Skew Bisubmodularity and Valued CSPs

Anna Huber, Andrei Krokhin, and Robert Powell
Durham University, Durham, DH1 3LE, UK
firstname.lastname@durham.ac.uk

Abstract

An instance of the (finite-)Valued Constraint Satisfaction Problem (VCSP) is given by a finite set of variables, a finite domain of values, and a sum of (rational-valued) functions, each function depending on a subset of the variables. The goal is to find an assignment of values to the variables that minimises the sum.

We study (assuming that **PTIME** $\neq$ **NP**) how the complexity of this very general problem depends on the functions allowed in the instances. The case when the variables can take only two values was classified by Cohen et al.: essentially, submodular functions give rise to the only tractable case, and any non-submodular function can be used to express, in a certain specific sense, the NP-hard Max Cut problem.

We investigate the case when the variables can take three values. We identify a new infinite family of conditions that includes bisubmodularity as a special case and which can collectively be called skew bisubmodularity. By a recent result of Thapper and Živný, this condition implies that the corresponding VCSP can be solved by linear programming.

We prove that submodularity with respect to a total order and skew bisubmodularity give rise to the only tractable cases, and, in all other cases, again, Max Cut can be expressed. We also show that our characterisation of tractable cases is tight, that is, none of the conditions can be omitted. Thus, our results provide a new dichotomy theorem in constraint satisfaction research, and lead to a whole series of intriguing open problems in submodularity research.

1 Introduction

What are the classes of discrete functions that admit an efficient minimisation algorithm? To answer this kind of general question in a meaningful way, one needs to fix a formal setting. One popular general setting considers classes of set functions (also known as pseudo-Boolean functions) $f : \{0,1\}^n \to \mathbb{R}$, or, more generally, classes of functions $f : D^n \to \mathbb{R}$ with a fixed finite set D, and the efficiency is measured in terms of n. One can consider the model in which functions are represented by a value-giving oracle, or the model where the functions are represented explicitly, but succinctly — say, as a sum of functions of small arity. Both models are actively studied (see, e. g. [17]). For example, submodular set functions can be efficiently minimised in the value-oracle model, while supermodular set functions cannot [18, 21, 32, 39], the standard argument for the latter fact coming from the hardness of the Max Cut problem which can be considered as a supermodular set function minimisation problem with explicitly represented objective function — in fact, a sum of binary supermodular functions (see Example 2). In this paper, we contribute towards the answer to the above question for functions $f : D^n \to \mathbb{Q}$ in the explicit representation model, by using the paradigm of the *valued constraint satisfaction problem* (VCSP) [11].

The constraint satisfaction problem (CSP) provides a framework in which it is possible to express, in a natural way, many combinatorial problems encountered in computer science and AI [12, 14, 16]. An instance of the CSP consists of a set of variables, a domain of values, and a set of constraints on combinations of values that can be taken by certain subsets of variables. The aim is then to find an assignment of values to the variables that satisfies the constraints. There are several natural optimisation versions of CSP: Max CSP (or Min CSP) where the goal is to find the assignment maximising the number of satisfied constraints (or minimising the number of unsatisfied constraints) [10, 14, 22, 23], problems like Max-Ones and Min-Hom where the constraints must be satisfied and some additional function of the assignment is to be optimised [14, 24, 40], and, the most general version, VCSP where each combination of values for variables in a constraint has a cost and the goal is to minimise the aggregate cost [7, 11]. Thus, an instance of the VCSP amounts to minimising a sum of functions, each depending on a subset of variables. If infinite costs are allowed then VCSP can model both feasibility and optimisation aspects and so generalises all the problems mentioned above [7, 11]. We will however allow *only finite costs* to concentrate on the optimisation aspect. Note that the VCSP has also been studied in various branches of computer science under different names such as Min-Sum, Gibbs energy minimisation, and Markov Random Fields (see, e. g. [13, 43]). We study the complexity of solving VCSPs to optimality.

We assume throughout the paper that **PTIME** $\neq$ **NP**. Since all the above problems are **NP**-hard in full generality, a major line of research in CSP tries to identify the tractable cases of such problems (see [14, 15]), the primary motivation being the general picture rather than specific applications. The two main ingredients of a constraint are (a) variables to which it is applied and (b) relations/functions specifying the allowed combinations of values or the costs for all combinations. Therefore, the main types of restrictions on CSP are (a) *structural* where the hypergraph formed by sets of variables appearing in individual constraints is restricted [19, 33], and (b) *language-based* where the constraint language, i. e. the set of relations/functions that can appear in constraints, is fixed (see, e. g. [6, 12, 14, 16]). The ultimate sort of results in these directions are dichotomy results, pioneered by [38], which characterise the tractable restrictions and show that the rest are as hard as the corresponding general problem (which cannot be generally taken for granted). The language-based direction is considerably more active than the structural one, there are many partial language-based dichotomy results, e. g. [4, 5, 11, 14, 22, 23, 27, 40], but many central questions are still open. In this paper, we study language-based restrictions for VCSP.

Related Work. Since VCSP is a very general problem and relatively new to the CSP dichotomy research, only a couple of earlier complexity classification results are known. The following cases have been classified: when the domain contains only two values [11], when the language contains all unary functions [27], when the domain is small and the language contains only 0-1-valued functions [22, 23]. On the hardness side, simulation of Max Cut has been a predominant idea. On the algorithmic side, most tractability results, e. g. [9, 10, 11, 23, 26, 29, 30], are based on various submodularity-like conditions. An adaption to VCSP of ideas from the algebraic approach to the CSP [6, 12] resulted in submodularity-inspired, but rather more general and abstract, algebraic properties called multimorphisms [11] and then, even more general, fractional polymorphisms [7, 8], which are certain families of operations of the same arity.

Fractional polymorphisms are known to be able to characterise all tractable constraint languages for VCSP [7, 8], and they have been recently used to characterise constraint languages such that the corresponding VCSP can be solved by the basic LP (linear programming) relaxation [28, 41]. Finally, Chapters 6 and 7 of [37] (where VCSP is called generalised CSP, or GCSP) give a condition such that, for constraint languages satisfying this condition, the basic SDP (semidefinite programming) relaxation finds the optimal value for each VCSP instance, while this task is UG-hard for all other languages. It should be noted, though, that the condition in question is not tangible except in some cases when fractional polymorphisms can be used (see Theorem 2.5).

