A Complete Characterization of Complexity for Boolean Constraint Optimization Problems

David Cohen[1], Martin Cooper[2], and Peter Jeavons[3]

[1] Department of Computer Science, Royal Holloway, University of London, UK
d.cohen@rhul.ac.uk
[2] IRIT, University of Toulouse III, France
cooper@irit.fr
[3] Computing Laboratory, University of Oxford, UK
peter.jeavons@comlab.ox.ac.uk

Abstract. We analyze the complexity of optimization problems expressed using valued constraints. This very general framework includes a number of well-known optimization problems such as MAX-SAT, and WEIGHTED MAX-SAT, as well as properly generalizing the classical CSP framework by allowing the expression of preferences. We focus on valued constraints over Boolean variables, and we establish a dichotomy theorem which characterizes the complexity of any problem involving a fixed set of constraints of this kind.

1 Introduction

In the classical constraint satisfaction framework each constraint allows some combinations of values and disallows others. A number of authors have suggested that the usefulness of this framework can be greatly enhanced by extending the definition of a constraint to assign different costs to different assignments, rather than simply allowing some and disallowing others [1]. Problems involving constraints of this form deal with *optimization* as well as feasibility: we seek an assignment of values to all of the variables having the least possible overall combined cost.

In this extended framework a constraint can be seen as a *cost function*, mapping each possible combination of values to a measure of undesirability. Several alternative mathematical frameworks for such cost functions have been proposed in the literature, including the very general frameworks of 'semi-ring based constraints' and 'valued constraints' [1]. For simplicity, we shall adopt the valued constraint framework here (although our results can easily be adapted to the semi-ring framework, for appropriate semi-ring structures). This very general framework includes a number of well-known optimization problems such as MAX-CSP, MAX-SAT, and WEIGHTED MAX-SAT, as well as properly generalizing the classical CSP framework by allowing the expression of preferences.

In general, optimization problems in this framework are NP-hard, so it is natural to investigate what restrictions can be imposed to make them tractable. One way to achieve this is to restrict the form of the cost functions which are allowed in problem instances. Such a restricted set of cost functions is called a **valued constraint language**. In this paper we investigate the complexity of problems involving different kinds of valued constraint languages. We focus on the Boolean case, where each variable can take just two different values, and we obtain a complete characterization of the complexity for all possible valued constraint languages over Boolean variables.

Our results generalize a number of earlier results for particular forms of optimization problem involving Boolean constraints. For example, Creignou *et al* obtained a complete characterization of the complexity of different constraint languages for the WEIGHTED MAX-SAT problem [2], where all costs are finite, and a cost is associated with each individual constraint (rather than with each individual combination of values for that constraint, as we allow here).

2 Definitions

In the valued constraint framework, a constraint is specified by a function which assigns a *cost* to each possible assignment of values for the variables it is constraining. In general, costs may be chosen from any *valuation structure*, satisfying the following definition.

Definition 1. *A* ***valuation structure****, χ, is a totally ordered set, with a minimum and a maximum element (denoted 0 and ∞), together with a commutative, associative binary* ***aggregation operator*** *(denoted +), such that for all $\alpha, \beta, \gamma \in \chi$*

$$\alpha + 0 = \alpha \tag{1}$$

$$\alpha + \gamma \geq \beta + \gamma \quad \textit{whenever} \quad \alpha \geq \beta. \tag{2}$$

In this paper we shall use the valuation structure $\overline{\mathbb{N}}$, consisting of the natural numbers together with infinity, with the usual ordering and the usual addition operation.

Definition 2. *An instance of the valued constraint satisfaction problem, VCSP, is a tuple $\mathcal{P} = \langle V, D, C, \chi \rangle$ where:*

- *V is a finite set of* ***variables;***
- *D is a finite set of possible* ***values*** *for these variables;*
- *χ 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 χ, called the* ***cost function*** *of c.*

Throughout the paper, the ith component of a tuple t will be denoted $t[i]$, and the length of t will be denoted $|t|$. For any two k-tuples u, v we will say that $u \leq v$ if and only if $u[i] \leq v[i]$ for $i = 1, 2, \ldots, k$.

