Computing Science

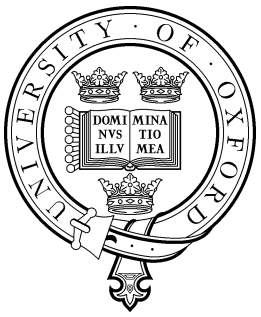
THE EXPRESSIVE POWER OF VALUED CONSTRAINTS: HIERARCHIES AND COLLAPSES

David A. Cohen,
Dept. of Computer Science, Royal Holloway,
University of London, UK
dave@cs.rhul.ac.uk

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

Stanislav Živný,
Computing Laboratory, University of Oxford, UK
stanislav.zivny@comlab.ox.ac.uk

CS-RR-07-03



Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford OX1 3QD

Abstract

In this paper we investigate the ways in which a fixed collection of valued constraints can be combined to express other valued constraints. We show that in some cases a large class of valued constraints, of all possible arities, can be expressed by using valued constraints of a fixed finite arity. We also show that some simple classes of valued constraints, including the set of all monotonic valued constraints with finite cost values, cannot be expressed by a subset of any fixed finite arity, and hence form an infinite hierarchy.

1 Introduction

Building a computational model of a combinatorial problem means capturing the requirements and optimisation criteria of the problem using the resources available in some given computational system. Modelling such problems using *constraints* means expressing the requirements and optimisation criteria using some combination of basic constraints provided by the system. In this paper we investigate what kinds of relationships and functions can be expressed using a given set of allowed constraint types.

The classical constraint satisfaction problem (CSP) model considers only the feasibility of satisfying a collection of simultaneous requirements. Various extensions have been proposed to this model to allow it to deal with different kinds of optimisation criteria or preferences between different feasible solutions. Two very general extended frameworks that have been proposed are the semi-ring CSP framework and the valued CSP (VCSP) framework [2]. The semi-ring framework is slightly more general, but the VCSP framework is simpler, and sufficiently powerful to describe many important classes of problems [7, 19].

In this paper we work with the VCSP framework. In this framework every constraint has an associated cost function which assigns a cost to every tuple of values for the variables in the scope of the constraint. The set of cost functions used in the description of the problem is called the *valued constraint language*.

As with all computing paradigms, it is desirable for many purposes to have a small language which can be used to describe a large collection of problems. Determining which problems can be expressed in a given language is therefore a central issue in assessing the flexibility and usefulness of a constraint system, and it is this question that we investigate here.

We make use of a number of algebraic tools that have been developed for this question [15], and for the related question of determining the complexity of a constraint language [4, 7]. By applying these tools to particular valued constraint languages, we show that some simple constraint classes provide infinite hierarchies of greater and greater expressive power, whereas other classes collapse to sets of cost functions of fixed arity which can express all the other cost functions in the class.

The paper is organised as follows. In Section 2, we define the standard valued constraint satisfaction problem and the notion of expressibility for valued constraints. In Section 3, we describe some algebraic techniques that have been developed for valued constraints in earlier papers and show how they can be used to investigate expressibility.

In Section 4, we show that relations of a fixed arity can express any relation of any arbitrary arity. We show the same result for max-closed relations. In Section 5, by contrast, we show that the *finite-valued* max-closed cost functions form an infinite hierarchy. In other words, finite-valued max-closed cost functions of different arities have different expressive power. In Section 6, we show a collapse to finite arity for general cost functions taking both finite and infinite values. Finally in Section 7, we summarise our results and suggest some important open questions.

2 Valued Constraints and Expressibility

In this section we define the valued constraint satisfaction problem and discuss how the cost functions used to define valued constraints can be combined to express other valued constraints. More detailed discussion of the valued constraint framework, and illustrative examples, can be found in [2, 7].

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**, $\oplus$, 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 $\mathcal{P} = \langle V, D, \mathcal{C}, \Omega \rangle$ where:*

- *V is a finite set of **variables**;*
- *D is a finite set of possible **values**;*
- *Ω is a valuation structure representing possible **costs**;*
- *$\mathcal{C}$ is a set of **valued constraints**. Each element of $\mathcal{C}$ is a pair $c = \langle \sigma, \phi \rangle$ where σ is a tuple of variables called the **scope** of c, and $\phi \in \Gamma$ is a mapping from $D^{|\sigma|}$ to Ω, called the **cost function** of c.*

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

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

*A **solution** to $\mathcal{P}$ is an assignment with minimal cost.*

The complexity of finding an optimal solution to a valued constraint problem will obviously depend on the forms of valued constraints which are allowed in the problem [7]. In order to investigate different families of valued constraint problems with different sets of allowed constraint types, we use the notion of a **valued constraint language**, which is simply a set of possible cost functions mapping D^k to Ω, for some fixed set D and some

fixed valuation structure Ω. The class of all VCSP instances where the cost functions of the valued constraints are all contained in a valued constraint language Γ will be denoted VCSP(Γ).

In any VCSP instance, the variables listed in the scope of each valued constraint are explicitly constrained, in the sense that each possible combination of values for those variables is associated with a given cost. Moreover, if we choose *any* subset of the variables, then their values are constrained implicitly in the same way, due to the combined effect of the valued constraints. This motivates the concept of **expressibility** for cost functions, which is defined as follows:

Definition 2.4. *For any* VCSP *instance* $\mathcal{P} = \langle V, D, \mathcal{C}, \Omega \rangle$*, and any list* $l = \langle v_1, \ldots, v_m \rangle$ *of variables of* $\mathcal{P}$*, the* ***projection*** *of* $\mathcal{P}$ *onto* l*, denoted* $\pi_l(\mathcal{P})$*, is the* m*-ary cost function defined as follows:*

$$\pi_l(\mathcal{P})(x_1, \ldots, x_m) \stackrel{\text{def}}{=} \min_{\{s:V \to D \mid \langle s(v_1), \ldots, s(v_m) \rangle = \langle x_1, \ldots, x_m \rangle\}} Cost_{\mathcal{P}}(s).$$

We say that a cost function ϕ *is* ***expressible*** *over a valued constraint language* Γ *if there exists an instance* $\mathcal{P} \in$ VCSP(Γ) *and a list* l *of variables of* $\mathcal{P}$ *such that* $\pi_l(\mathcal{P}) = \phi$*. We call the pair* $\langle \mathcal{P}, l \rangle$ *a* ***gadget*** *for expressing* ϕ *over* Γ*.*

In this paper we shall examine the expressibility of cost functions over three particular valuation structures which can be used to model a wide variety of problems [7]:

Definition 2.5. *Let* Ω *be a valuation structure and let* $\phi : D^m \to \Omega$ *be a cost function.*

- *If* $\Omega = \{0, \infty\}$*, then we call* ϕ *a* ***crisp*** *cost function.*
- *If* $\Omega = \mathbb{Q}_+$*, the set of non-negative rational numbers with the standard addition operation,* $+$*, then we call* ϕ *a* ***finite-valued*** *cost function.*
- *If* $\Omega = \overline{\mathbb{Q}}_+$*, the set of non-negative rational numbers together with infinity, with the standard addition operation (extended so that* $a + \infty = \infty$*, for every* $a \in \overline{\mathbb{Q}}_+$*), then we call* ϕ *a* ***general*** *cost function.*

Note that with any *relation* R over D we can associate a crisp cost function ϕ_R on D which maps tuples in R to 0 and tuples not in R to ∞. On the other hand, with any m-ary cost function ϕ we can associate a relation R_ϕ defined as $\langle x_1, \ldots, x_m \rangle \in R_\phi \Leftrightarrow \phi(x_1, \ldots, x_m) < \infty$, or equivalently an m-ary crisp cost function defined by:

$$\mathsf{Feas}(\phi)(x_1, \ldots, x_m) \stackrel{\text{def}}{=} \begin{cases} \infty & \text{if } \phi(x_1, \ldots, x_m) = \infty \\ 0 & \text{if } \phi(x_1, \ldots, x_m) < \infty. \end{cases}$$

In view of the close correspondence between crisp cost functions and relations we shall use these terms interchangeably in the rest of the paper.

3 Expressive Power and Algebraic Properties

Adding a finite constant to any cost function does not alter the relative costs. Hence, for any valued constraint language Γ with costs in Ω, we define the **expressive power** of Γ, denoted $\langle\Gamma\rangle$, to be the set of all cost functions ϕ such that $\phi + c$ is expressible over Γ for some constant $c \in \Omega$ where $c < \infty$.

Note that the notion of expressive power for crisp cost functions (=relations) corresponds to expressibility using conjunction and existential quantification (*primitive positive formulas*) [3].

A number of algebraic techniques to determine the expressive power of a given valued constraint language have been developed in earlier papers. To make use of these techniques, we first need to define some key terms.