Our work is concerned with classifying exact solvability of VCSPs. There is plenty of research in approximability of Max CSPs and VCSPs (e. g. [3, 14, 25, 37]), especially since the unique games conjecture (UGC) [25] concerns a special case of Max CSP. In fact, it is shown in [37] how to optimally approximate any VCSP assuming the UGC.

One of the main technical tools for identifying tractabililty in the VCSP, fractional polymorphisms, is a generalisation of submodularity. Submodular functions are a key concept in combinatorial optimisation [18, 32, 39], and their algorithmic aspects are being very actively studied (see e. g. [17, 21, 34, 35]).

Contributions. We classify the complexity of VCSPs with a fixed constraint language in the case of a three-element domain (see Theorem 3.3).

This result generalises the classification for the Boolean case [11] and the complexity (though not the approximability) classification for the case of a three-element domain and 0-1-valued functions from [22]. It is known in CSP research that generalising a dichotomy result from 2-element domains to 3-element domains is a crucial step in understanding the general picture and verifying the suitability of technical tools [4, 22]. Our result suggests a new possible characterisation of tractable VCSPs, which we discuss in Section 4. Moreover, we believe that the techniques used in this paper can be further extended to eventually lead to the full dichotomy for VCSPs on all finite domains. One interesting feature of our classification is that it is the first dichotomy result in CSP research when infinitely many conditions are definitely necessary to characterise tractable constraint languages in a version of CSP with a fixed domain.

It follows from our classification and [28, 41] that each tractable VCSP on a three-element domain can be solved by the basic LP relaxation. The classification raises the question whether other techniques, for example, the basic SDP relaxation, are ever necessary for solving VCSPs to optimality. This also raises the question whether tractable VCSPs can always be characterised by fractional polymorphisms of small arity, say arity 2 (when infinite values are allowed, fractional polymorphisms of arity 3 are necessary [11]). On the hardness side, our classification suggests that the ability to explicitly express Max Cut might be a unique general reason for a VCSP to be hard, just as the ability to explicitly express Not-All-Equal-Sat is conjectured

to be the only reason for a CSP with a fixed constraint language to be hard [6, 16].

Thapper and Živný asked [41] whether the VCSPs solvable by the basic LP relaxation can be characterised by a fixed-arity multimorphism. We answer this question negatively (see Proposition 3.5). Kolmogorov recently showed [28] that such VCSPs are characterised by binary (commutative) fractional polymorphisms.

We identify, for each $0 < \alpha < 1$, a new class of functions on $\{-1, 0, 1\}^n$ which we call α-bisubmodular. The ordinary bisubmodularity [1, 35, 36] would be 1-bisubmodularity in this notation. The new functions play a crucial role in our classification. Our results raise many new open questions in submodularity research which we discuss in Section 4.

2 Preliminaries

2.1 Valued Constraints Let D be a finite set. Let $\mathbb{Q}_{\geq 0}$ denote the set of all non-negative rational numbers. Let $F_D^{(m)}$ denote the set of all functions from D^m to $\mathbb{Q}_{\geq 0}$, and let $F_D = \bigcup_{m=1}^{\infty} F_D^{(m)}$. A *valued constraint language* on D is simply a subset of F_D.

Definition 1. *Let $V = \{x_1, \ldots, x_n\}$ be a set of variables. A* valued constraint *over V is an expression of the form $g(\mathbf{x})$ where $\mathbf{x} \in V^m$ and $g \in F_D^{(m)}$. The number m is the* arity *of the constraint.*

An instance $\mathcal{I}$ of VCSP *is a function*

$$f_{\mathcal{I}}(x_1, \ldots, x_n) = \sum_{i=1}^{q} w_i \cdot f_i(\mathbf{x}_i) \tag{1}$$

where, for each $i = 1, \ldots, q$, $f_i(\mathbf{x}_i)$ is a valued constraint over $V_{\mathcal{I}} = \{x_1, \ldots, x_n\}$ and $w_i \in \mathbb{Q}_{>0}$ is a weight. The goal is to find a mapping $\varphi : V_{\mathcal{I}} \to D$ that minimises $f_{\mathcal{I}}$. Let $\mathrm{Opt}(\mathcal{I})$ denote the set of all optimal solutions to $\mathcal{I}$.

For a valued constraint language $\Gamma \subseteq F_D$, let $\mathrm{VCSP}(\Gamma)$ denote the class of all VCSP *instances in which every valued constraint uses a function from Γ.*

We assume that each f_i in a VCSP instance is given by its full table of values. As usual in this line of research, we say that a language Γ is tractable if $\mathrm{VCSP}(\Gamma')$ is tractable for *each* finite $\Gamma' \subseteq \Gamma$, and it is **NP**-hard if $\mathrm{VCSP}(\Gamma')$ is **NP**-hard for *some* finite $\Gamma' \subseteq \Gamma$.

Example 1. (Submodularity [18, 32, 39]) *A function $\{0,1\}^n \to \mathbb{Q}$ is called* submodular *if*

$$f(\mathbf{a} \vee \mathbf{b}) + f(\mathbf{a} \wedge \mathbf{b}) \leq f(\mathbf{a}) + f(\mathbf{b}) \text{ for all } \mathbf{a}, \mathbf{b} \in \{0,1\}^n.$$

Here, $\vee$ and $\wedge$ denote the standard binary Boolean operations acting component-wise. If $\Gamma \subseteq F_{\{0,1\}}$ consists of submodular functions then $\mathrm{VCSP}(\Gamma)$ is tractable [11, 18, 32, 39]. A function f is called supermodular *if the function $-f$ is submodular.*

Example 2. (Max Cut) *In the* Max Cut *problem, one needs to partition the vertices of a given edge-weighted graph into two parts so as to maximise the total weight of edges with endpoints in different parts. This problem is well known to be* **NP***-hard.*

Let $f_{mc} : \{0,1\}^2 \to \mathbb{Q}$ be such that

$$f_{mc}(0,1) = f_{mc}(1,0) < f_{mc}(0,0) = f_{mc}(1,1).$$

Let $\Gamma_{mc} = \{f_{mc}\}$. It is easy to see that $\mathrm{VCSP}(\Gamma_{mc})$ is equivalent to Max Cut*. Indeed, variables in an instance of $\mathrm{VCSP}(\Gamma_{mc})$ can be seen as vertices of a graph, while constraints correspond to edges. An assignment of 0-1 values to the variables corresponds to a partition, and each constraint prefers a pair of different values to a pair of equal values. Thus, $\mathrm{VCSP}(\Gamma_{mc})$ is* **NP***-hard. Note that f_{mc} is supermodular.*

The main problem in this research direction is to identify *all* valued constraint languages Γ such that $\mathrm{VCSP}(\Gamma)$ is tractable, and to determine the complexity for the remaining constraint languages.

