

Only nodes in the subtree rooted at v have been removed and every child added to v has the same support as a previous leaf of this subtree. Every such leaf u that was not final was covered by a node above v in the tree. Hence, all leaves in the tree that are not final are still covered. $\square$

5. GENERAL-VALUED STRUCTURES

The valued structures we have dealt with so far were in fact *finite-valued*; i.e., for each function symbol $f \in \tau$, the range of f^A was $\mathbb{Q}_{\geq 0}$. We now discuss how our result can be extended to the *general-valued* case, in which for each function symbol $f \in \tau$, the range of f^A is $\overline{\mathbb{Q}}_{\geq 0} = \mathbb{Q}_{\geq 0} \cup \{\infty\}$. (We define $c + \infty = \infty + c = \infty$ for all $c \in \overline{\mathbb{Q}}_{\geq 0}$, and $0\infty = \infty 0 = 0$.)

Inspired by the OSAC algorithm [13], the algorithm for general-valued structures, denoted by BLP_g, works in two stages. Firstly, the instance is made arc consistent using a standard arc consistency algorithm [23].[7] Secondly, BLP (i.e., linear program (1) from Section 3) is solved.

Definition 4. *Let A be a general-valued τ-structure, and let $D = D(A)$. An m-ary operation $g : D^m \to D$ is a* polymorphism *of A if for every function symbol $f \in \tau$ and tuples $\bar{a}_1, \ldots, \bar{a}_m \in D^{ar(f)}$, it holds that $\text{Feas}(f^A(g(\bar{a}_1, \ldots, \bar{a}_m))) \leq \sum_{i=1}^m \text{Feas}(f^A(\bar{a}_i))$, where $\text{Feas}(\infty) = \infty$ and $\text{Feas}(c) = 0$ for any $c \in \mathbb{Q}_{\geq 0}$.*

From Definitions 1 and 4, if ω is a fractional polymorphism of a general-valued structure A, then every $g \in \text{supp}(\omega)$ is a polymorphism of A. In fact, any operation g that is generated by $\text{supp}(\omega)$ (i.e., g belongs to the clone generated by $\text{supp}(\omega)$) is a polymorphism of A. (For finite-valued structures, trivially, every operation g is a polymorphism.)

Arc consistency (also known as $(1, k)$-consistency) [23] is a decision procedure for precisely those $\{0, \infty\}$-valued structures A that are closed under a *set function* $g : 2^{D(A)} \setminus \{\emptyset\} \to D(A)$ [19, 21]. This condition has recently been shown to be equivalent to the requirement that A should have symmetric polymorphisms of all arities [39].

Our main theorem (Theorem 4.1) also holds for general-valued structures:

Theorem 5.1. *Let A be a general-valued structure over a finite signature. TFAE:*
- *(i) BLP_g solves VCSP(A).*
- *(ii) For every $m > 1$, $P^m(A) \to_f A$.*
- *(iii) For every $m > 1$, A has an m-ary totally symmetric fractional polymorphism.*
- *(iv) For every $n > 1$, A has a fractional polymorphism ω_n such that $\text{supp}(\omega_n)$ generates an n-ary symmetric operation.*

---

[7]The algorithm from [23] is sometimes called generalised arc consistency algorithm to emphasise the fact that it works for CSPs of arbitrary arities, not only for binary CSPs [41].

Proof. Note that (ii) implies that A has symmetric polymorphisms of all arities. This follows from Lemma 2.2, which holds for general-valued structures, and the above-mentioned fact that any $g \in \text{supp}(\omega)$, where ω is a fractional polymorphism of A, is a polymorphism of A. The same argument guarantees symmetric polymorphisms of all arities in (iii) and (iv); in (iv), we use that fact than any operation generated by $\text{supp}(\omega_n)$ is a polymorphism of A.