The i-th component of a tuple t will be denoted by $t[i]$. Note that any operation on a set D can be extended to tuples over the set D in a standard way, as follows. For any function $f : D^k \to D$, and any collection of tuples $t_1, \ldots, t_k \in D^m$, define $f(t_1, \ldots, t_k) \in D^m$ to be the tuple $\langle f(t_1[1], \ldots, t_k[1]), \ldots, f(t_1[m], \ldots, t_k[m])\rangle$.

Definition 3.1 ([9]). *Let R be an m-ary relation over a finite set D and let f be a k-ary operation on D. Then f is a **polymorphism** of R if $f(t_1, \ldots, t_k) \in R$ for all choices of $t_1, \ldots, t_k \in R$.*

A valued constraint language, Γ, which contains only crisp cost functions (= relations) will be called a crisp constraint language. We will say that f is a polymorphism of a crisp constraint language Γ if f is a polymorphism of every relation in Γ. The set of all polymorphisms of Γ will be denoted $\mathsf{Pol}(\Gamma)$.

It follows from the results of [13] that the expressive power of a crisp constraint language is fully characterised by its polymorphisms:

Theorem 3.2 ([13]). *For any crisp constraint language Γ over a finite set*

$$R \in \langle\Gamma\rangle \;\Leftrightarrow\; \mathsf{Pol}(\Gamma) \subseteq \mathsf{Pol}(\{R\}).$$

Hence, a crisp cost function ϕ is expressible over a crisp constraint language Γ if and only if it has all the polymorphisms of Γ.

We can extend the idea of polymorphisms to arbitrary valued constraint languages by considering the corresponding feasibility relations:

Definition 3.3 ([4]). *The **feasibility polymorphisms** of a valued constraint language Γ are the polymorphisms of the corresponding crisp feasibility cost functions, that is,*

$$\mathsf{FPol}(\Gamma) \;\stackrel{\text{def}}{=}\; \mathsf{Pol}(\{\mathsf{Feas}(\phi) \mid \phi \in \Gamma\}).$$

However, to fully capture the expressive power of valued constraint languages it is necessary to consider more general algebraic properties, such as the following:

Definition 3.4 ([5]). *A list of functions, $\langle f_1, \ldots, f_k \rangle$, where each f_i is a function from D^k to D, is called a k-ary **multimorphism** of a cost function $\phi : D^m \to \Omega$ if, for all $t_1, \ldots, t_k \in D^m$, we have*

$$\sum_{i=1}^{k} \phi(t_i) \quad \geq \quad \sum_{i=1}^{k} \phi(f_i(t_1, \ldots, t_k)).$$

For any valued constraint language Γ, we will say that $\mathcal{F} = \langle f_1, \ldots, f_k \rangle$ is a multimorphism of Γ if $\mathcal{F}$ is a multimorphism of every cost function in Γ. The set of all multimorphisms of Γ will be denoted $\mathsf{Mul}(\Gamma)$.

It is a simple consequence of the definitions that if $\{f_i\}_{1 \leq i \leq k}$ are polymorphisms of R, then $\langle f_1, \ldots, f_k \rangle$ is a multimorphism of the corresponding crisp cost function ϕ_R. Conversely, if $\langle f_1, \ldots, f_k \rangle$ is a multimorphism of ϕ, then $\{f_i\}_{1 \leq i \leq k}$ are polymorphisms of the corresponding relation R_ϕ.

The next result shows that the multimorphisms of a valued constraint language are preserved by all the cost functions expressible over that language.

Theorem 3.5 ([7]). *If $\mathcal{F}$ is a multimorphism of a valued constraint language Γ, then $\mathcal{F}$ is a multimorphism of $\langle \Gamma \rangle$.*

Hence, to show that a cost function ϕ is *not* expressible over a valued constraint language Γ it is sufficient to identify some multimorphism of Γ which is not a multimorphism of ϕ. This is the main technique that we shall use to establish inexpressibility results in the sections below.

It is currently an open question whether the set of multimorphisms of a valued constraint language completely characterizes the expressive power of that language. However, it was shown in [4] that the expressive power of a valued constraint language can be characterised by generalising the notion of multimorphism a little, to a property called a *fractional polymorphism*, which is essentially a multimorphism where each component function has an associated weight value.

Definition 3.6 ([4]). *A k-ary **weighted function** $\mathcal{F}$ on a set D is a set of the form $\{\langle r_1, f_1 \rangle, \ldots, \langle r_n, f_n \rangle\}$ where each r_i is a non-negative rational number such that $\sum_{i=1}^{n} r_i = k$ and each f_i is a distinct function from D^k to D.*

*For any m-ary cost function ϕ, we say that a k-ary weighted function $\mathcal{F}$ is a k-ary **fractional polymorphism** of ϕ if, for all $t_1, \ldots, t_k \in D^m$,*

$$\sum_{i=1}^{k} \phi(t_i) \;\geq\; \sum_{i=1}^{n} r_i \phi(f_i(t_1, \ldots, t_k)).$$

For any valued constraint language Γ, we will say that $\mathcal{F}$ is a fractional polymorphism of Γ if $\mathcal{F}$ is a fractional polymorphism of every cost function in Γ. The set of all fractional polymorphisms of Γ will be denoted $\mathsf{fPol}(\Gamma)$.

Note that multimorphisms can be viewed as fractional polymorphisms where all weights are natural numbers.

It was shown in [4] that the feasibility polymorphisms and fractional polymorphisms of a valued constraint language effectively determine its expressive power.

Theorem 3.7 ([4]). *Let Γ be a valued constraint language with costs in $\overline{\mathbb{Q}}_+$ such that, for all $\phi \in \Gamma$, and all $c \in \mathbb{Q}_+$, $c\phi \in \Gamma$ and $\mathsf{Feas}(\phi) \in \Gamma$.*

$$\phi \in \langle \Gamma \rangle \;\Leftrightarrow\; \mathsf{FPol}(\Gamma) \subseteq \mathsf{FPol}(\{\phi\}) \wedge \mathsf{fPol}(\Gamma) \subseteq \mathsf{fPol}(\{\phi\}).$$

Hence, a cost function ϕ is expressible over a valued constraint language Γ if and only if it has all the feasibility polymorphisms and fractional polymorphisms of Γ.

4 The Expressive Power of Arbitrary Relations and Max-Closed Relations

In this section we consider the expressive power of crisp constraint languages. We consider the languages containing all relations up to some fixed arity over some fixed domain, and we also consider an important subset of these relations defined for totally ordered domains, the so-called *max-closed* relations, which are defined below. In both cases we show that the relations of a fixed arity can express all relations of arbitrary arity.

Definition 4.1. *Let D be a fixed totally ordered set.*

- *The k-ary function on D which returns the largest of its k arguments in the given ordering of D is denoted* $\textsc{Max}_k$.
- *The k-ary function on D which returns the smallest of its k arguments in the given ordering of D is denoted* $\textsc{Min}_k$.
- *The k-ary function on D which returns the second largest of its $k \geq 2$ arguments in the given ordering of D is denoted* $\textsc{Second}_k$.

The function $\textsc{Max}_2$ will be denoted $\textsc{Max}$ and the function $\textsc{Min}_2$ will be denoted $\textsc{Min}$.

Definition 4.2. *A cost function ϕ is **max-closed** if $\langle \textsc{Max}, \textsc{Max} \rangle \in \mathsf{Mul}(\{\phi\})$.*

In this section we focus on *crisp* max-closed cost functions. This class of cost functions was first introduced (as a class of relations) in [16] and shown to be tractable. In other words, $\mathsf{VCSP}(\Gamma)$ is known to be polynomial-time solvable for every finite Γ consisting of max-closed relations. A number of examples of max-closed relations are given in [16].

Definition 4.3. *For every $d \geq 2$ we define the following:*

- $\mathbf{R}_{d,m}$ *denotes the set of all relations over a domain of size d of arity at most m, and $\mathbf{R}_d = \cup_{m \geq 0} \mathbf{R}_{d,m}$;*
- $\mathbf{R}^{\max}_{d,m}$ *denotes the set of all max-closed relations over an ordered domain of size d of arity at most m, and $\mathbf{R}^{\max}_d = \cup_{m \geq 0} \mathbf{R}^{\max}_{d,m}$.*

It is well-known that any relation can be expressed as a propositional formula in conjunctive normal form (CNF), hence we have the following characterisation of $\mathbf{R}_{d,m}$

Proposition 4.4. *A relation $R \in \mathbf{R}_{d,m}$ if and only if there is some formula ψ such that $\langle v_1, \ldots, v_m \rangle \in R \Leftrightarrow \psi(v_1, \ldots, v_m)$ and ψ is a conjunction of clauses of the form $(v_1 \neq a_1) \vee \ldots \vee (v_m \neq a_m)$ for some constants $a_1, \ldots, a_m$.*

Proof. Let $\overline{R} = D^m \setminus R = \{\langle e_{11}, \ldots, e_{1m} \rangle, \ldots, \langle e_{n1}, \ldots, e_{nm} \rangle\}$ be the complement of R. Define ψ such that each clause forbids one of the n disallowed tuples: $\psi(v_1, \ldots, v_m) \equiv \wedge_{1 \leq i \leq n} \neg[(v_1 = e_{i1}) \wedge \ldots \wedge (v_m = e_{im})] \equiv \wedge_{1 \leq i \leq n}[(v_1 \neq e_{i1}) \vee \ldots \vee (v_m \neq e_{im})]$. □

We also have a similar characterisation for $\mathbf{R}^{\max}_{d,m}$, adapted from Theorem 5.2 of [16].

