SIAM J. COMPUT.
Vol. 43, No. 3, pp. 1064–1084

SKEW BISUBMODULARITY AND VALUED CSPs*

ANNA HUBER†, ANDREI KROKHIN‡, AND ROBERT POWELL‡

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, with each function depending on a subset of the variables. The goal is to find an assignment of values to the variables that minimizes 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 characterization of tractable cases is tight; that is, none of the conditions can be omitted.

Key words. valued constraint satisfaction problem, optimization, computational complexity, dichotomy, submodularity

AMS subject classifications. 68Q25, 68Q17, 90C60, 90C27

DOI. 10.1137/120893549

1. Introduction.

What are the classes of discrete functions that admit an efficient minimization 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 [4]) $f : \{0,1\}^n \to \mathbb{R}$ 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., [18]). For example, submodular set functions can be efficiently minimized in the value-oracle model, while supermodular set functions cannot [19, 26, 37, 44], 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 minimization 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) [12].

The constraint satisfaction problem (CSP) provides a framework in which it is

*Received by the editors October 1, 2012; accepted for publication (in revised form) January 21, 2014; published electronically May 8, 2014. This work was supported by the UK EPSRC grant EP/J000078/1. A short version of this paper appeared in *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms* (*SODA*), New Orleans, LA, 2013, pp. 1296–1305.
http://www.siam.org/journals/sicomp/43-3/89354.html

†School of Computing and Mathematics, University of Derby, Kedleston Road, Derby DE22 1GB, UK (a.huber@derby.ac.uk).

‡School of Engineering and Computer Science, Durham University, Durham DH1 3LE, UK (andrei.krokhin@durham.ac.uk, robert.powell@durham.ac.uk).

possible to express, in a natural way, many combinatorial problems encountered in computer science and artificial intelligence [13, 15, 17]. 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 optimization versions of CSP: Max CSP (or Min CSP), where the goal is to find the assignment maximizing the number of satisfied constraints (or minimizing the number of unsatisfied constraints) [11, 15, 27, 28]; problems like Max-Ones and Min-Hom, where the constraints must be satisfied and some additional function of the assignment is to be optimized [15, 29, 45]; and, the most general version, VCSP, where each combination of values for variables in a constraint has a cost and the goal is to minimize the aggregate cost [8, 12]. Thus, an instance of the VCSP amounts to minimizing a sum of functions, with each depending on a subset of variables. If infinite costs are allowed, then VCSP can model both feasibility and optimization aspects and so generalizes all the problems mentioned above [8, 12]. We will, however, allow *only finite costs* to concentrate on the optimization aspect. Note that the VCSP has also been studied in various branches of computer science under different names such as Min-Sum, Gibbs energy minimization, and Markov random fields (see, e.g., [14, 50]). We study the complexity of solving VCSPs to optimality.

We assume throughout the paper that PTIME $\neq$ NP. Since all of the above problems are NP-hard in full generality, a major line of research in the CSP tries to identify the tractable cases of such problems (see [15, 16]), where the primary motivation is 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 [21, 38], 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., [7, 13, 15, 17]). The ultimate sort of results in these directions are dichotomy results, pioneered by [43], which characterize 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; see, e.g., [5, 6, 12, 15, 27, 28, 33, 45]), 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 were known (at the time of submission). The following cases have been classified: when the domain contains only two values [12], when the language contains all unary functions [33], and when the domain is small and the language contains only 0-1-valued functions [27, 28]. On the hardness side, simulation of Max Cut has been a predominant idea. On the algorithmic side, most tractability results, e.g., [10, 11, 12, 28, 31, 34, 35], are based on various submodularity-like conditions. An adaptation to VCSP of ideas from the algebraic approach to the CSP [7, 13] resulted in submodularity-inspired, but rather more general and abstract, algebraic properties called multimorphisms [12] and then, even more general, fractional polymorphisms [8, 9], which are certain families of operations of the same arity. Fractional polymorphisms are known to be able to characterize all tractable constraint languages for VCSP [8, 9], and they have been recently used to characterize constraint languages such that the corresponding

VCSP can be solved by the basic LP (linear programming) relaxation [32, 46]. After this paper was submitted, Thapper and Živný [48] proved that VCSPs that are not solvable by the basic LP relaxation are in fact NP-hard (they can simulate Max Cut), thus completing the classification of the complexity of finite-valued VCSPs. For applications of the theory of VCSP to other problems in discrete optimization, see, e.g., [22].

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, 15, 30, 42]), especially since the unique games conjecture (UGC) concerns a special case of Max CSP. In fact, it is shown in [42] how to optimally approximate any VCSP assuming the UGC.

One of the main technical tools for identifying tractability in the VCSP, fractional polymorphisms, is a generalization of submodularity. Submodular functions are a key concept in combinatorial optimization [19, 37, 44], and their algorithmic aspects are being actively studied (see, e.g., [18, 26, 39, 40]).

Contributions. We give a precise classification of the complexity of VCSPs with a fixed constraint language in the case when the domain consists of three values (see Theorem 8). Our classification is precise in the sense that the conditions describing the tractable cases are very specific and none of them can be omitted. This feature of our classification does not follow (immediately) from the general result of [48]. One interesting feature of this classification is that it is the first dichotomy result in CSP research when infinitely many conditions are definitely necessary to characterize tractable constraint languages in a version of CSP with a fixed domain. By contrast, the case of a 2-element domain has only eight tractable cases [12].

Thapper and Živný [47] asked whether the VCSPs solvable by the basic LP relaxation can be characterized by a fixed-arity multimorphism. We answer this question negatively (see Proposition 5).

We identify, for each $0 < \alpha < 1$, a new class of functions on $\{-1, 0, 1\}^n$ which we call α-bisubmodular. The definition can be found in section 2.3, Definition 4. The ordinary bisubmodularity [1, 40, 41] would be 1-bisubmodularity in this notation. The new functions play a crucial role in our classification. We have been informed by Hirai that α-bisubmodular functions can be captured in his framework of submodularity on modular semilattices [22, 23].