Definition 3. *For any VCSP instance $\mathcal{P} = \langle V, D, C, \chi \rangle$, an* ***assignment*** *for $\mathcal{P}$ is a mapping s from V to D. The* ***cost*** *of an assignment s, denoted $Cost_{\mathcal{P}}(s)$, is given by the sum (i.e., aggregation) of the costs for the restrictions of s onto each constraint scope, that is,*

$$Cost_{\mathcal{P}}(s) = \sum_{\langle\langle v_1, v_2, \ldots, v_m\rangle, \phi\rangle \in C} \phi(s(v_1), s(v_2), \ldots, s(v_m)).$$

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

*Example 1 (**SAT**).* For any instance $\mathcal{P}$ of the standard propositional satisfiability problem, SAT, 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 $\{0, \infty\}$. For each clause c of $\mathcal{P}$, we define a corresponding constraint $\widehat{c}$ of $\widehat{\mathcal{P}}$ with the same scope; the cost function of $\widehat{c}$ maps each tuple of values allowed by c to 0, and each tuple disallowed by c to ∞.

In this case the cost of an assignment s for $\widehat{\mathcal{P}}$ equals the minimal possible cost, 0, if and only if s satisfies all of the clauses of $\mathcal{P}$.

*Example 2 (**MAX-SAT**).* In the standard MAX-SAT problem the aim is to find an assignment to the variables which maximizes the number of satisfied clauses. For any instance $\mathcal{P}$ of MAX-SAT, 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 $\{0, 1\}$. For each clause c of $\mathcal{P}$, we define a corresponding constraint $c^{\#}$ of $\mathcal{P}^{\#}$ with the same scope; the cost function of $c^{\#}$ maps each tuple of values allowed by c to 0, and each tuple disallowed by c to 1.

In this case the cost of an assignment s for $\mathcal{P}^{\#}$ equals the total number of clauses of $\mathcal{P}$ which are violated by s. Hence a solution to $\mathcal{P}^{\#}$ corresponds to an assignment of $\mathcal{P}$ which violates the minimum number of clauses, and hence satisfies the maximum number of clauses.

A similar construction can be carried out for the weighted version of the MAX-SAT problem.

The problem of finding a solution to a valued constraint satisfaction problem is an NP optimization problem, that is, it lies in the complexity class NPO (see [2] for a formal definition of this class). It follows from Examples 1 and 2 that there is a polynomial-time reduction from some known NP-hard problems to the general VCSP. To achieve more tractable versions of VCSP, we will now consider the effect of restricting the forms of cost function allowed in the constraints.

Definition 4. *Let χ be a valuation structure. A* ***valued Boolean constraint language*** *with costs in χ is defined to be a set of functions, Γ, such that each $\phi \in \Gamma$ is a function from $\{0, 1\}^m$ to χ, for some natural number m, 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.*

For any valued Boolean constraint language Γ, if every instance in VCSP(Γ) can be solved in polynomial time then we will say that Γ is **tractable**. On the other hand, if there is a polynomial-time reduction from some NP-hard problem to VCSP(Γ), then we shall say that Γ is **NP-hard**.

*Example 3 (**SAT and MAX-SAT**).* Let Γ be any valued Boolean constraint language.

If we restrict Γ by only allowing functions with range $\{0, \infty\}$, as in Example 1, then each problem VCSP(Γ) corresponds precisely to a classical Boolean constraint satisfaction problem. Such problems are sometimes known as Generalized Satisfiability problems [3]. The complexity of VCSP(Γ) for such restricted sets Γ has been completely characterized, and the six tractable cases have been identified [3, 2].

Alternatively, if we restrict Γ by only allowing functions whose range has exactly two finite values including 0, as in Example 2, then each VCSP(Γ) corresponds precisely to a standard Weighted MAX-SAT problem [2], in which the aim is to find an assignment in which the total weight of satisfied clauses is maximized. The complexity of VCSP(Γ) for such restricted sets Γ has been completely characterized, and the three tractable cases have been identified (see Theorem 7.6 of [2]).

We note, in particular, that when Γ contains just the single binary function ϕ_{XOR} defined by

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

then VCSP(Γ) corresponds to the MAX-SAT problem for the exclusive-or predicate, which is known to be NP-hard (see Lemma 7.4 of [2]).

In an earlier paper [4], we introduced the idea of *expressing* a desired cost function by using a combination of available functions. The next two definitions formalize this idea .