Theorem 4.2 proves $(ii) \Rightarrow (i)$ since the assumption of having symmetric polymorphisms of all arities guarantees a feasible solution to (1). From the discussion above on arc consistency, if BLP_g solves $\text{VCSP}(A)$, then A has symmetric polymorphisms of all arities since arc consistency decides the existence of a finite-valued solution. Furthermore, the proof of Theorem 4.3 shows that having symmetric polymorphisms of all arities but not having a fractional homomorphism $P^m(A) \to_f A$ implies that BLP_g does not solve $\text{VCSP}(A)$. This gives $(i) \Rightarrow (ii)$. $(ii) \Leftrightarrow (iii)$ and $(iii) \Leftrightarrow (iv)$ are proved the same way as in Theorem 4.1, by Lemma 2.2 and Theorem 4.4, respectively. $\square$

Remark 1. *BLP_g for general-valued structures uses the arc consistency algorithm [23] and BLP. Since Kun et al [39] have shown that BLP solves CSPs (i.e., $\{0, \infty\}$-valued VCSPs), if represented by $\{0, 1\}$-valued structures, of width 1 – thus providing an alternative to the standard arc consistency algorithm [23] – a different approach is to combine Theorem 4.1 from this paper with [39] and solve general-valued structures using only BLP with an amended objective function which takes care of infinite costs using a large (but polynomial), instance-dependent constant.*

6. Tractable Valued Constraint Languages

As before, we denote $D = D(A)$. A *binary multimorphism* [11] of a valued structure A is a pair $\langle g_1, g_2 \rangle$ of binary functions $g_1, g_2 : D^2 \to D$ such that for every function symbol $f \in \tau$ and tuples $\bar{a}_1, \bar{a}_2 \in D^{ar(f)}$, it holds that

$$f^A(g_1(\bar{a}_1, \bar{a}_2)) + f^A(g_2(\bar{a}_1, \bar{a}_2)) \leq f^A(\bar{a}_1) + f^A(\bar{a}_2).$$

(Multimorphisms are a special case of fractional polymorphisms.) Since any semi-lattice operation[8] generates symmetric operations of all arities, we get:

Corollary 6.1 (of Theorem 5.1)**.** *Let A be a valued structure with a binary multimorphism $\langle g_1, g_2 \rangle$ where either g_1 or g_2 is a semi-lattice operation. Then A is tractable.*

We now give examples of valued structures (i.e., valued constraint languages) defined by such binary multimorphisms.

Example 1. Let $(D; \wedge, \vee)$ be an *arbitrary* lattice on D. Assume that a valued structure A has the multimorphism $\langle \wedge, \vee \rangle$. Then $\text{VCSP}(A)$ is tractable. The tractability of A was previously known only for distributive lattices [28, 48] and (finite-valued) diamonds [38], see also [37].

Example 2. A pair of operations $\langle g_1, g_2 \rangle$ is called a symmetric tournament pair (STP) if both g_1 and g_2 are commutative, conservative ($g_1(x, y) \in \{x, y\}$ and $g_2(x, y) \in \{x, y\}$ for all $x, y \in D$), and $g_1(x, y) \neq g_2(x, y)$ for all $x, y \in D$. Let A be a finite-valued structure with an STP multimorphism $\langle g_1, g_2 \rangle$. It is known that if a finite-valued structure admits an STP multimorphism, it also admits a submodularity multimorphism. This result is implicitly contained in [10].[9] Therefore, BLP solves any instance from $\text{VCSP}(A)$.

Example 3. Assume that a valued structure A is bisubmodular [24]. This means that $D = \{0, 1, 2\}$ and A has a multimorphism $\langle \min_0, \max_0 \rangle$ [11], where $\min_0(x, x) = x$ for all $x \in D$ and $\min_0(x, y) = 0$ for all $x, y \in D, x \neq y$; $\max_0(x, y) = 0$ if $0 \neq x \neq y \neq 0$ and $\max_0(x, y) = \max(x, y)$ otherwise,

---

[8]A semi-lattice operation is associative, commutative, and idempotent.

[9]Namely, the STP might contain cycles, but [10, Lemma 7.15] tells us that on cycles we have, in the finite-valued case, only unary cost functions. It follows that the cost functions admitting the STP must be submodular with respect to some total order.

where max returns the larger of its two arguments with respect to the normal order of integers. Since $\min_0$ is a semi-lattice operation, A is tractable. The tractability of (finite-valued) A was previously known only using a general algorithm for bisubmodular functions given by an oracle [24, 43].

