The Complexity of Soft Constraint Satisfaction

David A. Cohen,
Department of Computer Science,
Royal Holloway, University of London, UK
d.cohen@rhul.ac.uk

Martin C. Cooper,
IRIT, University of Toulouse III, France
cooper@irit.fr

Peter G. Jeavons,
Computing Laboratory, University of Oxford, UK
peter.jeavons@comlab.ox.ac.uk

Andrei A. Krokhin,
Department of Computer Science, University of Durham, UK
andrei.krokhin@durham.ac.uk

April 10, 2006

Abstract

Over the past few years there has been considerable progress in methods to systematically analyse the complexity of constraint satisfaction problems with specified constraint types. One very powerful theoretical development in this area links the complexity of a set of constraints to a corresponding set of algebraic operations, known as polymorphisms.

In this paper we extend the analysis of complexity to the more general framework of combinatorial optimisation problems expressed using various forms of soft constraints. We launch a systematic investigation of the complexity of these problems by extending the notion of a polymorphism to a more general algebraic operation, which we call a multimorphism. We show that many tractable sets of soft constraints, both established and novel, can be characterised by the presence of particular multimorphisms. We also show that a simple set of NP-hard constraints has very restricted multimorphisms. Finally, we use the notion of multimorphism to give a complete classification of complexity for the Boolean case which extends several earlier classification results for particular special cases.

Keywords: soft constraints, valued constraint satisfaction, combinatorial optimisation, submodular functions, tractability, multimorphism.

1 Introduction

In the standard constraint satisfaction framework [14, 38] a *constraint* is understood to be a predicate, or relation, specifying the allowed combinations of values for some fixed subset of variables: we will refer to such constraints here as *crisp* constraints. Problems with crisp constraints deal only with *feasibility*: no satisfying solution is considered better than any other.

A number of authors have suggested that the usefulness of the constraint satisfaction framework could be greatly enhanced by extending the definition of a constraint to include also *soft* constraints, which allow different measures of desirability to be associated with different combinations of values [1, 2, 43]. In this extended framework a constraint can be seen as a *cost function* defined on a fixed subset of the variables which maps each possible combination of values for those variables to a measure of desirability or undesirability.

Problems with soft constraints deal with *optimisation* as well as feasibility: the aim is to find an assignment of values to all of the variables having the best possible overall combined measure of desirability. In this paper we examine how limiting the choice of cost functions affects the complexity of this optimisation problem.

Example 1.1 Consider an optimisation problem where we have to choose sites for n service stations along a motorway of length L, subject to the following requirements:

- There are $r > n$ possible sites at distances $d_1, \ldots, d_r$ along the motorway.
- Each pair of consecutive service stations must be separated by a distance which is no less than A and no more than B.
- The service stations should be as equally spaced as possible.

One possible way to model this situation is as follows:

- Introduce variables $v_1, v_2, \ldots, v_n$ to represent the position of each service station, where each variable must be assigned a value from the set $\{d_1, \ldots, d_r\}$.
- Impose a binary constraint on each pair v_i, v_{i+1}, $i = 1, \ldots, n-1$, with cost function δ, where $\delta(x, y) = 0$ if $A \leq y - x \leq B$ and ∞ otherwise.
- Impose a binary constraint on each pair v_i, v_{i+1}, $i = 0, \ldots, n$, with cost function ζ, where $\zeta(x, y) = |x - y|^2$. Add a unary constraint on v_1 with cost function $\zeta(0, x)$, and a unary constraint on v_n, with cost function $\zeta(x, L)$. (Note that the sum of these functions in minimal when the values of these variables are equally spaced between 0 and L).

We would then seek an assignment of values from the set $D = \{d_1, \ldots, d_r\}$, to all of the variables, which minimises the sum of all these cost functions:

$$\sum_{i=1}^{n-1} \delta(v_i, v_{i+1}) + \zeta(0, v_1) + \sum_{i=1}^{n-1} \zeta(v_i, v_{i+1}) + \zeta(v_n, L).$$

$\square$

The cost of allowing additional flexibility in the specification of constraints, in order to model optimisation criteria as well as feasibility, is generally an increase in computational difficulty. For example, we establish below that the class of problems containing only unary constraints and a soft version of the binary equality constraint is NP-hard (see Example 2.11).

On the other hand, for certain types of soft constraint it is possible to solve the associated optimisation problems efficiently. For example, we establish below that optimisation problems of the form described in Example 1.1 can be solved in polynomial time (see Example 6.13).

In the case of crisp constraints there has been considerable progress in analysing the complexity of problems involving different types of constraints. This work has led to the identification of a number of classes of constraints which are *tractable*, in the sense that there exists a polynomial time algorithm to determine whether or not any collection of constraints from such a class can be simultaneously satisfied [15, 26, 33, 40, 42]. One powerful result in this area establishes that any tractable class of constraints over a finite domain must have relations which are all preserved by a non-trivial algebraic operation, known as a *polymorphism* [6, 26].

In the case of soft constraints there has been little detailed investigation of the tractable cases, except for certain special cases on a two-valued domain [10, 30], and a special case involving simple temporal constraints [31]. In an earlier paper [7] we identified a particular tractable class of binary soft constraints, and showed that this class was maximal, in the sense that adding any other soft binary constraint which is not in the class gives rise to a class of problems which is NP-hard. This class has recently been used to study the complexity of the MINIMUM COST HOMOMORPHISM problem [21], which has been used to model the "Level of Repair Analysis" problem from operations research [22] (see Example 2.7).

In this paper we take the first step towards a systematic analysis of the complexity of soft constraints of arbitrary arity over arbitrary finite domains. To do this we generalise the algebraic ideas used to study crisp constraints, and introduce a new algebraic operation which we call a *multimorphism*. Every cost function has an associated set of multimorphisms, and every multimorphism has an associated set of cost functions. We show that, for several different types of multimorphism, the associated collection of soft constraints is a maximal tractable class. In other words, we show that several maximal tractable classes of soft constraints can be precisely characterised as the collection of all soft constraints associated with a particular multimorphism. Furthermore, we show that a simple NP-hard class of soft constraints has very restricted multimorphisms.

Finally, we apply the techniques developed in the paper to the two-valued domain, where we obtain a new dichotomy theorem which classifies the complexity of any set of soft constraints over this domain (Theorem 7.1). This dichotomy theorem generalises several earlier results concerning the complexity of particular Boolean constraint problems, including the SATISFIABILITY problem [42], the MAX-SAT problem [9], the weighted MIN-ONES problem [10, 30], and the weighted MAX-ONES problem [10, 30] (see Corollary 7.12).

The examples given throughout the paper demonstrate that the framework we introduce here can be used to unify isolated results about tractable problem classes from many different application areas, as well as prompting the discovery of new tractable classes. For example, the notion of a multimorphism generalises the notion of a polymorphism (see Proposition 4.10), and so can be used to express earlier results concerning the characterisation of tractable subproblems of many different decision problems: in the case of the SATISFIABILITY problem these include the HORN-SAT and 2-SAT subproblems [19]; in the case of the standard crisp constraint satisfaction problem these include generalisations of HORN-SAT (such as the so-called 'max-closed' constraints [29, 26]), generalisations of 2-SAT (such as the so-called '0/1/all' or 'implicative' constraints [8, 25, 32]) and systems of linear equations [26]. The notion of a multimorphism can also be used to characterise tractable subproblems of optimisation problems: in the case of the optimisation problem MAX-SAT these include the '0-valid', '1-valid' and '2-monotone' constraints [10]; in the case of optimisation problems over sets these include the minimisation of submodular set functions [23, 39] and bisubmodular set functions [18].

2 Definitions

Several alternative mathematical frameworks for soft constraints have been proposed in the literature, including the very general frameworks of 'semi-ring based constraints' and 'valued constraints' [1, 2, 43]. For simplicity, we shall adopt the valued constraint framework here (the relationship with the semi-ring framework is discussed briefly in Section 8).

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

Definition 2.1 A **valuation structure**, Ω, is a totally ordered set, with a minimum and a maximum element (denoted 0 and ∞), together with a commutative, associative binary **aggregation operator** (denoted $\oplus$), such that for all $\alpha, \beta, \gamma \in \Omega$, $\alpha \oplus 0 = \alpha$ and $\alpha \oplus \gamma \geq \beta \oplus \gamma$ whenever $\alpha \geq \beta$.

Definition 2.2 An instance of the **valued constraint satisfaction problem,** VCSP, is a tuple $\mathcal{P} = \langle V, D, C, \Omega \rangle$ where:

- V is a finite set of **variables**;
- D is a finite set of possible **values**;
- Ω is a valuation structure representing possible **costs**;
- C is a set of **constraints**.

  Each element of C is a pair $c = \langle \sigma, \phi \rangle$ where σ is a tuple of variables called the **scope** of c, and ϕ is a mapping from $D^{|\sigma|}$ to Ω, called the **cost function** of c.

Definition 2.3 For any VCSP instance $\mathcal{P} = \langle V, D, C, \Omega \rangle$, an **assignment** for $\mathcal{P}$ is a mapping $s : V \to D$. The **cost** of an assignment s, denoted $\mathrm{Cost}_{\mathcal{P}}(s)$, is given by the aggregation of the costs for the restrictions of s onto each constraint scope, that is,

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

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

Our results in Sections 3 and 4 (except for Proposition 4.10) will apply to any valuation structure satisfying Definition 2.1. In Sections 5, 6 and 7, and in the examples of particular soft constraint problems given throughout the paper, we will focus on the valuation structure $\overline{\mathbb{R}}_+$, consisting of the non-negative real numbers together with infinity, with the usual ordering and the usual addition operation. (Possible extensions of our results to other valuation structures are discussed briefly in Section 8.)

The valuation structure $\overline{\mathbb{R}}_+$ is sufficiently flexible to allow us to express a wide range of problems as valued constraint satisfaction problems with costs in $\overline{\mathbb{R}}_+$, as the following examples indicate.

Example 2.4 [**Standard CSP**] In the standard constraint satisfaction problem with crisp constraints [14, 36] each constraint c is specified by a pair, $\langle \sigma, R \rangle$, where σ is the scope of c and R is a relation specifying the allowed combinations of values for the variables in σ.

For any standard constraint satisfaction problem instance $\mathcal{P}$, we can define a corresponding valued constraint satisfaction problem instance $\widehat{\mathcal{P}}$ in which the range of the cost functions of all the constraints is the set $\{0, \infty\} \subseteq \overline{\mathbb{R}}_+$. For each crisp constraint $\langle \sigma, R \rangle$ of $\mathcal{P}$, we define a corresponding valued constraint $\langle \sigma, \phi_R \rangle$ of $\widehat{\mathcal{P}}$; the cost function ϕ_R maps each tuple allowed by R to 0, and each tuple disallowed by R to ∞. The cost of an assignment s for $\widehat{\mathcal{P}}$ is computed as in Definition 2.3, so it equals the minimal possible cost, 0, if and only if s satisfies all of the crisp constraints in $\mathcal{P}$. □

Example 2.5 [**Max-CSP**] For any standard constraint satisfaction problem instance $\mathcal{P}$ with crisp constraints, we can define a corresponding valued constraint satisfaction problem instance $\mathcal{P}^{\#}$ in which the range of the cost functions of all the constraints is the set $\{0, 1\} \subseteq \overline{\mathbb{R}}_+$. For each crisp constraint $\langle \sigma, R \rangle$ of $\mathcal{P}$, we define a corresponding valued constraint $\langle \sigma, \chi_R \rangle$ of $\mathcal{P}^{\#}$; the cost function χ_R maps each tuple allowed by R to 0, and each tuple disallowed by R to 1.

The cost of an assignment s for $\mathcal{P}^{\#}$ is again computed as in Definition 2.3, so in this case it equals the total number of crisp constraints in $\mathcal{P}$ which are violated by s. Hence a solution to $\mathcal{P}^{\#}$ corresponds to an assignment which violates the minimal number of constraints of $\mathcal{P}$, and hence satisfies the maximal number of constraints of $\mathcal{P}$. Finding assignments of this kind is generally referred to as solving the MAX-CSP problem [17, 34]. □

Example 2.6 [**Minimum k-Terminal Cut and Min-Cut**] Let G be an undirected graph with vertices V and edges E, and let $\{v_1, v_2, \ldots, v_k\} \subseteq V$ be a set of k distinguished vertices. The problem of finding a smallest set of edges whose removal disconnects the distinguished vertices from each other is known as the MINIMUM k-TERMINAL CUT problem [12]; such a set of edges is called a *minimum k-terminal cut.* (In the special case when $k = 2$ this problem is known as the MIN-CUT problem [39].)

Each instance of the MINIMUM k-TERMINAL CUT problem can be formulated as a VCSP instance $\mathcal{P}_G$ with costs in $\overline{\mathbb{R}}_+$. The instance $\mathcal{P}_G$ is constructed as follows: the variables of $\mathcal{P}_G$ are the vertices V of G, and they take values in the set $D = \{1, 2, \ldots, k\}$. For each distinguished vertex $v_i \in \{v_1, v_2, \ldots, v_k\}$, impose a unary constraint on the variable v_i with cost function $\psi_i : \{0, 1\} \to \overline{\mathbb{R}}_+$, defined as follows:

$$\psi_i(x) = \begin{cases} 0 & \text{if } x = i \\ \infty & \text{otherwise} \end{cases}$$

For each edge $e \in E$, impose a binary constraint with scope e and cost function $\phi_{\mathrm{EQ}} : D^2 \to \overline{\mathbb{R}}_+$, defined as follows

$$\phi_{\mathrm{EQ}}(x, y) = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{otherwise.} \end{cases}$$

It is straightforward to check that the number of edges in a minimum k-terminal cut of G is equal to the cost of a solution to $\mathcal{P}_G$. □

Example 2.7 [**Level of Repair Analysis**] Level of Repair Analysis (LORA) is a prescribed procedure for defence logistics support planning [22]. For a complex engineering system containing perhaps thousands of assemblies, sub-assemblies, modules and components, LORA seeks to determine an optimal provision of repair and maintenance facilities to minimize overall life-cycle costs.

In the simple model of this problem presented in [22] the engineering system is modelled as a set of items V, together with a binary relation E on V such that $E(v, v')$ holds when item v is *contained in* item v'. Each item in V must be assigned a *repair level* (such as "central repair", "local repair", "discard") chosen from some fixed set of possible repair levels, D. There is a fixed cost c_{dv} associated with the assignment of repair level $d \in D$ to item $v \in V$ (which may be infinite if that repair level is not available for that item). There are also some restrictions on the allowed assignments for pairs of items related by E: for example, if an item is assigned "discard", then all items contained in that item must also be assigned this repair level. The question is to find an assignment of repair levels to items which minimises the total cost.

For any LORA instance of this kind, we can define a corresponding valued constraint satisfaction problem instance $\mathcal{P}$ with costs in $\overline{\mathbb{R}}_+$. For each item $v \in V$ we define a unary constraint $\langle\langle v\rangle, \kappa_v\rangle$ of $\mathcal{P}$, where the cost function κ_v maps each element $d \in D$ to c_{dv}. For each pair $\langle v, v'\rangle$ of items related by E, we define a valued constraint $\langle\langle v, v'\rangle, \phi\rangle$ of $\mathcal{P}$, where the cost function ϕ maps each pair of allowed repair levels to 0, and each pair of disallowed repair levels to ∞.

A solution to $\mathcal{P}$ corresponds to an assignment of repair levels which minimises the total cost. $\square$

The problem of finding a solution to a valued constraint satisfaction problem is an NP-optimisation problem, that is, it lies in the complexity class NPO (see [10] for a formal definition of this class).

For each valued constraint satisfaction problem there is a corresponding decision problem in which the question is to decide whether there is a solution with cost lower than some given threshold value. It is clear from Example 2.4 that there is a polynomial-time reduction to this decision problem from the standard constraint satisfaction problem, which is known to be NP-complete [36], so the general VCSP is NP-hard. In this paper we will consider the effect of restricting the forms of cost function allowed in the constraints; we will show that in some cases this results in more tractable versions of the VCSP.

Definition 2.8 Let D be a set and Ω a valuation structure. A **valued constraint language** over D with costs in Ω is defined to be a set, Γ, such that each $\phi \in \Gamma$ is a function from D^m to Ω, for some $m \in \mathbb{N}$, where m is called the arity of ϕ. The class VCSP(Γ) is defined to be the class of all VCSP instances where the cost functions of all constraints lie in Γ.

We will say that a *finite* valued constraint language Γ is **tractable** if every instance in VCSP(Γ) can be solved in polynomial time. We will say that an *infinite* valued constraint language is tractable if every finite subset[1] of it is tractable. Finally, we will say that a valued constraint language Γ is **NP-hard** if the decision problem corresponding to VCSP(Γ') is NP-complete, for some finite $\Gamma' \subseteq \Gamma$.

Example 2.9 [**SAT and Max-SAT**] Let Γ be any valued constraint language over a set D, where $|D| = 2$. In this case VCSP(Γ) is called a **Boolean** valued constraint satisfaction problem.

If we restrict Γ even further, by only allowing cost functions with range $\{0, \infty\} \subseteq \overline{\mathbb{R}}_+$, as in Example 2.4, then each VCSP(Γ) corresponds precisely to a standard Boolean constraint satisfaction problem with crisp constraints. Such problems are sometimes known as Generalised Satisfiability problems [19, 42]. The complexity of VCSP(Γ) for such restricted sets Γ has been completely characterised, and the six tractable cases have been identified [10, 19, 42].

Alternatively, if we restrict Γ by only allowing functions with range $\{0, 1\} \subseteq \overline{\mathbb{R}}_+$, as in Example 2.5, then each VCSP(Γ) corresponds precisely to a standard Boolean maximum satisfiability problem, in which the aim is to satisfy the maximum number of crisp constraints. Such problems are sometimes known as Max-Sat problems [10]. The complexity of VCSP(Γ) for such restricted sets Γ has been completely characterised, and the three tractable cases have been identified (see Theorem 7.6 of [10]). $\square$

[1]Defining tractability in terms of finite subsets ensures that the tractability of a valued constraint language is independent of whether the cost functions are represented *explicitly* (via tables of values) or *implicitly* (via oracles).

The next two examples indicate that generalising the constraint satisfaction framework to include valued constraints can indeed increase the computational complexity. For example, the standard 2-Satisfiability problem is well-known to be tractable [19], but the valued constraint satisfaction problem involving only the single binary Boolean function, ϕ_{XOR}, defined in Example 2.10, is NP-hard.

Example 2.10 Let Γ_{XOR} be the Boolean valued constraint language over $D = \{0,1\}$ which contains just the single binary function $\phi_{\mathrm{XOR}} : D^2 \to \overline{\mathbb{R}}_+$ defined by

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

The problem VCSP(Γ_{XOR}) corresponds to the Max-Sat problem for the exclusive-or predicate, which is known to be NP-hard (see Lemma 7.4 of [10]), so Γ_{XOR} is NP-hard. □

Similarly, the standard constraint satisfaction problem involving only crisp unary constraints and equality constraints is clearly trivial, but the valued constraint satisfaction problem involving only unary valued constraints and a soft version of the equality constraint, specified by the function ϕ_{EQ} defined in Example 2.6, is NP-hard.