Definition 5. *For any VCSP instance $\mathcal{P} = \langle V, D, C, \chi\rangle$, and any tuple of distinct variables $W = \langle v_1, \ldots, v_k\rangle$, the **cost function for** $\mathcal{P}$ **on** W, denoted $\Phi_{\mathcal{P}}^{W}$, is defined as follows:*

$$\Phi_{\mathcal{P}}^{W}(d_1, \ldots, d_k) = \min\{\, Cost_{\mathcal{P}}(s) \mid s : V \to D, \langle s(v_1), \ldots, s(v_k)\rangle = \langle d_1, \ldots, d_k\rangle\}$$

Definition 6. *A function ϕ is **expressible** over a valued constraint language Γ if there exists an instance $\mathcal{P} = \langle V, D, C, \chi\rangle$ 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 a valued constraint language Γ is denoted Γ^*.

The notion of expressibility has already been shown to be a key tool in analyzing the complexity of valued constraint languages, as the next result indicates.

Proposition 1 ([4]). *Let Γ and Γ' be valued constraint languages, with Γ' finite. If $\Gamma' \subseteq \Gamma^*$, then* VCSP($\Gamma'$) *is polynomial-time reducible to* VCSP(Γ).

The next result shows how Proposition 1 can be used to establish NP-hardness of a valued Boolean constraint language.

Corollary 1. *Let Γ be a valued Boolean constraint language, with costs in χ. If Γ^* contains a binary function ϕ_{XOR+} defined by*

$$\phi_{XOR+}(x,y) = \begin{cases} \alpha \text{ if } & x \neq y \\ \beta \text{ if } & x = y \end{cases}$$

for some $\alpha < \beta < \infty$, then VCSP(Γ) *is NP-hard.*

Proof. Lemma 7.4 of [2] states that the Boolean problem VCSP($\{\phi_{XOR}\}$) is NP-hard, where ϕ_{XOR} is the Boolean exclusive-or function, as defined in Example 3. 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, we conclude that VCSP($\{\phi_{XOR+}\}$) is also NP-hard. □

A similar notion of expressibility has been used extensively for classical constraints, which are specified using relations, rather than cost functions [5–7]. It has been shown that the relations expressible by a given set of relations are determined by certain algebraic invariance properties of those relations, known as *polymorphisms* [5, 6, 8]. A **polymorphism** of R, as defined in [5, 8], is a function $f : D^k \to D$, for some k, with the property that whenever $t_1, \ldots, t_k$ are in R then so is $\langle f(t_1[1], \ldots, t_k[1]), \ldots, f(t_1[m], \ldots, t_k[m])\rangle$.

The concept of a polymorphism is specific to *relations*, and cannot be applied directly to the *functions* in a valued constraint language. However, we now define a more general notion, introduced in [4], which we call a **multimorphism**, and which does apply directly to functions.

Definition 7. *Let D be a set, χ a valuation structure, and $\phi : D^m \to \chi$ a function.*

We extend the definition of ϕ in the following way: for any positive integer k, and any list of k-tuples, $t_1, t_2, \ldots, t_m$, over D, we define

$$\phi(t_1, t_2, \ldots, t_m) = \sum_{i=1}^{k} \phi(t_1[i], t_2[i], \ldots, t_m[i])$$

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*

$$\phi(F(t_1), F(t_2), \ldots, F(t_m)) \leq \phi(t_1, t_2, \ldots, t_m).$$

For any valued constraint language Γ we will say that Γ has a multimorphism F if and only if F is a multimorphism of ϕ for each $\phi \in \Gamma$.

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

Theorem 1 ([4]). *Every multimorphism of a valued constraint language Γ is also a multimorphism of Γ^*.*

In the rest of the paper we will usually denote a multimorphism $F : D^k \to D^k$ by listing explicitly the k separate *component* functions $F_i : D^k \to D$, given by $F_i(x_1, x_2, \ldots, x_k) = F(x_1, x_2, \ldots, x_k)[i]$.

It is shown in [4] that if $F : D^k \to D^k$ is a multimorphism of a function ϕ, then each of the component functions, F_i, is a polymorphism of the corresponding *feasibility relation*, $\mathrm{Feas}(\phi)$, which is defined as follows:

Definition 8. *For any function ϕ, with arity m, we define a relation known as the **feasibility relation** of ϕ, and denoted $\mathrm{Feas}(\phi)$, as follows:*

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

3 Dichotomy Theorem

In this section we will show that in the Boolean case all the *tractable* valued constraint languages are precisely characterized by the presence of certain forms of multimorphism. In fact we establish a dichotomy result: if a valued constraint language has one of eight types of multimorphism then it is tractable, otherwise it is NP-hard.