Since all constraints in this paper are valued, we often omit this word and say simply "constraint" or "constraint language". We note that functions in valued constraints take values in $\mathbb{Q}_{\geq 0}$ rather than in $\mathbb{Q}$, but, when infinite costs are disallowed, this restriction is not essential.

2.2 Expressive Power

Definition 2. *For a constraint language Γ, let $\langle\Gamma\rangle$ denote the set of all functions $f(x_1, \ldots, x_m)$ such that, for some instance $\mathcal{I}$ of $\mathrm{VCSP}(\Gamma)$ with objective function $f_{\mathcal{I}}(x_1, \ldots, x_m, x_{m+1}, \ldots, x_n)$, we have*

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

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

Definition 3. *If functions $f, f' \in F_D$ are such that f can be obtained from f' by* scaling *and* translating*, i. e. $f = a \cdot f' + b$ for some constants $a \in \mathbb{Q}_{>0}$ and $b \in \mathbb{Q}$, then we write $f \equiv f'$. For $\Gamma \subseteq F_D$, let $\Gamma_{\equiv} = \{f \mid f \equiv f' \text{ for some } f' \in \Gamma\}$.*

Theorem 2.1. *[8, 11] Let Γ and Γ' be constraint languages on D such that $\Gamma' \subseteq \langle\Gamma\rangle_{\equiv}$. If $\mathrm{VCSP}(\Gamma)$ is tractable then $\mathrm{VCSP}(\Gamma')$ is tractable. If $\mathrm{VCSP}(\Gamma')$ is* **NP***-hard then $\mathrm{VCSP}(\Gamma)$ is* **NP***-hard.*

The following condition, which we will call (MC), says that Γ can express Max Cut (see Example 2).

(MC) There exist distinct $a, b \in D$ such that $\langle\Gamma\rangle$ contains a unary function u with $\operatorname{argmin}(u) = \{a, b\}$ and a binary function h with $h(a, b) = h(b, a) < h(a, a) = h(b, b)$.

The role of function u in (MC) is to enforce that all optimal solutions to an instance of VCSP(Γ) take only values from $\{a, b\}$. The following lemma is effectively Lemma 5.1 of [11].

Lemma 2.1. *If a constraint language Γ satisfies condition (MC) then* VCSP(Γ) *is* **NP***-hard.*

All constraint languages Γ such that VCSP(Γ) is known to be **NP**-hard satisfy (MC) [11, 22, 23, 27].

2.3 Fractional Polymorphisms We denote by

$$O_D^{(k)} = \{F \mid F : D^k \to D\}$$

the set of all k-ary operations on D. Furthermore let $O_D = \bigcup_{k=1}^{\infty} O_D^{(k)}$. For $F \in O_D^{(k)}$ and (not necessarily distinct) tuples $\mathbf{a}_1, \ldots, \mathbf{a}_k \in D^n$, let $F(\mathbf{a}_1, \ldots, \mathbf{a}_k)$ denote the tuple in D^n obtained by applying F to $\mathbf{a}_1, \ldots, \mathbf{a}_k$ coordinate-wise.

A *fractional operation* of arity k on D is a probability distribution μ on $O_D^{(k)}$. Let

$$\operatorname{supp}(\mu) = \{F \mid \Pr_{\mu}[F] > 0\}.$$

For a function $f \in F_D^{(n)}$, μ is said to be a *fractional polymorphism* of f [8] if, for all $\mathbf{a}_1, \ldots, \mathbf{a}_k \in D^n$,

$$\mathbb{E}_{F\sim\mu}(f(F(\mathbf{a}_1, \ldots, \mathbf{a}_k))) \le \operatorname{avg}\{f(\mathbf{a}_1), \ldots, f(\mathbf{a}_k)\}, \tag{2}$$

or, in expanded form,

$$\sum_{F \in O_D^{(k)}} \Pr_{\mu}[F] \cdot f(F(\mathbf{a}_1, \ldots, \mathbf{a}_k)) \le \tfrac{f(\mathbf{a}_1)+\ldots+f(\mathbf{a}_k)}{k}. \tag{3}$$

Let $\operatorname{fPol}(f)$ denote the set of all fractional polymorphisms of f. For a constraint language Γ, let

$$\operatorname{fPol}(\Gamma) = \bigcap_{f\in\Gamma} \operatorname{fPol}(f).$$

A fractional polymorphism μ of arity k is a *multimorphism* [11] if the probability of each operation in μ is of the form l/k for some integer l. A k-ary multimorphism μ can be represented as a transformation $\mathbf{F} : D^k \to D^k$ given by a k-tuple $\mathbf{F} = (F_1, \ldots, F_k)$ of functions from $O_D^{(k)}$, where each operation $F \in O_D^{(k)}$ with $\Pr_{\mu}[F] = l/k$ appears l times in $\mathbf{F}$. The inequality (3) can then be written as

$$\sum_{i=1}^{k} f(F_i(\mathbf{a}_1, \ldots, \mathbf{a}_k)) \le \sum_{i=1}^{k} f(\mathbf{a}_i). \tag{4}$$

All important fractional polymorphisms identified earlier [9, 11, 23, 26] are in fact multimorphisms.