2. Preliminaries.

2.1. Valued constraints. Let D be a finite set. Let $\mathbb{Q}_{>0}$ denote the set of all positive rational numbers. Let $F_D^{(m)}$ denote the set of all functions from D^m to $\mathbb{Q}$, 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 $V = \{x_1, \dots, x_n\}$ 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 the VCSP *is a function*

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

where, for each $i = 1, \dots, q$, $f_i(\mathbf{x}_i)$ is a valued constraint over $V_{\mathcal{I}} = \{x_1, \dots, x_n\}$ and $w_i \in \mathbb{Q}_{>0}$ is a weight. The goal is to find a mapping $\varphi : V_{\mathcal{I}} \to D$ that minimizes $f_{\mathcal{I}}$. 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* VCSP(Γ) *denote the class of all* VCSP *instances in which every valued constraint uses a function from* Γ.

Note that, in some papers (especially when infinite values are allowed), functions taking negative values are not allowed in valued constraints. It is easy to see that our results would stay the same if we were to assume this restriction.

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 VCSP(Γ') is tractable for *each* finite $\Gamma' \subseteq \Gamma$, and it is NP-hard if VCSP(Γ') is NP-hard for *some* finite $\Gamma' \subseteq \Gamma$.

Example 1 (submodularity [19, 37, 44]). A function $\{0,1\}^n \to \mathbb{Q}$ is called *submodular* if

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

Here, $\vee$ and $\wedge$ denote the standard binary Boolean operations acting componentwise. If $\Gamma \subseteq F_{\{0,1\}}$ consists of submodular functions, then VCSP(Γ) is tractable [12, 19, 37, 44]. 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 maximize 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 $f_{mc}(0,1) = f_{mc}(1,0) < f_{mc}(0,0) = f_{mc}(1,1)$. Note that f_{mc} is a supermodular set function. Let $\Gamma_{mc} = \{f_{mc}\}$. One can see that VCSP(Γ_{mc}) is equivalent to Max Cut as follows. Variables in an instance of VCSP(Γ_{mc}) can be seen as vertices of a graph $G = (V, E)$, while constraints $w_e \cdot f_{mc}(x, y)$ correspond to edges $e = \{x, y\} \in E$ with weight $w_e > 0$. An assignment of 0-1 values to the variables corresponds to a partition of the graph, and minimizing the function

$$\sum_{e=\{x,y\} \in E} w_e \cdot f_{mc}(x, y)$$

corresponds to maximizing the total weight of cut edges, as each constraint prefers a pair of different values to a pair of equal values. Thus, VCSP(Γ_{mc}) is NP-hard.

The main problem in this research direction is to identify *all* valued constraint languages Γ such that VCSP(Γ) 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."

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* VCSP(Γ) *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 we 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 1 (see [9, 12]). *Let* Γ *and* Γ' *be constraint languages on* D *such that* $\Gamma' \subseteq \langle\Gamma\rangle_{\equiv}$*. If* VCSP($\Gamma$) *is tractable, then* VCSP(Γ') *is tractable. If* VCSP(Γ') *is*

NP-*hard, then* VCSP(Γ) *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 [12].

Lemma 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) [12, 27, 28, 33].

2.3. Fractional polymorphisms. Let $O_D^{(k)} = \{F \mid F : D^k \to D\}$ be the set of all k-ary operations on D, and let $O_D = \bigcup_{k=1}^{\infty} O_D^{(k)}$. For $F \in O_D^{(k)}$ and (not necessarily distinct) tuples $\mathbf{a}_1, \dots, \mathbf{a}_k \in D^n$, let $F(\mathbf{a}_1, \dots, \mathbf{a}_k)$ denote the tuple in D^n obtained by applying F to $\mathbf{a}_1, \dots, \mathbf{a}_k$ coordinatewise.

A *fractional operation* of arity k on D is a probability distribution μ on $O_D^{(k)}$. For a function $f \in F_D^{(n)}$, μ is said to be a *fractional polymorphism* of f [9] if, for all $\mathbf{a}_1, \dots, \mathbf{a}_k \in D^n$,

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

or, in expanded form,

$$\sum_{F \in O_D^{(k)}} \Pr_{\mu}[F] \cdot f(F(\mathbf{a}_1, \dots, \mathbf{a}_k)) \le \frac{1}{k}(f(\mathbf{a}_1) + \cdots + f(\mathbf{a}_k)). \tag{3}$$

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* [12] 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 $\mathbf{F} = (F_1, \dots, F_k)$ 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

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

All important fractional polymorphisms identified earlier [10, 12, 28, 31] 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 ∨) and a greatest lower bound (*meet*, denoted ∧).

A distributive lattice is one whose elements can be represented by subsets of a set so that join and meet are set-theoretic union and intersection, respectively.

Example 3 (submodularity on a lattice [11, 34, 35, 49]). Let $\mathcal{L} = (D, \vee, \wedge)$ be a (not necessarily distributive) lattice. A function $f : D^n \to \mathbb{Q}$ is called *submodular* on $\mathcal{L}$ if

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

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, 19, 40, 41]). 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:

$$1 \vee_0 -1 = -1 \vee_0 1 = 0 \text{ and } x \vee_0 y = \max(x, y) \text{ if } \{x, y\} \neq \{-1, 1\},$$
$$1 \wedge_0 -1 = -1 \wedge_0 1 = 0 \text{ and } x \wedge_0 y = \min(x, y) \text{ if } \{x, y\} \neq \{-1, 1\},$$

where max and min are with respect to the order $-1 > 0 < 1$. A function $f : D^n \to \mathbb{Q}$ is called *bisubmodular* if it has the binary multimorphism μ such that $\Pr_\mu[\vee_0] = \Pr_\mu[\wedge_0] = 1/2$, that is, if $f(\mathbf{a} \vee_0 \mathbf{b}) + f(\mathbf{a} \wedge_0 \mathbf{b}) \leq f(\mathbf{a}) + f(\mathbf{b})$ 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 [19, 31], can be described by suitable multimorphisms.

Fractional polymorphisms not only provide a useful way of describing classes of functions, they also characterize the expressive power of constraint languages.