Example 2.11 Let Γ_3 be the valued constraint language over $D = \{0,1,2\}$ consisting of the set of all unary functions with costs in $\overline{\mathbb{R}}_+$ together with the single binary function $\phi_{\mathrm{EQ}} : D^2 \to \overline{\mathbb{R}}_+$, defined in Example 2.6.

Even though Γ_3 is apparently simple, it can be shown that VCSP(Γ_3) is NP-hard, by reduction from the Minimum 3-Terminal Cut problem defined in Example 2.6, which is known to be NP-hard [12]. To obtain the reduction, we use the construction described in Example 2.6 to transform each instance of Minimum 3-Terminal Cut to an instance of VCSP(Γ_3) in polynomial time. □

In order to allow us to translate easily between relations and functions, as described in Example 2.4, we make the following definitions.

Definition 2.12 Any function ϕ which only takes values in the set $\{0,\infty\} \subseteq \Omega$ will be called a **crisp** function.

For any relation R, with arity m, we define an associated crisp function known as the **feasibility function** of R, and denoted ϕ_R, as follows:

$$\phi_R(x_1, x_2, \ldots, x_m) = \begin{cases} 0 & \text{if } \langle x_1, x_2, \ldots, x_m \rangle \in R \\ \infty & \text{otherwise} \end{cases}$$

For any m-ary function ϕ into any valuation structure Ω, we define a relation known as the **feasibility relation** of ϕ, and denoted Feas(ϕ), as follows:

$$\langle x_1, x_2, \ldots, x_m \rangle \in \mathrm{Feas}(\phi) \Leftrightarrow \phi(x_1, x_2, \ldots, x_m) < \infty.$$

A function ϕ will be called **essentially crisp** if ϕ takes at most one finite value, that is, there is some value α such that $\phi(x) = \beta < \infty \Rightarrow \beta = \alpha$. Any valued constraint language Γ containing essentially crisp functions only will be called an **essentially crisp language**.

Note that when Γ is an essentially crisp language any assignment with finite cost has the same cost as any other assignment with finite cost. Hence we can solve any instance of VCSP(Γ) for such languages by solving the corresponding standard constraint satisfaction problem in which each valued constraint $\langle \sigma, \phi \rangle$ is replaced by the crisp constraint $\langle \sigma, \mathrm{Feas}(\phi) \rangle$ (see Definition 2.12). We will use this observation a number of times in establishing the results below.

3 Expressibility

Let Γ be a valued constraint language, and consider an arbitrary instance $\mathcal{P}$ in VCSP(Γ). The variables in the scope of any constraint of $\mathcal{P}$ are explicitly constrained. What is more, *any* subset of the variables of $\mathcal{P}$ may be constrained *implicitly*, due to the combined effect of the constraints of $\mathcal{P}$. The cost function which describes this implicit constraint may or may not be an element of Γ, but can, in a sense, be *expressed* using elements of Γ.

The next two definitions formalise this idea of a function being expressible over a valued constraint language.

Definition 3.1 For any VCSP instance $\mathcal{P} = \langle V, D, C, \Omega \rangle$, and any tuple of distinct variables $W = \langle v_1, \ldots, v_k \rangle$, the **cost function of $\mathcal{P}$ on W**, denoted $\Phi_{\mathcal{P}}^{W}$, is defined as follows:

$$\Phi_{\mathcal{P}}^{W}(d_1, \ldots, d_k) \stackrel{\text{def}}{=} \min_{\{s:V \to D \ \mid \ \langle s(v_1), \ldots, s(v_k) \rangle = \langle d_1, \ldots, d_k \rangle\}} \mathrm{Cost}_{\mathcal{P}}(s)$$

Note that the cost function of $\mathcal{P}$ on W is a kind of *projection* of the overall cost function onto a specified subset of the variables.

Definition 3.2 A function ϕ is **expressible** over a valued constraint language Γ if there exists an instance $\mathcal{P} = \langle V, D, C, \Omega \rangle$ in VCSP(Γ) and a list W of variables from V such that $\phi = \Phi_{\mathcal{P}}^{W}$.

The set of all functions expressible over Γ will be denoted Γ^*.

In all cases $\Gamma^* \supseteq \Gamma$, but it is often the case that Γ^* contains many more functions than Γ, as the next example illustrates.

Example 3.3 Let $D = \{0, 1, 2, \ldots, |D| - 1\}$ be a subset of the integers, and let $\Gamma_1 = \{ \phi_0, \phi_1 \}$ be the valued constraint language over D consisting of the constant unary cost function $\phi_0 : D \to \overline{\mathbb{R}}_+$ defined by $\phi_0(x) = 1$ and the unary cost function $\phi_1 : D \to \overline{\mathbb{R}}_+$ defined by $\phi_1(x) = x$.

In this case the language $(\Gamma_1)^*$ also contains all cost functions defined by **linear expressions** with non-negative integer coefficients, since for any such cost function ϕ we have

$$\phi(x_1, x_2, \ldots, x_m) = a_0\phi_0(x_1) + \sum_{i=1}^{m} a_i\phi_1(x_i)$$

for some set of non-negative integers $a_0, a_1, \ldots, a_m$, and hence $\phi = \Phi_{\mathcal{P}}^{\langle x_1, \ldots, x_m \rangle}$ for some instance $\mathcal{P}$ of VCSP(Γ_1).

This much larger valued constraint language will be denoted Γ_{LIN}. □

The notion of expressibility is a key tool in analysing the complexity of valued constraint languages, as the next result shows.

Theorem 3.4 *Let Γ and Γ' be valued constraint languages with $\Gamma' \subseteq \Gamma^*$.*

- *If Γ is tractable, then Γ' is also tractable.*
- *If Γ' is NP-hard, then Γ is also NP-hard.*

Proof: Let Γ_0 be a finite subset of Γ', let $\mathcal{P} = \langle V, D, C, \Omega \rangle$ be any instance in VCSP(Γ_0), and let $c = \langle \sigma, \phi \rangle$ be a constraint in C.

Since $\Gamma' \subseteq \Gamma^*$ we know that ϕ is expressible over Γ, so there exists an instance $\mathcal{P}_\phi$ in VCSP(Γ), and a list of variables W of $\mathcal{P}_\phi$, such that $\Phi_{\mathcal{P}_\phi}^{W} = \phi$. Hence we can replace the constraint c in $\mathcal{P}$ with a copy of $\mathcal{P}_\phi$ (where the variables in the scope σ are identified with the list of variables W, and the remaining variables of $\mathcal{P}_\phi$ are disjoint from V) to obtain a new problem instance $\mathcal{P}'$. Note that the solutions to $\mathcal{P}'$, when restricted to V, correspond precisely to the original solutions to $\mathcal{P}$ and have the same costs.

By repeating this construction for each constraint c of $\mathcal{P}$, we can obtain an instance $\mathcal{P}''$ of VCSP(Γ) whose solutions, when restricted to V, correspond precisely to the original solutions to $\mathcal{P}$ and have the same costs. Since Γ_0 is finite, there is a bound on the size of the instances $\mathcal{P}_\phi$ used in the construction, and so the size of $\mathcal{P}''$ is bounded by a constant multiple of the size of $\mathcal{P}$.

If Γ is tractable, then we can solve $\mathcal{P}$ in polynomial time by carrying out this construction, using a polynomial time algorithm for VCSP(Γ), and then restricting the solutions obtained to the original variables V. This is sufficient to establish that Γ' is tractable.

If Γ' is NP-hard, then this construction establishes that Γ is also NP-hard. □

Example 3.5 Consider the languages Γ_1 and Γ_{LIN} defined in Example 3.3. Since Γ_1 contains only unary cost functions it is clearly tractable. Since $\Gamma_{\mathrm{LIN}} \subseteq (\Gamma_1)^*$, it follows from Theorem 3.4 that Γ_{LIN} is tractable. □

4 Multimorphisms

For crisp constraints, it has been shown that the expressive power of a set of relations is determined by certain algebraic invariance properties of those relations, known as *polymorphisms* [6, 26, 27, 28, 41, 46].

Throughout the rest of this paper, the ith component of a tuple t will be denoted $t[i]$.

Definition 4.1 A **polymorphism** of a relation $R \subseteq D^m$ is a function $f : D^k \to D$, for some k, such that whenever $t_1, \ldots, t_k$ are elements of R then so is $\langle f(t_1[1], \ldots, t_k[1]), \ldots, f(t_1[m], \ldots, t_k[m])\rangle$.

Example 4.2 Let $D = \{0, 1, 2, \ldots, |D| - 1\}$ be a subset of the integers, and let R be the ternary relation over D defined by $R = \{\langle x, y, z\rangle \mid ax + by \leq cz\}$, where a, b, c are positive constants. Consider the function $f : D^2 \to D$ defined by $f(x, y) = \mathrm{Min}(x, y)$. For any elements, t_1, t_2, of R we have that $at_1[1] + bt_1[2] \leq ct_1[3]$ and $at_2[1] + bt_2[2] \leq ct_2[3]$, which together imply that

$$a\,\mathrm{Min}(t1[1], t2[1]) + b\,\mathrm{Min}(t_1[2], t_2[2]) \leq c\,\mathrm{Min}(t_1[3], t_2[3]).$$

Hence f is a polymorphism of R, and we will say that R has the polymorphism Min. □

The concept of a polymorphism is specific to *relations*, and cannot be applied directly to the *functions* of a valued constraint language. However, we now introduce a more general notion, which we call a *multimorphism*, which does apply directly to functions (see Figure 1 for a concrete example).

Definition 4.3 Let D be a set, Ω a valuation structure, and $\phi : D^m \to \Omega$ a function.

We say that $F : D^k \to D^k$ is a **multimorphism** of ϕ if, for any list of k-tuples $t_1, t_2 \ldots, t_m$ over D we have

$$\bigoplus_{i=1}^{k} \phi(F(t_1)[i], F(t_2)[i], \ldots, F(t_m)[i]) \leq \bigoplus_{i=1}^{k} \phi(t_1[i], t_2[i], \ldots, t_m[i]). \qquad (1)$$

If F is a multimorphism of every function in a language Γ, then we will say that F is a multimorphism of Γ, and that Γ has the multimorphism F. The largest such language, consisting of *all* functions ϕ with costs in Ω which have F as a multimorphism, will be denoted $\mathrm{Imp}_{\Omega}(F)$.

The notation $\mathrm{Imp}_{\Omega}(F)$ is an abbreviation for "Improved by F"; this term was chosen because the functions for which F is a multimorphism are precisely those functions whose aggregated value is "improved" (i.e., lowered, or left unchanged) by applying the function F (co-ordinatewise) to any suitable collection of argument vectors, to obtain a new collection of argument vectors (Equation 1).

It will often be convenient to describe a multimorphism $F : D^k \to D^k$ by listing its k separate **component functions**, $F_i : D^k \to D$, defined by $F_i(x_1, \ldots, x_k) = F(x_1, \ldots, x_k)[i]$.

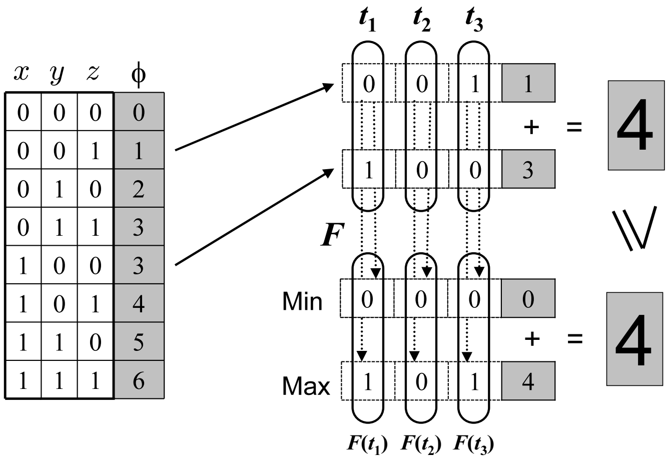

Figure 1: An example of the form of inequality that shows that the function $\phi : \{0,1\}^3 \to \overline{\mathbb{R}}_+$ defined by $\phi(x,y,z) = 3x + 2y + z$ has the multimorphism $F = \langle \text{Min}, \text{Max} \rangle$. (See Example 4.4.)

Example 4.4 Let $D = \{0, 1, 2, \ldots, |D| - 1\}$ be a subset of the integers, and let $\phi : D^3 \to \overline{\mathbb{R}}_+$ be the linear function defined by $\phi(x,y,z) = ax + by + cz$, where a, b, c are positive constants.

Consider the function $F : D^2 \to D^2$ defined by $F(x,y) = \langle \text{Min}(x,y), \text{Max}(x,y) \rangle$. For any list of pairs, t_1, t_2, t_3, over D we have:

$$
\begin{aligned}
&\bigoplus_{i=1}^{2} \phi(F(t_1)[i], F(t_2)[i], F(t_3)[i]) \\
&\quad= \bigoplus_{i=1}^{2} \phi(\langle \text{Min}(t_1[1], t_1[2]), \text{Max}(t_1[1], t_1[2]) \rangle [i], \ldots, \\
&\qquad\qquad\qquad \langle \text{Min}(t_3[1], t_3[2]), \text{Max}(t_3[1], t_3[2]) \rangle [i]) \\
&\quad= \phi(\text{Min}(t_1[1], t_1[2]), \text{Min}(t_2[1], t_2[2]), \text{Min}(t_3[1], t_3[2])) \\
&\qquad\quad \oplus\ \phi(\text{Max}(t_1[1], t_1[2]), \text{Max}(t_2[1], t_2[2]), \text{Max}(t_3[1], t_3[2])) \\
&\quad= a\,\text{Min}(t_1[1], t_1[2]) + b\,\text{Min}(t_2[1], t_2[2]) + c\,\text{Min}(t_3[1], t_3[2]) \\
&\qquad\quad +\ a\,\text{Max}(t_1[1], t_1[2]) + b\,\text{Max}(t_2[1], t_2[2]) + c\,\text{Max}(t_3[1], t_3[2]) \\
&\quad= a(t_1[1] + t_1[2]) + b(t_2[1] + t_2[2]) + c(t_3[1] + t_3[2]) \\
&\quad= \bigoplus_{i=1}^{2} \phi(t_1[i], t_2[i], t_3[i]).
\end{aligned}
$$

(A particular concrete example is illustrated in Figure 1.)

Hence F is a multimorphism of ϕ, and we will say that ϕ has the multimorphism $\langle \text{Min}, \text{Max} \rangle$. $\square$

The next result establishes that multimorphisms have the key property that they extend to all functions expressible over a given language.

Theorem 4.5 *If F is a multimorphism of a valued constraint language Γ, then F is also a multimorphism of Γ^*.*

Proof: Let F be a multimorphism of Γ, and let ϕ_1, ϕ_2 be arbitrary elements of Γ. By Definition 4.3, F is a multimorphism of $\phi_1 \oplus \phi_2$. Similarly, since Equation 1 holds for all choices of tuples t, F is a multimorphism of the function obtained by minimising ϕ_1 over any subset of its arguments. Hence, by Definition 3.2, F is a multimorphism of any function in Γ^*. □

We now show that some important classes of functions are characterised by the property of having a particular form of multimorphism.

Example 4.6 For any finite set V, a real-valued function ψ defined on subsets of V is called a **submodular function** [39] if, for all subsets S and T of V,

$$\psi(S \cap T) + \psi(S \cup T) \leq \psi(S) + \psi(T). \tag{2}$$

The problem of SUBMODULAR FUNCTION MINIMISATION consists in finding a subset S of V for which the value of $\psi(S)$ is minimal. Such problems arise in a number of different contexts [39]. For example, Cunningham [11] showed that finding the maximum flow in a network can be viewed as a special case of the general problem of submodular function minimisation.

It has been known for a long time that submodular functions can be minimised in polynomial time using the ellipsoid method [20]. Recently, several different strongly polynomial, combinatorial algorithms have been proposed for this problem [16, 23, 44].

Any function ψ defined on subsets of a set $V = \{v_1, \ldots, v_n\}$ can be associated with a function $\phi : \{0,1\}^n \rightarrow \overline{\mathbb{R}}_+$ defined as follows: for each tuple $t \in D^{|V|}$, set $\phi(t) = \psi(T)$, where $T = \{v_i \mid t[i] = 1\}$.

For any tuples s, t over $\{0,1\}$, if we set $S = \{v_i \mid s[i] = 1\}$ and $T = \{v_i \mid t[i] = 1\}$, then $S \cap T = \{v_i \mid \mathrm{Min}(s[i], t[i]) = 1\}$, where Min is the function returning the minimum of its two arguments. Similarly, $S \cup T = \{v_i \mid \mathrm{Max}(s[i], t[i]) = 1\}$, where Max is the function returning the maximum of its two arguments. Hence, comparing Equation 2 and Equation 1 (Definition 4.3), it follows that ψ is submodular if and only if the corresponding cost function ϕ has the multimorphism $\langle \mathrm{Min}, \mathrm{Max} \rangle$. □

Example 4.7 For any finite set V, a real-valued function ψ defined on pairs of disjoint subsets of V is called a **bisubmodular function** [18] if for all pairs $\langle S_1, S_2 \rangle$ and $\langle T_1, T_2 \rangle$ of disjoint subsets of V,

$$\psi(\langle S_1, S_2 \rangle \sqcap \langle T_1, T_2 \rangle) + \psi(\langle S_1, S_2 \rangle \sqcup \langle T_1, T_2 \rangle) \leq \psi(\langle S_1, S_2 \rangle) + \psi(\langle T_1, T_2 \rangle) \tag{3}$$

where

$$\langle S_1, S_2\rangle \sqcap \langle T_1, T_2\rangle = \langle S_1 \cap T_1, S_2 \cap T_2\rangle$$
$$\langle S_1, S_2\rangle \sqcup \langle T_1, T_2\rangle = \langle (S_1 \cup T_1) \setminus (S_2 \cup T_2), (S_2 \cup T_2) \setminus (S_1 \cup T_1)\rangle$$

It is known [18] that a bisubmodular function ψ which takes integer values only can be minimised in $\mathrm{O}(|V|^5 \log M)$ time, where M designates the maximum value of the function ψ.

Any function ψ defined on pairs of disjoint subsets of a set $V = \{v_1, \ldots, v_n\}$ can be associated with a function $\phi : \{0, 1, 2\}^n \to \overline{\mathbb{R}}_+$ defined as follows: for each tuple $t \in D^{|V|}$, set $\phi(t) = \psi(\langle T_1, T_2\rangle)$, where $T_1 = \{v_i \mid t[i] = 1\}$ and $T_2 = \{v_i \mid t[i] = 2\}$.

Arguing as in Example 4.6, if follows from Equation 3 that ψ is bisubmodular if and only if the corresponding cost function ϕ has the multimorphism $\langle min_0(x, y), max_0(x, y)\rangle$, where

$$min_0(x, y) = \begin{cases} \mathrm{Min}(x, y) & \text{if } \{x, y\} \neq \{1, 2\} \\ 0 & \text{otherwise} \end{cases}$$

$$max_0(x, y) = \begin{cases} \mathrm{Max}(x, y) & \text{if } \{x, y\} \neq \{1, 2\} \\ 0 & \text{otherwise} \end{cases}$$

□

In Section 6 we will show that a wide range of tractable optimisation problems with costs in $\overline{\mathbb{R}}_+$ are characterised by the presence of certain forms of multimorphism. In Section 7 we will show that in the Boolean case *every* such tractable optimisation problem of the form VCSP(Γ) is characterised by the presence of a particular multimorphism.