Recall that a *lattice* is a partially ordered set in which each pair of elements has a least upper bound (*join*, denoted $\vee$) and a greatest lower bound (*meet*, denoted $\wedge$). For the next example, see [10, 29, 30, 42].

Example 3. (Submodularity on a lattice) *Let $\mathcal{L} = (D, \vee, \wedge)$ be a lattice. A function $f : D^n \to \mathbb{Q}$ is called* submodular *on $\mathcal{L}$ if*

$$f(\mathbf{a} \vee \mathbf{b}) + f(\mathbf{a} \wedge \mathbf{b}) \le f(\mathbf{a}) + f(\mathbf{b}) \text{ for all } \mathbf{a}, \mathbf{b} \in D^n.$$

This inequality can be equivalently expressed by saying that f has the binary multimorphism μ with $\Pr_{\mu}[\vee] = \Pr_{\mu}[\wedge] = 1/2$. If $\mathcal{L}$ is a chain, *i. e. a total order, then $\vee$ and $\wedge$ are the usual* max *and* min *operations. For $D = \{0, 1\}$, submodularity on a chain is the same as ordinary submodularity.*

Example 4. (Bisubmodularity [1, 18, 35, 36]) *Let $D = \{-1, 0, 1\}$, and fix the order $-1 > 0 < 1$ on D. Define binary operations $\vee_0$ and $\wedge_0$ on D, as follows:*

$$\begin{aligned} &1 \vee_0 -1 = -1 \vee_0 1 = 0;\\ &x \vee_0 y = \max(x, y) \text{ if } \{x, y\} \neq \{-1, 1\};\\ &1 \wedge_0 -1 = -1 \wedge_0 1 = 0;\\ &x \wedge_0 y = \min(x, y) \text{ if } \{x, y\} \neq \{-1, 1\}, \end{aligned}$$

where maximum and minimum are taken with respect to the above order on D.

A function $f : D^n \to \mathbb{Q}$ is called bisubmodular *if it has the binary multimorphism μ with $\Pr_{\mu}[\vee_0] = \Pr_{\mu}[\wedge_0] = 1/2$, that is, if*

$$f(\mathbf{a} \vee_0 \mathbf{b}) + f(\mathbf{a} \wedge_0 \mathbf{b}) \le f(\mathbf{a}) + f(\mathbf{b})$$

holds for all $\mathbf{a}, \mathbf{b} \in D^n$.

Other well-known classes of discrete functions, such as $L^{\#}$-convex functions and tree-submodular functions [18, 26] can be described by suitable multimorphisms.

Fractional polymorphisms not only provide a useful way to describe classes of functions, they characterise the expressive power of constraint languages.

THEOREM 2.2. ([8]) *For any $\Gamma \subseteq F_D$ and any $f \in F_D$, we have $f \in \langle\Gamma\rangle_{\equiv}$ if and only if* $\mathrm{fPol}(\Gamma) \subseteq \mathrm{fPol}(f)$.

Together with Theorem 2.1, this implies that tractable valued constraint languages can be characterised by fractional polymorphisms, since any two languages with the same set of fractional polymorphisms must have the same complexity.

We now define a new type of (binary commutative) fractional polymorphisms on $\{-1, 0, 1\}$, which can collectively be called *skew bisubmodularity*. Recall the operations from Example 4, and, for $a \in \{-1, 1\}$ define the binary operation $\vee_a$ so that $1 \vee_a -1 = -1 \vee_a 1 = a$ and $x \vee_a y = x \vee_0 y = \max(x, y)$ otherwise.

DEFINITION 4. *Let $\alpha \in (0, 1]$. We say that a function $f : \{-1, 0, 1\}^n \to \mathbb{Q}$ is* α-bisubmodular (towards 1) *if it has the fractional polymorphism μ such that* $\Pr_\mu[\wedge_0] = 1/2$, $\Pr_\mu[\vee_0] = \alpha/2$, *and* $\Pr_\mu[\vee_1] = (1-\alpha)/2$.

In other words, a function f is α-bisubmodular (towards 1) if, for all $\mathbf{a}, \mathbf{b} \in \{-1, 0, 1\}^n$,

$$f(\mathbf{a} \wedge_0 \mathbf{b}) + \alpha \cdot f(\mathbf{a} \vee_0 \mathbf{b}) + (1-\alpha) \cdot f(\mathbf{a} \vee_1 \mathbf{b}) \leq f(\mathbf{a}) + f(\mathbf{b}). \tag{5}$$

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

Note that α-bisubmodular functions towards -1 can be defined by using $\vee_{-1}$ instead of $\vee_1$. In the rest of the paper, we assume that α-bisubmodular functions are skew towards 1, unless explicitly stated otherwise. Notice also that the 1-bisubmodular functions (towards 1 or -1) are the ordinary bisubmodular functions from Example 4.

2.4 Algorithms Some types of fractional polymorphisms are known to guarantee tractability of VCSP(Γ). An operation $F \in O_D^{(k)}$, $k \geq 1$ is called *idempotent* if

$$F(x, \ldots, x) = x$$

for all $x \in D$. An idempotent operation F is called *cyclic* if

$$F(x_1, x_2, \ldots, x_k) = F(x_2, \ldots, x_k, x_1)$$

for all $x_1, \ldots, x_k \in D$, and *symmetric* if

$$F(x_1, \ldots, x_k) = F(x_{\pi(1)}, \ldots, x_{\pi(k)})$$

for all $x_1, \ldots, x_k \in D$ and any permutation π on $\{1, \ldots, k\}$. Such operations play an important role in the algebraic approach to the standard CSP [2, 31]. Call a fractional operation μ idempotent, cyclic, or symmetric if each operation in $\mathrm{supp}(\mu)$ has the corresponding property.

