The complexity of conservative valued CSPs

(Extended Abstract)

Vladimir Kolmogorov*
Institute of Science and Technology (IST), Austria
vnk@ist.ac.at

Stanislav Živný†
University of Oxford, UK
standa.zivny@cs.ox.ac.uk

Abstract

We study the complexity of valued constraint satisfaction problems (VCSP). A problem from VCSP is characterised by a *constraint language*, a fixed set of cost functions over a finite domain. An instance of the problem is specified by a sum of cost functions from the language and the goal is to minimise the sum. Under the unique games conjecture, the approximability of finite-valued VCSPs is well-understood, see Raghavendra [FOCS'08]. However, there is no characterisation of finite-valued VCSPs, let alone general-valued VCSPs, that can be solved exactly in polynomial time, thus giving insights from a combinatorial optimisation perspective.

We consider the case of languages containing all possible unary cost functions. In the case of languages consisting of only $\{0, \infty\}$-valued cost functions (i.e. relations), such languages have been called *conservative* and studied by Bulatov [LICS'03] and recently by Barto [LICS'11]. Since we study valued languages, we call a language *conservative* if it contains all finite-valued unary cost functions. The computational complexity of conservative valued languages has been studied by Cohen *et al.* [AIJ'06] for languages over Boolean domains, by Deineko *et al.* [JACM'08] for $\{0, 1\}$-valued languages (a.k.a Max-CSP), and by Takhanov [STACS'10] for $\{0, \infty\}$-valued languages containing all finite-valued unary cost functions (a.k.a. Min-Cost-Hom).

We prove a Schaefer-like dichotomy theorem for conservative valued languages: if all cost functions in the language satisfy a certain condition (specified by a complementary combination of *STP and MJN multimorphisms*), then any instance can be solved in polynomial time (via a new algorithm developed in this paper), otherwise the language is NP-hard. This is the *first* complete complexity classification of *general-valued constraint languages* over non-Boolean domains. It is a common phenomenon that complexity classifications of problems over non-Boolean domains is significantly harder than the Boolean case. The polynomial-time algorithm we present for the tractable cases is a generalisation of the submodular minimisation problem and a result of Cohen *et al.* [TCS'08].

Our results generalise previous results by Takhanov [STACS'10] and (a subset of results) by Cohen *et al.* [AIJ'06] and Deineko *et al.* [JACM'08]. Moreover, our results do not rely on any computer-assisted search as in Deineko *et al.* [JACM'08], and provide a powerful tool for proving hardness of finite-valued and general-valued languages.

*Vladimir Kolmogorov was supported by the Royal Academy of Engineering/EPSRC, UK.

†Stanislav Živný is supported by a Junior Research Fellowship at University College, Oxford. Part of this work was done while the second author was visiting Microsoft Research Cambridge.

1 Introduction

The constraint satisfaction problem is a central generic problem in computer science. It provides a common framework for many theoretical problems as well as for many real-life applications, see [27] for a nice survey. An instance of the *constraint satisfaction problem* (CSP) consists of a collection of variables which must be assigned values subject to specified constraints. CSP is equivalent to the problem of evaluating conjunctive queries on databases [33], and to the homomorphism problem for relational structures [23].

An important line of research on the CSP is to identify all tractable cases; that is, cases that are recognisable and solvable in polynomial time. Most of this work has been focused on one of the two general approaches: either identifying structural properties of the way constraints interact which ensure tractability no matter what forms of constraints are imposed [21], or else identifying forms of constraints which are sufficiently restrictive to ensure tractability no matter how they are combined [10, 23].

The first approach has been used to characterise all tractable cases of bounded-arity CSPs: the *only* class of structures which ensures tractability (subject to a certain complexity theory assumption, namely FPT $\neq$ W[1]) are structures of bounded tree-width modulo homomorphic equivalence [19, 25, 26, 36]; and recently also for unbounded-arity CSPs [37]. The second approach has led to identifying certain algebraic properties known as polymorphisms [29] which are necessary for a set of constraint types to ensure tractability. A set of constraint types which ensures tractability is called a *tractable constraint language*.

Schaefer in his seminal work [41] gave a complete complexity classification of Boolean constraint languages. The algebraic approach based on polymorphisms [30] has been so far the most successful tool in generalising Schaefer's result to languages over a 3-element domain [9], languages with all unary relations [11, 4], languages comprised of a single binary relation without sources and sinks [3] (see also [6]), and languages comprised of a single binary relation that is a special triad [2]. The algebraic approach has also been essential in characterising the power of local consistency [5] and the "few subpowers property" [7], the two

main tools known for solving tractable CSPs. A major open question in this line of research is the *Dichotomy Conjecture* of Feder and Vardi, which states that every constraint language is either tractable or NP-hard [23]. We remark that there are other approaches to the dichotomy conjecture; see, for instance, [27] for a nice survey of Hell and Nešetřil, and [34] for a connection between the Dichotomy Conjecture and probabilistically checkable proofs.

Since in practice many constraint satisfaction problems are over-constrained, and hence have no solution, or are under-constrained, and hence have many solutions, *soft* constraint satisfaction problems have been studied [20]. In an instance of the soft CSP, every constraint is associated with a cost function (rather than a relation as in the CSP) which represents preferences among different partial assignments, and the goal is to find the best assignment. Several very general soft CSP frameworks have been proposed in the literature [42, 8]. In this paper we focus on one of the very general frameworks, the *valued* constraint satisfaction problem (VCSP) [42]. Throughout the paper, we use the term *constrain language* (or just *language*) for a set of cost functions over a finite domain. If all cost functions from a given language Γ are $\{0, \infty\}$-valued (i.e. relations), we call Γ a *crisp* language. (If necessary, to stress the fact that Γ is a language, but not a crisp language, we call Γ a *general-valued* language.)

Similarly to the CSP, an important line of research on the VCSP is to identify tractable cases which are recognisable in polynomial time. Is is well known that structural reasons for tractability generalise to the VCSP [20]. In the case of language restrictions, only a few conditions are known to guarantee tractability of a given language [14, 13].

Related work The problem of characterising the complexity of different languages has received significant attention in the literature. For some classes researchers have established a Schaefer-like dichotomy theorem of the following form: if language Γ admits certain *polymorphisms* or *multimorphisms* then it is tractable, otherwise it is NP-hard. Some of these classes are as follows: Boolean languages, i.e. languages with a 2-element domain (Cohen *et al.* [14]); crisp languages including all unary relations (Bulatov [11] and recently Barto [4]); crisp languages with a 3-element domain (Bulatov [9]); $\{0, 1\}$-valued languages including all unary cost functions (Deineko *et al.* [22]); crisp languages including additionally all finite-valued unary cost functions (Takhanov [43]); crisp languages including additionally a certain subset of finite-valued unary cost functions (Takhanov [44]).