A function $F : D^k \to D^k$ is called **conservative** if, for each possible choice of $x_1, x_2, \ldots, x_k$, the tuple $F(x_1, x_2, \ldots, x_k)$ contains the same multi-set of values $x_1, x_2, \ldots, x_k$ (in some order).

Example 4.8 For any totally ordered set D, the function $F : D^k \to D^k$ which returns its arguments in *sorted order* is conservative. For example, the function $F : D^2 \to D^2$ defined by $F(x, y) = \langle \mathrm{Min}(x, y), \mathrm{Max}(x, y)\rangle$ is conservative.

On the other hand, the function $F : D^2 \to D^2$ defined by $F(x, y) = \langle \mathrm{Max}(x, y), \mathrm{Max}(x, y)\rangle$ is not conservative. □

Lemma 4.9 *Any conservative function $F : D^k \to D^k$ is a multimorphism of all unary cost functions.*

Proof: If F is conservative, then Equation 1 of Definition 4.3 holds (with equality) for any unary function ϕ, so F is a multimorphism of any unary function. □

There is a close relationship between the *polymorphisms* of a relation R and the *multimorphisms* of the corresponding feasibility function ϕ_R, as the next result makes clear.

Proposition 4.10 *Let R be a relation of arity m, and let ϕ_R be the corresponding feasibility function with range $\{0, \infty\}$ defined in Definition 2.12.*

For any collection of polymorphisms $f_1, f_2, \ldots, f_k : D^k \to D$ of R, the function $F : D^k \to D^k$ is a multimorphism of ϕ_R, where

$$F(x_1, x_2, \ldots, x_k) = \langle f_1(x_1, x_2, \ldots, x_k), f_2(x_1, x_2, \ldots, x_k), \ldots, f_k(x_1, x_2, \ldots, x_k) \rangle$$

Furthermore, if $F : D^k \to D^k$ is a multimorphism of a function $\phi : D^m \to \overline{\mathbb{R}}_+$, then each of the k component functions of F is a polymorphism of the relation $\mathrm{Feas}(\phi)$.

Proof: Follows immediately from Definition 4.1 and Definition 4.3 restricted to the special case of crisp functions. □

Proposition 4.10 shows that, in the special case of *crisp* cost functions, a multimorphism can be seen as simply a collection of polymorphisms (which need not be distinct), and a polymorphism can be seen as simply a component function of a multimorphism. Hence the notion of a multimorphism can be viewed as an extension and generalisation of the notion of a polymorphism.

5 A Family of NP-hard Languages

In the remainder of the paper we will use the results obtained above to classify the complexity of a wide range of valued constraint languages with costs in $\overline{\mathbb{R}}_+$. We start by establishing a sufficient condition for such a language to be NP-hard.

Proposition 5.1 *Let Γ be a valued constraint language over a set D, with costs in $\overline{\mathbb{R}}_+$. If there exist $d, d' \in D$, and $\alpha, \beta \in \overline{\mathbb{R}}_+$, with $\alpha < \beta < \infty$, such that the binary function $\phi_{\mathrm{XOR}_\alpha^\beta}$ given by*

$$\phi_{\mathrm{XOR}_\alpha^\beta}(x, y) = \begin{cases} \alpha & \textit{if } \ x \neq y \ \wedge \ x, y \in \{d, d'\} \\ \beta & \textit{if } \ x = y \ \wedge \ x, y \in \{d, d'\} \\ \infty & \textit{otherwise} \end{cases}$$

is expressible over Γ, then $\mathrm{VCSP}(\Gamma)$ *is NP-hard.*

Proof: An assignment to an instance of $\mathrm{VCSP}(\{\phi_{\mathrm{XOR}_\alpha^\beta}\})$ has finite cost if and only if it assigns one of the two values d and d' to all (constrained) variables. Hence we may restrict all variables to these two values. Lemma 7.4 of [10] states that the two-valued problem $\mathrm{VCSP}(\{\phi_{\mathrm{XOR}}\})$ is NP-hard, where ϕ_{XOR} is the Boolean exclusive-or function, as defined in Example 2.10. Since adding a constant to all cost functions, and scaling all costs by a constant factor, does not affect the difficulty of solving a VCSP instance over the valuation structure $\overline{\mathbb{R}}_+$, we conclude that $\mathrm{VCSP}(\{\phi_{\mathrm{XOR}_\alpha^\beta}\})$ is also NP-hard. Hence, by Theorem 3.4, $\mathrm{VCSP}(\Gamma)$ is NP-hard. □

Next we show that the set of multimorphisms of any Boolean language which is shown to be NP-hard using Proposition 5.1 must be very restricted.

Definition 5.2 A function $f : D^k \to D$ is called **essentially unary** if there exists a non-constant unary function $g : D \to D$ and an index $i \in \{1, 2, \dots, k\}$ such that $f(d_1, d_2, \dots, d_k) = g(d_i)$ for all choices of $d_1, d_2, \dots, d_k$.

Definition 5.3 An injective multimorphism in which every component function is essentially unary will be called **trivial**.

Theorem 5.4 *A function $F : \{0,1\}^k \to \{0,1\}^k$ is a multimorphism of the valued Boolean constraint language Γ_{XOR} defined in Example 2.10 if and only if F is trivial.*

Proof: It is straightforward to check that any injective function $F : \{0,1\}^2 \to \{0,1\}^2$ where each component function is essentially unary is a multimorphism of $\Gamma_{\text{XOR}} = \{\phi_{\text{XOR}}\}$.

To establish the converse, let $D = \{0,1\}$ and let $F : D^k \to D^k$ be any multimorphism of ϕ_{XOR}. By Definition 4.3 we have

$$\forall s, t \in D^k, \quad \sum_{i=1}^{k} \phi_{\text{XOR}}(F(s)[i], F(t)[i]) \leq \sum_{i=1}^{k} \phi_{\text{XOR}}(s[i], t[i]).$$

For any pair of tuples s and t, we define the *Hamming distance* between s and t, denoted $H(s,t)$, to be the number of co-ordinate positions at which they differ. We can rewrite the above inequality using Hamming distances to obtain

$$\forall s, t \in D^k, \quad k - H(F(s), F(t)) \leq k - H(s,t),$$

and so

$$\forall s, t \in D^k, \quad H(F(s), F(t)) \geq H(s,t). \tag{4}$$

This implies that F is injective, and hence a bijection from D^k to D^k, so by summing over all elements of D^k we obtain

$$\sum_{s,t \in D^k} H(F(s), F(t)) = \sum_{s,t \in D^k} H(s,t). \tag{5}$$

From Equation 4 and Equation 5 it follows that

$$\forall s, t \in D^k, \quad H(F(s), F(t)) = H(s,t) \tag{6}$$

Now let $\mathbf{0}$ be the all zero k-tuple and define the function $P_F : D^k \to D^k$ by setting

$$P_F(s)[i] = \begin{cases} 1 - s[i] & \text{if} \quad F(\mathbf{0})[i] = 1 \\ s[i] & \text{otherwise} \end{cases}$$

Since $\phi_{\text{XOR}}(a,b) = \phi_{\text{XOR}}(1-a, 1-b)$, we have that

$$\sum_{i=1}^{k} \phi_{\text{XOR}}(P_F(F(s))[i], P_F(F(t))[i]) = \sum_{i=1}^{k} \phi_{\text{XOR}}(F(s)[i], F(t)[i]),$$

so $F \circ P_F$ is a multimorphism of ϕ_{XOR}. By construction, $P_F(F(\mathbf{0})) = \mathbf{0}$, and it follows from Equation 6 (by setting $t = \mathbf{0}$) that $F \circ P_F$ is conservative.

Let t_i be the k-tuple which is zero except at position i. We can re-order the components of the conservative function $F \circ P_F$ to obtain the function F' which fixes each t_i. Now consider a k-tuple s. The function F' is conservative, and by Equation 6 we have that $H(F'(s), t_i) = H(s, t_i)$, for each t_i. It follows that $F'(s)$ has ones exactly where s does, and so F' is the identity function. Hence $F \circ P_F$ simply returns its list of arguments in some fixed order.

Finally, since $F = (F \circ P_F) \circ P_F$, it follows that each component function of F is essentially unary. □

Corollary 5.5 *Let Γ be a valued constraint language over $\{0,1\}$, with costs in $\overline{\mathbb{R}}_+$.*

If the cost function $\phi_{\text{XOR}_\alpha^\beta}$ defined in Proposition 5.1 is expressible in Γ for some $\alpha, \beta \in \overline{\mathbb{R}}_+$, with $\alpha < \beta < \infty$, then every multimorphism of Γ is trivial.

Proof: Follows immediately from Theorem 5.4, Theorem 4.5, and the fact that the set of multimorphisms of any cost function with costs in $\overline{\mathbb{R}}_+$ is unchanged by adding a constant and scaling all values by a constant factor. □

6 Multimorphisms and Tractable Languages

In this section we will present several maximal tractable valued constraint languages with costs in $\overline{\mathbb{R}}_+$. Some of these are translations of known tractable optimisation problems into the VCSP framework, and others are novel tractable classes. In all cases we are able to give a characterisation of the tractable language in terms of a single multimorphism. Hence, in all cases we show that the presence of a certain kind of multimorphism is sufficient to guarantee tractability.

We first make the following observations:

- If Γ is a tractable valued constraint language with costs in $\overline{\mathbb{R}}_+$, then the set of relations $\{\text{Feas}(\phi) \mid \phi \in \Gamma\}$ must be a tractable crisp constraint language. By the results of [6, 26, 28], this implies that each relation $\text{Feas}(\phi)$ must have some fixed set of polymorphisms which guarantees the tractability of this set of relations.
- By Proposition 4.10, we know that if $F : D^k \to D^k$ is a multimorphism of a function ϕ, then each of the k component functions of F is a polymorphism of the relation $\text{Feas}(\phi)$.

Hence, in our search for tractable valued constraint languages with costs in $\overline{\mathbb{R}}_+$ a sensible place to start is by considering those multimorphisms whose component functions are polymorphisms which guarantee tractability. The most straightforward examples of such polymorphisms are constant functions, maximum and minimum functions on ordered sets, majority functions and minority functions [26]; the examples we give in this section are all obtained by combining these simple functions in various ways.

We will show in Section 7 that the examples considered in this section are sufficient to obtain a complete characterisation of the complexity of all valued Boolean constraint languages with costs in $\overline{\mathbb{R}}_+$.

6.1 Constant multimorphisms

The first example we consider is a rather straightforward family of tractable languages, characterised by the presence of a single unary multimorphism with a constant value.

Lemma 6.1 *A cost function ϕ has a unary multimorphism with constant value d if and only if the value of $\phi(d, d, \ldots, d)$ is the smallest value in the range of ϕ.*

Example 6.2 A constant cost function has all possible constant unary multimorphisms. □

Example 6.3 The valued constraint language Γ_{LIN} defined in Example 3.3 has the constant unary multimorphism with value 0. □

Although the proof of tractability for this case is trivial, the proof that every language characterised by a constant multimorphism is a *maximal* tractable language is more interesting, and provides a simple example of the techniques we shall use for other cases.

Theorem 6.4 *Let D be a set, and let $F : D \rightarrow D$ be a constant function.*

1. *The set of functions $\text{Imp}_{\overline{\mathbb{R}}_+}(F)$ is a tractable valued constraint language.*
2. *Any valued constraint language Γ such that $\Gamma \supset \text{Imp}_{\overline{\mathbb{R}}_+}(F)$ is NP-hard.*

Proof: Let d_F be the (constant) value of F.

1. Let ϕ be any function in $\text{Imp}_{\overline{\mathbb{R}}_+}(F)$, and let m be the arity of ϕ. Since F is a multimorphism of ϕ, we have that, for all $d_1, d_2, \ldots, d_m \in D$,

$$\phi(d_F, d_F, \ldots, d_F) \leq \phi(d_1, d_2, \ldots, d_m)$$

   Hence any instance $\mathcal{P}$ in VCSP($\text{Imp}_{\overline{\mathbb{R}}_+}(F)$) has a solution which assigns the value d_F to every variable, so VCSP($\text{Imp}_{\overline{\mathbb{R}}_+}(F)$) is tractable.

2. Now assume that $\Gamma \supset \mathrm{Imp}_{\overline{\mathbb{R}}_+}(F)$, and hence Γ contains a function ϕ of some arity m such that F is not a multimorphism of ϕ. Hence there exist $d_1, d_2, \ldots, d_m \in D$ such that $\phi(d_F, d_F, \ldots, d_F) > \phi(d_1, d_2, \ldots, d_m)$.

   If $\phi(d_F, \ldots, d_F) < \infty$, then set $\mu = (\phi(d_F, \ldots, d_F) - \phi(d_1, \ldots, d_m))/2$, otherwise set $\mu = 1$. Choose some i_0 such that $d_{i_0} \neq d_F$. Now define the functions δ and ψ as follows:

$$\delta(x_1, \ldots, x_m) = \begin{cases} 0 & \text{if } \langle x_1, \ldots, x_m \rangle \in \{\langle d_1, \ldots, d_m \rangle, \langle d_F, \ldots, d_F \rangle\} \\ \infty & \text{otherwise} \end{cases}$$

$$\psi(x_1, x_2, x_3) = \begin{cases} \mu & \text{if } \langle x_1, x_2, x_3 \rangle \in \{\langle d_{i_0}, d_{i_0}, d_{i_0} \rangle, \langle d_{i_0}, d_F, d_F \rangle\} \\ 0 & \text{otherwise} \end{cases}$$

   Note that $\delta, \psi \in \mathrm{Imp}_{\overline{\mathbb{R}}_+}(F) \subset \Gamma$.

   We can now construct the instance $\mathcal{P} \in \mathrm{VCSP}(\Gamma)$ with variables

$$\{X_1, \ldots, X_m, Y_1, \ldots, Y_m, Z_1, \ldots, Z_m\}$$

   and constraints

$$\begin{array}{ll} \langle\langle X_1, \ldots, X_m \rangle, \phi\rangle, & \langle\langle X_1, \ldots, X_m \rangle, \delta\rangle, \\ \langle\langle Y_1, \ldots, Y_m \rangle, \delta\rangle, & \langle\langle Z_1, \ldots, Z_m \rangle, \delta\rangle, \\ \langle\langle X_{i_0}, Y_{i_0}, Z_{i_0} \rangle, \psi\rangle. & \end{array}$$

   If we set $W = \langle Y_{i_0}, Z_{i_0} \rangle$, then it is straightforward to check that

$$\Phi_{\mathcal{P}}^{W}(x, y) = \begin{cases} \phi(d_1, d_2, \ldots, d_m) & \text{if } x \neq y \ \wedge \ x, y \in \{d_F, d_{i_0}\} \\ \mu + \phi(d_1, d_2, \ldots, d_m) & \text{if } x = y \ \wedge \ x, y \in \{d_F, d_{i_0}\} \\ \infty & \text{otherwise} \end{cases}$$

   Hence, by Proposition 5.1, VCSP(Γ) is NP-hard.

□

Example 6.5 Recall from Example 2.9 that the Max-Sat optimisation problem has just three maximal tractable classes, which are identified in [10]. Two of these can be characterised by having a constant function as a multimorphism; these are referred to in [10] as '0-valid' relations, and '1-valid' relations[2]. □

6.2 The multimorphism $\langle \mathrm{Min}, \mathrm{Max} \rangle$

The next example we consider is the family of valued constraint languages over a set D characterised by the presence of a single binary multimorphism, $\langle \mathrm{Min}, \mathrm{Max} \rangle$, where the binary operations Min and Max return the minimum and maximum values with respect to some fixed total ordering of D. These languages include the class of submodular set functions used in economics and operations research [39] (see Example 4.6).

[2]The third tractable class for the Max-Sat problem is discussed in Example 6.8, below.

Lemma 6.6 *Let D be a finite totally ordered set. A function $\phi : D^m \to \overline{\mathbb{R}}_+$ has the multimorphism $\langle \mathrm{Min}, \mathrm{Max} \rangle$ if and only if it satisfies the following two conditions:*

- *ϕ is **finitely submodular**, that is, for all m-tuples s, t, such that $\phi(s), \phi(t) < \infty$, we have that*

$$\phi(\mathrm{Min}(s,t)) + \phi(\mathrm{Max}(s,t)) \leq \phi(s) + \phi(t),$$

where the operations Min *and* Max *are applied co-ordinatewise.*

- $\mathrm{Feas}(\phi)$ *has the polymorphisms* Min *and* Max.

Proof: If ϕ has the multimorphism $\langle \mathrm{Min}, \mathrm{Max} \rangle$, then these two properties follow immediately from Definition 4.3 and Proposition 4.10.

Conversely, if ϕ is finitely submodular, then it satisfies Equation 1 of Definition 4.3 for all choices of t_1 and t_2. □

The second condition in Lemma 6.6 implies that the set of m-tuples on which ϕ is finite is a sublattice of the set of all m-tuples, where the lattice operations are the operations Min and Max applied co-ordinatewise. Theorem 49.2 of [45] states that any real-valued submodular function defined on such a sublattice can be extended to a submodular function on the full lattice. Hence, by Lemma 6.6, any function with the multimorphism $\langle \mathrm{Min}, \mathrm{Max} \rangle$ can be expressed as the sum of a finite-valued submodular function, and a crisp function ϕ_R associated with a relation R which has the polymorphisms Min and Max.

Theorem 6.7 *Let D be a finite totally ordered set, and let $F : D^2 \to D^2$ be the function defined by $F(d, d') = \langle \mathrm{Min}(d, d'), \mathrm{Max}(d, d') \rangle$.*

1. *The set of functions $\mathrm{Imp}_{\overline{\mathbb{R}}_+}(F)$ is a tractable valued constraint language.*
2. *Any valued constraint language Γ such that $\Gamma \supset \mathrm{Imp}_{\overline{\mathbb{R}}_+}(F)$ is NP-hard.*

Proof: Assume for simplicity that $D = \{0, 1, 2, \ldots, |D| - 1\}$ with the usual ordering.

1. To establish the tractability of the set of functions in $\mathrm{Imp}_{\overline{\mathbb{R}}_+}(F)$, we show that this problem can be reduced to the problem of minimising a real-valued submodular set function [39] over a special family of sets known as a ring family [44]. This problem can then be solved in polynomial time using an algorithm due to Schrijver [44].

   Let $\mathcal{P} = (V, D, C, \overline{\mathbb{R}}_+)$ be any instance of $\mathrm{VCSP}(\mathrm{Imp}_{\overline{\mathbb{R}}_+}(F))$. By Lemma 6.6, the feasibility relation corresponding to each constraint in $\mathcal{P}$ has the polymorphisms Min and Max. Hence the standard constraint satisfaction problem instance with these relations as crisp constraints can be solved in polynomial time, using the results of [29]. But this means that we can determine in polynomial-time whether or not there is an assignment for $\mathcal{P}$ with finite cost.