Example 4. Assume that a valued structure A is *weakly tree-submodular* on an *arbitrary* tree [35]. The meet (which is defined as the highest common ancestor) is again a semi-lattice operation. The same holds for *strongly tree-submodular* structures since strong tree-submodularity implies weak tree-submodularity [35]. The tractability of weakly tree-submodular valued structures was previously known only for chains and forks [35]. The tractability of strongly tree-submodular valued structures was previously known only for binary trees [35].

Example 5. Note that the previous example applies to *all* trees, not just binary ones. In particular, it applies to the tree consisting of one root with k children. This is equivalent to structures with $D = \{0, 1, \ldots, k\}$ and the multimorphism $\langle \min_0, \max_0 \rangle$ from Example 3. This is a natural generalisation of submodular ($k = 1$) and bisubmodular ($k = 2$) functions, known as k-submodular functions [26]. The tractability of k-submodular valued structures for $k > 2$ was previously open.

Example 6. Let b and c be two distinct elements of D and let $(D; <)$ be a partial order which relates all pairs of elements except for b and c. A pair $\langle g_1, g_2 \rangle$, where $g_1, g_2 : D^2 \to D$ are two binary operations, is a *1-defect* multimorphism if g_1 and g_2 are both commutative and satisfy the following conditions:
- If $\{x, y\} \neq \{b, c\}$, then $g_1(x, y) = x \wedge y$ and $g_2(x, y) = x \vee y$.
- If $\{x, y\} = \{b, c\}$, then $\{g_1(x, y), g_2(x, y)\} \cap \{x, y\} = \emptyset$, and $g_1(x, y) < g_2(x, y)$.

The tractability of valued structures that have a 1-defect multimorphism has recently been shown in [32]. We now show that valued structures with a 1-defect multimorphism are solvable by BLP_g.

Without loss of generality, we assume that $g_1(b, c) < b, c$ and write $g = g_1$. (Otherwise, $g_2(b, c) > b, c$, and g_2 is used instead.) Using g, we construct a symmetric m-ary operation $f(x_1, \ldots, x_m)$.

Let $f_1, \ldots, f_M$ be the $M = \binom{m}{2}$ terms $g(x_i, x_j)$. Let $f = g(f_1, g(f_2, \ldots, g(f_{M-1}, f_M) \ldots))$. There are three possible cases:
- $\{b, c\} \not\subseteq x_1, \ldots, x_m$. Then g acts as $\wedge$, which is a semi-lattice operation, hence so does f.
- $\{b, c\} \subseteq \{x_1, \ldots, x_m\}$ and $g(b, c) \leq x_1, \ldots, x_m$. Then $f_i = g(b, c)$ for some $1 \leq i \leq M$, and $g(f_i, f_j) = g(b, c)$ for all $1 \leq j \leq M$, so $f(x_1, \ldots, x_m) = g(b, c)$.
- $\{b, c\} \subseteq \{x_1, \ldots, x_m\}$ and there is a variable x_p for some $1 \leq p \leq m$ such that $x_p \leq g(b, c)$ and $x_p \leq x_1, \ldots, x_m$. Then $g(x_p, x_q) = x_p$ for all $1 \leq q \leq m$ so $f_i = x_p$ for some $1 \leq i \leq M$ and $g(f_i, f_j) = x_p$ for all $1 \leq j \leq M$, so $f(x_1, \ldots, x_m) = x_p$.

7. CONCLUSIONS

We have characterised precisely for which valued structures the basic linear programming relaxation (BLP) is a decision procedure. This implies tractability of several previously open classes of VCSPs including several generalisations of submodularity. In fact, BLP solves *all known* tractable finite-valued structures.

The main result does not give a decidability criterion for testing whether a valued structure is solvable by BLP. Interestingly, all known tractable finite-valued structures have a binary multimorphism. It is possible that every (finite-)valued structure solvable by BLP admits a fixed-arity multimorphism, which would give a polynomial-time checkable condition.