Theorem 4.5 ([16])**.** *A relation $R \in \mathbf{R}^{\max}_{d,m}$ if and only if there is some formula ψ such that $\langle v_1, \ldots, v_m \rangle \in R \Leftrightarrow \psi(v_1, \ldots, v_m)$ and ψ is a conjunction of clauses of the form $(v_1 > a_1) \vee \ldots \vee (v_m > a_m) \vee (v_i < b_i)$ for some constants $a_1, \ldots, a_m, b_i$.*

Note that in the special case of a Boolean domain (that is, when $d = 2$) this restricted form of clause is equivalent to a disjunction of literals with at most one negated literal; clauses of this form are sometimes called **anti-Horn** clauses.

It is well-known that for every $d \geq 2$, $\mathsf{Pol}(\mathbf{R}_d)$ is equal to the set of all possible projection operations [9]. We now characterise the polymorphisms of $\mathbf{R}^{\max}_d$.

Definition 4.6. *Let $I = \{i_1, \ldots, i_n\} \subseteq \{1, \ldots, k\}$ be a set of indices. Define the k-ary function*

$$\textsc{Max}_I(x_1, \ldots, x_k) \stackrel{\text{def}}{=} \textsc{Max}_n(x_{i_1}, \ldots, x_{i_n}).$$

For every k, there are exactly $2^k - 1$ functions of the form $\textsc{Max}_I$ for $\emptyset \neq I \subseteq \{1, \ldots, k\}$.

Proposition 4.7. *For all $d \geq 2$,*

$$\mathsf{Pol}(\mathbf{R}^{\max}_d) = \{\textsc{Max}_I \mid \emptyset \neq I \subseteq \{1, \ldots, k\}, k = 1, 2, \ldots\}.$$

Proof. When $|I| = 1$, the corresponding function $\textsc{Max}_I$ is just a projection operation, and every projection is a polymorphism of every relation [9].

If $\textsc{Max} \in \mathsf{Pol}(\{R\})$, then $\textsc{Max}_I \in \mathsf{Pol}(\{R\})$ for every $\emptyset \neq I \subseteq \{1, \ldots, k\}$. This is because every $\textsc{Max}_I$ can be obtained from $\textsc{Max}$ by composition and projections.

We now prove that the operations of the form $\textsc{Max}_I$ are the only polymorphisms of $\mathbf{R}^{\max}_d$. Suppose, for contradiction, that f is a k-ary polymorphism of $\mathbf{R}^{\max}_d$ which is different from $\textsc{Max}_I$ for every $\emptyset \neq I \subseteq \{1, \ldots, k\}$. It follows that, for each I such that $\emptyset \neq I \subseteq \{1, \ldots, k\}$, there is a k-tuple x_I such that $f(x_I) \neq \textsc{Max}_I(x_I)$. Let n be the total number of different tuples x_I, that is, $n = |\{x_I \mid \emptyset \neq I \subseteq \{1, \ldots, k\}\}| \leq 2^k - 1$ and denote these tuples by $x_1, \ldots, x_n$. Now consider the relation $R = \{\langle x_1[j], \ldots, x_n[j] \rangle\}_{1 \leq j \leq k}$. Define $R_0 = R$ and $R_{i+1} = R_i \cup \{\textsc{Max}(u, v) \mid u, v \in R_i\}$ for every $i \geq 0$. Clearly, $R_i \subseteq R_{i+1}$ and since there is only a finite number of different n-tuples, there is an l such that $R_l = R_{l+i}$ for every $i \geq 0$. Define R' to be the transitive closure of R under $\textsc{Max}$, that is, $R' = R_l$. Clearly, R' is max-closed and every tuple t of R' is of the form $t = \textsc{Max}_j(y_{i_1}, \ldots, y_{i_j})$ for some $j \geq 1$ and $y_{i_1}, \ldots, y_{i_j} \in R$. We have constructed R so that the application of f to the tuples $x_1, \ldots, x_n$ results in a tuple t which is different from every tuple of the former form and hence $t \notin R'$. Therefore, $f \notin \mathsf{Pol}(\mathbf{R}^{\max}_d)$. □

We now consider the expressive power of $\mathbf{R}_{d,m}$ and $\mathbf{R}^{\max}_{d,m}$.

It is clear that binary relations have greater expressive power than unary relations, so our first result is not unexpected, but it provides a simple illustration of the use of the algebraic approach.

Proposition 4.8. *For all $d \geq 2$, $\langle \mathbf{R}_{d,1} \rangle \subsetneq \langle \mathbf{R}_{d,2} \rangle$ and $\langle \mathbf{R}^{\max}_{d,1} \rangle \subsetneq \langle \mathbf{R}^{\max}_{d,2} \rangle$.*

Proof. Notice for example that MIN $\in \mathsf{Pol}(\mathbf{R}_{d,1})$ and MIN $\in \mathsf{Pol}(\mathbf{R}^{\max}_{d,1})$ but MIN $\notin \mathsf{Pol}(\mathbf{R}_{d,2})$ and MIN $\notin \mathsf{Pol}(\mathbf{R}^{\max}_{d,2})$. The result then follows from Theorem 3.5. □

4.1 Relations over a Boolean domain

As a first step, we now focus on the special case of a Boolean domain, that is, the case when $d = 2$. In this case we shall establish that ternary relations have fewer polymorphisms than binary relations, and hence have a greater expressive power. Similar remarks apply to the class of max-closed relations over a Boolean domain. The detailed results are as follows.

Proposition 4.9. MAJORITY $\in \mathsf{Pol}(\mathbf{R}_{2,2})$*, where* MAJORITY *is the unique ternary function on a 2-element set which returns the argument value that occurs most often.*

Proof. Let R be an arbitrary binary Boolean relation. Let $a = \langle a_1, a_2 \rangle$, $b = \langle b_1, b_2 \rangle$ and $c = \langle c_1, c_2 \rangle$ be three pairs belonging to R. Note that since the domain size is 2, the pair $\langle \textsc{Majority}(a_1, b_1, c_1), \textsc{Majority}(a_2, b_2, c_2) \rangle$ is equal to at least one of a, b, c, and hence belongs to R. □

Corollary 4.10. MAJORITY $\in \mathsf{Pol}(\mathbf{R}^{\max}_{2,2})$.

Proposition 4.11. MAJORITY $\notin \mathsf{Pol}(\mathbf{R}^{\max}_{2,3})$.

Proof. Consider the ternary Boolean max-closed relation R consisting of all triples except $\langle 0, 0, 0 \rangle$. To see that MAJORITY is not a polymorphism of R, consider the triples $\langle 0, 0, 1 \rangle$, $\langle 0, 1, 0 \rangle$ and $\langle 1, 0, 0 \rangle$. The application of MAJORITY to these tuples results in the triple $\langle 0, 0, 0 \rangle$ which is not in R. □

Corollary 4.12. MAJORITY $\notin \mathsf{Pol}(\mathbf{R}_{2,3})$.

However, we now show that ternary Boolean relations have the same expressive power as all Boolean relations. In other words, any Boolean relation of arbitrary arity is expressible by relations of arity at most three. We obtain this result by adapting the well-known reduction from SATISFIABILITY to 3-SATISFIABILITY.

Proposition 4.13. $\mathbf{R}_2 \subseteq \langle \mathbf{R}_{2,3} \rangle$.

Proof. By Proposition 4.4, any Boolean relation $R \in \mathbf{R}_2$ can be expressed as a CNF formula ψ. We now define a 3-CNF formula ψ' such that ψ is satisfiable if and only if ψ' is satisfiable.