A characterisation of valued constraint languages that can be solved by the basic LP relaxation has been obtained in [41]. For an instance $\mathcal{I}$ of VCSP of the form (1), let V_i be the subset of $V_\mathcal{I}$ involved in $\mathbf{x}_i$. The basic LP relaxation is the linear program on the variables

$\lambda_{i,\varphi_i} \in [0, 1]$ for each $i = 1, \ldots, q$ and $\varphi_i : V_i \to D$,
$\mu_{x,a} \in [0, 1]$ for each $x \in V_\mathcal{I}$ and $a \in D$,

given by the minimisation problem

$$\min \quad \sum_{i=1}^{q} \sum_{\varphi_i : V_i \to D} w_i \cdot f_i(\varphi_i(\mathbf{x}_i))) \cdot \lambda_{i,\varphi_i}$$

such that

$$\forall i = 1, \ldots, q, \forall x \in V_\mathcal{I}, \forall a \in D : \quad \sum_{\substack{\varphi_i : V_i \to D \\ \varphi_i(x) = a}} \lambda_{i,\varphi_i} = \mu_{x,a}$$

$$\forall x \in V_\mathcal{I} : \quad \sum_{a \in D} \mu_{x,a} = 1.$$

Since Γ is fixed, this relaxation has polynomial size (in $\mathcal{I}$). The integer programming formulation – i. e. we require the variables λ_{i,φ_i} and $\mu_{x,a}$ to be in $\{0, 1\}$ – is an integer programming formulation of $\mathcal{I}$:

$\mu_{x,a} = 1$ means variable x is assigned value a,
$\lambda_{i,\varphi_i} = 1$ means $\mathbf{x}_i$ is assigned $\varphi_i(\mathbf{x}_i)$.

THEOREM 2.3. ([41]) *The basic LP relaxation solves* VCSP(Γ) *in polynomial time if and only if Γ has symmetric fractional polymorphisms of all arities.*

Specifically, the basic LP relaxation finds the actual optimal value for each instance $\mathcal{I}$ [41], and then an optimal solution to $\mathcal{I}$ can be found by going through variables in some order and adding constraints $\mu_{x,a} = 1$ so that the LP optimum does not change.

THEOREM 2.4. ([28]) *If Γ has a fractional polymorphism μ of some arity $k > 1$ such that* $\mathrm{supp}(\mu)$ *contains a symmetric operation then Γ has symmetric fractional polymorphisms of all arities.*

In particular, it follows from Theorem 2.4 that the basic LP relaxation solves VCSP(Γ) in polynomial time if and only if Γ has a binary commutative (i. e. symmetric) fractional polymorphism.

One can also consider a basic SDP relaxation for VCSPs. The following theorem, which possibly provides a larger class of tractable languages, is implied by results in Chapters 6 and 7 of [37].

THEOREM 2.5. *If Γ has a cyclic fractional polymorphism of some arity $k > 1$ then the basic SDP relaxation solves* VCSP(Γ) *in polynomial time.*

3 Results

In this section we formally state our results. Most of the proofs are omitted due to space constraints, sometimes sketches are given. All proofs will be included in the full version of the paper.

3.1 Cores We say that a constraint language Γ on D is a *core* if, for each $a \in D$, there is an instance $\mathcal{I}_a$ of VCSP(Γ) such that a appears in every optimal solution to $\mathcal{I}$. The intuition is that if Γ is not a core then there is an element $a \in D$ such that any instance has an optimal solution not using a. In this case, we simply remove a from D, thus reducing the problem to a similar one on a smaller domain. Therefore, we can without loss of generality consider only cores. Note that α-bisubmodular functions can be defined for $\alpha = 0$, but it is not hard to check that the set of 0-bisubmodular functions is not a core.

The following proposition further reduces the class of cores that we need to consider. For a constraint language Γ, let Γ_c denote the constraint language obtained from Γ by adding all functions obtained from functions in Γ by fixing values for some variables, e. g. $g(x, y) = f(x, a, b, y) \in \Gamma_c$ if $f \in \Gamma$ and $a, b \in D$.

Proposition 3.1. *Let Γ be a core constraint language on an arbitrary finite set D. Then*

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

It follows that it is sufficient to consider only cores Γ which are closed under fixing values for a subset of variables and which, in addition, contain, for each $a \in D$, a unary function u_a with $\operatorname{argmin}(u_a) = \{a\}$. Note the last condition already implies that Γ is a core, and also that fPol(Γ_c) consists of the idempotent members of fPol(Γ). This algebraic condition proved to be extremely important in the algebraic approach to the CSP (see, e. g. [2, 6]). Note that the fractional polymorphisms describing submodularity and α-bisubmodularity are idempotent (and even symmetric).

3.2 A Characterisation of α-Bisubmodularity Let $\leq$ denote the partial order on $\{-1, 0, 1\}$ such that $0 \leq t$ for all t, and $-1, 1$ are incomparable. Let $\leq$ also denote the componentwise partial order on $\{-1, 0, 1\}^n$. For a tuple $\mathbf{c} \in \{-1, 1\}^n$, the *orthant* of $\mathbf{c}$ is the set

$$\mathbf{c}^{\downarrow} = \{\mathbf{x} \in \{-1, 0, 1\}^n \mid \mathbf{x} \leq \mathbf{c}\}.$$

Definition 5. *For every $\mathbf{c} \in \{-1, 1\}^n$, a function $f : \{-1, 0, 1\}^n \to \mathbb{Q}$ is called* submodular in the orthant of $\mathbf{c}$*, if the α-bisubmodularity inequality* (5) *holds for all $\mathbf{a}, \mathbf{b} \in \mathbf{c}^{\downarrow}$.*

Note that, in any fixed orthant, only one of -1 and 1 can appear in each coordinate, and so α-bisubmodularity becomes the ordinary submodularity inequality (with $0 < 1$ and $0 < -1$). Recall that a unary function f is α-bisubmodular if and only if $(1 + \alpha) \cdot f(0) \leq f(-1) + \alpha \cdot f(1)$.

Proposition 3.2. *Let $\alpha \in (0, 1]$. A function $f : \{-1, 0, 1\}^n \to \mathbb{Q}$ is α-bisubmodular if and only if the following two conditions hold:*

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

This proposition for the case $\alpha = 1$ was the main result of [1].

3.3 A Dichotomy Theorem We will generalise the following two theorems, the classification for the Boolean case [11] and the complexity classification for the case of a three-element domain and 0-1-valued functions [22], to a general classification for the case of a three-element domain.