Theorem 2. *For any valued Boolean constraint language Γ, 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. $\langle \max, \max \rangle$*, where* max *is the binary function returning the maximum of its arguments (i.e.,* $\max(x, y) = x \vee y$*);*
4. $\langle \min, \min \rangle$*, where* min *is the binary function returning the minimum of its arguments (i.e.,* $\min(x, y) = x \wedge y$*);*
5. $\langle \min, \max \rangle$*;*
6. $\langle \mathrm{Mjty}, \mathrm{Mjty}, \mathrm{Mjty} \rangle$*, where* Mjty *is the ternary* majority *function, (i.e., it satisfies the identity* $\mathrm{Mjty}(x, x, y) = \mathrm{Mjty}(x, y, x) = \mathrm{Mjty}(y, x, x) = x$*);*
7. $\langle \mathrm{Mnty}, \mathrm{Mnty}, \mathrm{Mnty} \rangle$*, where* Mnty *is the ternary* minority *function, (i.e., it satisfies the identity* $\mathrm{Mnty}(x, x, y) = \mathrm{Mnty}(x, y, x) = \mathrm{Mnty}(y, x, x) = y$*);*
8. $\langle \mathrm{Mjty}, \mathrm{Mjty}, \mathrm{Mnty} \rangle$*;*

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

We shall prove this result using a series of lemmas in Sections 3.1 and 3.2.

A cost function ϕ will be called **essentially classical** 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 classical cost functions only will be called an **essentially classical language**. Note that when Γ is an essentially classical language any assignment with finite cost has the same cost as any other assignment with finite cost. Hence we can solve any instance of $\mathrm{VCSP}(\Gamma)$ for such languages by solving the corresponding classical constraint satisfaction problem in which each valued constraint $\langle \sigma, \phi \rangle$ is replaced by the classical constraint $\langle \sigma, \text{Feas}(\phi) \rangle$ (see Definition 8). Hence the complexity of any essentially classical valued Boolean constraint language can be determined using Schaefer's Dichotomy Theorem for classical Boolean constraints [3, 6]. We will use this observation a number of times in the course of the proof.

3.1 Tractable cases

To establish the first part of Theorem 2, we must show that a valued Boolean constraint language which has one of the eight types of multimorphisms listed in the theorem is tractable.

We first note that the tractability of any valued constraint language (not necessarily Boolean) which has a multimorphism of one of the first two types listed in Theorem 2 was established in Theorem 2 of [4]. Furthermore, the tractability of any valued constraint language (not necessarily Boolean) which has a multimorphism of the third type listed in Theorem 2 was established in Theorem 4 of [4], and a symmetric argument (with the ordering reversed) establishes the tractability of any valued constraint language with a polymorphism of the fourth type. Finally, the tractability of any valued Boolean constraint language which has a multimorphism of the last type listed in Theorem 2 follows immediately from Theorem 5 of [4].

Hence, for the first part of the proof we only need to establish the tractability of valued constraint languages having one of the remaining three types of multimorphisms listed in Theorem 2. This is done in the next three lemmas.

Lemma 1. *Any valued Boolean constraint language which has the multimorphism* $\langle \text{Mjty}, \text{Mjty}, \text{Mjty} \rangle$ *is essentially classical, and tractable.*

Proof. Let ϕ be a k-ary cost function which has the multimorphism $\langle \text{Mjty}, \text{Mjty}, \text{Mjty} \rangle$. It follows from the definition of the Mjty function and the definition of a multimorphism 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. Hence ϕ is essentially classical, so Γ is essentially classical.

Furthermore, since for each $\phi \in \Gamma$, $\text{Feas}(\phi)$ has the polymorphism Mjty, it follows from Schaefer's Dichotomy Theorem [3, 6] that $\text{VCSP}(\Gamma)$ is tractable. $\square$

Lemma 2. *Any valued Boolean constraint language which has the multimorphism* $\langle \text{Mnty}, \text{Mnty}, \text{Mnty} \rangle$ *is essentially classical, and tractable.*

Proof. Let ϕ be a k-ary cost function which has the multimorphism $\langle \text{Mnty}, \text{Mnty}, \text{Mnty} \rangle$. It follows from the definition of the Mnty function and the definition of a multimorphism that for all $x, y \in D^k$, $3\phi(x) \leq \phi(x) + \phi(y) + \phi(y)$ 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. Hence ϕ is essentially classical, so Γ is essentially classical.