Let C be an arbitrary clause of ψ with literals $\{z_1, \ldots, z_l\}$. If $l \leq 3$, then let ψ' contain the same clause C. If $l > 3$, introduce additional variables $y_{C,1}, \ldots, y_{C,l-3}$ and replace C with the set of clauses, C', defined as follows:

$$C' = \{(z_1 \vee z_2 \vee \overline{y_{C,1}}), (y_{C,1} \vee z_3 \vee \overline{y_{C,2}}), (y_{C,2} \vee z_4 \vee \overline{y_{C,3}}), \ldots, (y_{C,l-3}, z_{l-1}, z_l)\}.$$

It is straightforward to check that any assignment of variables which satisfies ψ can be extended to the additional variables in such a way that it satisfies ψ'. On the other hand, given an assignment which satisfies ψ', the restriction to the original variables is a satisfying assignment for ψ. □

We now show that the same result holds for max-closed Boolean relations.

Corollary 4.14. $\mathbf{R}_2^{\max} \subseteq \langle \mathbf{R}_{2,3}^{\max} \rangle$.

Proof. By Theorem 4.5, any Boolean max-closed relation $R \in \mathbf{R}_2^{\max}$ can be expressed as a CNF formula ψ of the form shown in Theorem 4.5. The construction given in the proof of Proposition 4.13 preserves this special form of the clauses, and hence can be used to express R using ternary max-closed relations, by Theorem 4.5. □

Combining these results, we obtain the following result.

Theorem 4.15.

1. $\langle \mathbf{R}_{2,1} \rangle \subsetneq \langle \mathbf{R}_{2,2} \rangle \subsetneq \langle \mathbf{R}_{2,3} \rangle = \mathbf{R}_2$;
2. $\langle \mathbf{R}_{2,1}^{\max} \rangle \subsetneq \langle \mathbf{R}_{2,2}^{\max} \rangle \subsetneq \langle \mathbf{R}_{2,3}^{\max} \rangle = \mathbf{R}_2^{\max}$.

4.2 Relations over larger domains

For relations over a domain with 3 or more elements similar results can be obtained. In fact, in this case we can show that any relation can be expressed using *binary* relations.

Proposition 4.16. *For all* $d \geq 3$, $\mathbf{R}_d \subseteq \langle \mathbf{R}_{d,2} \rangle$.

Proof. By Proposition 4.4, any relation R over D, $|D| = d > 2$, can be expressed as a CNF formula ψ over D. Using a similar construction to the proof of Proposition 4.13, ψ can be expressed as a 3-CNF formula over D. This gives the weaker result that $\mathbf{R}_d \subseteq \langle \mathbf{R}_{d,3} \rangle$.

We now show how to express any 3-CNF formula over D as a 2-CNF formula over D. Let $D = \{e_1, e_2, e_3, \ldots\}$. Replace each ternary clause $C = (U_1(x_1) \vee U_2(x_2) \vee U_3(x_3))$ by a set of three clauses $C' = \{U_1(x_1) \vee N_1(y)), (U_2(x_2) \vee N_2(y)), (U_3(x_3) \vee N_3(y))\}$ where y is an additional variable and $N_1(y) = D \setminus \{e_1\}$ ("not 1"), $N_2(y) = D \setminus \{e_2\}$ and $N_3(y) = \{e_1, e_2\}$. Note that if C is satisfied by an assignment s to the variables x_1, x_2 and x_3, then s can be extended to satisfy C' as well. On the other hand, if s satisfies C' then at least one of $U_1(x_1)$, $U_2(x_2)$ and $U_3(x_3)$ is satisfied by s, so s restricted to x_1, x_2 and x_3 satisfies C.

It follows that $\langle \mathbf{R}_{d,3} \rangle \subseteq \langle \mathbf{R}_{d,2} \rangle$. □

Next we generalise Proposition 4.16 to show that max-closed relations over any domain can be expressed using ternary relations.

Proposition 4.17. *For all $d \geq 3$, $\mathbf{R}_d^{\max} \subseteq \langle \mathbf{R}_{d,3}^{\max} \rangle$.*

Proof. The proof is very similar to the proof of Proposition 4.16. By Theorem 4.5, any max-closed relation R over a set D can be expressed as a CNF formula ψ over D.

Let the elements of D be ordered $e_1 < e_2 < \ldots < e_d$. We use the same construction as in the proof of Proposition 4.13, except that instead of using a literal of the form $y_{C,i}$, where $y_{C,i}$ is an additional variable, we use the unary predicate $T(y_{C,i})$ where $T = \{\langle e_d \rangle\}$ and instead of using a literal of the form $\overline{y_{C,i}}$, we use the unary predicate $F(y_{C,i})$ where $F = \{\langle e_1 \rangle\}$. Note that $T(x)$ can be written as $x > e_{d-1}$ and $F(x)$ can be written as $x < e_2$. Hence, if the input formula is of the form shown in Theorem 4.5, then the output formula is of that form as well and so can be used to express R using ternary max-closed relations, by Theorem 4.5. □

By investigating polymorphisms of binary max-closed relations we can strengthen Proposition 4.17.

Theorem 4.18. *For all $d \geq 3$, $\mathbf{R}_d^{\max} \subseteq \langle \mathbf{R}_{d,2}^{\max} \rangle$.*

Proof. We show that $\mathsf{Pol}(\mathbf{R}_{d,2}^{\max}) \subseteq \mathsf{Pol}(\mathbf{R}_d^{\max})$. The result then follows from Theorem 3.2.

Without loss of generality, assume that $D = \{1, \ldots, M\}$, that is, $d = M$. Let $f \in \mathsf{Pol}(\mathbf{R}_{d,2}^{\max})$ be an arbitrary k-ary polymorphism. By Proposition 4.7, it is enough to show that $f = \textsc{Max}_I$ for some $\emptyset \neq I \subseteq \{1, \ldots, k\}$.

If $f = \textsc{Max}_{\{1,\ldots,k\}}$ we are done. Otherwise, there exist $a_1, \ldots, a_k \in D$ such that $a_i = \textsc{Max}_k(a_1, \ldots, a_k)$ and $a_i > f(a_1, a_2, \ldots, a_k) = a_j$. Without loss of generality in order to simplify our notation, assume that $i = 1$ and $j = 2$, that is, $a_1 = \textsc{Max}_k(a_1, \ldots, a_k)$ and $a_1 > f(a_1, a_2, \ldots, a_k) = a_2$. We show that f does not depend on its first parameter.

Claim 1. *f is conservative.*

Proof. For any subset $S \subseteq D$, we know that $R = \{(a, a) \mid a \in S\}$ is max-closed. □

For any fixed $x_2, \ldots, x_k \in D$, denote $\overline{x} = \langle x_2, \ldots, x_k \rangle$ and define the max-closed relation

$$R_{\overline{x}} = (\{a_2, \ldots, a_k\} \times \{x_2, \ldots, x_k\}) \cup (\{a_1\} \times D).$$

Now consider $g_{\overline{x}}(r) = f(r, x_2, \ldots, x_k)$. Note that $g_{\overline{x}}(r)$ is a restriction of f with all arguments except the first one fixed.

First we show that $g_{\overline{x}}(r)$ is not conservative.

Claim 2. *$g_{\overline{x}}(r) \in \{x_2, \ldots, x_k\}$.*

Proof. Note that $(a_1, r) \in R_{\overline{x}}$ and $\{(a_j, x_j) \mid j = 2, \ldots, k\} \subseteq R_{\overline{x}}$. Since f is a conservative polymorphism of $R_{\overline{x}}$ and $a_2 = f(a_1, a_2, \ldots, a_k)$, it follows from the definition of $R_{\overline{x}}$ that $g_{\overline{x}}(r) \in \{x_2, \ldots, x_k\}$. □

Define the max-closed relation

$$R'_{\overline{x}} = (\{M\} \times D) \cup \{(x_j, x_j) \mid j = 2, \ldots, k\}.$$

We show that if M, the biggest element of the domain, is not among $x_2, \ldots, x_k$, then $g_{\overline{x}}(r)$ is constant.

Claim 3. $M \notin \{x_2, \ldots, x_k\} \Rightarrow \forall r \in D, g_{\overline{x}}(r) = g_{\overline{x}}(M)$.

Proof. Note that $(M, r) \in R'_{\overline{x}}$ and $\{(x_j, x_j) \mid j = 2, \ldots, k\} \subseteq R'_{\overline{x}}$. By Claim 2, $g_{\overline{x}}(M) = x_i$ for some $2 \leq i \leq k$. From the definition of $R'_{\overline{x}}$ and the fact that f is a polymorphisms of $R'_{\overline{x}}$, $g_{\overline{x}}(r) = x_i = g_{\overline{x}}(M)$ for every $r \in D$. □

Next we generalise Claim 3 so that $g_{\overline{x}}(r)$ is constant whenever $x_2, \ldots, x_k$ does not contain all elements of the domain D.

Claim 4. $\{x_2, \ldots, x_k\} \neq D \Rightarrow \forall r \in D, g_{\overline{x}}(r) = g_{\overline{x}}(M)$.