Theorem 3.1. ([11]) *Let Γ be a core constraint language on $\{0, 1\}$. Either Γ consists of submodular functions and* VCSP(Γ) *is tractable, or Γ_c satisfies condition (MC) and* VCSP(Γ) *is* **NP***-hard.*

Theorem 3.2. ([22]) *Let $|D| = 3$ and let Γ be a core constraint language on D consisting of 0-1-valued functions. If the elements of D can be renamed $-1, 0, 1$ in such a way that each function in Γ is submodular on the chain $-1 < 0 < 1$, then* VCSP(Γ) *is tractable. Otherwise, Γ_c satisfies condition (MC) and* VCSP(Γ) *is* **NP***-hard.*

The next theorem is the main result of the paper.

Theorem 3.3. *Let $|D| = 3$ and let Γ be a core constraint language on D. If the elements of D can be renamed $-1, 0, 1$ in such a way that*

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

then VCSP(Γ) *is tractable. Otherwise, Γ_c satisfies condition (MC) and* VCSP(Γ) *is* **NP***-hard.*

The tractability part immediately follows from Theorems 2.3 and 2.4. (It can also be derived directly from Theorem 4.1 of [41].) For the hardness part, by Lemma 2.1 and Proposition 3.1, it suffices to show that Γ_c satisfies (MC). An outline of the proof is presented in Section 3.4.

We now argue that the tractable cases from Theorem 3.3 are pairwise distinct, except that submodularity on the chain $a < b < c$ is the same as submodularity on $c < b < a$, where $D = \{a, b, c\}$. It is easy to check that the function f such that $f(a, a) = f(c, c) = 0$ and $f(x, y) = 1$ otherwise is submodular on $a < b < c$ and $c < b < a$, but not on any other chain. The constraint language consisting of f and all 0-1 valued unary functions is submodular on $a < b < c$ and $c < b < a$, but not on any other chain, nor can it be α-bisubmodular under any renaming into $-1, 0, 1$. We will often represent a unary function f from $\{-1, 0, 1\}$ to $\mathbb{Q}$ by its vector of values $[f(-1), f(0), f(1)]$. It remains to prove the following statement.

PROPOSITION 3.3. *For every rational $\alpha \in (0, 1]$, there is a core constraint language Γ_α on $D = \{-1, 0, 1\}$ satisfying all of the following conditions:*

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

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

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

Let $\Gamma_\alpha = \{e, u_\alpha, v_\alpha, f_\alpha\}$. It can be directly checked that all functions in Γ_α are α-bisubmodular (Proposition 3.2 can also be used for checking f_α) and that the unary functions in Γ_α make it a core.

Notice that f_α is not submodular when restricted to $\{-1, 1\}$. Therefore Γ_α is not submodular on any chain on $\{-1, 0, 1\}$. It is easy to check that u_α is not α'-bisubmodular for any $\alpha' > \alpha$, and v_α is not α'-bisubmodular for any $\alpha' < \alpha$. It is also easy to check that the unary operations guarantee that any permutation of the names of elements $-1, 0, 1$ cannot make Γ_α α'-bisubmodular for any α', except swapping -1 and 1 when $\alpha = \alpha' = 1$. □

An easy corollary of Theorem 3.3 is the following:

THEOREM 3.4. *Let $|D| = 3$ and let Γ be a core constraint language on D. If Γ has a binary commutative fractional polymorphism, then* VCSP(Γ) *is tractable. Otherwise, Γ_c satisfies condition (MC) and* VCSP(Γ) *is* **NP***-hard.*

It is straightforward that the problem of deciding whether a given finite constraint language on a fixed set has a binary commutative fractional polymorphism can be expressed as an LP feasibility problem, and therefore is polynomially solvable. However, our characterisation of α-bisubmodular functions leads to a simple algorithm for recognising tractable cases.

PROPOSITION 3.4. *Let $|D| = 3$. There is a quadratic-time algorithm which, given a finite core constraint language Γ on D, decides whether* VCSP(Γ) *is tractable.*

Proof. Note that if Γ contains functions of arities $n_1, \ldots, n_k$ and M is the largest value taken by functions in Γ then the size of Γ is $(3^{n_1} + \ldots + 3^{n_k}) \cdot \log M$. For every renaming of the elements of D into $-1, 0, 1$ (there are six of them), we do the following. First we check whether each function in Γ is submodular on the chain $-1 < 0 < 1$. This can be done in quadratic time, by simply verifying the submodularity inequality for all pairs of tuples. If the above check succeeds then we stop and conclude that VCSP(Γ) is tractable. Otherwise, we use Proposition 3.2 to check whether there is $\alpha \in (0, 1]$ such that Γ is α-bisubmodular. First, we check whether each function in Γ is submodular in all orthants. The number of different orthants is linear, and the direct checking for each orthant is quadratic, as above. If some function fails the check then we conclude that VCSP(Γ) is not tractable and stop. Otherwise, we generate the set $\Gamma_c^{(1)}$ of unary functions in Γ_c. It is clear that it contains at most a quadratic number of functions. Each function $f \in \Gamma_c^{(1)}$ gives the inequality $(1+\alpha) \cdot f(0) \leq f(-1) + \alpha \cdot f(1)$ that restricts the possible values for α. One can go through this list of inequalities (just once), updating the possible values for α. At the end, we know whether the system of inequalities has a solution. If it does then Γ is α-bisubmodular and VCSP(Γ) is tractable, if it does not, for any renaming, then VCSP(Γ) is not tractable. □

3.4 Sketch of the proof of Theorem 3.3 In this section, we will outline the proof of our main result,

Theorem 3.3. Assume that $|D| = 3$, and that we have a constraint language Γ which is a core. By Proposition 3.1 and Theorem 2.1, we can assume that $\Gamma = \langle \Gamma_c \rangle_{\equiv}$. In particular, for each $a \in D$, Γ contains a unary function u_a with $\operatorname{argmin}(u_a) = \{a\}$. We have the following.