If every assignment for $\mathcal{P}$ has infinite cost, then we can return an arbitrary assignment as a solution, and we are done.

Otherwise, we define the set $Q = D \times V$, and associate each assignment s for $\mathcal{P}$ that has finite cost with a subset Q_s of Q, defined as follows:

$$Q_s = \{\langle d, v\rangle \in Q \mid v \in V \wedge d \leq s(v)\}$$

Now, it is straightforward to check that for any pair of assignments s and t with finite cost we have

$$\begin{aligned} Q_s \cup Q_t &= Q_{\mathrm{Max}(s,t)} \\ Q_s \cap Q_t &= Q_{\mathrm{Min}(s,t)}. \end{aligned}$$

Hence the subsets of Q associated with the assignments of finite cost form a collection $\mathcal{C}$ which is closed under union and intersection. Such a collection is referred to in [44] as a *ring family*.

Finally, we define the real-valued function ψ on $\mathcal{C}$, by setting

$$\psi(Q_s) = \mathrm{Cost}_{\mathcal{P}}(s)$$

Note that, since F is a multimorphism of every cost function in $\mathcal{P}$, for all $S, T \in \mathcal{C}$ we have

$$\psi(S \cup T) + \psi(S \cap T) \leq \psi(S) + \psi(T)$$

Hence, ψ is a real-valued submodular set function defined on the finite ring family $\mathcal{C}$, and so can be minimised in polynomial-time, using the algorithm described in [44]. The output of this algorithm is an element Q_s of $\mathcal{C}$ corresponding to a solution s to $\mathcal{P}$, so the problem is tractable.

2. Now assume that $\Gamma \supset \mathrm{Imp}_{\overline{\mathbb{R}}_+}(F)$, and hence Γ contains a function ϕ of some arity m such that F is not a multimorphism of ϕ. Hence, there exist $s, s' \in D^m$ such that

$$\phi(\mathrm{Min}(s, s')) + \phi(\mathrm{Max}(s, s')) > \phi(s) + \phi(s').$$

where the operators Min and Max are applied coordinatewise to the tuples s and s'.

It follows that we can find indexes i and j for which $s[i] > s'[i]$ and $s[j] < s'[j]$.

We define an m-ary function δ which takes the value 0 on the tuples $s, s', \mathrm{Max}(s, s')$ and $\mathrm{Min}(s, s')$, and ∞ in all other cases. Note that $\delta \in \mathrm{Imp}_{\overline{\mathbb{R}}_+}(F) \subset \Gamma$.

Define λ and μ as follows:

$$\begin{array}{rcl} \lambda & = & \min(\phi(\mathrm{Min}(s,s')), \phi(s)+\phi(s')+1) \\ \mu & = & \min(\phi(\mathrm{Max}(s,s')), \phi(s)+\phi(s')+1) \end{array}$$

It is straightforward to check that $\phi(s)+\phi(s') < \lambda+\mu < \infty$.

Now define the binary functions

$$\begin{array}{rcl} \zeta(x,y) & = & \begin{cases} \mu & \text{if} \quad (x,y)=(0,s'[i]) \\ \lambda & \text{if} \quad (x,y)=(1,s[i]) \\ \infty & \text{otherwise} \end{cases} \\ \kappa(x,y) & = & \begin{cases} 0 & \text{if} \quad (x,y)=(s[j],0) \\ \phi(s')+1 & \text{if} \quad (x,y)=(s[j],1) \\ \phi(s)+1 & \text{if} \quad (x,y)=(s'[j],0) \\ 0 & \text{if} \quad (x,y)=(s'[j],1) \\ \infty & \text{otherwise} \end{cases} \end{array}$$

Note that $\zeta, \kappa \in \mathrm{Imp}_{\overline{\mathbb{R}}_+}(F) \subset \Gamma$.

We can now construct the instance $\mathcal{P} \in \mathrm{VCSP}(\Gamma)$ with variables

$$\{X, Y, V_1, \ldots, V_m, W_1, \ldots, W_m, \}$$

and constraints

$$\begin{array}{ll} \langle\langle V_1,\ldots,V_m\rangle, \delta\rangle, & \langle\langle W_1,\ldots,W_m\rangle, \delta\rangle, \\ \langle\langle V_1,\ldots,V_m\rangle, \phi\rangle, & \langle\langle W_1,\ldots,W_m\rangle, \phi\rangle, \\ \langle\langle W_i, X\rangle, \kappa\rangle, & \langle\langle V_j, Y\rangle, \kappa\rangle, \\ \langle\langle X, V_i\rangle, \zeta\rangle, & \langle\langle Y, W_j\rangle, \zeta\rangle. \end{array}$$

If we set $W = \langle X, Y\rangle$, then it is straightforward to check that

$$\Phi^W_{\mathcal{P}}(x,y) = \begin{cases} \lambda+\mu+\phi(s)+\phi(s') & \text{if} \quad x \neq y \ \wedge \ x,y \in \{0,1\} \\ \lambda+\mu+\lambda+\mu & \text{if} \quad x = y \ \wedge \ x,y \in \{0,1\} \\ \infty & \text{otherwise} \end{cases}$$

Hence, by Proposition 5.1, VCSP(Γ) is NP-hard.

□

Example 6.8 Recall from Example 2.9 that the MAX-SAT optimisation problem has just three maximal tractable classes, which are identified in [10]. Two of these can be characterised by having a constant multimorphism (see Example 6.5). The third can be characterised by having the multimorphism $\langle \mathrm{Min}, \mathrm{Max}\rangle$; this class is referred to in [10] as the class of '2-monotone' relations, where it is defined as the class of relations definable by a logical expression of the form $(x_1 \wedge x_2 \wedge \cdots \wedge x_p) \vee (\overline{y_1} \wedge \overline{y_2} \wedge \cdots \overline{y_q})$ (where the x and y variables are not necessarily distinct). □

Example 6.9 It follows from Lemma 4.9 and Example 4.8 that every *unary* function has the multimorphism $\langle \text{Min}, \text{Max} \rangle$. $\square$

Example 6.10 Let $D = \{0, 1, \ldots, M\}$ be a set of integers. It follows from Example 6.9 and Theorem 4.5 that the language Γ_{LIN} defined in Example 3.3, consisting of all functions on D defined by linear expressions with positive integer coefficients, also has the multimorphism $\langle \text{Min}, \text{Max} \rangle$. $\square$

Example 6.11 A function $\phi : \{0, 1\}^m \to \mathbb{R}$ is called a **pseudo-Boolean function** [3]. It is straightforward to check from the table of values that the function ϕ defined by $\phi(x, y) = x(1 - y)$ has the multimorphism $\langle \text{Min}, \text{Max} \rangle$. It follows from Example 6.10 and Theorem 4.5 that the language Γ consisting of non-negative functions on $\{0, 1\}$ defined by expressions of the form $a_0 + \sum_i a_i x_i - \sum_{i,j} a_{ij} x_i x_j$, where the a_i and a_{ij} are non-negative integers, also has the multimorphism $\langle \text{Min}, \text{Max} \rangle$, and so is tractable by Theorem 6.7. $\square$

Example 6.12 It was shown in Example 2.6 that the Minimum k-Terminal Cut problem can be formulated as an instance of VCSP(Γ_k) for a language Γ_k consisting of crisp unary constraints and the cost function $\phi_{\text{EQ}} : \{0, 1, \ldots, k\}^2 \to \overline{\mathbb{R}}_+$ defined in Example 2.6.

In the special case when $k = 2$, it is straightforward to verify that the cost function ϕ_{EQ} has the multimorphism $\langle \text{Min}, \text{Max} \rangle$. Using this fact, and Example 6.9, Theorem 6.7 implies that the Min-Cut problem can be solved in polynomial time. (Compare with Example 2.11.) $\square$

Example 6.13 It follows immediately from Definition 4.3 that a *binary* function $\phi : D^2 \to \overline{\mathbb{R}}_+$ has the multimorphism $\langle \text{Min}, \text{Max} \rangle$ if and only if, for all $u, v, x, y \in D$, with $u < x$ and $v < y$, we have $\phi(u, v) + \phi(x, y) \leq \phi(u, y) + \phi(x, v)$.

Using this observation, it is straightforward to check that for any finite set of real values D the following *binary* functions all have the multimorphism $\langle \text{Min}, \text{Max} \rangle$, and hence any VCSP instance involving constraints with cost functions of these forms is tractable.

$$\begin{aligned}
\delta(x, y) &= \begin{cases} 0 & \text{if} \quad ax \leq by + c \text{ (for positive constants } a, b, c) \\ \infty & \text{otherwise} \end{cases} \\
\eta(x, y) &= \sqrt{x^2 + y^2} \\
\zeta(x, y) &= |x - y|^r \text{ (for } r \geq 1)
\end{aligned}$$

Using these observations, and Example 6.9, we conclude that the discrete optimisation problem described in Example 1.1 can be solved in polynomial time. (A more specialised algorithm for binary soft constraints of these kinds, which runs in cubic time, is given in our previous paper [7].) $\square$

6.3 The multimorphism $\langle \mathrm{Max}, \mathrm{Max} \rangle$

The next example we consider is the family of valued constraint languages over a set D characterised by the presence of a single binary multimorphism, $\langle \mathrm{Max}, \mathrm{Max} \rangle$, where the binary operation Max returns the maximum value with respect to some fixed total ordering of D. These languages generalise the crisp "max-closed" constraint languages introduced and shown to be tractable in [29].

We first show that any function with values in $\overline{\mathbb{R}}_+$ which has the multimorphism $\langle \mathrm{Max}, \mathrm{Max} \rangle$ satisfies some simple conditions. For any tuples u, v over an ordered set D, we will write $u \leq v$ if and only if $u[i] \leq v[i]$ for each co-ordinate position i.

Lemma 6.14 *A function $\phi : D^k \to \overline{\mathbb{R}}_+$ has the multimorphism $F : D^2 \to D^2$, where $F(d, d') = \langle \mathrm{Max}(d, d'), \mathrm{Max}(d, d') \rangle$ if and only if it satisfies the following two conditions:*

- *ϕ is **finitely antitone**, that is, for all tuples u, v with $\phi(u), \phi(v) < \infty$,*
$$u \leq v \Rightarrow \phi(u) \geq \phi(v).$$
- $\mathrm{Feas}(\phi)$ *has the polymorphism* Max.

Proof: If ϕ has the multimorphism F, then for all tuples u, v we have $\phi(u) + \phi(v) \geq 2\phi(\mathrm{Max}(u, v))$, which implies that both conditions hold.

Conversely, if ϕ does not have the multimorphism F, then there exist tuples u, w such that $\phi(u) + \phi(w) < 2\phi(\mathrm{Max}(u, w))$. Hence, without loss of generality, we may assume that $\phi(u) < \phi(\mathrm{Max}(u, w))$. Setting $v = \mathrm{Max}(u, w)$ we get $u < v$ and $\phi(u) < \phi(v)$. If $\phi(v) < \infty$ then the first condition in the lemma does not hold, and if $\phi(v) = \infty$, then the second condition fails to hold. □

By Lemma 6.14, any function with the multimorphism $\langle \mathrm{Max}, \mathrm{Max} \rangle$ can be expressed as the sum of a finite-valued antitone function, and a crisp function ϕ_R associated with a relation R which has the polymorphism Max.

Theorem 6.15 *Let D be a totally ordered finite set, and let $F : D^2 \to D^2$ be the function defined by $F(d, d') = \langle \mathrm{Max}(d, d'), \mathrm{Max}(d, d') \rangle$.*

1. *The set of functions $\mathrm{Imp}_{\overline{\mathbb{R}}_+}(F)$ is a tractable valued constraint language.*
2. *Any valued constraint language Γ such that $\Gamma \supset \mathrm{Imp}_{\overline{\mathbb{R}}_+}(F)$ is NP-hard.*

Proof: Assume for simplicity that $D = \{0, 1, 2, \ldots, |D| - 1\}$ with the usual ordering.

1. To establish the tractability of $\mathrm{Imp}_{\overline{\mathbb{R}}_+}(F)$, we will give an explicit polynomial-time algorithm for VCSP(Γ) for any fixed finite subset of $\mathrm{Imp}_{\overline{\mathbb{R}}_+}(F)$.

   Let Γ be a finite subset of $\mathrm{Imp}_{\overline{\mathbb{R}}_+}(F)$, and let $\mathcal{P} = (V, D, C, \overline{\mathbb{R}}_+)$ be any instance of VCSP(Γ). To each constraint $c = \langle \sigma, \phi \rangle \in C$ we can associate a crisp constraint $\bar{c} = \langle \sigma, \mathrm{Feas}(\phi) \rangle$ which allows precisely those tuples of values t for which $\phi(t) < \infty$. We can then establish arc consistency [37] in the constraint satisfaction problem formed by these associated crisp constraints. This is done by successively removing values from the domain of each variable if they are *unsupported*, that is, they cannot be extended to compatible values for all the other variables in the scope of each constraint containing v. Since Γ is finite, the arity of the constraint relations is bounded, so arc-consistency can be achieved in polynomial time [37].

For each variable v, let D_v be the domain of v after establishing arc consistency. If any of these domains are empty, then any assignment for $\mathcal{P}$ has cost ∞, and so any assignment is a solution. Otherwise, let $\bar{d}_v$ be the largest supported value for variable v. These values can be computed in polynomial time

By Lemma 6.14, each constraint of $\mathcal{P}$ is finitely antitone, so assigning $\bar{d}_v$ to each variable v is an optimal solution to $\mathcal{P}$.

2. Now assume that $\Gamma \supset \mathrm{Imp}_{\overline{\mathbb{R}}_+}(F)$, and hence Γ contains a function ϕ of some arity m such that F is not a multimorphism of ϕ. Hence, there exist $s, s' \in D^m$ such that

$$2\phi(\mathrm{Max}(s, s')) > \phi(s) + \phi(s'),$$

where the operator Max is applied coordinatewise to the tuples s and s'.

Set $s'' = \mathrm{Max}(s, s')$. We have to consider two cases depending on whether or not $\phi(s'')$ has cost ∞.

Case 1: $\phi(s'') < \infty$.

Without loss of generality we may assume that $\phi(s'') > \phi(s)$. In this case there must be at least one index i_0 for which $s[i_0] < s''[i_0]$.

We define an m-ary function δ which takes the value 0 on the tuples s and s'', and ∞ in all other cases. Note that $\delta \in \mathrm{Imp}_{\overline{\mathbb{R}}_+}(F) \subset \Gamma$.

We also define the binary function ψ as follows

$$\psi(x, y) = \begin{cases} 2\phi(s'') & \text{if } \langle x, y \rangle = \langle s[i_0], s[i_0] \rangle \\ 2\phi(s) & \text{if } \langle x, y \rangle \in \{\langle s[i_0], s''[i_0] \rangle, \langle s''[i_0], s[i_0] \rangle, \langle s''[i_0], s''[i_0] \rangle\} \\ \infty & \text{otherwise} \end{cases}$$

Note that $\psi \in \mathrm{Imp}_{\overline{\mathbb{R}}_+}(F) \subset \Gamma$.

We can now construct the instance $\mathcal{P} \in \mathrm{VCSP}(\Gamma)$ with variables

$$\{X_1, \ldots, X_m, Y_1, \ldots, Y_m\}$$

and constraints

$$\begin{array}{ll} \langle\langle X_1, \ldots, X_m \rangle, \phi \rangle, & \langle\langle Y_1, \ldots, Y_m \rangle, \phi \rangle, \\ \langle\langle X_1, \ldots, X_m \rangle, \delta \rangle, & \langle\langle Y_1, \ldots, Y_m \rangle, \delta \rangle, \\ \langle\langle X_{i_0}, Y_{i_0} \rangle, \psi \rangle. & \end{array}$$

If we set $W = \langle X_{i_0}, Y_{i_0} \rangle$, then it is straightforward to check that

$$\Phi_{\mathcal{P}}^{W}(x,y) = \begin{cases} \phi(s'') + 3\phi(s) & \text{if } x \neq y \ \wedge \ x,y \in \{s[i_0], s''[i_0]\} \\ 2(\phi(s'') + \phi(s)) & \text{if } x = y \ \wedge \ x,y \in \{s[i_0], s''[i_0]\} \\ \infty & \text{otherwise} \end{cases}$$

Hence, by Proposition 5.1, VCSP(Γ) is NP-hard.

Case 2: $\phi(s'') = \infty$.

Consider the relation Feas(ϕ) containing precisely those tuples for which the value of ϕ is finite. Since, by hypothesis, $\phi(s), \phi(s') < \infty$ and $\phi(s'') = \infty$, we have $s, s' \in$ Feas(ϕ) and $s'' =$ Max$(s, s') \notin$ Feas(ϕ). That is, the relation Feas(ϕ) does not have the polymorphism Max.

Now let L_{Max} be the crisp constraint language over D consisting of all relations which do have the polymorphism Max. It was shown in [29] that L_{Max} is a *maximal* tractable language, and hence the class of crisp constraint satisfaction problems with constraint relations chosen from $L_{\text{Max}} \cup \{\text{Feas}(\phi)\}$ is NP-complete. By representing these crisp constraints as valued constraints with the corresponding feasibility functions as cost functions, as described in Example 2.4, we can obtain a polynomial-time reduction from this problem to the decision problem associated with VCSP(Γ). Hence VCSP(Γ) is NP-hard.

□

Example 6.16 Let $D = \{0, 1, 2, \ldots, M\}$ be a subset of the integers, and let Γ_{AT} be the set of all antitone cost functions over D with costs in $\overline{\mathbb{R}}_+ \setminus \{\infty\}$. These cost functions can be used to express a preference for larger values of their arguments. For example, the ternary function ϕ, defined by $\phi(x, y, z) = 3M - \sqrt{x^2 + y^2 + z^2}$, can be used to select a point in D^3 which is as far as possible from the origin. By Lemma 6.14, Γ_{AT} has the multimorphism $\langle \text{Max}, \text{Max} \rangle$, and hence is tractable by Theorem 6.15 □

Example 6.17 The constraint programming language CHIP [47] incorporates a number of constraint solving techniques for arithmetic and other constraints. In particular it provides a constraint solver for a restricted class of crisp constraints over natural numbers, referred to as *basic constraints*. These basic constraints are of two kinds which are referred to as "domain constraints" and "arithmetic constraints". The domain constraints are unary constraints which restrict the value of a variable to some specified finite subset of the natural numbers. The arithmetic constraints are unary or binary constraints which have one of the following forms:

$$\begin{array}{rcl} aX & \neq & b \\ aX & = & bY + c \\ aX & \leq & bY + c \\ aX & \geq & bY + c \end{array}$$

where variables are represented by upper-case letters, and constants by lower case letters. All constants are non-negative, and a is non-zero.

If we represent these crisp constraints as valued constraints with the corresponding feasibility functions as cost functions, as described in Example 2.4, then it is easy to verify that they all have the multimorphism $\langle \mathrm{Max}, \mathrm{Max} \rangle$, and hence form a tractable valued constraint language, by Theorem 6.15.