Furthermore, since for each $\phi \in \Gamma$, $\text{Feas}(\phi)$ has the polymorphism Mnty, and the Mnty operation is the affine operation over the field with 2 elements, it follows from Schaefer's Dichotomy Theorem [3, 6] that $\text{VCSP}(\Gamma)$ is tractable. $\square$

The only remaining case for which we need to establish tractability is for valued Boolean constraint languages which have the multimorphism $\langle\min, \max\rangle$. It was shown in [4] that valued constraint languages with this multimorphism are closely connected with the submodular set functions used in economics and operations research [9]. Because of this connection, we shall refer to valued constraint languages with the multimorphism $\langle\min, \max\rangle$ as **submodular** languages.

It was established in Theorem 3 of [4] that finite-valued submodular languages are tractable. We now generalize this result in the Boolean case to include all submodular languages, including those where the cost functions take infinite values.

Lemma 3. *Any submodular valued Boolean constraint language is tractable.*

Proof. Let Γ be a submodular valued Boolean constraint language, and let $\mathcal{P}$ be any instance of VCSP(Γ).

Every cost function in $\mathcal{P}$ has the multimorphism $\langle\min, \max\rangle$, and so the corresponding feasibility relation has the polymorphisms min and max. Since the polymorphism Mjty can be obtained by composition from min and max, it follows that each of these feasibility relations has the polymorphism Mjty, and hence is decomposable into *binary* relations [10]. Hence we can determine whether or not $\mathcal{P}$ has a solution with finite cost by solving a corresponding instance of 2-SAT, which can be solved in polynomial-time.

If $\mathcal{P}$ has any finite cost solution, then we can find a solution with minimal cost by solving a submodular minimisation problem over the set of all solutions allowed by the feasibility relations of the constraints of $\mathcal{P}$. This set has the polymorphisms min and max, and hence forms a distributive lattice. A polynomial-time algorithm for minimising a submodular function over a distributive lattice is given in [11]. $\square$

3.2 Intractable cases

To establish the remaining part of Theorem 2, we must show that a valued Boolean constraint language which does not have any of the types of multimorphisms listed in the theorem is NP-hard. We first deal with essentially classical languages.

Lemma 4. *Any valued Boolean constraint language which is essentially classical and does not have any of the multimorphisms listed in Theorem 2 is NP-hard.*

Proof. If we replace each cost function ϕ in Γ with the relation Feas(ϕ) then we obtain a classical Boolean constraint language Γ' which does not have any of the polymorphisms $\mathbf{0}, \mathbf{1}, \min, \max, \mathrm{Mjty}$ or Mnty.

By Schaefer's Dichotomy Theorem [3, 6], 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 2 can express certain special functions, which we now define.

Definition 9. *A unary cost function σ 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 cost function ϕ is a* ***NEQ*** *function if*

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

– *A binary cost function ϕ is an* ***XOR*** *function if*

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

Lemma 5. *Let Γ be a valued Boolean constraint language which is not essentially classical.*

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 $\Phi^{\langle y\rangle}_{\mathcal{P}_0}$ 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 classical. 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 $\psi = \Phi^W_{\mathcal{P}}$.

Note 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 $\Phi^{\langle x_{01}\rangle}_{\mathcal{P}}$ 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 $\Phi^{\langle y\rangle}_{\mathcal{P}_2}$ 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. □

Lemma 6. *Let Γ be a valued Boolean constraint language which is not essentially classical, 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 cost function which does not have the multimorphism $\langle \mathbf{0} \rangle$, and $\phi_1 \in \Gamma$ be a cost 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 costs 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 $\langle \langle x_0 \rangle, M\sigma_0 \rangle$. 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 cost 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 5 to complete the proof. $\square$

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

Construction 1. *Let ϕ be any m-ary cost 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 costs 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 cost function $\phi_2 \stackrel{\text{def}}{=} \Phi_{\mathcal{P}}^{\langle x_{01}, x_{10} \rangle}$ will be called a ***compression*** *of ϕ by u and v.*

Lemma 7. *A function ϕ has a $\langle \max, \max \rangle$ multimorphism if and only if*

- *ϕ is* ***finitely antitone****, that is, for all tuples u, v with $\phi(u), \phi(v) < \infty$,*

$$u < v \Rightarrow \phi(u) \geq \phi(v).$$

- $\mathrm{Feas}(\phi)$ *has the polymorphism* max.

Proof. If ϕ has a $\langle \max, \max \rangle$ multimorphism, then for all tuples u, v we have $\phi(u) + \phi(v) \geq 2\phi(\max(u, v))$, so both conditions hold.