THEOREM 2 (see [9]). *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 1, this implies that tractable valued constraint languages can be characterized 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 $\alpha \in (0, 1]$. A function $f : \{-1, 0, 1\}^n \to \mathbb{Q}$ is* α-bisubmodular (towards 1) *if it has the fractional polymorphism μ with $\Pr_\mu[\wedge_0] = 1/2$, $\Pr_\mu[\vee_0] = \alpha/2$, 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 $\{-1, 0, 1\}^n$, we have

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

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 skewed 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

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

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

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

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

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

for all $x_1, \dots, x_k \in D$ and any permutation π on $\{1, \dots, k\}$. Such operations play an important role in the algebraic approach to the standard CSP [2, 36]. Call a fractional operation μ idempotent, cyclic, or symmetric if each operation in $\operatorname{supp} \mu = \{F \mid \Pr_\mu[F] > 0\}$ has the corresponding property.

A characterization of valued constraint languages that can be solved by the basic LP relaxation has been obtained in [46]. 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

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

given by the minimization 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

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

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

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 $\mathcal{I}$:

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

THEOREM 3 (see [46]). *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 $\mathcal{I}$ [46], and then an optimal solution of $\mathcal{I}$ 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. Adding constraints $\mu_{x,a} = 1$ corresponds to fixing values for some variables for functions in Γ. The use of functions obtained in that way does not affect the

tractabililty, as we will describe in detail in section 3.1. That is why this procedure leads indeed to an optimal solution of $\mathcal{I}$.

THEOREM 4 (see [32]). *If* Γ *has a fractional polymorphism* μ *of some arity* $k > 1$ *such that* $\operatorname{supp}\mu$ *contains a symmetric operation, then* Γ *has symmetric fractional polymorphisms of all arities.*

In particular, it follows from Theorem 4 that the basic LP relaxation solves VCSP(Γ) in polynomial time if and only if Γ has a binary idempotent commutative (i.e., symmetric) fractional polymorphism.

One can also consider a basic SDP (semidefinite programming) relaxation for VCSPs. The following theorem is implied by results in Chapters 6 and 7 of [42].

THEOREM 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.

3.1. Cores. In this section, we show that each constraint language has an equivalent "core" language, so it is enough to consider only cores. Results of this type play an important role in most CSP-related complexity classifications (e.g., [7, 27]).

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, without loss of generality we can 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. A different definition of a core was used in [48], but it was proved there to be equivalent to ours.

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 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* $\{u_a \mid a \in D\}$ *with* $\operatorname{argmin}(u_a) = \{a\}$.
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 from this proposition 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 that the last condition already implies that Γ is a core. It can easily be checked by using the definition of a fractional polymorphism 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, 7]). Note that the fractional polymorphisms describing submodularity and α-bisubmodularity are idempotent and commutative.

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

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

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

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

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

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

By Theorem 1, the problems VCSP(Γ_c) and VCSP($\Gamma_c \cup \{u_a \mid a \in D\}$) have the same complexity. Therefore, to prove items 2 and 3, it suffices to show that problems VCSP(Γ) and VCSP(Γ_c) have the same complexity. Fix an arbitrary finite subset $\Delta_c \subseteq \Gamma_c$. For each function f_c in Δ_c, fix a function f in Γ such that f_c is obtained from f by fixing values for some variables, and let Δ be the union of the set of all such functions f and the set of all functions from Γ that appear in the instance $\mathcal{I}_p$. Clearly, Δ is a finite subset of Γ. We claim that VCSP(Δ_c) reduces to VCSP(Δ) in polynomial time. Take an arbitrary instance $\mathcal{I}$ of VCSP(Δ_c) and assume without loss of generality that the sets of variables of $\mathcal{I}$ and $\mathcal{I}_p$ are disjoint. Replace each f_c in $\mathcal{I}$ by the corresponding f with constants, and then replace each constant a by the variable x_a. The new instance $\mathcal{I}_1$ is an instance of VCSP(Δ); denote its objective function by f_1. Consider the instance $\mathcal{I}_2$ of VCSP(Δ) with objective function $f_1 + W \cdot f_p$, where W is a large enough number to ensure that each optimal solution to $\mathcal{I}_2$, when restricted to $V_{\mathcal{I}_p} = \{x_a \mid a \in D\}$, is an optimal solution to $\mathcal{I}_p$. Since φ_{id} is an optimal solution to $\mathcal{I}_p$, the optimal solutions to $\mathcal{I}$ are precisely restrictions to $V_{\mathcal{I}}$ of those optimal solutions to $\mathcal{I}_2$ whose restriction on $V_{\mathcal{I}_p}$ is φ_{id}.

Each optimal solution φ to $\mathcal{I}$ gives rise to an optimal solution to $\mathcal{I}_2$ which coincides with φ on $V_{\mathcal{I}}$ and with φ_{id} on $V_{\mathcal{I}_p}$. In the other direction, let φ_2 be an optimal solution to $\mathcal{I}_2$. Its restriction to $V_{\mathcal{I}_p}$ is an optimal solution φ_2' to $\mathcal{I}_p$. By Lemma 2(1), the operation $s_{\varphi_2'}$ is a permutation on D. By applying Lemma 2 to $\mathcal{I}' = \mathcal{I}_p$, it follows

that each mapping ψ_t such that $s_{\psi_t} = s^t_{\varphi_2'}$, $t \geq 1$, is an optimal solution to $\mathcal{I}_p$. Choose t so that $s^t_{\varphi_2'} = s^{-1}_{\varphi_2'}$. Now apply Lemma 2(2) to ψ_t and $\varphi_2 \in \mathrm{Opt}(\mathcal{I}_2)$. It follows that $\varphi_2'' = s^{-1}_{\varphi_2'} \circ \varphi_2$ is an optimal solution to $\mathcal{I}_2$, and by construction, $\varphi_2''(x_a) = a$ for each $a \in D$. Hence, the restriction of φ_2'' to $V_{\mathcal{I}}$ is an optimal solution to $\mathcal{I}$. ∎

3.2. A characterization of α-bisubmodularity. In this section, we give a characterization of α-bisubmodularity which enables us to check for α-bisubmodularity by just verifying it on certain easy-to-check subsets.

Let $\leq$ denote the partial order on $\{-1, 0, 1\}$ such that $0 \leq t$ for all $t \in \{-1, 0, 1\}$ and $-1, 1$ are incomparable. Let $\leq$ also denote the componentwise partial order on $\{-1, 0, 1\}^n$. For every $\mathbf{c} \in \{-1, 1\}^n$, let

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

This set is called the *orthant* of $\mathbf{c}$.

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 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.*
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]. For the proof we will need the following lemma.