Proof. For any $p \in D \setminus \{M\}$, define

$$\Delta_p(x) = \begin{cases} x & \text{if } x \leq p, \\ x - 1 & \text{if } x > p. \end{cases}$$

We show that for every $p \in D$, if $p \notin \{x_2, \ldots, x_k\}$, then $g_{\overline{x}}(r) = g_{\overline{x}}(M)$ for every $r \in D$. Note that the case $p = M$ is already proved by Claim 3.

The relation $R_p = \{(d, \Delta_p(d) \mid d \in D\}$ is max-closed. Clearly, $(r, \Delta_p(r)) \in R_p$ and $\{(x_j, \Delta_p(x_j)) \mid j = 2, \ldots, k\} \subseteq R_p$. Since $g_{\overline{x}}$ is a polymorphism of R_p, we know that for every $r \in D$, $g_{\overline{x}}(r) \in \Delta_p^{-1}(g_{\overline{x}}(\Delta_p(r)))$.

Since $M \notin \{\Delta_p(d) \mid d \in D\}$, we know, by Claim 3, that $g_{\overline{x}}(\Delta_p(r))$ is constant. Say $g_{\overline{x}}(\Delta_p)(r)) = K_p$. If $K_p \neq p$, then $|\Delta_p^{-1}(K_p)| = 1$ and so $g_{\overline{x}}$ is constant. If $K_p = p$, then $\Delta_p^{-1}(K_p) = \{p, p + 1\}$. In this case if $p \notin \{x_2, \ldots, x_k\}$, then we know, by Claim 2, that $g_{\overline{x}}(r) \neq p$ and so $g_{\overline{x}}$ is again constant. □

Finally we show that $g_{\overline{x}}(r)$ is constant.

Claim 5. *$g_{\overline{x}}(r)$ is constant.*

Proof. Define

$$\Delta_+(x) = \begin{cases} x & \text{if } x \neq M, \\ x - 1 & \text{if } x = M, \end{cases}$$

and

$$\Delta_-(x) = \begin{cases} x & \text{if } x \neq 1, \\ x + 1 & \text{if } x = 1. \end{cases}$$

The relations $R_+ = \{(d, \Delta_+(d)) \mid d \in D\}$ and $R_- = \{(d, \Delta_-(d)) \mid d \in D\}$ are both max-closed. Define $\overline{y} = \langle \Delta_+(x_2), \ldots, \Delta_+(x_k) \rangle$ and $\overline{z} = \langle \Delta_-(x_2), \ldots, \Delta_-(x_k) \rangle$. Since $M \notin \{\Delta_+(d) \mid d \in D\}$ and $1 \notin \{\Delta_-(d) \mid d \in D\}$, we know, by Claim 4, that $g_{\overline{y}}$ and $g_{\overline{z}}$ are both constant.

If $g_{\overline{x}}$ is not constant, then since $g_{\overline{x}}$ is a polymorphism of R_+ it follows that for every $r \in D$, $g_{\overline{x}}(r) \in \{M, M-1\}$. Similarly, since $g_{\overline{x}}$ is a polymorphism of R_- it follows that $g_{\overline{x}}(r) \in \{1, 2\}$. Since $|D| > 2$ we know that this is not possible.[1] Therefore, $g_{\overline{x}}$ is constant. □

We have shown that if $a_1 = \max(a_1, \ldots, a_k)$ and $a_1 > f(a_1, \ldots, a_k) = a_2$, then f does not depend on its first parameter. Similarly, by repeating the same argument, we can show that if $f \neq \textsc{Max}_{\{2,\ldots,k\}}$, then f does not depend on its i-th parameter for some $2 \leq i \leq k$.

In general, if f does not depend on any parameter outside of $I \subseteq \{1, \ldots, k\}$ and $f \neq \textsc{Max}_I$, then f does not depend on all of the parameters whose index is in I.

Therefore, either there is some set $I \subseteq \{1, \ldots, k\}$ for which $f = \textsc{Max}_I$ or f is constant. Clearly, every subset $S \subseteq D$ of the domain D is a unary max-closed relation and hence f is a polymorphism of S. Therefore, f cannot be constant. □

Combining these results we obtain the following result.

Theorem 4.19. *For all $d \geq 3$,*

1. $\langle \mathbf{R}_{d,1} \rangle \subsetneq \langle \mathbf{R}_{d,2} \rangle = \mathbf{R}_d$;
2. $\langle \mathbf{R}_{d,1}^{\max} \rangle \subsetneq \langle \mathbf{R}_{d,2}^{\max} \rangle = \mathbf{R}_d^{\max}$.

5 Finite-valued Cost Functions

In this section we consider the expressive power of finite-valued constraint languages. In particular, we show that the finite-valued max-closed cost functions of any fixed arity *cannot* express all finite-valued max-closed cost-functions of any larger arity. Hence we identify an infinite hierarchy of finite-valued cost functions with ever-increasing expressive power.

The class of max-closed cost functions is discussed in more detail in [7] and shown to be tractable. A number of examples of max-closed cost functions are given in [7].

We will say that an m-tuple u *dominates* a m-tuple v, denoted $u \geq v$, if $u[i] \geq v[i]$ for $1 \leq i \leq m$.

Proposition 5.1 ([7])**.** *A m-ary cost function $\phi : D^m \to \Omega$ is max-closed if and only if* $\textsc{Max} \in \mathsf{FPol}(\{\phi\})$ *and ϕ is finitely antitone, that is, for all m-tuples u, v with $\phi(u), \phi(v) < \infty$, $u \leq v \Rightarrow \phi(u) \geq \phi(v)$.*

It follows from this that the finite-valued max-closed cost functions are simply the finite-valued antitone functions, that is, those functions whose values can only decrease as their arguments get larger. Note that for such functions the expressive power is likely to be rather limited because in any construction the "hidden variables" that are "projected out" can always be assigned the highest values in their domain in order to minimise the cost. Hence, using such hidden variables only adds a constant value to the total cost, and so does not allow more cost functions to be expressed.

[1] This is the only place where we have used the fact that $|D| \geq 3$.

Definition 5.2. *For all $d \geq 2$ we define the following:*

- $\mathbf{F}_{d,m}$ *denotes the set of all finite-valued cost functions (that is, cost functions whose valuation structure $\Omega = \mathbb{Q}_+$) of arity at most m over a domain of size d, and $\mathbf{F}_d = \cup_{m \geq 0} \mathbf{F}_{d,m}$;*
- $\mathbf{F}^{\max}_{d,m}$ *denotes the set of all max-closed finite-valued cost functions of arity at most m over an ordered domain of size d, and $\mathbf{F}^{\max}_d = \cup_{m \geq 0} \mathbf{F}^{\max}_{d,m}$.*

Proposition 5.3. *For all $d \geq 2$, $\langle \mathbf{F}_{d,1} \rangle \subsetneq \langle \mathbf{F}_{d,2} \rangle$ and $\langle \mathbf{F}^{\max}_{d,1} \rangle \subsetneq \langle \mathbf{F}^{\max}_{d,2} \rangle$.*

Proof. Consider the list of functions $\langle \textsc{Min}, \textsc{Max} \rangle$. It is straightforward to verify that $\langle \textsc{Min}, \textsc{Max} \rangle \in \mathsf{Mul}(\mathbf{F}_{d,1})$ and $\langle \textsc{Min}, \textsc{Max} \rangle \in \mathsf{Mul}(\mathbf{F}^{\max}_{d,1})$.

Now consider the binary max-closed finite-valued cost function ϕ over any domain containing $\{0,1\}$, defined by $\phi(\langle 0,0 \rangle) = 1$ and $\phi(\langle ., . \rangle) = 0$ otherwise. Note that ϕ is max-closed but $\langle \textsc{Min}, \textsc{Max} \rangle$ is *not* a multimorphism of ϕ. To see this, consider the application of $\langle \textsc{Min}, \textsc{Max} \rangle$ to the tuples $\langle 0,1 \rangle$ and $\langle 1,0 \rangle$ (see the figure below).

$$\begin{array}{rcc} & \begin{array}{cc} 0 & 1 \\ 1 & 0 \end{array} & \xrightarrow{\phi} \begin{array}{c} 0 \\ 0 \end{array} \Big\} \sum = 0 \\ \textsc{Min} & \begin{array}{cc} 0 & 0 \end{array} & \xrightarrow{\phi} \begin{array}{c} 1 \\ 0 \end{array} \Big\} \sum = 1 \\ \textsc{Max} & \begin{array}{cc} 1 & 1 \end{array} & \end{array}$$

The result then follows from Theorem 3.5. □

Clearly, we also have $\langle \mathbf{F}_{d,m} \rangle \subseteq \langle \mathbf{F}_{d,m+1} \rangle$ for $m \geq 2$, but we do not know whether these inclusions are strict. However, in the case of max-closed finite-valued cost functions we now prove a separation result.