An intriguing open question is whether our tractability results hold in the oracle-value model; that is, for objective functions which are not given explicitly as a sum of cost functions, but only by an oracle. For instance, the *maximisation problem* for submodular functions on distributive lattices, known to be NP-complete, allows for good approximation algorithms in both models [22].

References

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

[2] Libor Barto and Marcin Kozik. Constraint Satisfaction Problems of Bounded Width. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS'09)*, pages 461–471. IEEE Computer Society, 2009.

[3] Libor Barto and Marcin Kozik. Robust Satisfiability of Constraint Satisfaction Problems. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC'12)*. ACM, 2012.

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

[5] Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. Semiring-based Constraint Satisfaction and Optimisation. *Journal of the ACM*, 44(2):201–236, 1997.

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

[7] Andrei Bulatov, Andrei Krokhin, and Peter Jeavons. Classifying the Complexity of Constraints using Finite Algebras. *SIAM Journal on Computing*, 34(3):720–742, 2005.

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

[9] David A. Cohen, Martin C. Cooper, and Peter G. Jeavons. An Algebraic Characterisation of Complexity for Valued Constraints. In *Proceedings of the 12th International Conference on Principles and Practice of Constraint Programming (CP'06)*, volume 4204 of *Lecture Notes in Computer Science*, pages 107–121. Springer, 2006.

[10] David A. Cohen, Martin C. Cooper, and Peter G. Jeavons. Generalising submodularity and Horn clauses: Tractable optimization problems defined by tournament pair multimorphisms. *Theoretical Computer Science*, 401(1-3):36–51, 2008.

[11] David A. Cohen, Martin C. Cooper, Peter G. Jeavons, and Andrei A. Krokhin. The Complexity of Soft Constraint Satisfaction. *Artificial Intelligence*, 170(11):983–1016, 2006.

[12] David A. Cohen, Páidí Creed, Peter G. Jeavons, and Stanislav Živný. An algebraic theory of complexity for valued constraints: Establishing a Galois connection. In *Proceedings of the 36th International Symposium on Mathematical Foundations of Computer Science (MFCS'11)*, volume 6907 of *Lecture Notes in Computer Science*, pages 231–242. Springer, 2011.

[13] Martin C. Cooper, Simon de Givry, Martí Sánchez, Thomas Schiex, Matthias Zytnicki, and Tomáš Werner. Soft arc consistency revisited. *Artificial Intelligence*, 174(7–8):449–478, 2010.

[14] Martin C. Cooper and Thomas Schiex. Arc consistency for soft constraints. *Artificial Intelligence*, 154(1-2):199–227, 2004.

[15] Yves Crama and Peter L. Hammer. *Boolean Functions - Theory, Algorithms, and Applications*. Cambridge University Press, 2011.

[16] Páidí Creed and Stanislav Živný. On minimal weighted clones. In *Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming (CP'11)*, volume 6876 of *Lecture Notes in Computer Science*, pages 210–224. Springer, 2011.

[17] Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity Classification of Boolean Constraint Satisfaction Problems*, volume 7 of *SIAM Monographs on Discrete Mathematics and Applications*. SIAM, 2001.

[18] Víctor Dalmau and Andrei Krokhin. Robust satisfiability for CSPs: algorithmic and hardness results. 2011. In preparation.

[19] Víctor Dalmau and Justin Pearson. Set Functions and Width 1 Problems. In *Proceedings of the 5th International Conference on Constraint Programming (CP'99)*, volume 1713 of *Lecture Notes in Computer Science*, pages 159–173. Springer, 1999.

[20] Vladimir Deineko, Peter Jonsson, Mikael Klasson, and Andrei Krokhin. The approximability of Max CSP with fixed-value constraints. *Journal of the ACM*, 55(4), 2008.

[21] Tomás Feder and Moshe Y. Vardi. The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: A Study through Datalog and Group Theory. *SIAM Journal on Computing*, 28(1):57–104, 1998.