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

Lemma 8. *Let Γ be a valued Boolean constraint language which is not essentially classical, and does not have any of the multimorphisms $\langle \mathbf{0} \rangle$ or $\langle \mathbf{1} \rangle$ or $\langle \max, \max \rangle$ or $\langle \min, \min \rangle$.*

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

Proof. Let ϕ be a cost function in Γ which does not have a $\langle \max, \max \rangle$ multimorphism, and let ψ be a cost function in Γ which does not have a $\langle \min, \min \rangle$ multimorphism.

By Lemma 6, 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 $\langle \max, \max \rangle$ multimorphism, it follows from Lemma 7 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 1). 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 $\phi(\max(u,v)) = \infty$. 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 $\phi(u), \phi(v) \leq \phi(\min(u,v))$. This, together with the fact that $\phi(u)$ and $\phi(v)$ are finite whilst $\phi(\max(u,v))$ 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 $\phi(\max(u,v))$ and $\psi(\min(w,z))$ 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 (see Construction 1). We then have that $\rho(x,y) \stackrel{\mathrm{def}}{=} \phi_2(x,y) + \phi_2(y,x) + \psi_2(x,y) + \psi_2(y,x)$ is a NEQ function which is contained in Γ^*. We can now appeal to Lemma 5 to show that Γ^* contains an XOR function, and we are done. □

Lemma 9. *Let Γ be a valued Boolean constraint language which does not have the multimorphism $\langle \min, \max \rangle$.*

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

Proof. Suppose that $\sigma_0 \in \Gamma^*$ is a finite 0-selector, $\sigma_1 \in \Gamma^*$ is a finite 1-selector, and $\phi \in \Gamma$ is non-submodular (i.e., ϕ does not have the multimorphism $\langle \min, \max \rangle$).

Set $\lambda = \sigma_0(1) - \sigma_0(0)$ and $\mu = \sigma_1(0) - \sigma_1(1)$.

Choose u, v such that $\phi(u) + \phi(v) < \phi(\max(u,v)) + \phi(\min(u,v))$. Let ϕ_2 be a compression of ϕ by u and v. It is straightforward to check that ϕ_2 is also not submodular.

There are three cases to consider:

Case (1): $\phi_2(0,0), \phi_2(1,1) < \infty$.

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

$$\begin{array}{ll} \langle \langle x,y \rangle, 2\lambda\mu\phi_2 \rangle, & \\ \langle \langle x \rangle, \lambda(\phi_2(1,0) + \phi_2(1,1))\sigma_1 \rangle, & \langle \langle x \rangle, \mu(\phi_2(0,0) + \phi_2(0,1))\sigma_0 \rangle, \\ \langle \langle y \rangle, \lambda(\phi_2(0,1) + \phi_2(1,1))\sigma_1 \rangle, & \langle \langle y \rangle, \mu(\phi_2(0,0) + \phi_2(1,0))\sigma_0 \rangle. \end{array}$$

If we set $W = \langle x, y \rangle$, then it is straightforward to check that $\Phi_{\mathcal{P}}^W$ is an XOR function.

Case(2): *Exactly* one of $\phi_2(0,0)$ and $\phi_2(1,1)$ is finite.

First suppose that $\phi_2(0,0) = \infty > \phi_2(1,1)$. Let $\alpha = \max\{\phi_2(0,1) + \phi_2(1,0) - 2\phi_2(1,1) + 1, 0\}$, and construct an instance $\mathcal{P}_2 \in \text{VCSP}(\Gamma^*)$ with variables $\{x, u, v, y\}$, and constraints

$$\begin{array}{ll} \langle \langle x,u \rangle, \mu\phi_2 \rangle, & \langle \langle u,x \rangle, \mu\phi_2 \rangle, \\ \langle \langle u,v \rangle, \mu\phi_2 \rangle, & \langle \langle v,u \rangle, \mu\phi_2 \rangle, \\ \langle \langle v,y \rangle, \mu\phi_2 \rangle, & \langle \langle y,v \rangle, \mu\phi_2 \rangle, \\ \langle \langle x \rangle, \alpha\sigma_1 \rangle, & \langle \langle u \rangle, 2\alpha\sigma_1 \rangle, \\ \langle \langle v \rangle, 2\alpha\sigma_1 \rangle, & \langle \langle y \rangle, \alpha\sigma_1 \rangle. \end{array}$$

If we set $W = \langle x, y \rangle$, and $\eta = \Phi_{\mathcal{P}_2}^W$, then it is straightforward to verify that $\eta(0,1) = \eta(1,0)$, $\eta(0,0) = \eta(1,1)$, and

$$\eta(0,0) = \eta(0,1) + \mu(\alpha + 2\phi_2(1,1) - \phi_2(0,1) - \phi_2(1,0)),$$

and hence that η is an XOR function.