LEMMA 3.1. *For at least two distinct 2-element subsets $X \subseteq D$, Γ contains functions u_X with $\operatorname{argmin}(u_X) = X$.*

Let us rename the elements of D into $-1, 0, 1$ so that the two 2-element subsets guaranteed by Lemma 3.1 are $\{-1, 0\}$ and $\{0, 1\}$. From now on we assume that $D = \{-1, 0, 1\}$ and that Γ contains u_X for each non-empty subset $X \subseteq D$ except possibly $u_{\{-1,1\}}$. By translating and scaling, we can assume that $u_{\{-1,0\}} = [0, 0, 1]$ and $u_{\{0,1\}} = [1, 0, 0]$.

LEMMA 3.2. *One of the following holds.*

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

Let us consider first the case when $u_{\{-1,1\}} \in \Gamma$. Again, we can assume that $u_{\{-1,1\}} = [0, 1, 0]$. Clearly, any function from $F_D^{(1)}$ can be obtained as a positive linear combination of $u_{\{-1,0\}}$, $u_{\{0,1\}}$, and $u_{\{-1,1\}}$. This means that Γ contains all unary functions. Such constraint languages are called *conservative*, and their complexity is classified in [27]. This classification can be stated as follows: either Γ has a so-called STP multimorphism or else Γ satisfies (MC). If Γ has an STP multimorphism, then, by Theorem 19 of [28], Γ is submodular on a chain. Thus, in this case, the assertion of Theorem 3.3 holds.

Let us assume for the rest of this section that $u_{\{-1,1\}} \notin \Gamma$. By Lemma 3.2, we have that, for some $\alpha \in (0, 1]$, the unary functions in Γ are α-bisubmodular, all towards 1 or all towards -1. Let us assume that they are all α-bisubmodular towards 1, the other case is identical. If every function in Γ is α-bisubmodular then we are done. Otherwise, by Proposition 3.2, Γ contains a function which is not submodular in some orthant. The following lemma is well-known, see, e. g. [42]. The notion of submodularity from Example 3 can be naturally extended to the direct product of lattices, by defining the operations component-wise.

LEMMA 3.3. *Let $D_1, \ldots, D_n$ be finite chains. If a function $f : D_1 \times \ldots \times D_n \to \mathbb{Q}$ is not submodular then some binary function obtained from f by fixing all but two coordinates is not submodular.*

Since $\Gamma = \Gamma_c$, by Lemma 3.3 we can assume that Γ contains a binary function which is not submodular in some orthant. If Γ contains a binary function which is not submodular in the orthant of $(1, 1)$ or $(-1, -1)$ then, by Lemma 7.8 of [11], Γ satisfies (MC), with $u_{\{-1,0\}}$ or $u_{\{0,1\}}$, respectively, and then we are done. So let us assume that Γ contains a binary function f that is not submodular in the orthant of $(-1, 1)$.

If every function in Γ is submodular on the chain $-1 < 0 < 1$ then we are done. Otherwise, by Lemma 3.3, Γ contains a binary function g which is not submodular on this chain. We can assume that the function g is submodular both in the orthant of $(1, 1)$ and in the orthant of $(-1, -1)$, for we are done otherwise. If g satisfies both $g(1, 0) + g(0, -1) \leq g(0, 0) + g(1, -1)$ and $g(0, 1) + g(-1, 0) \leq g(0, 0) + g(-1, 1)$ then it can easily be checked that g is submodular on $-1 < 0 < 1$. Since this is not the case, at least one of the inequalities fails. We can assume, permuting the variables of g if necessary, that $g(1, 0) + g(0, -1) > g(0, 0) + g(1, -1)$. The following lemma finishes the proof of Theorem 3.3.

LEMMA 3.4. *If Γ contains binary functions f and g such as above then Γ contains a binary function which is not submodular in the orthant of $(-1, -1)$.*

3.5 Multimorphisms vs. Fractional Polymorphisms As we noted before, it was open whether tractability of valued constraint languages can be characterised by multimorphisms. We show that this is not the case because the set of 1/2-bisubmodular functions cannot be defined by multimorphisms. Clearly, not every unary function is 1/2-bisubmodular, so it suffices to prove the following.

PROPOSITION 3.5. *There is a finite set Γ of 1/2-bisubmodular functions such that each multimorphism of Γ is a multimorphism of every unary function on $\{-1, 0, 1\}$.*

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

$$\sum_{i=1}^{k} f(F_i(\mathbf{x}_1, \ldots, \mathbf{x}_k)) \leq \sum_{i=1}^{k} f(\mathbf{x}_i) \tag{6}$$

for all $\mathbf{x}_1, \ldots, \mathbf{x}_k \in D^n$. We define the function $\mathbf{F} : D^k \to D^k$ by $\mathbf{F} = (F_1, \ldots, F_k)$ and identify μ with $\mathbf{F}$. If $\mathbf{F}$ preserves each tuple in D^k as a multiset, we say that $\mathbf{F}$ preserves multisets. It is easy to see that in

this case the inequality (6) holds with equality for every unary function. So in order to prove Proposition 3.5 it suffices to show that if $\mathbf{F}$ is a multimorphism of Γ then $\mathbf{F}$ preserves multisets.

4 Conclusion and Discussion

We have classified the complexity of VCSPs (with finite costs) on a three-element domain. The tractable cases are described by simple fractional polymorphisms. Our results suggest the following open problem:

PROBLEM 1. *Are the following properties of a core constraint language Γ equivalent?*

1. *Γ_c does not satisfy condition (MC),*
2. *VCSP(Γ) is tractable,*
3. *fPol(Γ) contains a cyclic fractional operation of some arity $k > 1$,*
4. *fPol(Γ) contains a symmetric fractional operation of some arity $k > 1$,*
5. *fPol(Γ) contains a binary commutative fractional operation.*

In general, we have (5) $\Rightarrow$ (4) $\Rightarrow$ (3) $\Rightarrow$ (2) $\Rightarrow$ (1), where (3) $\Rightarrow$ (2) holds by Theorem 2.5, (2) $\Rightarrow$ (1) is by Lemma 2.1, and the implications (5) $\Rightarrow$ (4) $\Rightarrow$ (3) are trivial. The implication (3) $\Rightarrow$ (1) is not hard to show directly, without going through (2) and relying on $\mathbf{PTIME} \neq \mathbf{NP}$. In fact, (4) and (5) are equivalent by Theorem 2.4. However, by [11, 28, 41] and our results, all the properties (1) – (5) are indeed equivalent when $|D| \leq 3$. To solve the above problem, it might be helpful to develop a theory of "fractional" universal algebras aimed at VCSPs, similar to [6]. First steps towards this theory are made in [7, 8].