[22] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing Non-monotone Submodular Functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

[23] Eugene C. Freuder. Synthesizing Constraint Expressions. *Communications of the ACM*, 21(11):958–966, 1978.

[24] Satoru Fujishige and Satoru Iwata. Bisubmodular Function Minimization. *SIAM Journal on Discrete Mathematics*, 19(4):1065–1073, 2005.

[25] David Gale. *The Theory of Linear Economic Models*. McGraw-Hill, 1960.

[26] Anna Huber and Vladimir Kolmogorov. Towards Minimizing k-Submodular Functions. In *Proceedings of the 2nd International Symposium on Combinatorial Optimization (ISCO'12)*, 2012.

[27] Pawel M. Idziak, Petar Markovic, Ralph McKenzie, Matthew Valeriote, and Ross Willard. Tractability and learnability arising from algebras with few subpowers. *SIAM Journal on Computing*, 39(7):3023–3037, 2010.

[28] Satoru Iwata, Lisa Fleischer, and Satoru Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM*, 48(4):761–777, 2001.

[29] Peter G. Jeavons. On the Algebraic Structure of Combinatorial Problems. *Theoretical Computer Science*, 200(1-2):185–204, 1998.

[30] Peter G. Jeavons, David A. Cohen, and Marc Gyssens. Closure Properties of Constraints. *Journal of the ACM*, 44(4):527–548, 1997.

[31] Peter Jonsson, Andrei A. Krokhin, and Fredrik Kuivinen. Hard constraint satisfaction problems have hard gaps at location 1. *Theoretical Computer Science*, 410(38-40):3856–3874, 2009.

[32] Peter Jonsson, Fredrik Kuivinen, and Johan Thapper. Min CSP on Four Elements: Moving Beyond Submodularity. In *Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming (CP'11)*, volume 6876 of *Lecture Notes in Computer Science*, pages 438–453. Springer, 2011.

[33] Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David Williamson. The approximability of constraint satisfaction problems. *SIAM Journal on Computing*, 30(6):1863–1920, 2001.

[34] Subhash Khot. On the unique games conjecture (invited survey). In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity (CCC'10)*, pages 99–121. IEEE Computer Society, 2010.

[35] Vladimir Kolmogorov. Submodularity on a tree: Unifying $l^\sharp$-convex and bisubmodular functions. In *Proceedings of the 36th International Symposium on Mathematical Foundations of Computer Science (MFCS'11)*, volume 6907 of *Lecture Notes in Computer Science*, pages 400–411. Springer, 2011.

[36] Vladimir Kolmogorov and Stanislav Živný. The complexity of conservative valued CSPs. In *Proceedings of of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'12)*, pages 750–759. SIAM, 2012. Full version available on arXiv:1110.2809.

[37] Andrei Krokhin and Benoit Larose. Maximizing Supermodular Functions on Product Lattices, with Application to Maximum Constraint Satisfaction. *SIAM Journal on Discrete Mathematics*, 22(1):312–328, 2008.

[38] Fredrik Kuivinen. On the complexity of submodular function minimisation on diamonds. *Discrete Optimization*, 8(3):459–477, 2011.

[39] Gábor Kun, Ryan O'Donnell, Suguru Tamaki, Yuichi Yoshida, and Yuan Zhou. Linear programming, width-1 CSPs, and robust satisfaction. In *Proceedings of the 3rd Innovations in Theoretical Computer Science (ITCS'12)*, pages 484–495. ACM, 2012.

[40] Steffen L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

[41] Alan K. Mackworth. Consistency in Networks of Relations. *Artificial Intelligence*, 8:99–118, 1977.

[42] Olvi L. Mangasarian. *Nonlinear programming*. Classics in applied mathematics. SIAM, 1994.

[43] S. Thomas McCormick and Satoru Fujishige. Strongly polynomial and fully combinatorial algorithms for bisubmodular function minimization. *Mathematical Programming*, 122(1):87–120, 2010.

[44] Ugo Montanari. Networks of Constraints: Fundamental properties and applications to picture processing. *Information Sciences*, 7:95–132, 1974.

[45] Prasad Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC'08)*, pages 245–254. ACM, 2008.