Our proof exploits the results of Takhanov [43], who showed the existence of a majority polymorphism as a necessary condition for tractability of crisp languages including additionally all finite-valued unary cost functions. Other related work includes the work of Creignou *et al.* who studied various generalisations of the CSP to optimisation problems over Boolean domains [17], see also [18, 32]. Raghavendra [39] and Raghavendra and Steurer [40] have shown how to optimally approximate any finite-valued VCSP.

Contributions This paper focuses on valued languages containing all finite-valued unary cost functions; we call such languages *conservative*. Our main result is a dichotomy theorem for all conservative languages: if a conservative language Γ admits a complementary combination of *STP (symmetric tournament pair) and MJN (majority-majority-minority) multimorphisms*, then it is tractable, otherwise Γ is NP-hard. This is the first complete complexity classification of general-valued languages over non-Boolean domains, generalising previously obtained results in [14, 22, 43] as follows:

- Cohen *et al.* proved a dichotomy for arbitrary Boolean languages ($|D| = 2$). We generalise it to arbitrary domains ($|D| \geq 2$), although only for conservative languages.

- Deineko *et al.* [22] and Takhanov [43] proved a dichotomy for the following languages, respectively:

  - $\{0, 1\}$-valued languages containing additionally all unary cost functions;
  - $\{0, \infty\}$-valued languages containig additionaly all unary cost functions.

  In both of these case the languages are conservative, so these classifications are special cases of our result. Note, however, that Deineko *et al.* additionally give a dichotomy with respect to approximability (PO vs. APX-hard), even when the number of occurances of variables in instances is bounded; this part of [22] does not follow from our classification.

Moreover, our results provide a new powerful tool and do not rely on a computer-assisted search as in [22]. Building on techniques from this paper, Jonsson *et al.* [31] have recently shown that the same approach can be also used for certain non-conservative languages.

Since the complexity of Boolean conservative languages is known, we start, similarly to Bulatov and Takhanov [11, 43], by exploring the interactions between different 2-element subdomains. Given a conservative language Γ, we will investigate properties of a certain graph G_Γ associated with the language and cost functions expressible over Γ. We link the complexity of Γ to certain properties of the graph G_Γ.

First, we show that if G_Γ does not satisfy certain properties, then Γ is intractable. Second, using G_Γ, we construct a (partial) *STP multimorphism* and a (partial) *MJN multimorphism*. Finally, we show that any language which admits

a complementary combination of *STP and MJN multimorphisms* is tractable, thus generalising a tractable class of Cohen *et al.* [13], which in turn is a generalisation of the submodular minimisation problem. Thus we obtain a dichotomy theorem. The general-valued case is much more involved than the finite-valued case, and requires different techniques compared to previous results.

Given a finite language Γ, the graph G_Γ is finite as well, but depends on the expressive power of Γ (see Section 2 for precise definitions), which is infinite. In order to test whether Γ is tractable, we do not need to construct the graph G_Γ as it follows from our result that we just need to test for the existence of a complementary combination of two multimorphisms, which can be established in polynomial time.

Our results are formulated using the terminology of valued constraint satisfaction problems, but they apply to various other optimisation frameworks that are equivalent to valued constraint satisfaction problems such as Gibbs energy minimisation, Markov Random Fields, Min-Sum problems, and other models [35, 46].

Organisation of the paper The rest of the paper is organised as follows. In Section 2, we define valued constraint satisfaction problems (VCSPs), conservative languages, multimorphisms and other necessary definitions needed throughout the paper. We state our results in Section 3. Finally, Section 4 gives an overview of the omitted proofs, which will be given in full detail in the full version of this paper.

2 Background and notation

We denote by $\mathbb{Q}_+$ the set of all non-negative rational numbers. We define $\overline{\mathbb{Q}}_+ = \mathbb{Q}_+ \cup \{\infty\}$ with the standard addition operation extended so that for all $a \in \mathbb{Q}_+$, $a + \infty = \infty$. Members of $\overline{\mathbb{Q}}_+$ are called *costs*. Throughout the paper, we denote by D any fixed finite set, called a *domain*. Elements of D are called *domain values* or *labels*.

A function f from D^m to $\overline{\mathbb{Q}}_+$ will be called a *cost function* on D of *arity* m. If the range of f lies entirely within $\mathbb{Q}_+$, then f is called a *finite-valued* cost function. If the range of f is $\{0, \infty\}$, then f is called a *crisp* cost function. If the range of a cost function f includes non-zero finite costs and infinity, we emphasise this fact by calling f a *general-valued* cost function. Let $f : D^m \to \overline{\mathbb{Q}}_+$ be an m-ary cost function f. We denote $\mathtt{dom} f = \{\boldsymbol{x} \in D^m \mid f(\boldsymbol{x}) < \infty\}$ to be the effective domain of f. The argument of f is called an *assignment* or a *labelling*. Functions f of arity $m = 2$ are called *binary*.

A *language* is a set of cost functions with the same domain D. Language Γ is called finite-valued (crisp, general-valued respectively) if all cost functions in Γ are finite-valued (crisp, general-valued respectively). A language Γ is *Boolean* if $|D| = 2$.

DEFINITION 2.1. *An instance $\mathcal{I}$ of the* valued constraint satisfaction problem *(VCSP) is a function $D^V \to \overline{\mathbb{Q}}_+$ given by*

$$\mathit{Cost}_{\mathcal{I}}(\boldsymbol{x}) = \sum_{t \in T} f_t\left(x_{i(t,1)}, \ldots, x_{i(t,m_t)}\right)$$

It is specified by a finite set of nodes V, finite set of terms (also known as constraints) T, cost functions $f_t : D^{m_t} \to \overline{\mathbb{Q}}_+$ or arity m_t and indices $i(t,k) \in V$ for $t \in T$, $k = 1, \ldots, m_t$. A solution *to $\mathcal{I}$ is an assignment $\boldsymbol{x} \in D^V$ with the minimum cost.*

We denote by $\mathsf{VCSP}(\Gamma)$ the class of all VCSP instances whose terms f_t belong to Γ. A finite language Γ is called *tractable* if $\mathsf{VCSP}(\Gamma)$ can be solved in polynomial time, and *intractable* if $\mathsf{VCSP}(\Gamma)$ is NP-hard. An infinite language Γ is tractable if every finite subset $\Gamma' \subseteq \Gamma$ is tractable, and intractable if there is a finite subset $\Gamma' \subseteq \Gamma$ that is intractable.