It will be convenient to use a shorthand notation for multiple copies of the same function within a list. For a k-ary function f, we write:

$$n * f(x_1, \ldots, x_k) \stackrel{\text{def}}{=} \overbrace{f(x_1, \ldots, x_k), \ldots, f(x_1, \ldots, x_k)}^{n\text{-times}}.$$

Proposition 5.4. *For all $m \geq 3$, $\langle (m-1) * \textsc{Max}_m, \textsc{Second}_m \rangle \in \mathsf{Mul}(\mathbf{F}^{\max}_{d,m-1})$.*

Proof. Let ϕ be an arbitrary $(m-1)$-ary max-closed finite-valued cost function. Let $t_1, \ldots, t_m$ be $(m-1)$-tuples. We show that there is an i such that the tuple $s = \textsc{Second}_m(t_1, \ldots, t_m)$ dominates t_i, that is, $s[j] \geq t_i[j]$ for $1 \leq j \leq m-1$. To show this we count the number of tuples which can fail to be dominated by s. If a tuple t_p is not dominated by s, for some $1 \leq p \leq m$, it means that there is a position $1 \leq j \leq m-1$ such that $t_p[j] > s[j]$. But since $\textsc{Second}_m$ returns the second biggest value, for every $1 \leq j \leq m-1$, there is at most one tuple which is not dominated by s. Since there are $m \geq 3$ tuples, there must be an i such that t_i is dominated by s. Moreover, $\textsc{Max}_m(t_1, \ldots, t_m)$ clearly dominates all $t_1, \ldots, t_m$. By Proposition 5.1, ϕ is antitone and therefore $\langle (m-1) * \textsc{Max}_m, \textsc{Second}_m \rangle$ is a multimorphism of ϕ, by Definition 3.4. □

Proposition 5.5. *For all $m \geq 3$, $\langle (m-1) * \textsc{Max}_m, \textsc{Second}_m \rangle \notin \mathsf{Mul}(\mathbf{F}^{\max}_{d,m})$.*

Proof. Let ϕ be the m-ary max-closed finite-valued cost function defined by $\phi(\langle 0,\ldots,0\rangle) = 1$ and $\phi(\langle .,\ldots,.\rangle) = 0$ otherwise. To show that $\langle (m-1)*\textsc{Max}_m, \textsc{Second}_m\rangle$ is *not* a multimorphism of ϕ, consider the m-tuples $\langle 0,\ldots,0,1\rangle$, $\langle 0,\ldots,0,1,0\rangle,\ldots,\langle 1,0,\ldots,0\rangle$. Each of them is assigned the cost 0 by ϕ. But applying the functions $\langle (m-1)*\textsc{Max}_m, \textsc{Second}_m\rangle$ co-ordinatewise results in $m-1$ tuples $\langle 1,\ldots,1\rangle$, which are assigned cost 0 by ϕ, and one tuple $\langle 0,\ldots,0\rangle$, which is assigned cost 1 by ϕ (see the figure below).

$$
\begin{array}{lcccccc}
 & 0 & 0 & \ldots & 0 & 0 & 1 \\
 & 0 & 0 & \ldots & 0 & 1 & 0 \\
 & & & \vdots & & & \\
 & 1 & 0 & \ldots & 0 & 0 & 0 \\
\hline
\textsc{Max}_m & 1 & 1 & \ldots & 1 & 1 & 1 \\
\vdots & & & \vdots & & & \\
\textsc{Max}_m & 1 & 1 & \ldots & 1 & 1 & 1 \\
\textsc{Second}_m & 0 & 0 & \ldots & 0 & 0 & 0
\end{array}
\quad
\begin{array}{c}
\\ \\ \xrightarrow{\phi} \\ \\ \\ \hline \\ \\ \xrightarrow{\phi} \\ \\ \\
\end{array}
\quad
\begin{array}{l}
0 \\ 0 \\ \vdots \\ 0 \\ \hline 0 \\ \vdots \\ 0 \\ 1
\end{array}
\quad
\begin{array}{l}
\Big\} \sum = 0 \\ \\ \\ \\ \Big\} \sum = 1
\end{array}
$$

□

Theorem 5.6. *For all $d \geq 2$, $\langle \mathbf{F}^{\max}_{d,1}\rangle \subsetneq \langle \mathbf{F}^{\max}_{d,2}\rangle \subsetneq \langle \mathbf{F}^{\max}_{d,3}\rangle \subsetneq \langle \mathbf{F}^{\max}_{d,4}\rangle \cdots$*

Proof. By Propositions 5.3, 5.4 and 5.5 and Theorem 3.5. □

6 General Cost Functions

In this section we show that general cost functions of a fixed arity can express cost functions of arbitrary arities. Comparing this result with the results of the previous section provides a striking example of the way in which allowing infinite cost values in a valued constraint language can drastically affect the expressibility of cost functions over that language, including finite-valued cost functions.

Definition 6.1. *For all $d \geq 2$ we define the following:*

- *$\mathbf{G}_{d,m}$ denotes the set of all general cost functions (that is, cost functions whose valuation structure $\Omega = \overline{\mathbb{Q}}_+$) of arity at most m over a domain of size d, and $\mathbf{G}_d = \cup_{m\geq 0}\mathbf{G}_{d,m}$;*
- *$\mathbf{G}^{\max}_{d,m}$ denotes the set of all general max-closed cost functions of arity at most m over an ordered domain of size d, and $\mathbf{G}^{\max}_d = \cup_{m\geq 0}\mathbf{G}^{\max}_{d,m}$.*

Once again it is straightforward to establish a separation between unary and binary general cost functions.

Proposition 6.2. *$\langle \mathbf{G}_{d,1}\rangle \subsetneq \langle \mathbf{G}_{d,2}\rangle$ and $\langle \mathbf{G}^{\max}_{d,1}\rangle \subsetneq \langle \mathbf{G}^{\max}_{d,2}\rangle$.*

Proof. Identical to the proof of Proposition 5.3. □

Proposition 6.3. $\langle \mathbf{G}^{\max}_{2,2} \rangle \subsetneq \langle \mathbf{G}^{\max}_{2,3} \rangle$.

Proof. By Proposition 5.5, $\langle \textsc{Max}_3, \textsc{Max}_3, \textsc{Second}_3 \rangle \notin \mathsf{Mul}(\mathbf{F}^{\max}_{2,3})$ and therefore, $\langle \textsc{Max}_3, \textsc{Max}_3, \textsc{Second}_3 \rangle \notin \mathsf{Mul}(\mathbf{G}^{\max}_{2,3})$.

We will now show that $\langle \textsc{Max}_3, \textsc{Max}_3, \textsc{Second}_3 \rangle \in \mathsf{Mul}(\mathbf{G}^{\max}_{2,2})$. The result then follows from Theorem 3.5.

Let ϕ be an arbitrary binary max-closed general cost function over a Boolean domain D. To show that $\langle \textsc{Max}_3, \textsc{Max}_3, \textsc{Second}_3 \rangle \in \mathsf{Mul}(\{\phi\})$, we need to show that, for all pairs p_1, p_2, p_3 over D, $\phi(p_1) + \phi(p_2) + \phi(p_3) \geq \phi(p) + \phi(p) + \phi(s)$, where $p = \textsc{Max}_3(p_1, p_2, p_3)$ and $s = \textsc{Second}_3(p_1, p_2, p_3)$.

If $\phi(p_i) = \infty$ for some $1 \leq i \leq 3$, then the inequality is trivially satisfied, so we may assume that each $\phi(p_i)$ is finite.

Since ϕ is max-closed, $\textsc{Max} \in \mathsf{FPol}(\{\phi\})$. Hence, $\textsc{Max}_3 \in \mathsf{FPol}(\{\phi\})$, because $\textsc{Max}_3$ can be obtained from $\textsc{Max}_2$ by composition. Because the domain is Boolean, $\textsc{Second}_3 = \textsc{Majority}$, so, by Proposition 4.10, $\textsc{Second}_3 \in \mathsf{FPol}(\{\phi\})$. It follows from Definition 3.3 that if each $\phi(p_i) < \infty$, then $\phi(p) < \infty$ and $\phi(s) < \infty$.

The same argument as in the proof of Proposition 5.4 shows that there is an i such that the tuple s dominates p_i. Moreover, the tuple p clearly dominates p_i, $1 \leq i \leq 3$. Hence, if $\phi(p)$ and $\phi(s)$ are both finite, then by Proposition 5.1, the desired inequality is satisfied. □

Clearly, we also have $\langle \mathbf{G}_{d,m} \rangle \subseteq \langle \mathbf{G}_{d,m+1} \rangle$ for $m \geq 2$, but we do not know whether these inclusions are strict. However, in the case of max-closed general cost functions we now prove a collapse result.