Moreover, this tractable language can be extended, as shown in [29], to also include the feasibility functions of the following non-binary relations, which also have the multimorphism $\langle \mathrm{Max}, \mathrm{Max} \rangle$.

$$\begin{array}{rcl} a_1X_1 + a_2X_2 + \ldots + a_rX_r & \geq & bY + c \\ aX_1X_2 \ldots X_r & \geq & bY + c \\ (a_1X_1 \geq b_1) \vee (a_2X_2 \geq b_2) \vee \ldots \vee (a_rX_r \geq b_r) & \vee & (aY \leq b) \end{array}$$

The tractable language consisting of all crisp constraint functions with the multimorphism $\langle \mathrm{Max}, \mathrm{Max} \rangle$ will be denoted Γ_{MC}. $\square$

Example 6.18 By Lemma 6.14 and Theorem 3.4, we can combine the tractable languages Γ_{AT} (defined in Example 6.16) and Γ_{MC} (defined in Example 6.17) to obtain the much larger tractable language $(\Gamma_{\mathrm{AT}} \cup \Gamma_{\mathrm{MC}})^*$. In fact, we have $\mathrm{Imp}_{\overline{\mathbb{R}}_+}(\langle \mathrm{Max}, \mathrm{Max} \rangle) = (\Gamma_{\mathrm{AT}} \cup \Gamma_{\mathrm{MC}})^*$.

This larger tractable language includes functions such as the binary function $\phi : D^2 \to \overline{\mathbb{R}}_+$ defined by

$$\phi(x, y) = \begin{cases} (M - x)(M - y) & \text{if } x < y \\ \infty & \text{if } x \geq y \end{cases}$$

This function can be expressed as the sum of the antitone function $\psi(x, y) = (M - x)(M - y)$, and the function $\phi_{R_<}$, where $R_< = \{\langle x, y \rangle \mid x < y\}$. It can be used to express a preference for larger values for x, y provided $x < y$. $\square$

6.4 Majority and minority multimorphisms

The next example we consider is the family of valued constraint languages over a set D characterised by the presence of a single ternary multimorphism, $\langle F_1, F_2, F_3 \rangle$, where each component function F_i is a *majority* operation, defined as follows.

Definition 6.19 A function $f : D^3 \to D$ is called a **majority** operation if, for all $x, y \in D$,

$$f(x, x, y) = f(x, y, x) = f(y, x, x) = x.$$

Languages with a multimorphism of this kind can be shown to be essentially crisp, and hence their complexity can be determined by using techniques developed for the standard constraint satisfaction problem with crisp constraints. In fact, problems involving such languages can be viewed as a generalisation of the standard tractable 2-SATISFIABILITY problem to larger finite domains.

Proposition 6.20 *Any valued constraint language with costs in $\overline{\mathbb{R}}_+$ which has a multimorphism $\langle F_1, F_2, F_3 \rangle$, where each F_i is a majority operation, is an essentially crisp language, and is tractable.*

Proof: Let Γ be a valued constraint language which has the multimorphism $\langle F_1, F_2, F_3 \rangle$, and let ϕ be a k-ary cost function in Γ.

If each F_i is a majority operation, then it follows from Definition 6.19 and Definition 4.3 that for all $x, y \in D^k$, $3\phi(x) \leq \phi(x) + \phi(x) + \phi(y)$ and $3\phi(y) \leq \phi(y) + \phi(y) + \phi(x)$. Hence, if both $\phi(x)$ and $\phi(y)$ are finite, then we have $\phi(x) \leq \phi(y)$ and $\phi(y) \leq \phi(x)$, so they must be equal, which means that ϕ is essentially crisp.

Furthermore, for each $\phi \in \Gamma$, the relation $\mathrm{Feas}(\phi)$ has the polymorphism F_1, which is a majority operation, so it follows from Theorem 5.7 of [26] that $\mathrm{VCSP}(\Gamma)$ is tractable. $\square$

Similar arguments can be used for *minority* operations, defined as follows:

Definition 6.21 A function $f : D^3 \to D$ is called a **minority** operation if, for all $x, y \in D$,

$$f(x, x, y) = f(x, y, x) = f(y, x, x) = y.$$

Proposition 6.22 *Any valued constraint language with costs in $\overline{\mathbb{R}}_+$ which has a multimorphism $\langle F_1, F_2, F_3 \rangle$, where each F_i is a minority operation, is an essentially crisp language, and is tractable.*

Proof: Let Γ be a valued constraint language which has the multimorphism $\langle F_1, F_2, F_3 \rangle$, and let ϕ be a k-ary cost function in Γ.

If each F_i is a minority operation, then for all $x, y \in D^k$, we have $3\phi(x) \leq \phi(x) + \phi(y) + \phi(y)$ and $3\phi(y) \leq \phi(y) + \phi(x) + \phi(x)$. Hence, if both $\phi(x)$ and $\phi(y)$ are finite, then we have $\phi(x) \leq \phi(y)$ and $\phi(y) \leq \phi(x)$, so they must be equal, which means that ϕ is essentially crisp.

Furthermore, for each $\phi \in \Gamma$, the relation $\mathrm{Feas}(\phi)$ has the polymorphism F_1, which is a minority operation, and hence a *Mal'tsev operation* (see [13]), so it follows from Theorem 1 of [13] that $\mathrm{VCSP}(\Gamma)$ is tractable. $\square$

6.5 The multimorphism $\langle \mathrm{Mjrty}_1, \mathrm{Mjrty}_2, \mathrm{Mnrty}_3 \rangle$

The final example we consider is the valued constraint language with costs in $\overline{\mathbb{R}}_+$ which is characterised by the presence of the single ternary multimorphism

$\langle \text{Mjrty}_1, \text{Mjrty}_2, \text{Mnrty}_3 \rangle$, where

$$\text{Mjrty}_1(x,y,z) = \begin{cases} y & \text{if } \quad y = z \\ x & \text{otherwise.} \end{cases}$$

$$\text{Mjrty}_2(x,y,z) = \begin{cases} x & \text{if } \quad x = z \\ y & \text{otherwise.} \end{cases}$$

$$\text{Mnrty}_3(x,y,z) = \begin{cases} x & \text{if } \quad y = z \ \wedge \ z \neq x \\ y & \text{if } \quad x = z \ \wedge \ z \neq y \\ z & \text{otherwise.} \end{cases}$$

Note that Mjrty_1 and Mjrty_2 are both majority operations[3] and Mnrty_3 is a minority operation (see Definitions 6.19 and 6.21).

We will show in this section that any function taking values in $\overline{\mathbb{R}}_+$ which has this multimorphism has a very simple form. The proof of this fact is rather involved, but we include it here largely because the result turns out to be essential for the complete classification of the Boolean case in Section 7. Despite the simplicity of the associated constraint language, we will show that this multimorphism again defines a maximal tractable class.

We first need a technical lemma. For any m-tuple s over a set D, we will write $s[i \leftarrow d]$ to denote the tuple with $d \in D$ **substituted** at position i. In other words, $s[i \leftarrow d]$ is the m-tuple which is identical to s except (possibly) at position i, where it is equal to d.

Lemma 6.23 *A function $\phi : D^m \to \overline{\mathbb{R}}_+$ can be expressed as a sum of unary functions if and only if, for all tuples $s, t \in D^m$, and all $i = 1, \ldots, m$ we have that*

$$\phi(s) + \phi(t) = \phi(s[i \leftarrow t[i]]) + \phi(t[i \leftarrow s[i]]). \tag{7}$$

Proof: Suppose that ϕ can be expressed as a sum of unary functions. This means there exist $\phi_1, \ldots, \phi_m$ such that, for all tuples $s = \langle s_1, \ldots, s_m \rangle$ and $t = \langle t_1, \ldots, t_m \rangle$,

$$\phi(s) + \phi(t) = \sum_{i=1}^{m} (\phi_i(s_i) + \phi_i(t_i))$$

By rearranging the terms in the summation we get Equation 7.

Conversely, suppose that ϕ satisfies Equation 7. We will now show that this implies that ϕ can be expressed as a sum of unary functions.

Let $s_0 = \langle s_{01}, \ldots, s_{0m} \rangle$ be an m-tuple on which ϕ achieves its minimum cost, that is

$$\forall s \in D^m, \quad \phi(s_0) \leq \phi(s). \tag{8}$$

If $\phi(s_0) = \infty$ then ϕ never takes a finite value so $\phi(x_1, \ldots, x_m) = \sum_{i=1}^{m} \zeta(x_i)$ where $\zeta(x) = \infty$ and the result holds. So we may assume that $\phi(s_0) < \infty$.

For $i = 1, \ldots, m$, let μ_i be the unary cost function defined by

$$\mu_i(x) = \min\{\phi(x_1, \ldots, x_m) \mid x_i = x\}.$$

[3]The operation Mjrty_1 is sometimes known as the *dual discriminator* operation [46].

and for each $x \in D$ choose a witness $w_i^x \in D^m$ such that $w_i^x[i] = x$, and $\phi(w_i^x) = \mu_i(x)$.

Note that for all tuples $\langle x_1, \ldots, x_m \rangle$ with $x_i = x$, we have

$$\mu_i(x) \leq \phi(x_1, \ldots, x_m). \tag{9}$$

We now have, for all $x \in D$,

$$\begin{array}{rrcll}
 & \mu_i(x) + \phi(s_0) & = & \phi(w_i^x) + \phi(s_0) & \text{by choice of } w_i^x \\
 & & = & \phi(w_i^x[i \leftarrow s_{0i}]) + \phi(s_0[i \leftarrow w_i^x[i]]) & \text{by Equation 7} \\
 & & = & \phi(w_i^x[i \leftarrow s_{0i}]) + \phi(s_0[i \leftarrow x]) & \text{by choice of } w_i^x \\
 & & \geq & \phi(s_0) + \phi(s_0[i \leftarrow x]) & \text{by Equation 8} \\
\text{so} & \mu_i(x) & \geq & \phi(s_0[i \leftarrow x]) & \text{cancelling } \phi(s_0) < \infty \\
\text{but} & \mu_i(x) & \leq & \phi(s_0[i \leftarrow x]) & \text{by Equation 9} \\
\text{and so} & \mu_i(x) & = & \phi(s_0[i \leftarrow x]) &
\end{array}$$

Now consider an arbitrary tuple $s = \langle x_1, \ldots, x_m \rangle$. By applying Equation 7 $m - 1$ times we obtain:

$$\phi(s) + (m-1)\phi(s_0) = \sum_{i=1}^{m} \phi(s_0[i \leftarrow x_i]) = \sum_{i=1}^{m} \mu_i(x_i)$$

Equation 8 ensures that choosing $\phi_i(x) = \mu_i(x) - \phi(s_0), i = 2, \ldots, m$ is well defined. Finally, choosing $\phi_1(x) = \mu_1(x)$ gives the result. $\square$

Using this result we now show that any function which has the multimorphism $\langle \mathrm{Mjrty}_1, \mathrm{Mjrty}_2, \mathrm{Mnrty}_3 \rangle$ can be expressed as a sum of unary functions and binary functions of the following kind.

Definition 6.24 Let D be a set, and Ω a valuation structure. A crisp binary function $\phi : D^2 \to \Omega$ will be called a **permutation restriction** if

$$\forall x \in D, \quad |\{y \mid \phi(x,y) = 0\}| \leq 1 \quad \text{and} \quad |\{y \mid \phi(y,x) = 0\}| \leq 1.$$

Theorem 6.25 *Let D be a finite set, and let $F : D^3 \to D^3$ be the function defined by $F(x,y,z) = \langle \mathrm{Mjrty}_1(x,y,z), \mathrm{Mjrty}_2(x,y,z), \mathrm{Mnrty}_3(x,y,z) \rangle$.*

A k-ary function ϕ belongs to the set $\mathrm{Imp}_{\overline{\mathbb{R}}_+}(F)$ if and only if it can be expressed as a sum of unary functions and permutation restrictions.

Proof: By Theorem 4.5, to show that any function which can be expressed as the sum of unary functions and permutation restrictions is in $\mathrm{Imp}_{\overline{\mathbb{R}}_+}(F)$ it is sufficient to show that all unary functions and all permutation restrictions are in $\mathrm{Imp}_{\overline{\mathbb{R}}_+}(F)$.

Since F is conservative, we know by Lemma 4.9 that F is a multimorphism of all unary functions.

Now let π be an arbitrary permutation restriction, and consider the arbitrary triples $t_1, t_2 \in D^3$. If any $\pi(t_1[i], t_2[i]) = \infty$ then F trivially satisfies the inequality $\oplus_{i=1}^{3} \pi(F(t_1)[i], F(t_2)[i]) \leq \oplus_{i=1}^{3} \pi(t_1[i], t_2[i])$, so consider the case where each $\pi(t_1[i], t_2[i]) < \infty$. In this case each $t_2[i]$ is determined by the corresponding $t_1[i]$, because π is a permutation restriction. Suppose that two of the pairs $\langle t_1[i], t_2[i]\rangle$ are equal, say they are both $\langle p, q\rangle$, and that the third pair is $\langle r, s\rangle$. Then, by the definition of F, we have that $\langle F(t_1)[1], F(t_2)[1]\rangle = \langle F(t_1)[2], F(t_2)[2]\rangle = \langle p, q\rangle$ and that $\langle F(t_1)[3], F(t_2)[3]\rangle = \langle r, s\rangle$, so F again satisfies the inequality $\oplus_{i=1}^{3} \pi(F(t_1)[i], F(t_2)[i]) \leq \oplus_{i=1}^{3} \pi(t_1[i], t_2[i])$ (with equality). The only remaining case to consider is when each pair $\langle t_1[i], t_2[i]\rangle$ is distinct, but in this case the definition of F gives $\langle F(t_1)[i], F(t_2)[i]\rangle = \langle t_1[i], t_2[i]\rangle, i = 1, 2, 3$, and so again $\oplus_{i=1}^{3} \pi(F(t_1)[i], F(t_2)[i]) = \pi(t_1[i], t_2[i])$. Hence F is a multimorphism of any permutation restriction.

Conversely, suppose that ϕ is a k-ary function in $\mathrm{Imp}_{\overline{\mathbb{R}}_+}(F)$. In the remainder of the proof we shall establish that ϕ can be expressed as a sum of unary functions and permutation restrictions.

Consider the k-ary relation $\mathrm{Feas}(\phi)$. It follows from Proposition 4.10 that $\mathrm{Feas}(\phi)$ must have the three polymorphisms $\mathrm{Mjrty}_1, \mathrm{Mjrty}_2$ and Mnrty_3. Any relation with a majority operation (such as Mjrty_1) as a polymorphism is known to be *decomposable* into its binary projections [25, 46]. This means that $\langle x_1, \ldots, x_k\rangle \in \mathrm{Feas}(\phi)$ exactly when

$$\forall i, j \in \{1, \ldots, k\}, \quad \langle x_i, x_j\rangle \in R_{ij},$$

where

$$R_{ij} = \{\langle x_i, x_j\rangle \mid \exists \langle x_1, x_2, \ldots, x_k\rangle \in \mathrm{Feas}(\phi)\}.$$

Furthermore, polymorphisms are preserved under taking projections [26], so each of the binary relations R_{ij} also has the three polymorphisms $\mathrm{Mjrty}_1, \mathrm{Mjrty}_2$ and Mnrty_3.

Binary relations with the polymorphism Mjrty_1 have previously been characterised [46], and any such relation is known to have one of the following forms:

- $\mathrm{Feas}(\mu_1 + \mu_2)$, where μ_1, μ_2 are unary functions;
- $\mathrm{Feas}(\pi)$, where π is a permutation restriction;
- $\{\langle x, y\rangle \in D_1 \times D_2 \mid (x = d_1) \vee (y = d_2)\}$, for some $d_1, d_2 \in D$, and some $D_1, D_2 \subseteq D$ with $|D_1| > 1$ and $|D_2| > 1$.

Of these three, it is straightforward to check that only the first two have Mnrty_3 as a polymorphism. Therefore ϕ can be expressed as a sum of functions of the following form:

$$\phi(x_1, \ldots, x_k) = \psi(x_1, \ldots, x_k) + \sum_{i \in I} \pi_i(x_{a_i}, x_{b_i}) + \sum_{j \in J} \mu_j(x_{c_j}), \qquad (10)$$

where ψ is a cost function taking only finite values, each π_i is a permutation restriction, and each μ_j is a crisp unary function.

Let G be the graph with vertices $\{1,\ldots,k\}$ and edges $\{\langle a_i,b_i\rangle \mid i \in I\}$. Choose a set $M = \{m_1,\ldots,m_r\}$ containing one representative from each connected component of G, and define the function η as follows:

$$\eta(y_1,\ldots,y_r) \stackrel{\text{def}}{=} \min\{\phi(x_1,\ldots,x_k) \mid x_{m_i} = y_i, i = 1,\ldots,r\}. \tag{11}$$

By the choice of M, every vertex $1,\ldots,k$ is connected in G to exactly one m_i. Hence, for any $\langle y_1,\ldots,y_r\rangle \in D^r$, we have

$$|\{\langle x_1,\ldots,x_k\rangle \mid \phi(x_1,\ldots,x_k) < \infty \text{ and } x_{m_i} = y_i, i = 1,\ldots,r\}| \leq 1. \tag{12}$$

Let $\langle x_1,\ldots,x_k\rangle \in D^k$, and set $\langle y_1,\ldots,y_r\rangle = \langle x_{m_1},\ldots,x_{m_r}\rangle$. We have, by Equation 12, that $\eta(y_1,\ldots,y_r) \leq \phi(x_1,\ldots,x_k)$, with equality if $\phi(x_1,\ldots,x_k)$ is finite.

It only remains to prove that η can be expressed as a sum of unary functions. Let $1 \leq j \leq r$ and $s = \langle s_1,\ldots,s_r\rangle, t = \langle t_1,\ldots,t_r\rangle \in D^r$.

First suppose that $\eta(s),\eta(t) < \infty$. Since $\phi \in \mathrm{Imp}_{\overline{\mathbb{R}}_+}(F)$, and η is expressible over $\{\phi\}$, we know by Theorem 4.5 that $\eta \in \mathrm{Imp}_{\overline{\mathbb{R}}_+}(F)$, and hence

$$\eta(s) + \eta(t[j \leftarrow s_j]) + \eta(s[j \leftarrow t_j] \geq \eta(s) + \eta(s) + \eta(t).$$

Cancelling $\eta(s) < \infty$, and using symmetry, we obtain,

$$\eta(s[j \leftarrow t_j]) + \eta(t[j \leftarrow s_j]) = \eta(s) + \eta(t). \tag{13}$$

Otherwise, without loss of generality we may assume that $\eta(s) = \infty$, and hence $\phi(x_1,\ldots,x_k) = \infty$ for all $x_1,\ldots,x_k$ with $\langle x_{m_1},\ldots,x_{m_r}\rangle = s$. Using Equation 10, this implies there is some single index i such that $\phi(x_1,\ldots,x_k) = \infty$ for all $x_1,\ldots,x_k$ with $x_{m_i} = s_i$. Hence Equation 13 holds in this case also, since both sides equal ∞.