The idea behind conservative languages is to contain all possible unary cost functions: Bulatov has called a crisp language Γ conservative if Γ contains all unary relations [11]. We are interested in valued languages containing all possible unary cost functions and hence define conservative languages as follows:

DEFINITION 2.2. *Language Γ is called* conservative *if Γ contains all $\{0,1\}$-valued unary cost functions $u : D \to \{0,1\}$.*

Such languages have been studied by Deineko *et al.* [22] and Takhanov [43]. Note, we could have defined Γ to be conservative if it contains all possible general-valued unary cost functions $u : D \to \overline{\mathbb{Q}}_+$. However, the weaker definition 2.2 will be sufficient for our purposes: as shown in the full version of this paper, adding all possible unary cost functions $u : D \to \overline{\mathbb{Q}}_+$ to a conservative language Γ does not change the complexity of Γ.

DEFINITION 2.3. *A mapping $F : D^k \to D$, $k \geq 1$ is called a* polymorphism *of a cost function $f : D^m \to \overline{\mathbb{Q}}_+$ if*

$$F(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k) \in \mathtt{dom} f \qquad \forall \boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \in \mathtt{dom} f$$

where F is applied component-wise. F is a polymorphism of a language Γ if F is a polymorphism of every cost function in Γ.

Multimorphisms [14] are generalisations of polymorphisms. To make the paper easier to read, we only define binary and ternary multimorphisms as we will not need multimorphisms of higher arities.

DEFINITION 2.4. *Let $\langle \sqcap, \sqcup \rangle$ be a pair of operations, where $\sqcap, \sqcup : D \times D \to D$, and let $\langle F_1, F_2, F_3 \rangle$ be a triple of operations, where $F_i : D \times D \times D \to D$, $1 \leq i \leq 3$.*

- *Pair* $\langle \sqcap, \sqcup \rangle$ *is called a (binary)* multimorphism *of cost function* $f : D^m \to \overline{\mathbb{Q}}_+$ *if*

$$(2.1)\quad f(\boldsymbol{x} \sqcap \boldsymbol{y}) + f(\boldsymbol{x} \sqcup \boldsymbol{y}) \leq f(\boldsymbol{x}) + f(\boldsymbol{y}) \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathtt{dom}\, f$$

where operations $\sqcap, \sqcup$ *are applied component-wise.* $\langle \sqcap, \sqcup \rangle$ *is a multimorphism of language* Γ *if* $\langle \sqcap, \sqcup \rangle$ *is a multimorphism of every* f *from* Γ.

- *Triple* $\langle F_1, F_2, F_3 \rangle$ *is called a (ternary)* multimorphism *of cost function* $f : D^m \to \overline{\mathbb{Q}}_+$ *if*

$$(2.2)\quad f(F_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})) + f(F_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})) + f(F_3(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}))$$
$$\leq\ f(\boldsymbol{x}) + f(\boldsymbol{y}) + f(\boldsymbol{z}) \qquad \forall \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathtt{dom}\, f$$

where operations F_1, F_2, F_3 *are applied component-wise.* $\langle F_1, F_2, F_3 \rangle$ *is a multimorphism of language* Γ *if* $\langle F_1, F_2, F_3 \rangle$ *is a multimorphism of every* f *from* Γ.

- *Operation* $F : D^k \to D$ *is called* conservative *if* $F(x_1, \ldots, k_k) \in \{x_1, \ldots, x_k\}$ *for all* $x_1, \ldots, x_k \in D$.

- *Pair* $\langle \sqcap, \sqcup \rangle$ *is called* conservative *if* $\{\{a \sqcap b, a \sqcup b\}\} = \{\{a, b\}\}$ *for all* $a, b \in D$*, where* $\{\{\ldots\}\}$ *denotes a* multiset*, i.e. in the case of repeatitions elements' multiplicities are taken into account. Similarly, triple* $\langle F_1, F_2, F_3 \rangle$ *is called* conservative *if* $\{\{F_1(a,b,c), F_2(a,b,c), F_3(a,b,c)\}\} = \{\{a, b, c\}\}$ *for all* $a, b, c \in D$*. In other words, applying* $\langle F_1, F_2, F_3 \rangle$ *to* (a, b, c) *should give a permutation of* (a, b, c).

- *Pair* $\langle \sqcap, \sqcup \rangle$ *is called a* symmetric tournament pair (STP) *if it is conservative and both operations* $\sqcap, \sqcup$ *are commutative, i.e.* $a \sqcap b = b \sqcap a$ *and* $a \sqcup b = b \sqcup a$ *for all* $a, b \in D$.

- *An operation* $\mathtt{Mj} : D^3 \to D$ *is called a* majority operation *if for every tuple* $(a, b, c) \in D^3$ *with* $|\{a, b, c\}| = 2$ *operation* $\mathtt{Mj}$ *returns the unique majority element among* a, b, c *(that occurs twice). An operation* $\mathtt{Mn} : D^3 \to D$ *is called a* minority operation *if for every tuple* $(a, b, c) \in D^3$ *with* $|\{a, b, c\}| = 2$ *operation* $\mathtt{Mn}$ *returns the unique minority element among* a, b, c *(that occurs once).*

- *Triple* $\langle \mathtt{Mj}_1, \mathtt{Mj}_2, \mathtt{Mn}_3 \rangle$ *is called an* MJN *if it is conservative,* $\mathtt{Mj}_1, \mathtt{Mj}_2$ *are (possibly different) majority operations, and* $\mathtt{Mn}_3$ *is a minority operation.*

We say that $\langle \sqcap, \sqcup \rangle$ is a multimorphism of language Γ, or Γ admits $\langle \sqcap, \sqcup \rangle$, if all cost functions $f \in \Gamma$ satisfy (2.1). Using a polynomial-time algorithm for minimising submodular functions, Cohen *et al.* have obtained the following result:

THEOREM 2.1. ([13]) *If a language* Γ *admits an STP, then* Γ *is tractable.*

The existence of an MJN multimorphism also leads to tractability. This was shown for a specific choice of an MJN by Cohen *et al.* [14].

Our tractability result, presented in the next section, will include both above-mentioned tractable classes as special cases.

Expressibility Finally, we define the important notion of expressibility, which captures the idea of introducing auxiliary variables in a VCSP instance and the possibility of minimising over these auxiliary variables. (For crisp languages, this is equivalent to *implementation* [18].)

DEFINITION 2.5. *A cost function* $f : D^m \to \overline{\mathbb{Q}}_+$ *is* expressible *over a language* Γ *if there exists an instance* $\mathcal{I} \in \mathit{VCSP}(\Gamma)$ *with the set of nodes* $V = \{1, \ldots, m, m+1, \ldots, m+k\}$ *where* $k \geq 0$ *such that*

$$f(\boldsymbol{x}) = \min_{\boldsymbol{y} \in D^k} \mathit{Cost}_{\mathcal{I}}(\boldsymbol{x}, \boldsymbol{y}) \qquad \forall \boldsymbol{x} \in D^m$$

We define Γ^* *to be the* expressive power *of* Γ*; that is, the set of all cost functions* f *such that* f *is expressible over* Γ.