Many efficient algorithms exist for minimising submodular functions (see, e. g. [18, 21, 34, 39]). Lovász asked in [32] whether there is a generalisation of submodularity that preserves the nice properties such as efficient minimisation, and this question led to the discovery of bisubmodularity in [36] (where it is called directed submodularity), and subsequent efficient minimisation algorithms for it (see, e. g. [35, 36]). Our results suggest that it would be interesting to study classes of functions defined by fractional polymorphisms and try to find efficient minimisation algorithms for them in the value-oracle model.

PROBLEM 2. *Which fractional operations μ guarantee an efficient minimisation algorithm, in the value-oracle model, for the class of all functions f with $\mu \in \mathrm{fPol}(f)$?*

For binary multimorphisms, this problem was introduced as MFM (multimorphism function minimisation) in [23]. Some initial results in this direction can be found in [20, 29, 30]. We crucially use α-bisubmodular functions in our classification. In the VCSP model, they can be efficiently minimised by the basic LP relaxation [28, 41]. However, it is not clear whether they can also be efficiently minimised in the value-oracle model. Perhaps the shortest way to efficient algorithms for minimising submodular and bisubmodular functions is via the convexity of their Lovász extension [32, 36]. This approach does not work, at least directly, for α-bisubmodular functions with $\alpha < 1$. Problem 2 can be approached for specific fractional polymorphisms, e. g. those from Example 3 or k-submodularity from [20], as well as for general types of operations such as those in Problem 1. Once some efficient algorithms are discovered (if they exist), one could naturally try to design strongly polynomial or (fully) combinatorial algorithms. In short, one can try to extend many aspects of submodularity research to classes of functions defined by fractional polymorphisms.

Acknowledgements

We would like to thank Vladimir Kolmogorov for showing us an example that eventually led to the notion of α-bisubmodularity, Prasad Raghavendra for patiently explaining how Theorem 2.5 follows from [37], and Johan Thapper for suggesting the notion of core for VCSP that we use in the paper.

References

[1] K. Ando, S. Fujishige, and T. Naitoh. A characterization of bisubmodular functions. *Discrete Mathematics*, 148:299–303, 1996.

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

[3] L. Barto and M. Kozik. Robust satisfiability of constraint satisfaction problems. In *Proceedings of STOC'12*, pages 931–940, 2012.

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

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

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

[7] D. Cohen, M. Cooper, P. Creed, P. Jeavons, and S. Živný. An algebraic theory of complexity for discrete optimisation. *CoRR*, Technical Report abs/1207.6692, 2012.

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

[9] D. Cohen, M. Cooper, and P. Jeavons. Generalising submodularity and Horn clauses: Tractable optimization problems defined by tournament pair multimorphisms. *Theoretical Computer Science*, 401(1):36–51, 2008.

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

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

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

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

[14] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*, SIAM. 2001.

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

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

[17] U. Feige, V. Mirrokni, and J. Vondrák. Maximizing non-monotone submodular functions. *SIAM J. Comput.*, 40:1133–1153, 2011.

[18] S. Fujishige. *Submodular Functions and Optimization*. Elsevier, 2nd edition, 2005.

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

[20] A. Huber and V. Kolmogorov. Towards minimizing k-submodular functions. In *Proceedings of ISCO'12*, pages 451–462, 2012.

[21] S. Iwata and J. Orlin. A simple combinatorial algorithm for submodular function minimization. In *Proceedings of SODA'09*, pages 1230–1237, 2009.

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

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

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

[25] S. Khot. On the unique games conjecture. In *Proceedings of CCC'10*, pages 99–121, 2010.

[26] V. Kolmogorov. Submodularity on a tree: Unifying $L^{\#}$-convex and bisubmodular functions. In *Proceedings of MFCS'11*, pages 400–411, 2011.

[27] V. Kolmogorov and S. Živný. The complexity of conservative valued CSPs. In *Proceedings of SODA'12*, pages 750–759, 2012.

[28] V. Kolmogorov. The power of linear programming for valued CSPs: A constructive characterization. *CoRR*, Technical Report abs/1207.7213, 2012.

[29] A. Krokhin and B. Larose. Maximizing supermodular functions on product lattices, with application to maximum constraint satisfaction. *SIAM J. Discr. Math.*, 22(1):312–328, 2008.

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

[31] G. Kun, R. O'Donnell, S. Tamaki, Y. Yoshida, an Y. Zhou. Linear programming, width-1 CSPs, and robust satisfaction. In *Proceedings of ITCS'12*, pages 484–495, 2012.

[32] L. Lovász. Submodular functions and convexity. In A. Bachem, M. Grötschel, and B. Korte, editors, *Mathematical Programming: The State of the Art*, pages 235–257. Springer, 1983.

[33] D. Marx. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. In *Proceedings of STOC'10*, pages 735–744, 2010.

[34] S.T. McCormick. Submodular function minimization. In K. Aardal, G. Nemhauser, and R. Weismantel, editors, *Handbook on Discrete Optimization*, chapter 7, pages 321–391. Elsevier, 2006.

[35] S.T. McCormick and S. Fujishige. Strongly polynomial and fully combinatorial algorithms for bisubmodular function minimization. *Mathematical Programming, Ser.A*, 122:87–120, 2010.

[36] L. Qi. Directed submodularity, ditroids and directed submodular flows. *Mathematical Programming*, 42(1–3):579–599, 1988.

[37] P. Raghavendra. *Approximating NP-hard problems: Efficient algorithms and their limits*. PhD thesis, University of Washington, 2009.

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

[39] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, 2003.

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

[41] J. Thapper and S. Živný. The power of linear programming for valued CSPs. In *Proceedings of FOCS'12*, 2012. To appear; available at arXiv:1204.1079v2.

[42] D. Topkis. Minimizing a submodular function on a lattice. *Operations Res.*, 26(2):305–321, 1978.

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