Hence, in all cases, by Lemma 6.23, η can be expressed as a sum of unary functions. □

Corollary 6.26 *A function $\phi : D^m \to \overline{\mathbb{R}}_+$ has the multimorphism* $\langle \mathrm{Mjrty}_1, \mathrm{Mjrty}_2, \mathrm{Mnrty}_3\rangle$ *if and only if it satisfies the following two conditions:*

- *ϕ is **finitely modular**, that is, for all m-tuples s,t, and all $i = 1,\ldots,m$ such that $\phi(s)$, $\phi(t)$, $\phi(s[i \leftarrow t[i]])$, $\phi(t[i \leftarrow s[i]]) < \infty$, we have that*

$$\phi(s) + \phi(t) = \phi(s[i \leftarrow t[i]]) + \phi(t[i \leftarrow s[i]]).$$

- $\mathrm{Feas}(\phi)$ *has the polymorphisms* Mjrty_1 *and* Mnrty_3.

We will now prove that the set of all functions with the multimorphism $\langle \mathrm{Mjrty}_1, \mathrm{Mjrty}_2, \mathrm{Mnrty}_3\rangle$ is a maximal tractable valued constraint language.

Theorem 6.27 *Let D be a finite set, and let $F : D^3 \to D^3$ be the function defined by $F(x, y, z) = \langle \mathrm{Mjrty}_1(x, y, z), \mathrm{Mjrty}_2(x, y, z), \mathrm{Mnrty}_3(x, y, z)\rangle$.*

1. *The set $\mathrm{Imp}_{\overline{\mathbb{R}}_+}(F)$ is a tractable valued constraint language.*
2. *Any valued constraint language Γ such that $\Gamma \supset \mathrm{Imp}_{\overline{\mathbb{R}}_+}(F)$ is NP-hard.*

Proof:

1. This is a straightforward application of Theorem 6.25. To solve any instance of $\mathrm{VCSP}(\mathrm{Imp}_{\overline{\mathbb{R}}_+}(F))$ we can simply merge each pair of variables constrained by a permutation restriction (combining the associated unary constraints appropriately). The resulting VCSP instance has only unary constraints and so can be solved trivially.

2. Now assume that $\Gamma \supset \mathrm{Imp}_{\overline{\mathbb{R}}_+}(F)$, and hence Γ contains a function ϕ of some arity m such that F is not a multimorphism of ϕ. By Corollary 6.26, there are 3 cases to consider.

   Case 1: ϕ is not finitely modular

   In this case, there exist $j \in \{1, \ldots, m\}, s = \langle s_1, \ldots, s_m\rangle$, and $t = \langle t_1, \ldots, t_m\rangle \in D^m$ such that

   $$\phi(s) + \phi(t) < \phi(s[j \leftarrow t_j]) + \phi(t[j \leftarrow s_j])$$

   and all values in the inequality are finite.

   For $i = 1, 2, \ldots, m$, we define the following permutation restrictions:

   $$\zeta_i(x, y) = \begin{cases} 0 & \text{if} \quad x = s_1, y = s_i \\ 0 & \text{if} \quad x = t_1, y = t_i \\ \infty & \text{otherwise} \end{cases} \qquad \kappa_i(x, y) = \begin{cases} 0 & \text{if} \quad x = s_1, y = t_i \\ 0 & \text{if} \quad x = t_1, y = s_i \\ \infty & \text{otherwise} \end{cases}$$

   Note that each ζ_i and each $\kappa_i \in \mathrm{Imp}_{\overline{\mathbb{R}}_+}(F) \subset \Gamma$.

   We can now construct the instance $\mathcal{P} \in \mathrm{VCSP}(\Gamma)$ with variables

   $$\{X_1, \ldots, X_m, Y_1, \ldots, Y_m, Z\}$$

   and constraints

   $$\begin{array}{ll} \langle\langle X_1, Y_1\rangle, \kappa_1\rangle, & \\ \langle\langle X_1, \ldots, X_m\rangle, \phi\rangle, & \langle\langle Y_1, \ldots, Y_m\rangle, \phi\rangle, \\ \langle\langle Z, X_j\rangle, \kappa_j\rangle, & \langle\langle Z, Y_j\rangle, \zeta_j\rangle, \\ \langle\langle X_1, X_i\rangle, \zeta_i\rangle & (i = 1, 2, \ldots, j-1, j+1, \ldots, m), \\ \langle\langle Y_1, Y_i\rangle, \zeta_i\rangle & (i = 1, 2, \ldots, j-1, j+1, \ldots, m). \end{array}$$

   If we set $W = \langle X_1, Z\rangle$, then it is straightforward to check that

   $$\Phi^W_{\mathcal{P}}(x, y) = \begin{cases} \phi(s) + \phi(t) & \text{if} \quad x \neq y \ \wedge \ x, y \in \{s_1, t_1\} \\ \phi(s[j \leftarrow t_j]) + \phi(t[j \leftarrow s_j]) & \text{if} \quad x = y \ \wedge \ x, y \in \{s_1, t_1\} \\ \infty & \text{otherwise.} \end{cases}$$

   Hence, by Proposition 5.1, $\mathrm{VCSP}(\Gamma)$ is NP-hard.

Case 2: Feas(ϕ) **does not have the polymorphism** Mjrty_1

Let $\text{Feas}(\Gamma) = \{\text{Feas}(\psi) \mid \psi \in \Gamma\}$, and let P be the set of all polymorphisms of $\text{Feas}(\Gamma)$. Since Γ contains all permutation restrictions, the algebra whose set of operations is P is *homogeneous*, as defined in [46]. A complete description of all homogeneous finite algebras is given in Chapter 5 of [46] and it is straightforward to verify[4] from this that if P does not contain the operation Mjrty_1, then every element of P is a polymorphism of the relation $R = \{\langle d_0, d_0, d_0\rangle, \langle d_0, d_1, d_1\rangle, \langle d_1, d_0, d_1\rangle, \langle d_1, d_1, d_0\rangle\}$, for some $d_0, d_1 \in D$.

Hence, by Theorem 4.10 of [24], the relation R can be expressed using some finite combination of relations from $\text{Feas}(\Gamma)$. This implies that Γ^* contains a function ϕ such that $\phi(s) < \infty$ exactly when $s \in R$.

Now set

$$\begin{aligned} \alpha &= \psi(d_0, d_1, d_1) + \psi(d_1, d_0, d_1) < \infty \\ \beta &= \psi(d_1, d_1, d_0) + \psi(d_0, d_0, d_0) < \infty. \end{aligned}$$

We define the binary permutation restriction π and the unary function μ as follows:

$$\pi(x,y) = \begin{cases} 0 & \text{if} \quad x = d_0, y = d_1 \quad \text{or} \quad x = d_1, y = d_0 \\ \infty & \text{otherwise.} \end{cases}$$

$$\mu(x) = \begin{cases} \alpha + 1 & \text{if} \quad x = d_0 \\ 0 & \text{if} \quad x = d_1 \\ \infty & \text{otherwise.} \end{cases}$$

We can now construct the instance $\mathcal{P} \in \text{VCSP}(\Gamma)$ with variables

$$\{X, Y, Z, X', Y'\}$$

and constraints

$$\begin{array}{ll} \langle\langle X, Y, Z\rangle, \psi\rangle, & \langle\langle X', Y', Z\rangle, \psi\rangle, \\ \langle\langle X, X'\rangle, \pi\rangle, & \langle\langle Y, Y'\rangle, \pi\rangle, \\ \langle\langle Z\rangle, \mu\rangle. & \end{array}$$

If we set $W = \langle X, Y\rangle$, then it is straightforward to check that

$$\Phi_{\mathcal{P}}^{W}(x,y) = \begin{cases} \alpha & \text{if} \quad x \neq y \quad \wedge \quad x, y \in \{d_0, d_1\} \\ \alpha + \beta + 1 & \text{if} \quad x = y \quad \wedge \quad x, y \in \{d_0, d_1\} \\ \infty & \text{otherwise.} \end{cases}$$

Hence, by Proposition 5.1, VCSP(Γ) is NP-hard.

[4] This is stated explicitly in Lemma 5.6 of [46] for the case when $|D| \geq 5$; the remaining 3 cases can be checked individually.

Case 3: Feas(ϕ) **has the polymorphism** Mjrty_1**, but not** Mnrty_3**.**

As indicated in the proof of Theorem 6.25, relations with the polymorphism Mjrty_1 are known to be decomposable into binary relations of 3 distinct types [46], and the only one of these types which does not have the polymorphism Mnrty_3 is the set of relations of the form $\{\langle x, y\rangle \in D_1 \times D_2 \mid (x = d_1) \vee (y = d_2)\}$, for some $d_1, d_2 \in D$, and some $D_1, D_2 \subseteq D$ with $|D_1| > 1$ and $|D_2| > 1$.

Now define the binary relation $R_{ij} = \{\langle x_i, x_j\rangle \mid \exists\langle x_1, x_2, \ldots, x_m\rangle \in \mathrm{Feas}(\phi)\}$. It follows from the observations just made that we can choose a pair of indices i and j and $a, b, c, d \in D$ with $a \neq b$, $c \neq d$, such that $\langle a, c\rangle, \langle b, d\rangle, \langle b, c\rangle \in R_{ij}$, and $\langle a, d\rangle \notin R_{ij}$. Hence, if we define the function ϕ' by setting

$$\phi'(x, y) = \min\{\phi(z_1, \ldots, z_m) \mid x = z_i, y = z_j\}$$

then we have $\phi'(a, c), \phi'(b, d), \phi'(b, c) < \infty$, and $\phi'(a, d) = \infty$.

Now define the functions:

$$\zeta(x) = \begin{cases} \phi'(b, d) & \text{if} \quad x = a \\ \phi'(a, c) + \phi'(b, d) + 1 & \text{if} \quad x = b \\ \infty & \text{otherwise} \end{cases}$$

$$\kappa(x) = \begin{cases} 2(\phi'(b, d) + 1) & \text{if} \quad x = c \\ 0 & \text{if} \quad x = d \\ \infty & \text{otherwise} \end{cases}$$

$$\tau(x, y) = \begin{cases} 0 & \text{if} \quad \{x, y\} = \{a, b\} \\ \infty & \text{otherwise} \end{cases}$$

Note that $\tau, \kappa, \zeta \in \mathrm{Imp}_{\overline{\mathbb{R}}_+}(F) \subset \Gamma$ and $\phi' \in \Gamma^*$.

We can now construct the instance $\mathcal{P} \in \mathrm{VCSP}(\Gamma^*)$ with variables

$$\{X, Y, Z, \overline{Z}\}$$

and constraints

$$\begin{array}{ll} \langle\langle X, Y\rangle, \phi'\rangle, & \langle\langle Z, Y\rangle, \phi'\rangle, \\ \langle\langle X\rangle, \zeta\rangle, & \langle\langle Z\rangle, \zeta\rangle, \\ \langle\langle Y\rangle, \kappa\rangle, & \langle\langle \overline{Z}, Z\rangle, \tau\rangle. \end{array}$$

If we set $W = \langle X, \overline{Z}\rangle$, then it is straightforward to check that

$$\Phi_{\mathcal{P}}^{W}(x, y) = \begin{cases} 2\phi'(a, c) + 4\phi'(b, d) + 2 & \text{if} \quad x \neq y, \ x, y \in \{a, b\} \\ 2\phi'(a, c) + 4\phi'(b, d) + \phi'(b, c) + 3 & \text{if} \quad x = y, \ x, y \in \{a, b\} \\ \infty & \text{otherwise.} \end{cases}$$

Hence, by Proposition 5.1, VCSP(Γ) is NP-hard.

□

7 The Boolean Case

Recall from Example 2.9 that a valued constraint language over the set $\{0, 1\}$ is called a valued Boolean constraint language. In this section we will show that *every* tractable valued Boolean constraint language with costs in $\overline{\mathbb{R}}_+$ is characterized by the presence of a certain form of multimorphism. In fact we establish a dichotomy result: if a valued Boolean constraint language with costs in $\overline{\mathbb{R}}_+$ has one of eight specified multimorphisms then it is tractable, otherwise it is NP-hard.

Theorem 7.1 *Let Γ be a valued Boolean constraint language with costs in $\overline{\mathbb{R}}_+$. If Γ has one of the following multimorphisms then* $\mathrm{VCSP}(\Gamma)$ *is tractable:*

1. $\langle \mathbf{0} \rangle$*, where* $\mathbf{0}$ *is the constant unary function returning the value 0;*
2. $\langle \mathbf{1} \rangle$*, where* $\mathbf{1}$ *is the constant unary function returning the value 1;*
3. $\langle \mathrm{Max}, \mathrm{Max} \rangle$*, where* Max *is the binary function returning the maximum of its arguments (i.e.,* $\mathrm{Max}(x, y) = x \vee y$*);*
4. $\langle \mathrm{Min}, \mathrm{Min} \rangle$*, where* Min *is the binary function returning the minimum of its arguments (i.e.,* $\mathrm{Min}(x, y) = x \wedge y$*);*
5. $\langle \mathrm{Min}, \mathrm{Max} \rangle$*;*
6. $\langle \mathrm{Mjrty}, \mathrm{Mjrty}, \mathrm{Mjrty} \rangle$*, where* Mjrty *is the unique ternary majority function on the set* $\{0, 1\}$*;*
7. $\langle \mathrm{Mnrty}, \mathrm{Mnrty}, \mathrm{Mnrty} \rangle$*, where* Mnrty *is the unique ternary minority function on the set* $\{0, 1\}$*;*
8. $\langle \mathrm{Mjrty}, \mathrm{Mjrty}, \mathrm{Mnrty} \rangle$*;*

In all other cases $\mathrm{VCSP}(\Gamma)$ *is NP-hard.*

To establish the first part of Theorem 7.1, we must show that a valued Boolean constraint language which has one of the eight types of multimorphisms listed in the theorem is tractable.

The tractability of any valued constraint language which has the multimorphism $\langle \mathbf{0} \rangle$ or $\langle \mathbf{1} \rangle$ was established in Theorem 6.4. Furthermore, the tractability of any valued constraint language which has the multimorphism $\langle \mathrm{Max}, \mathrm{Max} \rangle$ was established in Theorem 6.15, and a symmetric argument (with the domain ordering reversed) establishes the tractability of any valued constraint language with the multimorphism $\langle \mathrm{Min}, \mathrm{Min} \rangle$. The tractability of any valued constraint language which has the multimorphism $\langle \mathrm{Min}, \mathrm{Max} \rangle$ was established in Theorem 6.7. The tractability of any valued constraint language which has the multimorphism $\langle \mathrm{Mjrty}, \mathrm{Mjrty}, \mathrm{Mjrty} \rangle$ was established in Proposition 6.20, and the tractability of any valued constraint language which has the multimorphism $\langle \mathrm{Mnrty}, \mathrm{Mnrty}, \mathrm{Mnrty} \rangle$ was established in Proposition 6.22. Finally, the tractability of any valued Boolean constraint language which has the multimorphism $\langle \text{Mjrty}, \text{Mjrty}, \text{Mnrty} \rangle$ follows immediately from Theorem 6.27.

To establish the remaining part of Theorem 7.1, we must show that a valued Boolean constraint language with costs in $\overline{\mathbb{R}}_+$ which does *not* have any of the types of multimorphisms listed in the theorem is NP-hard. We first deal with essentially crisp languages.

Lemma 7.2 *Any valued Boolean constraint language which is essentially crisp and does not have any of the multimorphisms listed in Theorem 7.1 is NP-hard.*

Proof: If we replace each cost function ϕ in Γ with the relation $\text{Feas}(\phi)$ then we obtain a crisp Boolean constraint language Γ' which does not have any of the polymorphisms $\mathbf{0}, \mathbf{1}, \text{Min}, \text{Max}, \text{Mjrty}$ or Mnrty.

By Schaefer's Dichotomy Theorem [42, 26], Γ' is NP-complete, and hence Γ is NP-hard. $\square$

For the remaining languages, our strategy will be to show that any language which does not have one of the multimorphisms listed in Theorem 7.1 can express certain special functions, which we now define.

Definition 7.3

- A unary function σ on the set $\{0, 1\}$ is a **0-selector** if

$$\sigma(0) < \sigma(1)$$

  and it is a **finite 0-selector** if, in addition, $\sigma(1) < \infty$.

  A **(finite) 1-selector** is defined analogously. A **selector** is either a 1-selector or a 0-selector.

- A binary function ϕ on the set $\{0, 1\}$ is a **NEQ** function if

$$\phi(0, 1) = \phi(1, 0) < \phi(1, 1) = \phi(0, 0) = \infty.$$

- A binary function ϕ on the set $\{0, 1\}$ is an **XOR** function if

$$\phi(0, 1) = \phi(1, 0) < \phi(1, 1) = \phi(0, 0) < \infty.$$

Lemma 7.4 *Let Γ be a valued Boolean constraint language with costs in $\overline{\mathbb{R}}_+$ which is not essentially crisp.*

If Γ^* contains a NEQ function, then either Γ^* contains both a finite 0-selector and a finite 1-selector, or else Γ^* contains an XOR function.

Proof: Let $\nu \in \Gamma^*$ be a NEQ function.

First we show that if Γ^* contains a finite 0-selector σ_0, then it also contains a finite 1-selector. To see this, simply construct the instance $\mathcal{P}_0$ with variables

$\{x, y\}$ and constraints $\{\langle\langle x\rangle, \sigma_0\rangle, \langle\langle x, y\rangle, \nu\rangle\}$, and note that $\Phi_{\mathcal{P}_0}^{\langle y\rangle}$ is a finite 1-selector. Similarly, if Γ^* contains a finite 1-selector, then it also contains a finite 0-selector.

Now let $\zeta \in \Gamma$ be a cost function of arity m which is not essentially crisp. Choose tuples u, v such that $\zeta(u)$ and $\zeta(v)$ are as small as possible with $\zeta(u) < \zeta(v) < \infty$. Let $\mathcal{P}$ be the VCSP instance with four variables: $\{x_{00}, x_{01}, x_{10}, x_{11}\}$, and three constraints:

$$\langle\langle x_{u[1]v[1]}, \ldots, x_{u[m]v[m]}\rangle, \zeta\rangle, \quad \langle\langle x_{00}, x_{11}\rangle, \nu\rangle, \quad \langle\langle x_{01}, x_{10}\rangle, \nu\rangle.$$

Let $W = \langle x_{01}, x_{11}\rangle$, and $\psi = \Phi_{\mathcal{P}}^W$.

Note that the arity-m cost function ζ is applied to only four variables by repeating arguments. Note also that $\psi(0,1) = \zeta(u) + 2\nu(0,1)$ and $\psi(1,1) = \zeta(v) + 2\nu(0,1)$. If $\psi(0,1) \neq \psi(1,0)$, then, by the choice of u, $\psi(0,1) < \psi(1,0)$, and $\psi(0,1) < \psi(1,1) < \infty$, so $\Phi_{\mathcal{P}}^{\langle x_{01}\rangle}$ is a finite 0-selector.