The importance of expressibility is in the following result:

THEOREM 2.2. ([14]) *For any language* Γ*,* Γ *is tractable iff* Γ^* *is tractable.*

It is easy to observe and well known that any polymorphism (multimorphism) of Γ is also a polymorphism (multimorphism) of Γ^* [14].

3 Our results

In this section, we relate the complexity of a conservative language Γ to properties of a certain graph G_Γ associated with Γ.

Given a conservative language Γ, let $G_\Gamma = (P, E)$ be the graph with the set of nodes $P = \{(a, b) | a, b \in D, a \neq b\}$ and the set of edges E defined as follows: there is an edge between $(a, b) \in P$ and $(a', b') \in P$ iff there exists binary cost function $f \in \Gamma^*$ such that

$$(3.3)\qquad f(a, a') + f(b, b') > f(a, b') + f(b, a')\ ,$$
$$(a, b'), (b, a') \in \mathtt{dom}\, f$$

Note that G_Γ may have self-loops. For node $p \in P$ we denote the self-loop by $\{p, p\}$. We say that edge $\{(a, b), (a', b')\} \in E$ is *soft* if there exists binary $f \in \Gamma^*$ satisfying (3.3) such that at least one of the assignments (a, a'), (b, b') is in $\mathtt{dom}\, f$. Edges in E that are not soft are called *hard*. For node $p = (a, b) \in P$ we denote $\bar{p} = (b, a) \in P$. Note, a somewhat similar graph (but not the same) was used

by Takhanov [43] for languages Γ containing crisp functions and finite unary cost functions.[1]

We denote $M \subseteq P$ to be the set of vertices $(a, b) \in P$ without self-loops, and $\overline{M} = P - M$ to be the complement of M. It follows from the definition that set M is *symmetric*, i.e. $(a, b) \in M$ iff $(b, a) \in M$. We will write $\{a, b\} \in M$ to indicate that $(a, b) \in M$; this is consistent due to the symmetry of M. Similarly, we will write $\{a, b\} \in \overline{M}$ if $(a, b) \in \overline{M}$, and $\{a, b\} \in P$ if $(a, b) \in P$, i.e. $a, b \in D$ and $a \neq b$.

DEFINITION 3.1. *Let $\langle \sqcap, \sqcup \rangle$ and $\langle \mathtt{Mj}_1, \mathtt{Mj}_2, \mathtt{Mn}_3 \rangle$ be binary and ternary operations respectively.*

- *Pair $\langle \sqcap, \sqcup \rangle$ is an* STP on M *if $\langle \sqcap, \sqcup \rangle$ is conservative on $P \cup \{\{a\} \mid a \in D\}$ and commutative on M.*
- *Triple $\langle \mathtt{Mj}_1, \mathtt{Mj}_2, \mathtt{Mn}_3 \rangle$ is an* MJN on $\overline{M}$ *if it is conservative and for each triple $(a, b, c) \in D^3$ with $\{a, b, c\} = \{x, y\} \in \overline{M}$ operations $\mathtt{Mj}_1(a, b, c)$, $\mathtt{Mj}_2(a, b, c)$ return the unique majority element among a, b, c (that occurs twice) and $\mathtt{Mn}_3(a, b, c)$ returns the remaining minority element.*

Our main results are given by the following three theorems.

THEOREM 3.1. *Let Γ be a conservative language.*

(a) If G_Γ has a soft self-loop then Γ is NP-hard.

(b) If G_Γ does not have soft self-loops then Γ admits a pair $\langle \sqcup, \sqcap \rangle$ which is an STP on M and satisfies additionally $a \sqcap b = a, a \sqcup b = b$ for $\{a, b\} \in \overline{M}$.

THEOREM 3.2. *Let Γ be a conservative language. If Γ does not admit an MJN on $\overline{M}$ then it is NP-hard.*

THEOREM 3.3. *Suppose language Γ admits an STP on M and an MJN on $\overline{M}$, for some choice of symmetric $M \subseteq P$. Then Γ is tractable.*

Theorems 3.1-3.3 give the dichotomy result for conservative languages:

COROLLARY 3.1. *If a conservative language Γ admits an STP on M and an MJN on $\overline{M}$ for some symmetric $M \subseteq P$ then Γ is tractable. Otherwise Γ is NP-hard.*

Proof. The first part follows from Theorem 3.3; let us show the second part. Suppose that the precondition of the corollary does not hold, then one of the following cases must be true (we assume below that M is the set of nodes without self-loops in G_Γ):

- G_Γ has a soft self-loop. Then Γ is NP-hard by Theorem 3.1(a).
- G_Γ does not have soft self-loops and Γ does not admit an STP on M. This is a contradiction by Theorem 3.1(b).
- G_Γ does not have soft self-loops and Γ does not admit an MJN on $\overline{M}$. Then Γ is NP-hard by Theorem 3.2.

In the finite-valued case, we get a simpler tractability criterion:

COROLLARY 3.2. *If a conservative finite-valued language Γ admits an STP then Γ is tractable. Otherwise Γ is NP-hard.* [2]

Proof. Consider the graph G_Γ associated with Γ. If G_Γ contains a soft self-loop, then, by Theorem 3.1(a), Γ is NP-hard. Suppose that G_Γ does not contain soft self-loops. As Γ is finite-valued, G_Γ cannot have hard self-loops. Therefore, $\overline{M}$ is empty and $M = P$. By Theorem 3.1(b), Γ admits an STP. The tractability then follows from Theorem 3.3.

4 Overview of the proofs

In this section we sketch proofs of our results. Complete proofs of Theorems 3.1-3.3 will be given in the full version of this paper.

First, we show that we can strengthen the definition of conservative languages without loss of generality. Let $\bar{\Gamma}$ be the language obtained from Γ by adding all possible general-valued unary cost functions $u : D \to \overline{\mathbb{Q}}_+$. Note, $\bar{\Gamma}$ may be different from Γ since Γ is only guaranteed to have all possible $\{0, 1\}$-valued unary cost functions.