[46] Thomas J. Schaefer. The Complexity of Satisfiability Problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC'78)*, pages 216–226. ACM, 1978.

[47] Thomas Schiex, Hélène Fargier, and Gérard Verfaillie. Valued Constraint Satisfaction Problems: Hard and Easy Problems. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, 1995.

[48] Alexander Schrijver. A Combinatorial Algorithm Minimizing Submodular Functions in Strongly Polynomial Time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355, 2000.

[49] Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

Appendix A. Proof of Lemma 2.2

Lemma A.1. *Let A be a valued structure and $m > 1$. Then $P^m(A) \to_f A$ if and only if A has an m-ary totally symmetric fractional polymorphism.*

Proof. Let $D = D(A)$.

($\Rightarrow$) Let ω be a fractional homomorphism from $P^m(A)$ to A. Let $h : D^m \to \left(\binom{D}{m}\right)$ be the function that sends a tuple $(a_1, \ldots, a_m) \in D^m$ to its multiset $[a_1, \ldots, a_m]$. Then the composition $\omega \circ h$ is the desired fractional polymorphism.

($\Leftarrow$) Let ω be an m-ary totally symmetric fractional polymorphism. Every operation $g \in \mathrm{supp}(\omega)$ induces a function $g' : \left(\binom{D}{m}\right) \to D$. Define ω' as the fractional homomorphism with $\omega'(g') = \omega(g)$. Let f be a k-ary function symbol in the signature of A, let $\alpha_1, \ldots, \alpha_k \in \left(\binom{D}{m}\right)$ be arbitrary, and pick $\bar{a}_i$ with $[\bar{a}_i] = \alpha_i$ that minimises $\sum_{i=1}^{m} f^A(\bar{a}_1[i], \ldots, \bar{a}_k[i])$, for $1 \leq i \leq k$.

$$
\begin{aligned}
\sum_{g'} \omega'(g') f^A(g'(\alpha_1), \ldots, g'(\alpha_k)) &= \sum_{g} \omega(g) f^A(g(\bar{a}_1), \ldots, g(\bar{a}_k)) \\
&\leq \frac{1}{m} \sum_{i=1}^{m} f^A(a_1[i], \ldots, a_k[i]) \\
&= \frac{1}{m} \min_{\bar{t}_i \in D^m : [\bar{t}_i] = \alpha_i} \sum_{i=1}^{m} f^A(\bar{t}_1[i], \ldots, \bar{t}_k[i]) \\
&= \frac{1}{m} \sum_{i=1}^{m} f^{P^m(A)}(\alpha_1, \ldots, \alpha_k).
\end{aligned}
$$

Hence, $P^m(A) \to_f A$. $\qquad\square$

Appendix B. Proof of Theorem 4.3

The following variant of Farkas' Lemma is due to Gale [25] (cf. Mangasarian [42]).

Lemma B.1. *Let $A \in \mathbb{R}^{m \times n}$ and $\bar{b} \in \mathbb{R}^m$. Then exactly one of the two holds:*
- $A\bar{x} \leq \bar{b}$ *for some $\bar{x} \in \mathbb{R}^n$; or*
- $A^T \bar{y} = 0$, $\bar{b}^T \bar{y} = -1$ *for some $\bar{y} \in \mathbb{R}_{\geq 0}$.*

Theorem B.2. *Let A be a valued structure and assume that BLP solves $\mathrm{VCSP}(A)$. Then $P^m(A) \to_f A$ for every $m > 1$.*

Proof. Let τ be the signature of A and let $D = D(A)$. We prove the contrapositive. Assume that there is an integer $m > 1$ such that $P^m(A)$ does not have a fractional homomorphism to A. Let Ω denote the set of functions from $\left(\binom{D}{m}\right)$ to D. We are assuming that the following system of inequalities does not have a solution $\omega : \Omega \to \mathbb{Q}_{\geq 0}$.