LEMMA 3. *Assume that $f : \{-1, 0, 1\}^n \to \mathbb{Q}$ is submodular in every orthant and that every unary function obtained from f by fixing values for all but one variable is α-bisubmodular. Then every unary function obtained from f by fixing values for and identifying variables is α-bisubmodular.*

Proof. Assume that $f : \{-1, 0, 1\}^n \to \mathbb{Q}$ satisfies the conditions of the lemma. Let $1 \leq m \leq n$, and let $g : \{-1, 0, 1\}^m \to \mathbb{Q}$ be obtained from f by fixing values for some of (possibly none of) the variables. Note that g is also submodular in each orthant. Let $h : \{-1, 0, 1\} \to \mathbb{Q}$ be obtained from g by identifying all remaining variables, i.e., $h(x) = g(x, \ldots, x)$. We have to show that $(1 + \alpha) \cdot h(0) \leq h(-1) + \alpha \cdot h(1)$.

We use induction on m. The case $m = 1$ holds by the assumptions of the lemma. Assume the result holds for $m - 1$.

The induction hypothesis, applied to $g(x, \ldots, x, 1)$, gives

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

Submodularity in the orthant of $(-1, \ldots, -1, 1)$ gives

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

and the assumption on unary functions, applied to $g(-1, \ldots, -1, x)$, gives

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

If we multiply the second inequality by $(1+\alpha)$ and add all three inequalities, then we get the required $h(-1)+\alpha\cdot h(1)\geq(1+\alpha)\cdot h(0)$. ∎

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

$$x\vee_{a,b}y=x\vee_0 y \text{ if } \{x,y\}\neq\{-1,1\},\qquad -1\vee_{a,b}1=a \text{ and } 1\vee_{a,b}-1=b.$$

Proof of Proposition 2. The "only if" direction follows easily from the definitions, so let us prove the other one. Let f satisfy conditions 1 and 2 from the proposition. By Lemma 3 we can assume that every unary function obtained from f by fixing and identifying variables is α-bisubmodular.

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

$$f(\mathbf{x})+f((\mathbf{x}\vee_0\mathbf{y})\wedge_0\mathbf{y})\geq f(\mathbf{x}\wedge_0\mathbf{y})+f((\mathbf{x}\vee_0\mathbf{y})\vee_0\mathbf{x}). \tag{6}$$

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

$$f(\mathbf{y})+f(\mathbf{x}\vee_0\mathbf{y})\geq f((\mathbf{x}\vee_0\mathbf{y})\wedge_0\mathbf{y})+f((\mathbf{x}\vee_0\mathbf{y})\vee_0\mathbf{y}). \tag{7}$$

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

$$f(\mathbf{x}\vee_{0,1}\mathbf{y})+f(\mathbf{x}\vee_{1,0}\mathbf{y})\geq f(\mathbf{x}\vee_0\mathbf{y})+f(\mathbf{x}\vee_1\mathbf{y}). \tag{8}$$

Applying α-bisubmodularity of the unary function obtained from f by identifying all those variables with indices i such that $\mathbf{x}_i=-1$ and $\mathbf{y}_i=1$ (i.e., the coordinates where $(\mathbf{x}\vee_0\mathbf{y})\vee_0\mathbf{x}$, $\mathbf{x}\vee_1\mathbf{y}$, and $\mathbf{x}\vee_{0,1}\mathbf{y}$ differ) and fixing all other variables gives

$$f((\mathbf{x}\vee_0\mathbf{y})\vee_0\mathbf{x})+\alpha f(\mathbf{x}\vee_1\mathbf{y})\geq(1+\alpha)f(\mathbf{x}\vee_{0,1}\mathbf{y}). \tag{9}$$

The same argument with identifying all those variables with indices $i\in[n]$ such that $\mathbf{x}_i=1$ and $\mathbf{y}_i=-1$ (i.e., the coordinates where $(\mathbf{x}\vee_0\mathbf{y})\vee_0\mathbf{y}$, $\mathbf{x}\vee_1\mathbf{y}$, and $\mathbf{x}\vee_{1,0}\mathbf{y}$ differ) and fixing all other variables gives

$$f((\mathbf{x}\vee_0\mathbf{y})\vee_0\mathbf{y})+\alpha f(\mathbf{x}\vee_1\mathbf{y})\geq(1+\alpha)f(\mathbf{x}\vee_{1,0}\mathbf{y}). \tag{10}$$

We then have the following chain of inequalities:

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

where the first inequality follows from (6) and (7), the second from (9) and (10), and the last from (8). By comparing the first and last expressions in the above chain, we get the required inequality

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

3.3. A dichotomy theorem. In this section we state and discuss our main theorem, Theorem 8, which is the classification of the complexity of VCSPs with a fixed constraint language in the case of a 3-element domain. The proof will be given in the next section. Theorem 8 generalizes the following two theorems, which are the classification for the Boolean case [12] and the complexity classification for the case of a 3-element domain and 0-1-valued functions [27].

THEOREM 6 (see [12]). *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 7 (see [27]). *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 following theorem is the main result of our paper.

THEOREM 8. *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*

(i) *each function in Γ is submodular on the chain $-1 < 0 < 1$, or*

(ii) *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 3 and 4. (It can also be derived directly from Theorem 4.1 of [46].) For the hardness part, by Lemma 1 and Proposition 1, it suffices to show that Γ_c satisfies (MC), which we do in section 3.4.

We now show that the tractable cases from Theorem 8 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, and it cannot 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. *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 being 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:

(i) unary $e = [1, 0, 1]$, $u_\alpha = [p + q, q, 0]$, and $v_\alpha = [0, p, p + q]$.

(ii) binary f_α such that $f_\alpha(1, -1) = f_\alpha(-1, 1) = 1$, $f_\alpha(0, -1) = f_\alpha(-1, 0) = 1 + q$, $f_\alpha(-1, -1) = 1 + p + q$, and $f(x, y) = 0$ on the remaining pairs (x, y).

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

Notice that f_α is not submodular when restricted to $\{-1, 1\}$. Therefore Γ_α is not submodular on any chain on $\{-1, 0, 1\}$. It is easy to check that u_α is not α'-

bisubmodular for any $\alpha' > \alpha$, and v_α is not α'-bisubmodular for any $\alpha' < \alpha$. 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 $\alpha = \alpha' = 1$. ∎

The problem of deciding whether a given finite constraint language Γ on a fixed set D has a binary idempotent commutative fractional polymorphism can be solved by LP; see [32]. However, our characterization of α-bisubmodular functions leads to a simple algorithm for recognizing tractable cases.