PROPOSITION 4.1. *(a) Graphs G_Γ and $G_{\bar{\Gamma}}$ are the same: if $\{p, q\}$ is a soft (hard) edge in G_Γ then it is also a soft (hard) edge in $G_{\bar{\Gamma}}$, and vice versa. (b) If $\bar{\Gamma}$ is NP-hard then so is Γ.*

This shows that it suffices to prove Theorems 3.1 and 3.2 for language $\bar{\Gamma}$. Indeed, consider Theorem 3.1 for a conservative language G. If G_Γ has a soft self-loop then by Proposition 4.1(a) so does $G_{\bar{\Gamma}}$. Theorem 3.1(a) for language $\bar{\Gamma}$ would imply that $\bar{\Gamma}$ is NP-hard, and therefore Γ is also NP-hard by Proposition 4.1(b). If G_Γ does not have soft self-loops then neither does $G_{\bar{\Gamma}}$. Theorem 3.1(b) for language $\bar{\Gamma}$ would imply that $\bar{\Gamma}$ admits the appropriate multimorphism

[1] Roughly speaking, the graph structure in [43] was defined via a "min" polymorphism rather than a $\langle \min, \max \rangle$ multimorphism, so the property $\{p, q\} \in E \Rightarrow \{\bar{p}, \bar{q}\} \in E$ (that we prove for our graph in the next section) might not hold in Takhanov's case. Also, in [43] edges were not classified as being soft or hard.

[2] It can be shown that if a finite-valued language admits an STP multimorphism, it also admits a submodularity multimorphism. This result is implicitly contained in [13]. Namely, after reducing the domains as in [13, Theorem 8.3], the STP might contain cycles. [13, Lemma 7.15] tells us that on cycles we have, in the finite-valued case, only unary cost functions. It follows that the cost functions admitting the STP must be submodular w.r.t. some total order [16].

This simplifies the tractability critertion in the finite-valued case (though we do not exploit this fact anywhere in the paper).

$\langle \sqcup, \sqcap \rangle$. Since $\Gamma \subseteq \bar{\Gamma}$, $\langle \sqcup, \sqcap \rangle$ is also a multimorphism of Γ. A similar argumentation holds for Theorem 3.2.

We can therefore make the following assumption without loss of generality when proving Theorems 3.1 and 3.2:

Assumption 1. Γ contains all general-valued unary cost functions $u : D \to \overline{\mathbb{Q}}_+$.

4.1 Sketch of the proof of Theorem 3.1 Using unary cost functions, Theorem 3.1(a) can be proved via a reduction from the Max-SAT problem with XOR clauses and the Independent Set problem. We focus on Theorem 3.1(b). Graph G_Γ and its properties play a crucial role in the proofs.

In the lemma below, a *path* of length k is a sequence of edges $\{p_0, p_1\}, \{p_1, p_2\}, \ldots, \{p_{k-1}, p_k\}$, where $\{p_{i-1}, p_i\} \in E$. Note that we allow edge repetitions. A path is *even* iff its length is even. A path is a *cycle* if $p_0 = p_k$. If $X \subseteq P$ then $(X, E[X])$ denotes the subgraph of (P, E) induced by X.

LEMMA 4.1. *Graph $G_\Gamma = (P, E)$ satisfies the following properties:*

(a) $\{p, q\} \in E$ implies $\{\bar{p}, \bar{q}\} \in E$ and vice versa. The two edges are either both soft or both hard.

(b) Suppose that $\{p, q\} \in E$ and $\{q, r\} \in E$, then $\{p, \bar{r}\} \in E$. If at least one of the first two edges is soft then the third edge is also soft.

(c) For each $p \in P$, nodes p and $\bar{p}$ are either both in M or both in $\overline{M}$.

(d) There are no edges from M to $\overline{M}$.

(e) Graph $(M, E[M])$ does not have odd cycles.

(f) If node p is not isolated (i.e. it has at least one incident edge $\{p, q\} \in E$) then $\{p, \bar{p}\} \in E$.

(g) Nodes $p \in \overline{M}$ do not have incident soft edges.

We now construct a pair of operations $\langle \sqcap, \sqcup \rangle$ for Γ that behaves as an STP on M and as a multi-projection (returning its two arguments in the same order) on $\overline{M}$.

LEMMA 4.2. *There exists an assignment $\sigma : M \to \{-1, +1\}$ such that (i) $\sigma(p) = -\sigma(q)$ for all edges $\{p, q\} \in E$, and (ii) $\sigma(p) = -\sigma(\bar{p})$ for all $p \in M$.*

Proof. By Lemma 4.1(e) graph $(M, E[M])$ does not have odd cycles. Therefore, by Harary's Theorem, graph $(M, E[M])$ is bipartite and there exists an assignment $\sigma : M \to \{-1, +1\}$ that satisfies property (i). Let us modify this assignment as follows: for each isolated node $p \in M$ (i.e. node without incident edges) set $\sigma(p)$, $\sigma(\bar{p})$ so that $\sigma(p) = -\sigma(\bar{p}) \in \{-1, +1\}$. (Note, if p is isolated then by Lemma 4.1(a) so is $\bar{p}$). Clearly, property (i) still holds. Property (ii) holds for each node $p \in M$ as well: if p is isolated then (ii) holds by construction, otherwise by Lemma 4.1(f) there exists edge $\{p, \bar{p}\} \in E$, and so (ii) follows from property (i).

Given assignment σ constructed in Lemma 4.2, we now define operations $\sqcap, \sqcup : D^2 \to D$ as follows:

- $a \sqcap a = a \sqcup a = a$ for $a \in D$.
- If $(a, b) \in M$ then $a \sqcap b$ and $a \sqcup b$ are the unique elements of D satisfying $\{a \sqcap b, a \sqcup b\} = \{a, b\}$ and $\sigma(a \sqcap b, a \sqcup b) = +1$.
- If $(a, b) \in \overline{M}$ then $a \sqcap b = a$ and $a \sqcup b = b$.

LEMMA 4.3. *For any binary cost function $f \in \Gamma^*$ and any $\boldsymbol{x}, \boldsymbol{y} \in \mathtt{dom}\, f$ there holds*

$$f(\boldsymbol{x} \sqcap \boldsymbol{y}) + f(\boldsymbol{x} \sqcup \boldsymbol{y}) \leq f(\boldsymbol{x}) + f(\boldsymbol{y}) \tag{4.4}$$

In order to proceed, we introduce the following notation. Given a cost function f of arity m, we denote by V the set of variables corresponding to the arguments of f, with $|V| = m$. For two assignments $\boldsymbol{x}, \boldsymbol{y} \in D^m$ we denote $\Delta(\boldsymbol{x}, \boldsymbol{y}) = \{i \in V \mid x_i \neq y_i\}$ to be the set of variables on which $\boldsymbol{x}$ and $\boldsymbol{y}$ differ.

Having Lemma 4.3 as a base case and using induction on $|V|$, we can prove the following

LEMMA 4.4. *Condition* (4.4) *holds for any cost function $f \in \Gamma^*$ and assignments $\boldsymbol{x}, \boldsymbol{y} \in \mathtt{dom}\, f$ with $|\Delta(\boldsymbol{x}, \boldsymbol{y})| \leq 2$.*

Having Lemma 4.4 as a base case and using induction on $|\Delta(\boldsymbol{x}, \boldsymbol{y})|$, we can prove the following

LEMMA 4.5. *Condition* (4.4) *holds for any cost function $f \in \Gamma^*$ and any $\boldsymbol{x}, \boldsymbol{y} \in \mathtt{dom}\, f$.*

The previous lemma finishes the proof of Theorem 3.1(b).