First we show that general max-closed cost functions of a fixed arity have the same feasibility polymorphisms as cost functions of arbitrary arities.

Proposition 6.4. *For all $d \geq 3$, $\mathsf{FPol}(\mathbf{G}^{\max}_{d,2}) = \mathsf{FPol}(\mathbf{G}^{\max}_{d})$. Moreover, $\mathsf{FPol}(\mathbf{G}^{\max}_{2,3}) = \mathsf{FPol}(\mathbf{G}^{\max}_{2})$.*

Proof. Assume for contradiction that there is an $f \in \mathsf{FPol}(\mathbf{G}^{\max}_{d,2})$ such that $f \notin \mathsf{FPol}(\mathbf{G}^{\max}_{d})$. By Definition 6.1, $\mathbf{R}^{\max}_{d} = \{\mathsf{Feas}(\phi) \mid \phi \in \mathbf{G}^{\max}_{d}\}$. Therefore, such an f would contradict Theorem 4.19 since $\mathsf{Pol}(\mathbf{R}^{\max}_{d,2}) = \mathsf{Pol}(\mathbf{R}^{\max}_{d})$.

Similarly, assume that there is an $f \in \mathsf{FPol}(\mathbf{G}^{\max}_{2,3})$ such that $f \notin \mathsf{FPol}(\mathbf{G}^{\max}_{2})$. This would contradict Theorem 4.15 since $\mathsf{Pol}(\mathbf{R}^{\max}_{2,3}) = \mathsf{Pol}(\mathbf{R}^{\max}_{2})$. □

We now prove that general max-closed cost functions of a fixed arity have the same fractional polymorphisms as cost functions of arbitrary arities. First we characterise feasibility polymorphisms of general cost functions.

Proposition 6.5. *For all $d \geq 2$,*

$$\mathsf{FPol}(\mathbf{G}^{\max}_{d}) = \{\textsc{Max}_I \mid \emptyset \neq I \subseteq \{1, \ldots, k\}, k = 1, 2, \ldots\}.$$

Proof. It follows from Definition 6.1 that $\mathbf{R}^{\max}_{d} = \{\mathsf{Feas}(\phi) \mid \phi \in \mathbf{G}^{\max}_{d}\}$. Therefore, $\mathsf{FPol}(\mathbf{G}^{\max}_{d}) = \mathsf{FPol}(\mathbf{R}^{\max}_{d})$ and the result follows from Proposition 4.7. □

Next we characterise fractional polymorphisms of general cost functions.

Definition 6.6. *Let $\mathcal{F} = \{(r_1, \textsc{Max}_{S_1}), \ldots, (r_n, \textsc{Max}_{S_n})\}$ be a k-ary weighted function and $S \subseteq \{1, \ldots, k\}$.*

We define

$$\mathrm{supp}_{\mathcal{F}} S \stackrel{\text{def}}{=} \{i \mid S_i \cap S \neq \emptyset\}$$

and

$$\mathrm{wt}_{\mathcal{F}}(S) \stackrel{\text{def}}{=} \sum_{i \in \mathrm{supp}_{\mathcal{F}}(S)} r_i$$

Theorem 6.7. *Let $\mathcal{F} = \{(r_1, \textsc{Max}_{S_1}), \ldots, (r_n, \textsc{Max}_{S_n})\}$ be a k-ary weighted function. The following are equivalent:*

1. *$\mathcal{F} \in \mathsf{fPol}(\mathbf{G}_d^{\max})$.*
2. *$\mathcal{F} \in \mathsf{fPol}(\mathbf{G}_{d,1}^{\max})$.*
3. *For every subset $S \subseteq \{1, \ldots, k\}$, $\mathrm{wt}_{\mathcal{F}}(S) \geq |S|$.*

Proof. We first show that $\neg(3) \Rightarrow \neg(2) \Rightarrow \neg(1)$.

First suppose that there exists an $S \subseteq \{1, \ldots, k\}$ such that $\mathrm{wt}_{\mathcal{F}}(S) < |S|$. Let $\{a, b\} \subseteq D$ be the two biggest elements of D and $a < b$. Consider the unary cost function ϕ where

$$\phi(x) = \begin{cases} 0 & \text{if } x = b, \\ 1 & \text{if } x = a, \\ \infty & \text{otherwise.} \end{cases}$$

Certainly $\phi \in \mathbf{G}_{d,1}^{\max}$.

Now let

$$x_i = \begin{cases} b & \text{if } i \in S, \\ a & \text{if } i \notin S. \end{cases}$$

We have that

$$\sum_{i=1}^{k} \phi(x_i) = k - |S|, \text{ and}$$

$$\sum_{j=1}^{n} r_j \phi(\textsc{Max}_{S_j}(x_1, \ldots, x_k)) = k - \mathrm{wt}_{\mathcal{F}}(S)$$

$$> k - |S|, \text{ by assumption.}$$

So $\mathcal{F}$ is not a fractional polymorphism of $\mathbf{G}_{d,1}^{\max}$ and so not a fractional polymorphism of $\mathbf{G}_d^{\max}$.

To complete the proof we will show that (3) $\Rightarrow$ (1).

Suppose that, for every subset $S \subseteq \{1, \ldots, k\}$, $\mathrm{wt}_{\mathcal{F}}(S) \geq |S|$.

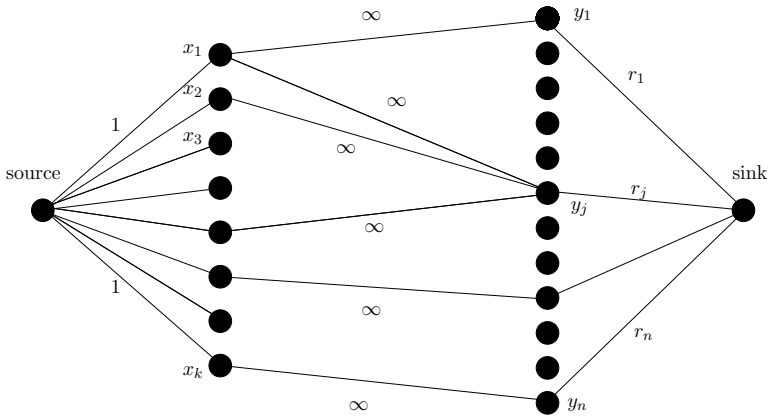


Figure 1: The flow from x_i to y_j in a maximum flow is the value of p_{ji}

We will first show the existence of a set of non-negative values p_{ji} for $j = 1, \ldots, n$ and $i = 1, \ldots, k$ where

$$\sum_{i=1}^{k} p_{ji} = r_j,$$
$$\sum_{j=1}^{n} p_{ji} = 1 \text{ and}$$
$$p_{ji} = 0 \text{ if } i \notin S_j.$$

Consider the network in Figure 1. The capacity from the source to any node x_i is one. The capacity from node y_j to the sink is r_j. There is an arc from node x_i to node y_j precisely when $i \in S_j$, and the capacity of these arcs is infinite.

We will use the Min-Cut Max-Flow theorem to generate the p_{ji}.

Suppose that we have a minimum cut of this network. Let R be those arcs in this cut from the source to any node x_i. Let $S = \{1, \ldots, k\} - \{i \mid x_i \in R\}$. Since we have a cut we must (at least) cut every arc from the nodes $\{y_j \mid j \in \mathrm{supp}_{\mathcal{F}}(S)\}$ to the sink. By assumption $\mathrm{wt}_{\mathcal{F}}(S) \geq |S|$ and so this cut has total cost at least k. Certainly there is a cut of cost exactly k (cut all arcs from the source) and so the max-flow through this network is precisely k. Such a flow can only be achieved if each arc from the source and each arc to the sink is filled to its capacity. The flow along the arc from x_i to y_j then gives the required value for p_{ji}.

Now we will use these values p_{ji} to show that $\mathcal{F}$ is indeed a fractional polymorphism of $\mathbf{G}_d^{\max}$.

Let $x_1, \ldots, x_k$ be m-ary tuples and $\phi \in \mathbf{G}_{d,m}^{\max}$ be an m-ary cost function. We have to show the following:

$$\sum_{i=1}^{k} \phi(x_i) \geq \sum_{j=1}^{n} r_j \phi(\textsc{Max}_{S_j}(x_1, \ldots, x_k)). \tag{1}$$

If any $\phi(x_i)$ is infinite, then this inequality clearly holds.

By Proposition 6.5, all $\textsc{Max}_{S_j}$, $1 \leq j \leq n$, are feasibility polymorphisms of $\mathbf{G}_d^{\max}$. Therefore, if all $\phi(x_i)$ are finite, then all $\phi(\textsc{Max}_{S_j}(x_1, \ldots, x_k))$ are finite as well.