PROPOSITION 4. *Let $|D| = 3$. There is a cubic-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 $n_1, \dots, n_k$ and M is the value taken by functions in Γ that has the largest representation (as a rational number p/q with $\gcd(p, q) = 1$), then the size of Γ is $(3^{n_1} + \cdots + 3^{n_k}) \cdot \log M$. 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 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 in the size of Γ (since a function of arity n_i has 2^{n_i} orthants), 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 $(1+\alpha) \cdot f(0) \le f(-1) + \alpha \cdot f(1)$ 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. Proof of Theorem 8. In this section, we will prove our main result, Theorem 8. Assume that $|D| = 3$ and that we have a constraint language Γ which is a core. By Proposition 1 and Theorem 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 $\operatorname{argmin}(u_a) = \{a\}$.

In the next two proofs, when we add unary functions, we always assume that they depend on the same variable.

LEMMA 4. *For at least two distinct 2-element subsets $X \subseteq D$, Γ contains functions u_X with* $\operatorname{argmin}(u_X) = X$.

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

$$[1, \delta, 0] + (1 - \delta)[0, 1, \beta] = [1, 1, (1 - \delta)\beta],$$
$$\frac{\beta - 1}{\delta}[1, \delta, 0] + [0, 1, \beta] = \left[\frac{\beta - 1}{\delta}, \beta, \beta\right].$$

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

Now consider

$$[1,\delta,0]+\frac{\delta-1}{\gamma}[\gamma,0,1]=\left[\delta,\delta,\frac{\delta-1}{\gamma}\right],$$
$$(1-\gamma)[1,\delta,0]+[\gamma,0,1]=[1,(1-\gamma)\delta,1].$$

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

Finally, consider

$$[0,1,\beta]+(1-\beta)[\gamma,0,1]=[(1-\beta)\gamma,1,1],$$
$$\frac{\gamma-1}{\beta}[0,1,\beta]+[\gamma,0,1]=\left[\gamma,\frac{\gamma-1}{\beta},\gamma\right].$$

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

Thus we can assume that $(1-\delta)\beta\le 1$, $(1-\gamma)\delta\le 1$, and $(1-\beta)\gamma\le 1$. It is impossible for more than one of these inequalities to be an equality when $\delta,\beta,\gamma>0$. For example, if the first two are equalities, then $1-\delta>0$, so $\delta<1$, so $1-\gamma>1$, and $\gamma<0$. Hence, at least two of these inequalities are strict, and then we can generate two required functions as follows:

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

Let us rename the elements of D into $-1,0,1$ so that the two 2-element subsets guaranteed by Lemma 4 are $\{-1,0\}$ and $\{0,1\}$. From now on we assume that $D=\{-1,0,1\}$ and that Γ contains u_X for each nonempty 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 5. *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.

Proof. Recall that we assume that $\Gamma=\langle\Gamma_c\rangle_{\equiv}$. If Γ contains a unary function h with $h(-1)\le h(0)\ge h(1)$, where at least one inequality is strict, then one gets $u_{\{-1,1\}}$ as follows. If only one inequality is strict, then, by adding h and one of u_{-1}, u_1 with a suitable coefficient, one gets a function where both inequalities are strict. If both inequalities are strict, then, by adding h and one of $u_{\{-1,0\}}$, $u_{\{0,1\}}$ (with a suitable coefficient), one gets a function u' with $\operatorname{argmin}(u')=\{-1,1\}$, which can be taken as $u_{\{-1,1\}}$.

So, assuming that condition 1 from the lemma does not hold, we can now make the following assumption:

(*) No unary function $h\in\Gamma$ satisfies $h(-1)\le h(0)\ge h(1)$ unless $h(-1)=h(0)=h(1)$.

We need to show that there exists $\alpha\in(0,1]$ such that the unary functions in Γ are α-bisubmodular, all towards 1 or all towards -1. Let Λ be the set of all unary functions from D to $\mathbb{Q}$ obtained by translating and scaling each function in Γ so

that each function $g \in \Lambda$ satisfies $g(0) = 0$ and $g(-1) \in \{-1, 0, 1\}$. Notice that the property of being α-bisubmodular is not affected when scaling and translating. Therefore, it suffices to show that there is an $\alpha \in (0, 1]$ such that all functions $g \in \Lambda$ are α-bisubmodular, all towards 1 or all towards -1. That is, we need to show that there is an $\alpha \in (0, 1]$ such that all $g \in \Lambda$ satisfy $0 \le \alpha \cdot g(1) + g(-1)$ or all $g \in \Lambda$ satisfy $0 \le \alpha \cdot g(-1) + g(1)$.

Assume that each function u_X is scaled to be in Λ. In particular, $u_1(-1) \in \{0, 1\}$ and $u_1(1) < 0$. Also, we have $u_{-1}(-1) = -1$ and $u_{-1}(1) > 0$.

If all of the unary functions in Λ satisfy $f(-1) + f(1) \ge 0$, then all of them are 1-bisubmodular.

Let f be a unary function in Λ with $f(-1) + f(1) < 0$. We do a case analysis on the three possible values for $f(-1)$.

First case: $f(-1) = 0$. Then $f(1) < 0$, which contradicts assumption (*).

Second case: $f(-1) = 1$. Let Z_1 be the set of all unary functions g in Λ with $g(-1) + g(1) < 0$ and $g(-1) = 1$. Then $g(1) < -1$ for all $g \in Z_1$. Note that $f \in Z_1$, so Z_1 is nonempty.