We remark that the idea behind the previous three lemmas comes from the proof that a k-ary finite-valued cost function f is submodular iff every binary projection of f is submodular [45]. However, this is known to not hold for general-valued cost functions (i.e. cost functions taking on both finite and infinite costs) [12], and hence our proofs are more elaborate.

4.2 Sketch of the proof of Theorem 3.2 First, we introduce the following simplifying assumptions:

Assumption 2. Γ admits a majority polymorphism.

Assumption 3. G_Γ does not have soft self-loops.

(If these assumptions do not hold then it can be shown that Γ is NP-hard, using results of Takhanov [43] and Theorem 3.1(a)). Our goal is then to construct an MJN multimorphism on $\overline{M}$ under assumptions 1-3.

In order to do this, we introduce a function μ which maps every set $\{a,b,c\} \subseteq D$ with $|\{a,b,c\}| = 3$ to a subset of $\{a,b,c\}$. This subset is defined as follows: $c \in \mu(\{a,b,c\})$ iff there exists binary function $f \in \Gamma^*$ and a pair $(a',b') \in \overline{M}$ such that

$$\mathtt{dom}\, f = \{(a,a'),(b,a'),(c,b')\}$$

LEMMA 4.6. *Set $\mu(\{a,b,c\})$ contains at most one label. Furthermore, if $\mu(\{a,b,c\}) = \{c\}$ then $(a,c) \in \overline{M}$ and $(b,c) \in \overline{M}$.*

For convenience, we define $\mu(\{a,b,c\}) = \varnothing$ if $|\{a,b,c\}| \leq 2$. We are now ready to construct operation $\mathtt{MJN} = \langle \mathtt{Mj}_1, \mathtt{Mj}_2, \mathtt{Mn}_3 \rangle$. Given a tuple $(a,b,c) \in D^3$, we define

$$\mathtt{MJN}(a,b,c) \;=$$

$$\begin{cases} (x,x,y) & \text{if } \{\{a,b,c\}\} = \{\{x,x,y\}\}, \{x,y\} \in \overline{M} \quad (4.5a) \\ (b \sqcap c, b \sqcup c, a) & \text{if } \mu(\{a,b,c\}) = \{a\} \quad (4.5b) \\ (a \sqcap c, a \sqcup c, b) & \text{if } \mu(\{a,b,c\}) = \{b\} \quad (4.5c) \\ (a \sqcap b, a \sqcup b, c) & \text{in any other case} \quad (4.5d) \end{cases}$$

where $\{\{\ldots\}\}$ denotes a *multiset*, i.e. elements' multiplicities are taken into account.

To prove Theorem 3.2, we need to prove for every $f \in \Gamma^*$ and for every $\boldsymbol{x},\boldsymbol{y},\boldsymbol{z} \in \mathtt{dom}\, f$,

$$f(\mathtt{Mj}_1(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})) + f(\mathtt{Mj}_2(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})) + f(\mathtt{Mn}_3(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})) \leq f(\boldsymbol{x}) + f(\boldsymbol{y}) + f(\boldsymbol{z}) \quad (4.6)$$

We say that an instance $(f,\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$ is *valid* if $f \in \Gamma^*$ and $\boldsymbol{x},\boldsymbol{y},\boldsymbol{z} \in \mathtt{dom}\, f$. It is *satisfiable* if (4.6) holds, and *unsatisfiable* otherwise. For a triple $\boldsymbol{x},\boldsymbol{y},\boldsymbol{z} \in D^V$ denote $\delta(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) = \sum_{i \in V} |\{x_i,y_i,z_i\}|$, $\Delta(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) = \{i \in V \mid x_i \neq y_i\}$ and $\Delta^M(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) = \{i \in \Delta(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) \mid \{x_i,y_i,z_i\} = \{a,b\} \in M\}$.

Suppose that an unsatisfiable instance exists. Let $(f,\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$ be a lowest unsatisfiable instance with respect to the partial order $\preceq$ defined as the lexicographical order with components

$$\begin{aligned} (\;& \delta(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}), \\ & |\Delta(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})|, \\ & |\Delta^M(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})|, \\ & |\{i \in V \mid \mu(\{x_i,y_i,z_i\}) = \{x_i\}\}| \;) \end{aligned} \quad (4.7)$$

(the first component is more significant). We denote $\delta_{\min} = \delta(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$. Thus, we can assume that all valid instances $(f,\boldsymbol{x}',\boldsymbol{y}',\boldsymbol{z}')$ with $(\boldsymbol{x}',\boldsymbol{y}',\boldsymbol{z}') \prec (\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$ (and in particular with $\delta(\boldsymbol{x}',\boldsymbol{y}',\boldsymbol{z}') < \delta_{\min}$) are satisfiable, while the instance $(f,\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$ is unsatisfiable.

By analysing this minimal counter-example, we show that none of the cases (4.5a)-(4.5c) are possible. The arguments that we use are more complicated versions of the induction arguments used for proving theorem 3.1; additionally, for one of the cases we exploit the fact that f admits a majority polymorphism.

We obtain that $\mathtt{MJN}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$ is defined via (4.5d) for all nodes. But then inequality (4.6) follows from the fact that $\langle \sqcap, \sqcup \rangle$ is a multimorphism of f, contradicting to the choice of $(f,\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$. This proves Theorem 3.2.

4.3 Sketch of the proof of Theorem 3.3 In this section we present an algorithm for minimising instances from $\mathsf{VCSP}(\Gamma)$. The idea for the algorithm and some of the proof techniques have been influenced by the techniques used by Takhanov [43] for proving the absence of *arithmetical deadlocks* in certain instances. However, the algorithm itself is very different from Takhanov's approach. (The latter does not rely on submodular minimization algorithms; instead, it performs a reduction to an optimization problem in a perfect graph).

Let $f : \mathcal{D} \to \overline{\mathbb{Q}}_+$ be the function to be minimised, V be the set of its variables (which we will also call nodes), and D_i be the domain of variable $i \in V$ with $\mathcal{D} = \times_{i \in V} D_i$. In the beginning all domains are the same ($D_i = D$), but as the algorithm progresses we will allow D_i to become different for different $i \in V$. As a consequence, operations $\sqcap, \sqcup$ may act differently on different components of vectors $\boldsymbol{x},\boldsymbol{y} \in \mathcal{D}$ We denote $\sqcap_i, \sqcup_i : D_i \times D_i \to D_i$ to be the i-th operations of $\langle \sqcap, \sqcup \rangle$. Similarly, we denote by $\mathtt{Mj}_{1i}, \mathtt{Mj}_{2i}, \mathtt{Mn}_{3i} : D_i \times D_i \times D_i \to D_i$ to be the i-th operations of $\langle \mathtt{Mj}_1, \mathtt{Mj}_2, \mathtt{Mn}_3 \rangle$.