$$
\begin{aligned}
\sum_{g \in \Omega} \omega(g) f^A(g(\bar{\alpha})) &\leq f^{P^m(A)}(\bar{\alpha}) \quad \forall f \in \tau, \bar{\alpha} \in \left(\binom{D}{m}\right)^{ar(f)} \\
\sum_{g \in \Omega} \omega(g) &= 1 \\
\omega(g) &\geq 0 \quad \forall g \in \Omega.
\end{aligned}
$$

In order to apply Lemma B.1, we rewrite the equality $\sum_g \omega(g) = 1$ into two inequalities $\sum_g \omega(g) \leq 1$ and $-\sum_g \omega(g) \leq -1$. The last set of inequalities are rewritten to the form $-\omega(g) \leq 0$ for each $g \in \Omega$. We have one variable for each inequality, i.e., $y(f, \bar{\alpha})$ for $f \in \tau$, and $\bar{\alpha} \in \left(\binom{D}{m}\right)^{ar(f)}$.

Additionally, we have two variables z_+, z_- for the two inequalities involving the constant 1 and one variable $w(g)$ for each $g \in \Omega$.

$$\sum_{f,\bar{\alpha}} y(f,\bar{\alpha}) f^A(g(\bar{\alpha})) + z_+ - z_- - w(g) = 0$$

$$\sum_{f,\bar{\alpha}} y(f,\bar{\alpha}) f^{Pm(A)}(\bar{\alpha}) + z_+ - z_- = -1$$

$$y, z_+, z_-, w \geq 0$$

We can isolate $z_+ + z_-$ in the last equality,

$$z_+ + z_- = -1 - \sum_{f,\bar{\alpha}} y(f,\bar{\alpha}) f^{Pm(A)}(\bar{\alpha}),$$

which substituted into the first set of equalities implies that there is a solution $y(f,\bar{\alpha}), w(g)$ such that, for each $g \in \Omega$,

$$\sum_{f,\bar{\alpha}} y(f,\bar{\alpha}) f^A(g(\bar{\alpha})) = w(g) + 1 + \sum_{f,\bar{\alpha}} y(f,\bar{\alpha}) f^{Pm(A)}(\bar{\alpha}).$$

We therefore find that there is a solution to the following system:

$$\sum_{f,\bar{\alpha}} y(f,\bar{\alpha}) f^A(g(\bar{\alpha})) > \sum_{f,\bar{\alpha}} y(f,\bar{\alpha}) f^{Pm(A)}(\bar{\alpha}) \quad \forall g \in \Omega$$

$$y(f,\bar{\alpha}) \geq 0 \quad \forall f, \bar{\alpha}.$$

Let I be the instance on variables $\left(\!\binom{D}{m}\!\right)$. For each k-ary function symbol $f \in \tau$, and $\bar{\alpha} \in \left(\!\binom{D}{m}\!\right)^{ar(f)}$, define

$$f^I(\bar{\alpha}) = y(f, \bar{\alpha}).$$

We now give a solution λ, μ to the basic LP (1) with an objective value equal to the right-hand side of (3). Each variable $\mu_\alpha(a)$ is assigned the value of the multiplicity of a in α divided by m. Given $f, \bar{\alpha}$, let $\bar{t}_1, \ldots, \bar{t}_k \in D^m$ be such that $f^{Pm(A)}(\bar{\alpha}) = \frac{1}{m} \sum_{i=1}^m f^A(\bar{t}_1[i], \ldots, \bar{t}_k[i])$, and assign values to the λ-variables as follows:

$$\lambda_{f,\bar{\alpha},\sigma} = \frac{1}{m} |\{i \mid \sigma(\bar{\alpha}[j]) = \bar{t}_j[i] \text{ for all } j\}|$$