Let $\alpha = \inf_{g \in Z_1} -\frac{1}{g(1)}$. Note that $0 \le \alpha < 1$. If $\alpha = 0$, then there exists a $g' \in Z_1$ with $g'(1) < -u_{-1}(1)$, but then the function $g' + u_{-1} \in \Gamma$ contradicts assumption (*). If $\alpha > 0$, all unary functions g in Λ with $g(-1) = 1$ are α-bisubmodular towards 1. So either all unary functions in Λ are α-bisubmodular towards 1 and we are done, or there is a unary function h in Λ with $h(-1) \in \{-1, 0\}$ and $\alpha \cdot h(1) + h(-1) < 0$. If $h(-1) = 0$, then $h(1) < 0$, which contradicts assumption (*). Let $h(-1) = -1$. Then we have $h(1) > 0$ by assumption (*), and so $\alpha \cdot h(1) + h(-1) < 0$ yields $\alpha < \frac{1}{h(1)}$. By the definition of α, there is a $g' \in Z_1$ with $-\frac{1}{g'(1)} < \frac{1}{h(1)}$. The function $g' + h \in \Gamma$ contradicts assumption (*).

Third case: $f(-1) = -1$. Let Z_{-1} be the set of all unary functions g in Λ with $g(-1) = -1$. By assumption (*), we have $g(1) > 0$ for all $g \in Z_{-1}$.

Let $\alpha := \inf_{g \in Z_{-1}} g(1)$. Since $f \in Z_{-1}$ and $f(1) < 1$, we have $0 \le \alpha < 1$. If $\alpha = 0$, that means that there is a $g \in Z_{-1}$ with $g(1) < -u_1(1)$, but then the function $g + u_1 \in \Gamma$ contradicts assumption (*). So we can assume $\alpha > 0$. All elements of Z_{-1} are α-bisubmodular towards -1. Similarly to the second case above, one can show that all functions in Λ must be α-bisubmodular towards -1, or one gets a contradiction with assumption (*) again. ∎

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 [33]. Theorem 3.6 in [33] states (referring to [32] for a formal proof) that either Γ is submodular on a chain or Γ is NP-hard. The hardness is shown, again, by satisfying (MC). Thus, in this case, the assertion of Theorem 8 holds.

Let us assume for the rest of this section that $u_{\{-1,1\}} \notin \Gamma$. By Lemma 5, 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 2, Γ contains a function which is not submodular in some orthant. The following lemma is well known; see, e.g., [49]. The notion of submodularity from Example 3 can be naturally extended to the direct product of lattices by defining the operations componentwise.

LEMMA 6. *Let $D_1, \dots, D_n$ be finite chains. If a function $f : D_1 \times \cdots \times D_n \to \mathbb{Q}$*

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 6 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 [12], Γ satisfies (MC), with $u_{\{0,1\}}$ or $u_{\{-1,0\}}$, 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 6, 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 $(-1,-1)$; we are done otherwise. If g satisfies both $g(1,0)+g(0,-1) \le g(0,0)+g(1,-1)$ and $g(0,1)+g(-1,0) \le g(0,0)+g(-1,1)$, 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 8.

LEMMA 7. *If Γ contains binary functions f and g as above, then Γ contains a binary function which is not submodular in the orthant of $(-1,-1)$.*

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

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

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

We define the binary function g' through

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

We have $g' \in \Gamma$, $g'(1,0) = g'(0,-1)$, and also $g'(1,-1) < g'(1,0)+g'(0,-1)$. If $g'(1,-1) \le 0$, let $C > -g'(1,-1)$, and define the binary function g'' through

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

for all $x, y \in D$; otherwise we keep $g'' = g'$. We have $g'' \in \Gamma$, $g''(1,0) = g''(0,-1)$, and $0 < g''(1,-1) < g''(1,0)+g''(0,-1)$. Now we can obtain, by scaling, a binary function g''' in Γ with $g'''(0,0) = 0$, $g'''(0,-1) = g'''(1,0) = 1$, and $0 < g'''(1,-1) < 2$. We can assume that $g = g'''$ from the beginning.

Note that $f(-1,1) - 2 > 0$ and $g(1,-1) > 0$. By scaling the unary function u_1, we can obtain a unary function u_1' in Γ such that

$$g(1,-1) < u_1'(0) < \min\{2, f(-1,1)+g(1,-1)-2\} \text{ and } u_1'(1) = 0.$$

By adding $u_{\{0,1\}}$ with a large enough coefficient, we can obtain a unary function v in Γ, which still fulfills

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

but where the value $v(-1)$ is as large as we want, and for our purposes

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

is enough. Now, Γ also contains the binary function s defined by

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

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

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

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

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

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

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

3.5. Multimorphisms versus fractional polymorphisms. While it has been known that fractional polymorphisms characterize expressive power [9], it was open whether tractability of valued constraint languages could be characterized 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 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 $\{-1,0,1\}$.*

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

$$\sum_{i=1}^{k} f(F_i(\mathbf{x}_1, \dots, \mathbf{x}_k)) \leq \sum_{i=1}^{k} f(\mathbf{x}_i) \tag{11}$$

for all $\mathbf{x}_1, \dots, \mathbf{x}_k \in D^n$. We define the function $\mathbf{F} : D^k \to D^k$ by $\mathbf{F} = (F_1, \dots, F_k)$ and identify μ with $\mathbf{F}$. For the proof of Proposition 5 we will use the following 1/2-bisubmodular functions:

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

Let $\Gamma = \{u_{\{-1,0\}}, u_{\{0,1\}}, v_1, v_{-1}, b\}$. It is easy to check (by using Definition 4) that these functions are indeed 1/2-bisubmodular.

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 inequality (11) holds with equality for every unary function. So the following lemma finishes the proof of Proposition 5.

LEMMA 8. *If $\mathbf{F}$ is a multimorphism of Γ, then $\mathbf{F}$ preserves multisets.*

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

Let $\mathbf{d} = (d_1, \dots, d_k) \in D^k$, let ℓ be the number of 1's in $\mathbf{d}$, and let m be the number of -1's. Let ℓ' be the number of 1's in $\mathbf{F}(\mathbf{d})$, and let m' be the number of -1's. Applying (11) with $u_{\{-1,0\}}$ and $u_{\{0,1\}}$ gives $\ell' \le \ell$ and $m' \le m$, and applying (11) with v_1 and v_{-1} gives

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

Now we will show, using the binary function $b \in \Gamma$, that any multiset has to be preserved by $\mathbf{F}$. Without loss of generality, let $d_1 = \cdots = d_\ell = 1$ and $F_1(\mathbf{d}) = \cdots = F_{\ell'}(\mathbf{d}) = 1$. We have $\ell' = \ell - x$ and $m' = m - 2x$ for some nonnegative integer x.