A symmetric argument clearly works when $\phi_2(1,1) = \infty > \phi_2(0,0)$.

Case (3): $\phi_2(0,0) = \phi_2(1,1) = \infty$.

In this case the function $\phi_2(x,y) + \phi_2(y,x)$ is a NEQ function which is clearly contained in Γ^*.

□

Lemma 10. *A Boolean function ϕ has a $\langle \text{Mjty}, \text{Mjty}, \text{Mnty} \rangle$ multimorphism if and only if:*

– ϕ *is **finitely modular**, that is, for all tuples* u, v *with* $\phi(u)$, $\phi(v)$, $\phi(\max(u,v))$, $\phi(\min(u,v)) < \infty$,

$$\phi(u) + \phi(v) = \phi(\max(u,v)) + \phi(\min(u,v)).$$

– $\mathrm{Feas}(\phi)$ *has the polymorphisms* Mjty *and* Mnty.

Proof. Follows immediately from the characterization of valued Boolean constraint languages with a $\langle \mathrm{Mjty}, \mathrm{Mjty}, \mathrm{Mnty} \rangle$ multimorphism given in Theorem 4.17 of [12]. $\square$

Lemma 11. *Let Γ be a valued Boolean constraint language which does not have the multimorphism* $\langle \mathrm{Mjty}, \mathrm{Mjty}, \mathrm{Mnty} \rangle$.

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 $\langle \mathrm{Mjty}, \mathrm{Mjty}, \mathrm{Mnty} \rangle$. We have to show that Γ^* also contains an XOR function.

By Lemma 10 there are 2 cases: either ϕ is not finitely modular, or $\mathrm{Feas}(\phi)$ does not have both polymorphisms Mjty and Mnty.

In the first case, choose tuples u, v such that $\phi(u) + \phi(v) \neq \phi(\min(u,v)) + \phi(\max(u,v))$. 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 $\mathrm{Feas}(\phi)$ has the polymorphism Mjty but not Mnty. In this case, by Theorem 3.5 of [10], $\mathrm{Feas}(\phi)$ is decomposable into binary relations (in other words, it is equal to the relational join of its binary projections). Since $\mathrm{Feas}(\phi)$ does not have the Mnty polymorphism, this implies that one of its binary projections does not have the Mnty polymorphism. The only binary Boolean relations which do not have the Mnty polymorphism have exactly three tuples. Therefore, by projection, it is possible to construct from ϕ a binary cost 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 cost function η is non-submodular and exactly one of $\eta(0,0)$ and $\eta(1,1)$ are infinite, and so, by Case 2 of Lemma 9, Γ^* contains an XOR function.

Suppose now that $\mathrm{Feas}(\phi)$ has the polymorphism Mnty but not Mjty. Since $\mathrm{Feas}(\phi)$ has the polymorphism Mnty, it is an *affine* relation [2] 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 [2]. 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 Mjty and Mnty polymorphisms. Thus Feas(ϕ) is affine but not of width 2. Hence, by Lemma 5.34 of [2], Feas(ϕ) 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 cost 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 non-submodular function where both $\eta(0,0)$ and $\eta(1,1)$ are finite. Hence, by Case 1 of Lemma 9, the result follows in this case also.

Finally, if Feas(ϕ) has neither the polymorphism Mnty nor Mjty, then the set of Boolean relations $\{\text{Feas}(\phi), \text{Feas}(\nu)\}$ can be shown to have essentially unary polymorphisms only (see Theorem 4.12 of [7]). By Theorem 4.10 of [7], this implies that in this case Feas(ϕ) can again be used to construct the 4-ary affine constraint $w+x+y+z=0$, and we can proceed as above. □

Lemma 12. *Let Γ be a valued Boolean constraint language which does not have any of the multimorphisms listed in Theorem 2.*

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

Proof. Suppose that Γ is not essentially classical and has none of the multimorphisms listed in Theorem 2. By Lemmas 9 and 8, 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 11 we know that Γ^* contains an XOR function. □

Combining Lemmas 4 and 12, together with Corollary 1, establishes the NP-hardness of any valued Boolean constraint language having none of the multimorphisms listed in Theorem 2, and so completes the proof of Theorem 2.