We denote by P the collection of sets $P = (P_i)_{i \in V}$ where $P_i = \{\{a,b\} \mid a,b \in D_i, a \neq b\}$. We denote by M a collection of subsets $M = (M_i)_{i \in V}$, $M_i \subseteq P_i$, and $\overline{M} = (\overline{M}_i)_{i \in V}$, $\overline{M}_i = P_i - M_i$. We now extend Definition 3.1 as follows.

DEFINITION 4.1. *Let $\langle \sqcap, \sqcup \rangle$ and $\langle \mathtt{Mj}_1, \mathtt{Mj}_2, \mathtt{Mn}_3 \rangle$ be collections of binary and ternary operations respectively.*

- *Pair $\langle \sqcap, \sqcup \rangle$ is an* STP *on M if for all $i \in V$ pair $\langle \sqcap_i, \sqcup_i \rangle$ is an STP on M_i.*
- *Triple $\langle \mathtt{Mj}_1, \mathtt{Mj}_2, \mathtt{Mn}_3 \rangle$ is an* MJN *on $\overline{M}$ if for all $i \in V$ triple $\langle \mathtt{Mj}_{1i}, \mathtt{Mj}_{2i}, \mathtt{Mn}_{3i} \rangle$ is an MJN on $\overline{M}_i$.*

We will assume without loss of generality that $\langle \sqcap_i, \sqcup_i \rangle$ is non-commutative on any $\{a,b\} \in \overline{M}_i$ (if not, we can simply add such $\{a,b\}$ to M_i).

We are now ready to present the algorithm; it will consist of three stages.

Stage 1: Decomposition into binary relations It can be shown that the instance admits a majority polymorphism, and hence every cost function f can be decomposed [1] into unary relations $\rho_i \subseteq D_i$, $i \in D_i$ and binary relations $\rho_{ij} \subseteq D_i \times D_j$, $i, j \in V$, $i \neq j$ such that

$$\boldsymbol{x} \in \mathtt{dom}\, f \quad \Leftrightarrow$$

$$[x_i \in \rho_i \ \forall i \in V] \quad \text{and} \quad [(x_i, x_j) \in \rho_{ij} \ \forall i, j \in V, i \neq j]$$

We will always assume that binary relations are symmetric, i.e. $(x, y) \in \rho_{ij} \Leftrightarrow (y, x) \in \rho_{ji}$. We use the following notation for relations:

- If $\rho_{ij} \in D_i \times D_j$, $X \subseteq D_i$ and $Y \subseteq D_j$ then

$$\begin{aligned} \rho_{ij}(X, \cdot) &= \{y \mid \exists x \in X \text{ s.t. } (x, y) \in \rho_{ij}\} \\ \rho_{ij}(\cdot, Y) &= \{x \mid \exists y \in Y \text{ s.t. } (x, y) \in \rho_{ij}\} \end{aligned}$$

  If $X = \{x\}$ and $Y = \{y\}$ then these two sets will be denoted as $\rho_{ij}(x, \cdot)$ and $\rho_{ij}(\cdot, y)$ respectively.

In the first stage we establish *strong 3-consistency* using the standard constraint-processing techniques [15] so that the resulting relations satisfy *arc-consistency*:

$$\{x \mid (\exists y)(x, y) \in \rho_{ij}\} = \rho_i \qquad \forall \text{ distinct } i, j \in V$$

and *path-consistency*:

$$\rho_{ik}(x, \cdot) \cap \rho_{jk}(y, \cdot) \neq \varnothing \quad \forall \text{ distinct } i, j, k \in V, (x, y) \in \rho_{ij}$$

It is known that in the presence of a majority polymorphism strong 3-consistency is equivalent to global consistency [28]; that is $\mathtt{dom}\, f$ is empty iff all ρ_i and ρ_{ij} are empty. Using this fact, it is not difficult to show that the strong 3-consistency relations ρ_i, ρ_{ij} are uniquely determined by f via

$$\rho_i = \{x_i \mid \boldsymbol{x} \in \mathtt{dom}\, f\} \qquad \rho_{ij} = \{(x_i, x_j) \mid \boldsymbol{x} \in \mathtt{dom}\, f\}$$

The second equation implies that any polymorphism of f is also a polymorphism of ρ_{ij}.

From now on we will assume that $D_i = \rho_i$ for all $i \in V$. This can be achieved by reducing sets D_i if necessary. We will also assume that all sets D_i are non-empty.

Stage 2: Modifying M and $\langle \sqcap, \sqcup \rangle$ The second stage of the algorithm works by iteratively growing sets M_i and simultaneously modifying operations $\langle \sqcap_i, \sqcup_i \rangle$ so that (i) $\langle \sqcap_i, \sqcup_i \rangle$ is still a conservative pair which is commutative on M_i and non-commutative on $\overline{M}_i$, and (ii) $\langle \sqcap, \sqcup \rangle$ is a multimorphism of f. It stops when we get $M_i = P_i$ for all $i \in V$.

We now describe one iteration. First, we identify subset $U \subseteq V$ and subsets $A_i, B_i \subseteq D_i$ for each $i \in U$ using the following algorithm:

1: pick node $k \in V$ and pair $\{a, b\} \in \overline{M}_k$. (If they do not exist, terminate and go to Stage 3.)
2: set $U = \{k\}$, $A_k = \{a\}$, $B_k = \{b\}$
3: **while** there exists $i \in V - U$ such that $\rho_{ki}(A_k, \cdot) \cap \rho_{ki}(B_k, \cdot) = \varnothing$ **do**
4: add i to U, set $A_i = \rho_{ki}(A_k, \cdot)$, $B_i = \rho_{ki}(B_k, \cdot)$
// compute "closure" of sets A_i for $i \in U$
5: **while** there exists $a \in D_k - A_k$ s.t. $a \in \rho_{ki}(\cdot, A_i)$ for some $i \in U - \{k\}$ **do**
6: add a to A_k
set $A_j = \rho_{kj}(A_k, \cdot)$ for all $j \in U - \{k\}$
7: **end while**
// compute "closure" of sets B_i for $i \in U$
8: **while** there exists $b \in D_k - B_k$ s.t. $b \in \rho_{ki}(\cdot, B_i)$ for some $i \in U - \{k\}$ **do**
9: add b to B_k
set $B_j = \rho_{kj}(B_k, \cdot)$ for all $j \in U - \{k\}$
10: **end while**
// done
11: **end while**
12: **return** set $U \subseteq V$ and sets $A_i, B_i \subseteq D_i$ for $i \in U$

LEMMA 4.7. *Sets U and A_i, B_i for $i \in U$ produced by the algorithm have the following properties:*

(a) Sets A_i and B_i for $i \in U$ are disjoint.