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

For every $p \in [\ell]$, let $\mathbf{e}^{(p)} \in D^k$ be defined as follows: $\mathbf{e}_p^{(p)} := -1$ and $\mathbf{e}_i^{(p)} := 1$ for every $i \in [k] \setminus \{p\}$. We have

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

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

If $F_{j^{(p)}}(\mathbf{d}) = -1$, we have

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

which is a contradiction to (11).

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

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

which also is a contradiction to (11). So for every $p \in [\ell]$ we have $F_{j^{(p)}}(\mathbf{d}) = 1$. This yields $\{j^{(p)} \mid p \in [\ell]\} = [\ell']$, and, if $\ell' < \ell$, there must be two different indices $p, q \in [\ell]$ such that $j^{(p)} = j^{(q)}$.

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

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

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

contradicting to (11). So we cannot have $\ell' < \ell$, and thus we have $\ell' = \ell$ and $m' = m$. ∎

4. Conclusion and discussion. We have classified the complexity of VCSPs (with finite costs) on a 3-element domain. The tractable cases are described by simple fractional polymorphisms. After the submission of this paper, Thapper and Živný proved in [48] that, for a core constraint language Γ, VCSP(Γ) is NP-hard unless Γ has a binary idempotent commutative fractional polymorphism (in which case the problem is tractable by [32, 46]). It would be interesting to refine this result by being more specific about which fractional polymorphisms one needs to take there. As a possible first step, one might try to derive our classification from the general classification of Thapper and Živný.

Many efficient algorithms exist for minimizing submodular functions (see, e.g., [19, 26, 39, 44]). Lovász asked in [37] whether there is a generalization of submodularity that preserves the nice properties such as efficient minimization, and this question led to the discovery of bisubmodularity in [41] (where it is called directed submodularity) and subsequent efficient minimization algorithms for it (see, e.g., [40, 41]). The following interesting problem was mentioned in [28, 46].

PROBLEM 1. *Which fractional operations μ guarantee an efficient minimization algorithm, in the value-oracle model, for the class of all functions f with $\mu \in \mathrm{fPol}(f)$?*

Some initial results in this direction can be found in [24, 34, 35]. At the time of submission it was open whether α-bisubmodular functions can be efficiently minimized in the value-oracle model, but this question was recently answered in the positive in [20, 25]. One possible first step to approaching Problem 1 is to consider specific fractional polymorphisms, e.g., those from Example 3 or k-submodularity from [24]. Once some efficient algorithms are discovered (if they exist), one could naturally try to design strongly polynomial or (fully) combinatorial algorithms.

Acknowledgments. 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 5 follows from [42], and Johan Thapper for suggesting the notion of core for VCSP that we use in the paper. We also thank the referees for their useful comments.

REFERENCES

[1] K. ANDO, S. FUJISHIGE, AND T. NAITOH, *A characterization of bisubmodular functions*, Discrete Math., 148 (1996), pp. 299–303.

[2] L. BARTO AND M. KOZIK, *Absorbing subalgebras, cyclic terms and the constraint satisfaction problem*, Log. Methods Comput. Sci., 8 (2012), 1:07.

[3] L. BARTO AND M. KOZIK, *Robust satisfiability of constraint satisfaction problems*, in Proceedings of the 44th ACM Symposium on Theory of Computing (STOC'12), ACM, New York, 2012, pp. 931–940.

[4] E. Boros and P. L. Hammer, *Pseudo-Boolean optimization*, Discrete Appl. Math., 123 (2002), pp. 155–225.
[5] A. Bulatov, *A dichotomy theorem for constraint satisfaction problems on a* 3*-element set*, J. ACM, 53 (2006), pp. 66–120.
[6] A. Bulatov, *Complexity of conservative constraint satisfaction problems*, ACM Trans. Comput. Logic, 12 (2011), 24.
[7] A. Bulatov, P. Jeavons, and A. Krokhin, *Classifying the complexity of constraints using finite algebras*, SIAM J. Comput., 34 (2005), pp. 720–742.
[8] D. A. Cohen, M. C. Cooper, P. Creed, P. G. Jeavons, and S. Živný, *An algebraic theory of complexity for discrete optimization*, SIAM J. Comput., 42 (2013), pp. 1915–1939.
[9] D. A. Cohen, M. C. Cooper, and P. G. Jeavons, *An algebraic characterisation of complexity for valued constraints*, in Principles and Practice of Constraint Programming (CP'06), Lecture Notes in Comput. Sci. 4204, Springer, Berlin, 2006, pp. 107–121.
[10] D. Cohen, M. Cooper, and P. Jeavons, *Generalising submodularity and Horn clauses: Tractable optimization problems defined by tournament pair multimorphisms*, Theoret. Comput. Sci., 401 (2008), pp. 36–51.
[11] D. Cohen, M. Cooper, P. Jeavons, and A. Krokhin, *Supermodular functions and the complexity of Max CSP*, Discrete Appl. Math., 149 (2005), pp. 53–72.
[12] D. Cohen, M. Cooper, P. Jeavons, and A. Krokhin, *The complexity of soft constraint satisfaction*, Artificial Intelligence, 170 (2006), pp. 983–1016.
[13] D. Cohen and P. Jeavons, *The complexity of constraint languages*, in Handbook of Constraint Programming, F. Rossi, P. van Beek, and T. Walsh, eds., Elsevier, Dordrecht, The Netherlands, 2006, pp. 245–280.
[14] Y. Crama and P. L. Hammer, *Boolean Functions—Theory, Algorithms and Applications*, Cambridge University Press, Cambridge, UK, 2011.
[15] N. Creignou, S. Khanna, and M. Sudan, *Complexity Classifications of Boolean Constraint Satisfaction Problems*, Discrete Math. Appl. 7, SIAM, Philadelphia, 2001.
[16] N. Creignou, Ph. G. Kolaitis, and H. Vollmer, eds., *Complexity of Constraints*, Lecture Notes in Comput. Sci. 5250, Springer, Berlin, 2008.
[17] 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 (1998), pp. 57–104.
[18] U. Feige, V. S. Mirrokni, and J. Vondrák, *Maximizing non-monotone submodular functions*, SIAM J. Comput., 40 (2011), pp. 1133–1153.
[19] S. Fujishige, *Submodular Functions and Optimization*, 2nd ed., Ann. Discrete Math. 58, Elsevier, Dordrecht, The Netherlands, 2005.
[20] S. Fujishige, S. Tanigawa, and Y. Yoshida, *Generalized Skew Bisubmodularity: A Characterization and a Min-Max Theorem*, Technical Report RIMS-1781, Kyoto University, Kyoto, Japan, 2013.
[21] G. Gottlob, G. Greco, and F. Scarcello, *Tractable optimization problems through hypergraph-based structural restrictions*, in Proceedings of the 36th International Colloquium on Automata, Languages, and Programming (ICALP'09), Lecture Notes in Comput. Sci. 5556, Springer, Berlin, 2009, pp. 16–30.
[22] H. Hirai, *Discrete convexity and polynomial solvability in minimum* 0*-extension problems*, in Proceedings of the Twenty-Fourth ACM-SIAM Symposium on Discrete Algorithms (SODA'13), ACM, New York, SIAM, Philadelphia, 2013, pp. 1770–1788.
[23] H. Hirai, *Discrete convexity for multiflows and* 0*-extensions*, in Proceedings of the 8th Japanese-Hungarian Symposium on Discrete Mathematics and Its Applications, 2013.
[24] A. Huber and V. Kolmogorov, *Towards minimizing k-submodular functions*, in Proceedings of the 2nd International Symposium on Combinatorial Optimization (ISCO'12), Athens, 2012, pp. 451–462.
[25] A. Huber and A. Krokhin, *Oracle Tractability of Skew Bisubmodular Functions*, preprint, arXiv:1308.6505, 2013.
[26] S. Iwata and J. Orlin, *A simple combinatorial algorithm for submodular function minimization*, in Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'09), ACM, New York, SIAM, Philadelphia, 2009, pp. 1230–1237.
[27] P. Jonsson, M. Klasson, and A. Krokhin, *The approximability of three-valued max CSP*, SIAM J. Comput., 35 (2006), pp. 1329–1349.
[28] P. Jonsson, F. Kuivinen, and J. Thapper, *Min CSP on four elements: Moving beyond submodularity*, in Principles and Practice of Constraint Programming (CP'11), Lecture Notes in Comput. Sci. 6876, Springer, Berlin, 2011, pp. 438–453.
[29] P. Jonsson and G. Nordh, *Introduction to the maximum solution problem*, in Complexity of Constraints, Lecture Notes in Comput. Sci. 5250, Springer, Berlin, 2008, pp. 255–282.