Note that $\sum_{\sigma:\sigma(\bar{\alpha}[j])=a} \lambda_{f,\bar{\alpha},\sigma} = \mu_{\bar{\alpha}[j]}(a)$ for all $1 \leq j \leq k$ and $a \in D$. Furthermore, λ is defined so that $f^{Pm(A)}(\bar{\alpha}) = \sum_{\sigma:\{\bar{\alpha}\}\to D} f^A(\sigma(\bar{\alpha})) \lambda_{f,\bar{\alpha},\sigma}$. Hence, the variables λ, μ satisfy the basic LP (1), and we have

$$(3) \qquad BLP(I,A) \leq \sum_{f,\bar{\alpha}} f^I(\bar{\alpha}) \sum_{\sigma:\{\bar{\alpha}\}\to D} f^A(\sigma(\bar{\alpha})) \lambda_{f,\bar{\alpha},\sigma} = \sum_{f,\bar{\alpha}} f^I(\bar{\alpha}) f^{Pm(A)}(\bar{\alpha}),$$

where the sums are over $f \in \tau$ and $\bar{\alpha} \in \left(\!\binom{D}{m}\!\right)^{ar(f)}$.

It now follows from (3) and (3) that the measure of any solution $g : \left(\!\binom{D}{m}\!\right) \to D$ to I is strictly greater than $BLP(I, A)$. Consequently, BLP does not solve $VCSP(A)$. $\qquad\square$

Appendix C. Optimal Soft Arc Consistency

In this section we define optimal soft arc consistency, which is closely related to BLP given by (1) in Section 3.

Let I and A be valued structures over a common finite signature. Let $X = D(I)$ and $D = D(A)$. We will group the terms of an instance with respect to their *scope*. Let $S \subseteq X$. The terms of this scope are those of the form $f^I(\bar{x}) f^A(\sigma(\bar{x}))$, where $\{\bar{x}\} = S$, and $ar(f) = |\bar{x}|$. For each scope S, $x \in S$, and $\sigma : S \to D$, we have a variable $y_{S,x}(\sigma(x))$. For each $x \in X$, we have a variable z_x.

Establishing *optimal soft arc consistency* (OSAC) amounts to solving the following linear program [13]:

$$
(4) \quad
\begin{aligned}
\max \quad & \sum_x z_x \\
\text{s.t.} \quad & \sum_{\{\bar{x}\}=S,f} f^I(\bar{x}) f^A(\sigma(\bar{x})) - \sum_{x \in S} y_{S,x}(\sigma(x)) \geq 0 && \forall S \subseteq X, \sigma : S \to D \\
& \sum_u u^I(x) u^A(\sigma(x)) - z_x + \sum_{S:x \in S} y_{S,x}(\sigma(x)) \geq 0 && \forall x \in X, \sigma : \{x\} \to D
\end{aligned}
$$

We refer the reader to [13] for more details, but the idea behind (4) is that it gives the maximum lower bound on $\mathsf{Opt}_A(I)$ among all arc-consistency closures of the given instance I, where the closure is obtained by repeated calls of three basic operations called Extend, Project, and UnaryProject.

We will be interested in the dual of (4). The dual has variables $\lambda_{S,\sigma}$ for $S \subseteq X$ and $\sigma : S \to D$, and variables $\mu_x(a)$ for $x \in X, a \in D$.

$$
(5) \quad
\begin{aligned}
\min \quad & \sum_{S \subseteq X, \sigma} \Big(\sum_{\{\bar{x}\}=S,f} f^I(\bar{x}) f^A(\sigma(\bar{x})) \Big) \lambda_{S,\sigma} + \sum_{x \in X, \sigma} \Big(\sum_u u^I(x) u^A(\sigma(x)) \Big) \mu_x(\sigma(x)) \\
\text{s.t.} \quad & \sum_{\sigma : \sigma(x)=a} \lambda_{S,\sigma} = \mu_x(a) && \forall S \subseteq X, x \in S, a \in D \\
& \sum_{a \in D} \mu_x(a) = 1 && \forall x \in X \\
& \lambda, \mu \geq 0
\end{aligned}
$$

Note that (5) is a tighter relaxation than (1) as it has only one variable λ for all constraints with the same scope (seen as a set) of variables. In (1), different constraints have different variables λ even if the scopes (seen as sets) are the same. Consequently, OSAC solves all problems solved by BLP. Moreover, since the basic SDP relaxation of a VCSP(A) instance is tighter than BLP [45], the basic SDP relaxation also solves all tractable cases identified in this paper.