Hence we may assume that $\psi(0,1) = \psi(1,0)$. If $\psi(0,0) \neq \psi(1,1)$, then if $\psi(0,0) < \infty$ the function $\psi(x,x)$ is a finite selector, and hence Γ^* contains both a finite 0-selector and a finite 1-selector. On the other hand, if $\psi(0,0) = \infty$ then construct the instance $\mathcal{P}_2$ with variables $\{x, y\}$ and constraints $\{\langle\langle x, x\rangle, \psi\rangle, \langle\langle x, y\rangle, \psi\rangle\}$. In this case $\Phi_{\mathcal{P}_2}^{\langle y\rangle}$ is a finite 0-selector, and hence Γ^* again contains both a finite 0-selector and a finite 1-selector.

Otherwise we may assume that $\psi(0,1) = \psi(1,0)$ and $\psi(0,0) = \psi(1,1)$. By construction, we have $\psi(0,1) = \zeta(u) + 2\nu(0,1) < \zeta(v) + 2\nu(0,1) = \psi(1,1) < \infty$. So in this case ψ is an XOR function. $\square$

Lemma 7.5 *Let Γ be a valued Boolean constraint language with costs in $\overline{\mathbb{R}}_+$ which is not essentially crisp, and does not have either of the multimorphisms $\langle\mathbf{0}\rangle$ or $\langle\mathbf{1}\rangle$.*

Either Γ^* contains a 0-selector and a 1-selector, or else Γ^* contains an XOR function.

Proof: Let $\phi_0 \in \Gamma$ be a function which does not have the multimorphism $\langle\mathbf{0}\rangle$, and $\phi_1 \in \Gamma$ be a function which does not have the multimorphism $\langle\mathbf{1}\rangle$, and let m be the arity of ϕ_0. Choose a tuple r such that $\phi_0(r)$ is the minimal value of ϕ_0. By the choice of ϕ_0, we have $\phi_0(r) < \phi_0(0, 0, \ldots, 0)$.

Suppose first that Γ^* contains a 0-selector σ_0. Let M be a finite natural number which is larger than all finite values in the range of ϕ_0. We construct the instance $\mathcal{P} \in \mathrm{VCSP}(\Gamma)$ with two variables $\{x_0, x_1\}$, and two constraints $\langle\langle x_{r[1]}, \ldots, x_{r[m]}\rangle, \phi_0\rangle$ and $\langle\langle x_0\rangle, M\sigma_0\rangle$. (The cost function $M\sigma_0$ is simply equivalent to taking M copies of a constraint with cost function σ_0.) It is straightforward to check that $\Phi_{\mathcal{P}}^{\langle x_1\rangle}(1) < \Phi_{\mathcal{P}}^{\langle x_1\rangle}(0)$, and so in this case Γ^* contains a 1-selector. A similar argument, using ϕ_1, shows that if Γ^* contains a 1-selector, then it also contains a 0-selector.

Hence, we need to show that either Γ^* contains a selector, or it contains an XOR function. If $\phi_0(0,\ldots,0) \neq \phi_0(1,\ldots,1)$ then the unary function $\sigma(x) = \phi_0(x,\ldots,x)$ in Γ^* is clearly a selector, and the result holds.

Otherwise, we construct the instance $\mathcal{P}' \in \mathrm{VCSP}(\Gamma)$ with two variables $\{x_0, x_1\}$ and the single constraint $\langle\langle x_{r[1]},\ldots,x_{r[m]}\rangle,\phi_0\rangle$. Now, by considering the costs of all four possible assignments, we can verify that either $\Phi_{\mathcal{P}'}^{\langle x_0\rangle}$ or $\Phi_{\mathcal{P}'}^{\langle x_1\rangle}$ is a selector, or else $\nu = \Phi_{\mathcal{P}'}^{\langle x_0,x_1\rangle}$ is either an XOR function, or a NEQ function.

If ν is an XOR function we are done, otherwise we appeal to Lemma 7.4 to complete the proof. $\square$

Many of the remaining lemmas in this Section use the following construction which combines a given function ϕ of arbitrary arity with a pair of selectors, in order to express a *binary* function with some similar properties.

Construction 7.6 Let $\phi : D^m \to \overline{\mathbb{R}}_+$ be an m-ary function which is not identically infinite, and let σ_0 be a 0-selector and σ_1 a 1-selector. Let u, v be two m-tuples, and let M be a natural number larger than all finite values in the range of ϕ.

Let $\mathcal{P}$ be a VCSP instance with variables $\{x_{00}, x_{01}, x_{10}, x_{11}\}$, and constraints:

$$\langle\langle x_{u[1]v[1]},\ldots,x_{u[m]v[m]}\rangle,\phi\rangle, \quad \langle\langle x_{00}\rangle, M\sigma_0\rangle, \quad \langle\langle x_{11}\rangle, M\sigma_1\rangle.$$

The binary function $\phi_2 \stackrel{\mathrm{def}}{=} \Phi_{\mathcal{P}}^{\langle x_{01},x_{10}\rangle}$ will be called a **compression** of ϕ by u and v.

Lemma 7.7 *Let Γ be a valued Boolean constraint language with costs in $\overline{\mathbb{R}}_+$ which is not essentially crisp, and does not have any of the multimorphisms $\langle\mathbf{0}\rangle$ or $\langle\mathbf{1}\rangle$ or $\langle\mathrm{Max},\mathrm{Max}\rangle$ or $\langle\mathrm{Min},\mathrm{Min}\rangle$.*

Either Γ^* contains a finite 0-selector and a finite 1-selector, or else Γ^* contains an XOR function.

Proof: Let ϕ be a function in Γ which does not have a $\langle\mathrm{Max},\mathrm{Max}\rangle$ multimorphism, and let ψ be a function in Γ which does not have a $\langle\mathrm{Min},\mathrm{Min}\rangle$ multimorphism.

By Lemma 7.5, either Γ^* contains an XOR function and we have nothing to prove, or else Γ^* contains a 0-selector, σ_0, and a 1-selector, σ_1.

Since ϕ does not have a $\langle\mathrm{Max},\mathrm{Max}\rangle$ multimorphism, it follows from Lemma 6.14 that either ϕ is not finitely antitone, or else the relation $\mathrm{Feas}(\phi)$ does not have the polymorphism Max.

For the first case, choose two tuples u and v, with $u < v$ with $\phi(u) < \phi(v) < \infty$, and let ϕ_2 be a compression of ϕ by u and v (see Construction 7.6). It is straightforward to check that $\phi_2(0,0) < \phi_2(1,1) < \infty$, which means that $\phi_2(x,x)$ is a finite 0-selector belonging to Γ^*.

On the other hand suppose that ϕ is finitely antitone, and that Γ^* contains a finite 1-selector τ. In this case we know that $\mathrm{Feas}(\phi)$ does not have the polymorphism Max, so we can choose u, v such that $\phi(u), \phi(v) < \infty$ and $\phi(\mathrm{Max}(u,v)) = \infty$. Let ϕ_2 be a compression of ϕ by u and v, and construct the instance $\mathcal{P} \in \mathrm{VCSP}\,\Gamma^*$ with variables $\{x, y\}$, and constraints:

$$\langle\langle x,y\rangle,\phi_2\rangle, \qquad \langle\langle y,x\rangle,\phi_2\rangle, \qquad \langle\langle y\rangle,\tau\rangle.$$

The fact that ϕ is finitely antitone gives $\phi(u), \phi(v) \leq \phi(\mathrm{Min}(u,v))$. This, together with the fact that $\phi(u)$ and $\phi(v)$ are finite whilst $\phi(\mathrm{Max}(u,v))$ is infinite, is enough to show that $\Phi_{\mathcal{P}}^{\langle x\rangle}$ is a finite 0-selector.

So, we have shown that if Γ^* contains a finite 1-selector, then it contains a finite 0-selector whether or not ϕ is finitely antitone. A symmetric argument, exchanging 0 and 1, Max and Min, and ϕ and ψ, shows that if Γ^* contains a finite 0-selector, then it contains a finite 1-selector.

Hence, to complete the proof we may assume that Γ^* contains no finite selectors. In this case we know that $\mathrm{Feas}(\phi)$ does not have the polymorphism Max and $\mathrm{Feas}(\psi)$ does not have the polymorphism Min, so we may choose tuples u, v, w, z such that $\phi(u), \phi(v), \psi(w)$ and $\psi(z)$ are all finite, but $\phi(\mathrm{Max}(u,v))$ and $\psi(\mathrm{Min}(w,z))$ are both infinite. Now let ϕ_2 be a compression of ϕ by u and v, and ψ_2 a compression of ψ by w and z We then have that $\rho(x,y) \stackrel{\text{def}}{=} \phi_2(x,y) + \phi_2(y,x) + \psi_2(x,y) + \psi_2(y,x)$ is a NEQ function which is contained in Γ^*. We can now appeal to Lemma 7.4 to show that Γ^* contains an XOR function, and we are done. □

Lemma 7.8 *Let Γ be a valued Boolean constraint language with costs in $\overline{\mathbb{R}}_+$ which does not have the multimorphism $\langle \mathrm{Min}, \mathrm{Max}\rangle$.*

If Γ^* contains both a finite 0-selector and a finite 1-selector, then Γ^* contains a NEQ function or an XOR function.

Proof: Let ϕ be a function in Γ that does not have the multimorphism $\langle \mathrm{Min}, \mathrm{Max}\rangle$. Choose u, v such that $\phi(\mathrm{Min}(u,v)) + \phi(\mathrm{Max}(u,v)) > \phi(u) + \phi(v)$. Let ϕ_2 be a compression of ϕ by u and v. It is straightforward to check that the binary function ϕ_2 also does not have the multimorphism $\langle \mathrm{Min}, \mathrm{Max}\rangle$.

It follows that

$$\phi_2(0,0) + \phi_2(1,1) \;>\; \phi_2(0,1) + \phi_2(1,0). \tag{14}$$

Without loss of generality, suppose that $\phi_2(0,0) \geq \phi_2(1,1)$. (The proof for the case $\phi_2(0,0) > \phi_2(1,1)$ is symmetrically equivalent.)

From Equation (14), we have

$$2\phi_2(0,0) - [\phi_2(0,1) + \phi_2(1,0)] \;>\; [\phi_2(0,1) + \phi_2(1,0)] - 2\phi_2(1,1)$$

with $2\phi_2(0,0) - [\phi_2(0,1) + \phi_2(1,0)] > 0$.

Now let $\sigma_0 \in \Gamma^*$ be a finite 0-selector, and set $\lambda = \sigma_0(1) - \sigma_0(0)$. Since $\lambda > 0$, it is possible to choose a non-negative rational number $\frac{N}{M}$ such that

$$2\phi_2(0,0) - [\phi_2(0,1) + \phi_2(1,0)] \;>\; \frac{N}{M}\lambda \;>\; [\phi_2(0,1) + \phi_2(1,0)] - 2\phi_2(1,1).$$

Construct an instance $\mathcal{P} \in \text{VCSP}(\Gamma^*)$ with variables $\{x, u, v, y\}$, and constraints

$$\begin{array}{ll}
\langle\langle x, u\rangle, M\phi_2\rangle, & \langle\langle u, x\rangle, M\phi_2\rangle, \\
\langle\langle u, v\rangle, M\phi_2\rangle, & \langle\langle v, u\rangle, M\phi_2\rangle, \\
\langle\langle v, y\rangle, M\phi_2\rangle, & \langle\langle y, v\rangle, M\phi_2\rangle, \\
\langle\langle x\rangle, N\sigma_0\rangle, & \langle\langle u\rangle, 2N\sigma_0\rangle, \\
\langle\langle v\rangle, 2N\sigma_0\rangle, & \langle\langle y\rangle, N\sigma_0\rangle.
\end{array}$$

If we set $W = \langle x, y\rangle$, and $\eta = \Phi_{\mathcal{P}}^{W}$, then it is straightforward to verify that $\eta(0,1) = \eta(1,0)$, $\eta(0,0) = \eta(1,1)$, and

$$\begin{aligned}
\eta(0,0) &= \eta(0,1) + M\min\{2\phi_2(0,0) - [\phi_2(0,1) + \phi_2(1,0)] - \frac{N}{M}\lambda, \\
&\qquad\qquad \frac{N}{M}\lambda - [\phi_2(0,1) + \phi_2(1,0)] - 2\phi_2(1,1)\} \\
&> \eta(0,1).
\end{aligned}$$

If $\phi_2(1,1) = \infty$, then $\eta(0,0) = \infty$ and hence η is a NEQ function. If $\phi_2(1,1) < \infty$, then $\eta(0,0) < \infty$ and hence η is an XOR function. $\square$

Lemma 7.9 *Let Γ be a valued Boolean constraint language with costs in $\overline{\mathbb{R}}_+$ which does not have the multimorphism* $\langle \text{Mjrty}, \text{Mjrty}, \text{Mnrty}\rangle$*.*

If Γ^* contains a finite 0-selector, a finite 1-selector, and a NEQ function, then Γ^* contains an XOR function.

Proof: Suppose that $\sigma_0 \in \Gamma^*$ is a finite 0-selector, $\sigma_1 \in \Gamma^*$ is a finite 1-selector, $\nu \in \Gamma^*$ is a NEQ function, and $\phi \in \Gamma$ does not have the multimorphism $\langle \text{Mjrty}, \text{Mjrty}, \text{Mnrty}\rangle$. We have to show that Γ^* also contains an XOR function.

By Corollary 6.26 there are 2 cases: either ϕ is not finitely modular, or $\text{Feas}(\phi)$ does not have both polymorphisms Mjrty and Mnrty.

In the first case, choose tuples u, v such that $\phi(u) + \phi(v) \neq \phi(\text{Min}(u,v)) + \phi(\text{Max}(u,v))$. Let ϕ_2 be a compression of ϕ by u and v. It is straightforward to check that ϕ_2 is also not finitely modular. Now construct the instance $\mathcal{P}$ with variables $\{w, x, y, z\}$, and constraints

$$\langle\langle x, w\rangle, \nu\rangle, \qquad \langle\langle z, y\rangle, \nu\rangle, \qquad \langle\langle x, z\rangle, \phi_2\rangle, \qquad \langle\langle w, y\rangle, \phi_2\rangle.$$

It is straightforward to check that either $\Phi_{\mathcal{P}}^{\langle x,y\rangle}$ or $\Phi_{\mathcal{P}}^{\langle w,y\rangle}$ is an XOR function.

Next, suppose that $\text{Feas}(\phi)$ has the polymorphism Mjrty but not Mnrty. In this case, by Theorem 3.5 of [25], $\text{Feas}(\phi)$ is decomposable into binary relations (in other words, it is equal to the relational join of its binary projections). Since $\text{Feas}(\phi)$ does not have the Mnrty polymorphism, this implies that one of its binary projections does not have the Mnrty polymorphism. The only binary Boolean relations which do not have the Mnrty polymorphism have exactly three tuples. Therefore, by projection, it is possible to construct from ϕ a binary function ψ such that exactly three of $\psi(0,0), \psi(0,1), \psi(1,0), \psi(1,1)$ are finite. If $\psi(0,1)$ or $\psi(1,0)$ is infinite, then let η be the projection onto variables x, y of $\psi(x,v) + \nu(v,y)$, otherwise let $\eta = \psi$. The function η does not have the multimorphism $\langle \mathrm{Min}, \mathrm{Max} \rangle$, and exactly one of $\eta(0,0)$ and $\eta(1,1)$ are infinite, and so, by the construction in the proof of Lemma 7.8, Γ^* contains an XOR function.

Suppose now that $\mathrm{Feas}(\phi)$ has the polymorphism Mnrty but not Mjrty. Since $\mathrm{Feas}(\phi)$ has the polymorphism Mnrty, it is an *affine* relation [10] over the finite field with 2 elements, GF(2), and can be expressed as a system of linear equations over GF(2). Creignou et al. define a Boolean relation to be *affine with width 2* if it can be expressed as a system of linear equations over GF(2), with at most two variables per equation [10]. In fact, linear equations over GF(2) with one variable correspond to the unary relations, and linear equations over GF(2) with two variables correspond to the binary equality and disequality relations. The unary relations, and the binary equality and disequality relations all have both the Mjrty and Mnrty polymorphisms. Thus $\mathrm{Feas}(\phi)$ is affine but not of width 2. Hence, by Lemma 5.34 of [10], $\mathrm{Feas}(\phi)$ can be used to construct the 4-ary affine constraint $w + x + y + z = 0$. In other words, there is some $\psi \in \Gamma^*$ such that $\psi(w,x,y,z) < \infty$ iff $w + x + y + z = 0$.

Now set $\lambda = \psi(0,0,1,1) + \psi(0,1,0,1) + 1$ and construct the VCSP instance $\mathcal{P}$ with variables $\{w, x, y, z\}$, and constraints

$$\langle \langle w, x, y, z \rangle, \psi \rangle, \qquad \langle \langle w \rangle, 3M\sigma_0 \rangle, \qquad \langle \langle z \rangle, \lambda\sigma_1 \rangle$$

where M is a natural number larger than the square of any finite value in the range of ψ or σ_1. Let $\eta = \Phi_{\mathcal{P}}^{\langle x,y \rangle}$. It is straightforward to verify that η is a binary function where both $\eta(0,0)$ and $\eta(1,1)$ are finite, and it does not have the multimorphism $\langle \mathrm{Min}, \mathrm{Max} \rangle$. Hence, by the construction in the proof of Lemma 7.8, the result follows in this case also.

Finally, if $\mathrm{Feas}(\phi)$ has neither the polymorphism Mnrty nor Mjrty, then the set of Boolean relations $\{\mathrm{Feas}(\phi), \mathrm{Feas}(\nu)\}$ can be shown to have essentially unary polymorphisms only (see Theorem 4.12 of [24]). By Theorem 4.10 of [24], this implies that in this case $\mathrm{Feas}(\phi)$ can again be used to construct the 4-ary affine constraint $w + x + y + z = 0$, and we can proceed as above. □

Lemma 7.10 *Let Γ be a valued Boolean constraint language with costs in $\overline{\mathbb{R}}_+$ which does not have any of the multimorphisms listed in Theorem 7.1.*

Either Γ is essentially crisp, or else Γ^* contains an XOR function.

Proof: Suppose that Γ is not essentially crisp and has none of the multimorphisms listed in Theorem 7.1. By Lemmas 7.8 and 7.7, either Γ^* contains an XOR function, or else Γ^* contains a NEQ function and a finite 0-selector and a finite 1-selector. In the latter case, by Lemma 7.9 we know that Γ^* contains an XOR function. □

Combining Lemmas 7.2 and 7.10, together with Proposition 5.1, establishes the NP-hardness of any valued Boolean constraint language having none of the multimorphisms listed in Theorem 7.1, and so completes the proof of Theorem 7.1.

For valued Boolean constraint languages taking *finite values only*, some of the tractable cases identified in Theorem 7.1 coincide, as the next result indicates.