[30] S. Khot, *On the unique games conjecture*, in Proceedings of the 25th Annual IEEE Conference on Computational Complexity (CCC'10), IEEE, Piscataway, NJ, 2010, pp. 99–121.
[31] V. Kolmogorov, *Submodularity on a tree: Unifying $L^{\#}$-convex and bisubmodular functions*, in Proceedings of Mathematical Foundations of Computer Science (MFCS'11), Lecture Notes in Comput. Sci. 6907, Springer, Heidelberg, 2011, pp. 400–411.
[32] V. Kolmogorov, *The power of linear programming for valued CSPs: A constructive characterization*, in Proceedings of the 40th International Colloquium on Automata, Languages, and Programming (ICALP'13), Lecture Notes in Comput. Sci. 7965, Springer, Berlin, 2013, pp. 625–636.
[33] V. Kolmogorov and S. Živný, *The complexity of conservative valued CSPs*, J. ACM, 60 (2013), 10.
[34] A. Krokhin and B. Larose, *Maximizing supermodular functions on product lattices, with application to maximum constraint satisfaction*, SIAM J. Discrete Math., 22 (2008), pp. 312–328.
[35] F. Kuivinen, *On the complexity of submodular function minimisation on diamonds*, Discrete Optim., 8 (2011), pp. 459–477.
[36] G. Kun, R. O'Donnell, T. Suguru, Y. Yoshida, and Y. Zhou, *Linear programming, width-1 CSPs, and robust satisfaction*, in Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS'12), ACM, New York, 2012, pp. 484–495.
[37] L. Lovász, *Submodular functions and convexity*, in Mathematical Programming: The State of the Art, A. Bachem, M. Grötschel, and B. Korte, eds., Springer, Berlin, 1983, pp. 235–257.
[38] D. Marx, *Tractable hypergraph properties for constraint satisfaction and conjunctive queries*, in Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC'10), ACM, New York, 2010, pp. 735–744.
[39] S. T. McCormick, *Submodular function minimization*, in Handbook on Discrete Optimization, K. Aardal, G. Nemhauser, and R. Weismantel, eds., Elsevier, Dordrecht, The Netherlands, 2006, pp. 321–391.
[40] S. T. McCormick and S. Fujishige, *Strongly polynomial and fully combinatorial algorithms for bisubmodular function minimization*, Math. Programming Ser. A, 122 (2010), pp. 87–120.
[41] L. Q. Qi, *Directed submodularity, ditroids and directed submodular flows*, Math. Programming Ser. B, 42 (1988), pp. 579–599.
[42] P. Raghavendra, *Approximating NP-Hard Problems: Efficient Algorithms and Their Limits*, Ph.D. Thesis, University of Washington, Seattle, WA, 2009.
[43] T. J. Schaefer, *The complexity of satisfiability problems*, in Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC'78), ACM, New York, 1978, pp. 216–226.
[44] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*, Springer, Berlin, 2003.
[45] R. Takhanov, *A dichotomy theorem for the general minimum cost homomorphism problem*, in Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS'10), Nancy, France, 2010, pp. 657–668.
[46] J. Thapper and S. Živný, *The power of linear programming for valued CSPs*, in Proceedings of the 53rd IEEE Symposium on Foundations of Computer Science (FOCS'12), IEEE, Piscataway, NJ, 2012, pp. 669–678.
[47] J. Thapper and S. Živný, *The Power of Linear Programming for Valued CSPs*, preprint, arXiv:1204.1079v2, 2012.
[48] J. Thapper and S. Živný, *The complexity of finite-valued CSPs*, in Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC'13), ACM, New York, 2013, pp. 695–704.
[49] D. Topkis, *Minimizing a submodular function on a lattice*, Oper. Res., 26 (1978), pp. 305–321.
[50] M. J. Wainwright and M. I. Jordan, *Graphical models, exponential families, and variational inferences*, Found. Trends Mach. Learn., 1 (2008), pp. 1–305.