(b) $\{a, b\} \in \overline{M}_i$ for all $i \in U$, $a \in A_i$, $b \in B_i$.

(c) $\rho_{ki}(A_k, \cdot) = A_i$, $\rho_{ki}(B_k, \cdot) = B_i$, $\rho_{ki}(\cdot, A_i) = A_k$, $\rho_{ki}(\cdot, B_i) = B_k$ for all $i \in U - \{k\}$ where k is the node chosen in line 1.

(d) Suppose that $i \in U$ and $j \in \overline{U} \equiv V - U$. If $(c, x) \in \rho_{ij}$ where $c \in A_i \cup B_i$ and $x \in D_j$ then $(d, x) \in \rho_{ij}$ for all $d \in A_i \cup B_i$.

To complete the iteration, we modify sets M_i and operations $\sqcap_i, \sqcup_i$ for each $i \in U$ as follows:

- add all pairs $\{a, b\}$ to M_i where $a \in A_i$, $b \in B_i$.
- redefine $a \sqcap_i b = b \sqcap_i a = a$, $a \sqcup_i b = b \sqcup_i a = b$ for all $a \in A_i$, $b \in B_i$

LEMMA 4.8. *The new pair of operations $\langle \sqcap, \sqcup \rangle$ is a multimorphism of f.*

The two lemmas imply that all steps are well-defined, and upon termination the algorithm produces a pair $\langle \sqcap, \sqcup \rangle$ which is an STP multimorphism of f.

Stage 3: Reduction to a submodular minimisation problem At this stage we have an STP multimorphism. Hence, the instance can be solved by Theorem 2.1.

References

[1] K.A. Baker and A.F. Pixley. Polynomial Interpolation and the Chinese Remainder Theorem. *Mathematische Zeitschrift*, 143(2):165–174, 1975.

[2] L. Barto, M. Kozik, M. Maróti, and T. Niven. CSP dichotomy for special triads. *Proceedings of the American Mathematical Society*, 137(9):2921–2934, 2009.

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

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

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

[6] Libor Barto and Marcin Kozik. New Conditions for Taylor Varieties and CSP. In *Proceedings of the 25th IEEE Symposium on Logic in Computer Science (LICS'10)*, pages 100–109, 2010.

[7] J. Berman, P. Idziak, P. Marković, R. McKenzie, M. Valeriote, and R. Willard. Varieties with few subalgebras of powers. *Transactions of the American Mathematical Society*, 362(3):1445–1473, 2010.

[8] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based Constraint Satisfaction and Optimisation. *Journal of the ACM*, 44(2):201–236, 1997.

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

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

[11] Andrei A. Bulatov. Tractable Conservative Constraint Satisfaction Problems. In *Proceedings of the 18th IEEE Symposium on Logic in Computer Science (LICS'03)*, pages 321–330, 2003.

[12] R.E. Burkard, B. Klinz, and R. Rudolf. Perspectives of Monge Properties in Optimization. *Discrete Applied Mathematics*, 70(2):95–161, 1996.

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

[14] D. A. Cohen, M. C. Cooper, P. G. Jeavons, and A. A. Krokhin. The Complexity of Soft Constraint Satisfaction. *Artificial Intelligence*, 170(11):983–1016, 2006.

[15] M.C. Cooper. An Optimal k-consistency Algorithm. *Artificial Intelligence*, 41(1):89–95, 1989.

[16] M.C. Cooper. Personal communication. 2011.

[17] N. Creignou. A dichotomy theorem for maximum generalized satisfiability problems. *Journal of Computer and System Sciences*, 51(3):511–522, 1995.

[18] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classification of Boolean Constraint Satisfaction Problems*, volume 7 of *SIAM Monographs on Discrete Mathematics and Applications*. SIAM, 2001.

[19] Víctor Dalmau, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint Satisfaction, Bounded Treewidth, and Finite-Variable Logics. In *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming (CP'02)*, volume 2470 of *Lecture Notes in Computer Science*, pages 310–326. Springer, 2002.

[20] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[21] R. Dechter and J. Pearl. Network-based Heuristics for Constraint Satisfaction Problems. *Artificial Intelligence*, 34(1):1–38, 1988.

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

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

[24] M. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.

[25] Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *Journal of the ACM*, 54(1):1–24, 2007.

[26] Martin Grohe and Dániel Marx. Constraint solving via fractional edge covers. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'06)*, pages 289–298, 2006.

[27] Pavol Hell and Jaroslav Nešetřil. Colouring, constraint satisfaction, and complexity. *Computer Science Review*, 2(3):143–163, 2008.

[28] P. Jeavons, D. Cohen, and M. C. Cooper. Constraints, Consistency and Closure. *Artificial Intelligence*, 101(1–2):251–265, 1998.

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

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

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

[32] S. Khanna, M. Sudan, L. Trevisan, and D. Williamson. The approximability of constraint satisfaction problems. *SIAM Journal on Computing*, 30(6):1863–1920, 2001.

[33] Ph.G. Kolaitis and M.Y. Vardi. Conjunctive-Query Containment and Constraint Satisfaction. *Journal of Computer and System Sciences*, 61(2):302–332, 2000.

[34] Gabor Kun and Mario Szegedy. A New Line of Attack on the Dichotomy Conjecture. In *Proceedings of the 41st Annual*

ACM Symposium on Theory of Computing (STOC'09), pages 725–734, 2009.

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

[36] Dániel Marx. Approximating fractional hypertree width. *ACM Transactions on Algorithms*, 6(2), Article 29, 2010.

[37] Dániel Marx. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC'10)*, pages 735–744, 2010.

[38] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, Approximation, and Complexity Classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991.

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

[40] Prasad Raghavendra and David Steurer. How to Round Any CSP. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS'09)*, pages 586–594, 2009.

[41] T.J. Schaefer. The Complexity of Satisfiability Problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC'78)*, pages 216–226, 1978.

[42] T. Schiex, H. Fargier, and G. Verfaillie. Valued Constraint Satisfaction Problems: Hard and Easy Problems. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, 1995.

[43] Rustem 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)*, pages 657–668, 2010.

[44] Rustem Takhanov. Extensions of the Minimum Cost Homomorphism Problem. In *Proceedings of the 16th International Computing and Combinatorics Conference (COCOON'10)*, pages 328–337, 2010.

[45] D. Topkis. *Supermodularity and Complementarity*. Princeton University Press, 1998.

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