Corollary 7.11 *Let Γ be a valued Boolean constraint language where all costs are finite real values. If Γ has one of the multimorphisms $\langle \mathbf{0} \rangle$, $\langle \mathbf{1} \rangle$, or $\langle \mathrm{Min}, \mathrm{Max} \rangle$, then $\mathrm{VCSP}(\Gamma)$ is tractable. In all other cases $\mathrm{VCSP}(\Gamma)$ is NP-hard.*

Proof: Let ϕ be a function taking finite values in $\overline{\mathbb{R}}_+$ only. By Lemma 6.14, if ϕ has the multimorphism $\langle \mathrm{Max}, \mathrm{Max} \rangle$, then ϕ is antitone, and hence has the multimorphism $\langle \mathbf{1} \rangle$. By a symmetric argument, if ϕ has the multimorphism $\langle \mathrm{Min}, \mathrm{Min} \rangle$, then ϕ is monotone, and hence has the multimorphism $\langle \mathbf{0} \rangle$. By Proposition 6.20, if ϕ has the multimorphism $\langle \mathrm{Mjrty}, \mathrm{Mjrty}, \mathrm{Mjrty} \rangle$, then ϕ is constant, and hence has the multimorphism $\langle \mathbf{0} \rangle$. Similarly, by Proposition 6.22, if ϕ has the multimorphism $\langle \mathrm{Mnrty}, \mathrm{Mnrty}, \mathrm{Mnrty} \rangle$, then ϕ is again constant, and hence has the multimorphism $\langle \mathbf{0} \rangle$. By Corollary 6.26, if ϕ has the multimorphism $\langle \mathrm{Mjrty}, \mathrm{Mjrty}, \mathrm{Mnrty} \rangle$, then ϕ is modular, and hence it has the multimorphism $\langle \mathrm{Min}, \mathrm{Max} \rangle$. The result now follows from Theorem 7.1. □

We now show that Theorem 7.1 generalises a number of earlier dichotomy results for particular Boolean problems [10, 30, 42]. Let S be a set of Boolean relations: the problem $\textsc{Sat}(S)$ is the problem of deciding whether there exists an assignment $s : V \rightarrow \{0, 1\}$ which satisfies a given collection of crisp constraints with relations chosen from S. The problem $\textsc{Max-Sat}(S)$ is the problem of finding an assignment which maximises the number of constraints from such a collection which are simultaneously satisfied. The problem $\textsc{Min-Ones}(S)$ is the problem of deciding whether there exists an assignment which satisfies a given collection of crisp constraints with relations chosen from S, and if so finding such an assignment which minimises the number of variables taking the value 1. In the slightly more general *weighted* $\textsc{Min-Ones}(S)$ problem the aim is to minimise a specified weighted sum, $\sum_{v \in V} w_v s(v)$, where the w_v are non-negative integers [10, 30]. Similarly, the problem $\textsc{Max-Ones}(S)$ is the problem of deciding whether there exists an assignment which satisfies a given collection of crisp constraints with relations chosen from S, and if so finding such an assignment which maximises the number of variables taking the value 1. In the *weighted* $\textsc{Max-Ones}(S)$ problem the aim is to maximise a specified weighted sum, $\sum_{v \in V} w_v s(v)$, where the w_v are non-negative integers [10, 30].

Corollary 7.12 *Let S be a set of Boolean relations and let $\Gamma_S = \{\phi_R \mid R \in S\}$ be the corresponding crisp valued constraint language over $\{0, 1\}$.*

1. $\textsc{Sat}(S)$ *can be solved in polynomial time if S has one of the polymorphisms* $\mathbf{0}, \mathbf{1}, \mathrm{Min}, \mathrm{Max}, \mathrm{Mnrty}$, *or* Mjrty. *Otherwise it is NP-complete.*

2. $\textsc{Max-Sat}(S)$ *can be solved in polynomial time if* Γ_S *has one of the multimorphisms* $\langle \mathbf{0} \rangle$, $\langle \mathbf{1} \rangle$, *or* $\langle \mathrm{Min}, \mathrm{Max} \rangle$. *Otherwise it is NP-hard.*

3. *Weighted* $\textsc{Min-Ones}(S)$ *can be solved in polynomial time if* Γ_S *has one of the multimorphisms* $\langle \mathbf{0} \rangle$, $\langle \mathrm{Min}, \mathrm{Min} \rangle$, *or* $\langle \mathrm{Mjrty}, \mathrm{Mjrty}, \mathrm{Mnrty} \rangle$. *Otherwise it is NP-hard.*

4. *Weighted* $\textsc{Max-Ones}(S)$ *can be solved in polynomial time if* Γ_S *has one of the multimorphisms* $\langle \mathbf{1} \rangle$, $\langle \mathrm{Max}, \mathrm{Max} \rangle$, *or* $\langle \mathrm{Mjrty}, \mathrm{Mjrty}, \mathrm{Mnrty} \rangle$. *Otherwise it is NP-hard.*

Proof:

1. Follows immediately from Theorem 7.1 and Proposition 4.10.

2. Follows from Corollary 7.11 and Example 2.9.

3. Let $\phi_1 : \{0,1\} \rightarrow \overline{\mathbb{R}}_+$ be the function defined by $\phi_1(x) = x$. By Example 3.3, the problem VCSP($\{\phi_1\}$) is equivalent to the problem of minimising a linear expression of the form $\sum_{v \in V} w_v s(v)$, where the w_v are non-negative integers. Hence, weighted $\textsc{Min-Ones}(S)$ can be expressed as VCSP($\Gamma_S \cup \{\phi_1\}$). The function ϕ_1 is contained in exactly 4 of the tractable classes identified in Theorem 7.1 (cases 1,4,5 and 8), so the problem VCSP($\Gamma_S \cup \{\phi_1\}$) is tractable when Γ_S has one of the multimorphisms $\langle \mathbf{0} \rangle$, $\langle \mathrm{Min}, \mathrm{Min} \rangle$, $\langle \mathrm{Min}, \mathrm{Max} \rangle$, or $\langle \mathrm{Mjrty}, \mathrm{Mjrty}, \mathrm{Mnrty} \rangle$, and NP-hard otherwise. Finally, by Proposition 4.10, if a *crisp* language has the multimorphism $\langle \mathrm{Min}, \mathrm{Max} \rangle$ then it also has the multimorphism $\langle \mathrm{Min}, \mathrm{Min} \rangle$.

4. Similar to (3), but using the function ϕ_1' defined by $\phi_1'(x) = 1 - x$.

□

Corollary 7.12 gives an alternative and more unified description of the tractable cases for these problems to the ones given previously in [10, 30, 42].

Finally, we note that the dichotomy described in Theorem 7.1 can be expressed in a more concise form using earlier results about crisp Boolean constraints and Theorem 5.4.

Corollary 7.13 *Let* Γ *be a valued Boolean constraint language with costs in* $\overline{\mathbb{R}}_+$. *If* Γ *has a non-trivial multimorphism then it is tractable. Otherwise it is NP-hard.*

Proof: Earlier results about crisp Boolean constraint languages show that a crisp Boolean language is tractable if it has a polymorphism which is not essentially unary, and NP-complete otherwise (see, for example, Corollary 2.29 of [6]). Using the relationship between polymorphisms and multimorphisms set out in Proposition 4.10, and the fact that multimorphisms are preserved by addition of a constant, this implies that the result holds when Γ is essentially crisp.

If Γ is not essentially crisp, then by Lemma 7.10, either Γ has a non-trivial multimorphism, and is tractable for one of the reasons described earlier, or else Γ^* contains an XOR function.

If Γ^* contains an XOR function, then by Corollary 5.5, every multimorphism of Γ is trivial. □

8 Conclusions and Future Work

In this paper we have begun a systematic investigation of the complexity of the optimisation problems resulting from different forms of soft constraint. Since soft constraints are specified by functions, we have introduced an algebraic property of a function, which we call a multimorphism, and shown that in a range of cases the presence of such a property is sufficient to ensure tractability.

Moreover, we have shown that the presence of a multimorphism precisely characterises a number of tractable problem classes that appear on the surface to be very different. These tractable classes are listed in Section 6; as indicated by the examples given in that section, they are overlapping, but incomparable, in the sense that none is contained in any of the others (see Figure 2). In the Boolean case, when the costs are real-valued or infinite, we have shown that the presence of one of eight forms of multimorphism characterises each of the possible tractable cases, and that all other cases are NP-hard. This result generalises earlier complexity classifications for the Satisfiability, Max-Sat, Min-Ones and Max-Ones problems.

On the basis of the results presented here, we conjecture that the multimorphisms of a valued constraint language over a finite set completely determine its expressive power, and hence its complexity. If this is true, then multimorphisms are likely to play a central role in the analysis of complexity for soft constraints, just as the related notion of a polymorphism does in the analysis of complexity for crisp constraints [4, 6, 5, 26, 27, 28].

To define any form of soft constraint we must specify the set of possible values for the costs, and the way in which these are combined. In this paper we have adopted the valued constraint framework [1, 43], where the costs are chosen from some totally ordered set. For our concrete classification results in Sections 5, 6 and 7 we have fixed this set to be $\overline{\mathbb{R}}_+$, the set of non-negative real numbers together with infinity, combined using standard addition. One possible direction in which to extend our results would be to investigate the complexity of valued constraint languages with other valuation structures.

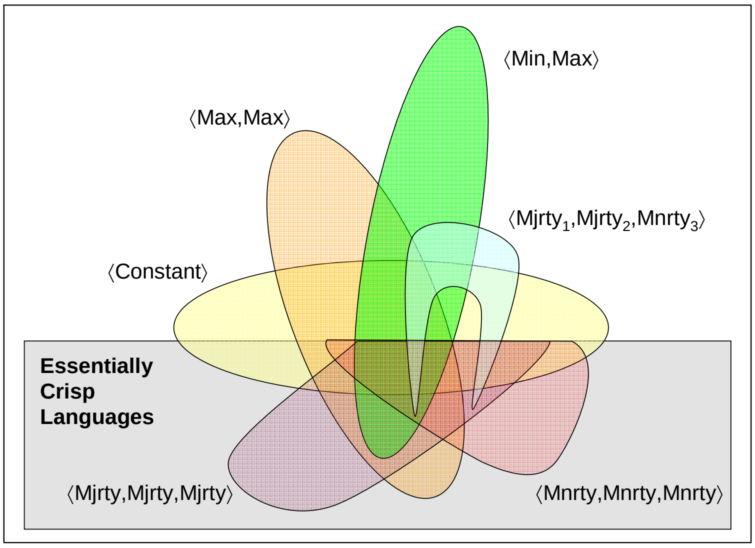

Figure 2: The tractable classes identified in Section 6

Example 8.1 Consider the set of integers $\{0, 1, \ldots, M\}$ for some fixed $M \geq 1$. We can define a valuation structure, Ω_M, on this set by taking the standard ordering, and defining the aggregation operation to be the addition-with-ceiling operation $+_M$, defined as follows:

$$\forall a, b \in \{0, 1, \ldots, M\} \qquad a +_M b \;=\; \min\{a + b, M\}$$

This valuation structure has been shown to be useful to express problems where all solutions which violate M or more constraints are considered equally bad [35]. $\square$

Changing the valuation structure can change the set of multimorphisms associated with a set of functions, as the next example indicates.

Example 8.2 Let Γ be a valued constraint language over a finite set D containing all unary cost functions with range $\{0, 1\}$. For each $d \in D$, Γ contains the unary cost function χ_d, defined as follows:

$$\chi_d(x) = \begin{cases} 1 & \text{if} \quad x = d \\ 0 & \text{otherwise} \end{cases}$$

Hence if $F : D^k \to D^k$ is a multimorphism of Γ, then

$$\forall x_1, \ldots, x_k \in D, \quad \bigoplus_{i=1}^{k} \chi_d(F(x_1, \ldots, x_k)[i]) \leq \bigoplus_{i=1}^{k} \chi_d(x_i). \tag{15}$$

It was shown in Lemma 4.9 that every conservative function is a multimorphism of Γ. If the costs taken by the functions in Γ are defined to be elements of $\overline{\mathbb{R}}_+$, then the converse result also holds: every multimorphism of Γ is conservative. To see this, note that in this case Equation 15 implies that for each $d \in D$, the k-tuple $F(x_1, \ldots, x_k)$ contains at most as many co-ordinate positions equal to d as the tuple $\langle x_1, \ldots, x_k \rangle$. Since this is true for each $d \in D$, it follows that we have equality for each $d \in D$, which means that F is conservative.

However, if the costs are defined to be elements of the valuation structure Ω_M defined in Example 8.1 then this argument no longer holds when $k > M$. For example, when $M = 1$, Γ is the language containing all crisp unary cost functions, which has the multimorphism $\langle \mathrm{Max}, \mathrm{Max} \rangle$, which is not conservative. $\square$

Another possible extension of the results obtained here would be to allow the costs to be chosen from a *partially* ordered set. This additional flexibility is allowed by the semiring-based framework for soft constraints [1, 2]. This framework also allows for other operations to be used in defining what constitutes the preferred cost, rather than simply the minimum. Further investigation is needed to determine whether the notion of a multimorphism can be used to characterise interesting tractable constraint languages in this more general framework.

Other future developments to this work could include the study of approximability properties for optimisation problems involving soft constraints over arbitrary finite sets. This would build on and extend the detailed and successful investigation of approximability properties which has already been completed for Max-Sat and related problems in the Boolean case [10, 30].

References

[1] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, and G. Verfaillie. Semiring-based CSPs and valued CSPs: Frameworks, properties, and comparison. *Constraints*, 4:199–240, 1999.

[2] S. Bistarelli, U. Montanari, and F. Rossi. Constraint solving over semirings. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 624–630, Montreal, Canada, 1995.

[3] E. Boros and P.L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123:155–225, 2002.

[4] A.A. Bulatov. A dichotomy theorem for constraints on a three-element set. In *Proceedings 43rd IEEE Symposium on Foundations of Computer Science, FOCS'02*, pages 649–658. IEEE Computer Society, 2002.

[5] A.A. Bulatov, A.A. Krokhin, and P.G. Jeavons. The complexity of maximal constraint languages. In *Proceedings 33rd ACM Symposium on Theory of Computing, STOC'01*, pages 667–674, 2001.

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

[7] D. Cohen, M. Cooper, P. Jeavons, and A. Krokhin. A maximal tractable class of soft constraints. *Journal of Artificial Intelligence Research*, 22:1–22, 2004.

[8] M.C. Cooper, D.A. Cohen, and P.G. Jeavons. Characterising tractable constraints. *Artificial Intelligence*, 65:347–361, 1994.

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

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

[11] W.H. Cunningham. Minimum cuts, modular functions,and matroid polyhedra. *Networks*, 15(2):205–215, 1985.

[12] E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23(4):864–894, 1994.

[13] V. Dalmau. Generalized majority-minority operations are tractable. In *Proceedings 20th IEEE Symposium on Logic in Computer Science, (LICS 2005)*, pages 438–447. IEEE Computer Society, 2005.

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

[15] 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*, 28:57–104, 1998.

[16] L. Fleischer and S. Iwata. Improved algorithms for submodular function minimization and submodular flow. In *Proceedings of the 32th Annual ACM Symposium on Theory of Computing*, pages 107–116, 2000.

[17] E.C. Freuder and R. Wallace. Partial constraint satisfaction. *Artificial Intelligence*, 58:21–70, 1992.

[18] S. Fujishige and S. Iwata. Bisubmodular function minimization. In *Integer Programming and Combinatorial Optimization: 8th International IPCO Conference*, volume 2081 of *Lecture Notes in Computer Science*, pages 160–169, 2001.

[19] M. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, CA., 1979.

[20] M. Grötschel, L. Lovasz, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–198, 1981.

[21] G. Gutin, A. Rafiey, and A. Yeo. Minimum cost and list homomorphisms to semicomplete digraphs. To appear in Discrete Applied Mathematics.

[22] G. Gutin, A. Rafiey, A. Yeo, and M. Tso. Level of repair analysis and minimum cost homomorphisms of graphs. In *Proceedings of Algorithmic Applications in Management - (AAIM 2005)*, volume 3521 of *Lecture Notes in Computer Science*, pages 427–439. Springer-Verlag, 2005.

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

[24] P.G. Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200:185–204, 1998.

[25] P.G. Jeavons, D.A. Cohen, and M.C. Cooper. Constraints, consistency and closure. *Artificial Intelligence*, 101(1–2):251–265, 1998.

[26] P.G. Jeavons, D.A. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the ACM*, 44:527–548, 1997.

[27] P.G. Jeavons, D.A. Cohen, and M. Gyssens. How to determine the expressive power of constraints. *Constraints*, 4:113–131, 1999.

[28] P.G. Jeavons, D.A. Cohen, and J.K. Pearson. Constraints and universal algebra. *Annals of Mathematics and Artificial Intelligence*, 24:51–67, 1998.

[29] P.G. Jeavons and M.C. Cooper. Tractable constraints on ordered domains. *Artificial Intelligence*, 79(2):327–339, 1995.

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

[31] L. Khatib, P.H. Morris, R.A. Morris, and F. Rossi. Temporal constraint reasoning with preferences. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 322–327, Seattle, US, 2001.

[32] L. Kirousis. Fast parallel constraint satisfaction. *Artificial Intelligence*, 64:147–160, 1993.

[33] A. Krokhin, A. Bulatov, and P. Jeavons. The complexity of constraint satisfaction: an algebraic approach. In *Proceedings of the NATO Advanced Study Institute on Structural Theory of Automata, Semigroups and Universal Algebra*, volume 207 of *NATO Science Series II: Mathematics, Physics and Chemistry*, pages 181–213. Springer-Verlag, 2005.

[34] J. Larrosa, P. Meseguer, and T. Schiex. Maintaining reversible DAC for Max-CSP. *Artificial Intelligence*, 107:149–163, 1999.

[35] J. Larrosa and T. Schiex. Solving weighted CSP by maintaining arc consistency. *Artificial Intelligence*, 159:1–26, 2004.

[36] A.K. Mackworth and E.C. Freuder. The complexity of constraint satisfaction revisited. *Artificial Intelligence*, 59:57–62, 1993.

[37] R. Mohr and G. Masini. Good old discrete relaxation. In Y. Kodratoff, editor, *Proceedings 8th European Conference on Artificial Intelligence — ECAI'88*, pages 651–656. Pitman, 1988.

[38] U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Information Sciences*, 7:95–132, 1974.

[39] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & Sons, 1988.

[40] J.K. Pearson and P.G. Jeavons. A survey of tractable constraint satisfaction problems. Technical Report CSD-TR-97-15, Royal Holloway, University of London, July 1997.

[41] R. Pöschel and L.A. Kalužnin. *Funktionen- und Relationenalgebren.* DVW, Berlin, 1979.

[42] T.J. Schaefer. The complexity of satisfiability problems. In *Proceedings 10th ACM Symposium on Theory of Computing, STOC'78*, pages 216–226, 1978.

[43] 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)*, pages 631–637, Montreal, Canada, 1995.

[44] A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *JCTB: Journal of Combinatorial Theory, Series B*, 80:346–355, 2000.

[45] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics.* Springer, 2003.

[46] A. Szendrei. *Clones in Universal Algebra*, volume 99 of *Seminaires de Mathematiques Superieures.* University of Montreal, 1986.

[47] P. van Hentenryck, Y. Deville, and C-M. Teng. A generic arc-consistency algorithm and its specializations. *Artificial Intelligence*, 57:291–321, 1992.