4 Some Special Cases

Corollary 2. *Let Γ be a valued Boolean constraint language Γ where all costs are finite. If Γ has one of the multimorphisms $\langle\mathbf{0}\rangle$, $\langle\mathbf{1}\rangle$, or $\langle\min,\max\rangle$, then* VCSP(Γ) *is tractable. In all other cases* VCSP(Γ) *is NP-hard.*

Proof. Let ϕ be a cost function taking *finite values only.* By Lemma 7, if ϕ has the multimorphism $\langle\max,\max\rangle$, then ϕ is antitone, and hence has the multimorphism $\langle\mathbf{1}\rangle$. By a symmetric argument, if ϕ has the multimorphism $\langle\min,\min\rangle$, then ϕ is monotone, and hence has the multimorphism $\langle\mathbf{0}\rangle$. By Lemma 1, if ϕ has the multimorphism $\langle \text{Mjty}, \text{Mjty}, \text{Mjty} \rangle$, then ϕ is constant, and hence has the multimorphism $\langle \mathbf{0} \rangle$. By Lemma 2, if ϕ has the multimorphism $\langle \text{Mnty}, \text{Mnty}, \text{Mnty} \rangle$, then ϕ is again constant, and hence has the multimorphism $\langle \mathbf{0} \rangle$. By Lemma 10, if ϕ has the multimorphism $\langle \text{Mjty}, \text{Mjty}, \text{Mnty} \rangle$, then ϕ is modular, and hence submodular, that is, it has the multimorphism $\langle \min, \max \rangle$.

The result now follows from Theorem 2. □

Using the construction given in Example 2, this immediately gives a dichotomy theorem for the MAX-SAT problem for any Γ corresponding to a set of relations.

Corollary 3. *If Γ has one of the multimorphisms $\langle \mathbf{0} \rangle$, $\langle \mathbf{1} \rangle$, or $\langle \min, \max \rangle$, then MAX-SAT($\Gamma$) is tractable. In all other cases MAX-SAT(Γ) is NP-hard.*

This result gives an alternative description to the one given in Theorem 7.6 of [2] for the three tractable cases of MAX-SAT.

References

1. Bistarelli, S., Fargier, H., Montanari, U., Rossi, F., Schiex, T., Verfaillie, G.: Semiring-based CSPs and valued CSPs: Frameworks, properties, and comparison. Constraints **4** (1999) 199–240
2. Creignou, N., Khanna, S., Sudan, M.: Complexity Classification of Boolean Constraint Satisfaction Problems. Volume 7 of SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics (2001)
3. Schaefer, T.: The complexity of satisfiability problems. In: Proceedings 10th ACM Symposium on Theory of Computing, STOC'78. (1978) 216–226
4. Cohen, D., Cooper, M., Jeavons, P., Krokhin, A.: Soft constraints: complexity and multimorphisms. In: Proceedings of 9th International Conference on Principles and Practice of Constraint Programming (CP'03). Volume 2833 of Lecture Notes in Computer Science., Springer-Verlag (2003) 244–258
5. Bulatov, A., Krokhin, A., Jeavons, P.: Constraint satisfaction problems and finite algebras. In: Proceedings ICALP'00. Volume 1853 of Lecture Notes in Computer Science., Springer-Verlag (2000) 272–282
6. Jeavons, P., Cohen, D., Gyssens, M.: Closure properties of constraints. Journal of the ACM **44** (1997) 527–548
7. Jeavons, P.: On the algebraic structure of combinatorial problems. Theoretical Computer Science **200** (1998) 185–204
8. Pöschel, R., Kalužnin, L.: Funktionen- und Relationenalgebren. DVW, Berlin (1979)
9. Nemhauser, G., Wolsey, L.: Integer and Combinatorial Optimization. John Wiley & Sons (1988)
10. Jeavons, P., Cohen, D., Cooper, M.: Constraints, consistency and closure. Artificial Intelligence **101** (1998) 251–265
11. Iwata, S.: A faster scaling algorithm for minimizing submodular functions. SIAM Journal on Computing **32** (2003) 833–840
12. Cohen, D., Cooper, M., Jeavons, P., Krokhin, A.: An investigation of the multimorphisms of tractable and intractable classes of valued constraints. Technical Report CSD-TR-03-03, CS department, Royal Holloway, University of London (2003)