By definition of p_{ji} and using that $p_{ji} = 0$ whenever $i \notin S_j$ we have that

$$\sum_{j=1}^{n} r_j \phi(\textsc{Max}_{S_j}(x_1, \ldots, x_k)) = \sum_{j=1}^{n} \sum_{i \in S_j} p_{ji} \phi(\textsc{Max}_{S_j}(x_1, \ldots, x_k)).$$

Now, since ϕ is antitone, we have

$$\sum_{j=1}^{n} \sum_{i \in S_j} p_{ji} \phi(\textsc{Max}_{S_j}(x_1, \ldots, x_k)) \leq \sum_{j=1}^{n} \sum_{i \in S_j} p_{ji} \phi(x_i)$$

Since $p_{ji} = 0$ whenever $i \notin S_j$ we have that

$$\sum_{j=1}^{n} \sum_{i \in S_j} p_{ji} \phi(x_i) = \sum_{j=1}^{n} \sum_{i=1}^{k} p_{ji} \phi(x_i)$$

Finally, since $\sum_{j=1}^{n} p_{ji} = 1$ we have established Inequality (1). □

Theorem 6.8. *For all $d \geq 3$, $\mathsf{fPol}(\mathbf{G}_{d,2}^{\max}) = \mathsf{fPol}(\mathbf{G}_d^{\max})$. Moreover, $\mathsf{fPol}(\mathbf{G}_{2,3}^{\max}) = \mathsf{fPol}(\mathbf{G}_2^{\max})$.*

Proof. By Proposition 6.4, $\mathbf{G}_{d,2}^{\max}$ and $\mathbf{G}_d^{\max}$ have the same feasibility polymorphisms. Also, $\mathbf{G}_{2,3}^{\max}$ and $\mathbf{G}_2^{\max}$ have the same feasibility polymorphisms. By Proposition 6.5, these feasibility polymorphisms are of the form "max-on-a-subset". Clearly, each component function of a fractional polymorphism has to be a feasibility polymorphism. Therefore, the result follows from Theorem 6.7. □

Theorem 6.9. *For all $d \geq 3$, $\langle \mathbf{G}_{d,1}^{\max} \rangle \subsetneq \langle \mathbf{G}_{d,2}^{\max} \rangle = \mathbf{G}_d^{\max}$. Moreover, $\langle \mathbf{G}_{2,1}^{\max} \rangle \subsetneq \langle \mathbf{G}_{2,2}^{\max} \rangle \subsetneq \langle \mathbf{G}_{2,3}^{\max} \rangle = \mathbf{G}_2^{\max}$.*

Proof. Strict inclusions follow from Propositions 6.2 and 6.3. Note that for every $d \geq 2$, $m \geq 1$ and $c \in \mathbb{Q}_+$, $\mathbf{G}_{d,m}^{\max}$ is closed under scaling by c. Therefore, using Theorem 3.7 the collapses follow from Proposition 6.4 and Theorem 6.8. □

Note that the proof shows a slightly stronger result: $\mathbf{G}_d^{\max} = \langle \mathbf{R}_{d,2}^{\max} \cup \mathbf{F}_{d,1}^{\max} \rangle$ for every $d \geq 3$ and $\mathbf{G}_2^{\max} = \langle \mathbf{R}_{2,3}^{\max} \cup \mathbf{F}_{2,1} \rangle$.

Example 6.10. *Consider the ternary max-closed finite-valued cost function ϕ over $D = \{0, 1, 2\}$ which is defined as*

$$\phi(t) = \begin{cases} 1 & \text{if } t = \langle 0, 0, 0 \rangle, \\ 0 & \text{otherwise.} \end{cases}$$

By Proposition 5.5, $\phi \notin \langle \mathbf{F}_{3,2}^{\max} \rangle$. In other words, ϕ is not expressible using only max-closed finite-valued cost functions of arity at most 2. However, by Theorem 6.9,

$\phi \in \langle \mathbf{G}^{\max}_{3,2} \rangle$. *We now show how* ϕ *can be expressed using max-closed general cost functions of arity at most 2.*

Let ϕ_0 *be the binary finite-valued max-closed cost function defined as follows:*

$$\phi_0(t) = \begin{cases} 1 & \text{if } t = \langle 0, 0 \rangle, \\ 0 & \text{otherwise.} \end{cases}$$

Next, define two binary crisp[2] *max-closed cost functions*

$$\phi_1(t) = \begin{cases} \infty & \text{if } t = \langle 0, 1 \rangle, \\ 0 & \text{otherwise,} \end{cases}$$

and

$$\phi_2(t) = \begin{cases} \infty & \text{if } t = \langle 0, 2 \rangle, \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mathcal{P} = \langle V, D, \mathcal{C} \rangle$ *where* $V = \{x, y, z, u, v\}$ *and*

$$\mathcal{C} = \{\langle \langle x, u \rangle, \phi_1 \rangle, \langle \langle y, u \rangle, \phi_2 \rangle, \langle \langle y, v \rangle, \phi_1 \rangle, \langle \langle z, v \rangle, \phi_2 \rangle, \langle \langle u, v \rangle, \phi_0 \rangle\}.$$

We claim that $\langle \mathcal{P}, \langle x, y, z \rangle \rangle$ *is a gadget for expressing* ϕ *over* $\mathbf{G}^{\max}_{3,2}$. *(See Figure 2.) If any of* x, y, z *is non-zero, then at least one of the variables* u, v *can be assigned a non-zero value and the cost of such an assignment is zero. If all* x, y *and* z *are assigned zero, then the minimal cost assignment assigns* u *and* v *zero as well.*

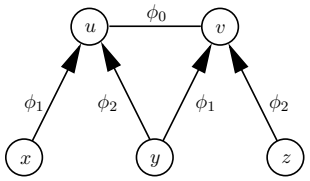


Figure 2: $\mathcal{P}$, an instance of $\mathbf{G}^{\max}_{3,2}$ expressing ϕ from Example. 6.10.

We now show another gadget for expressing ϕ *using only max-closed crisp cost functions of arity at most 2 and max-closed finite-valued cost functions of arity at most 1.*

Let μ *be a unary max-closed finite-valued cost function defined as*

$$\mu(x) = \begin{cases} 1 & \text{if } x = 0, \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mathcal{P}' = \langle V, D, \mathcal{C} \rangle$ *where* $V = \{x, y, z, u, v, w\}$ *and*

$$\mathcal{C} = \{\langle \langle x, u \rangle, \phi_1 \rangle, \langle \langle y, u \rangle, \phi_2 \rangle, \langle \langle y, v \rangle, \phi_1 \rangle, \langle \langle z, v \rangle, \phi_2 \rangle, \langle \langle u, w \rangle, \phi_1 \rangle, \langle \langle v, w \rangle, \phi_2 \rangle, \langle w, \mu \rangle\}.$$

See Figure 3. Similarly to the argument above, $\langle \mathcal{P}, \langle x, y, z \rangle \rangle$ *is a gadget for expressing* ϕ.

[2]Note that "a finite variant of ϕ_1" defined as $\phi_1(\langle 0, 1 \rangle) = M$ for some finite $M < \infty$ and $\phi_1(\langle ., . \rangle) = 0$ otherwise is not max-closed. The infinite cost is necessary.

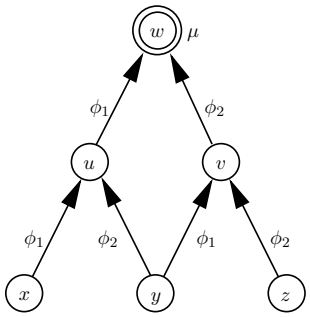


Figure 3: $\mathcal{P}'$, an instance of $\mathbf{G}^{\max}_{3,2}$ expressing ϕ from Example 6.10.

Example 6.11. *Consider a ternary max-closed finite-valued cost function ϕ over $D = \{0,1,2\}$ defined as $\phi = (\#0)^2$, that is, the number of zeros in the input tuple squared. We show a gadget for expressing ϕ using only cost functions from $\mathbf{R}^{\max}_{3,2} \cup \mathbf{F}^{\max}_{3,1} \subseteq \mathbf{G}^{\max}_{3,2}$.*

Define three binary crisp max-closed cost functions as follows:

$$\phi_0(t) = \begin{cases} \infty & \text{if } t = \langle 0,1\rangle, \\ \infty & \text{if } t = \langle 0,2\rangle, \\ 0 & \text{otherwise,} \end{cases}$$

$$\phi_1(t) = \begin{cases} \infty & \text{if } t = \langle 0,1\rangle, \\ 0 & \text{otherwise,} \end{cases}$$

and

$$\phi_2(t) = \begin{cases} \infty & \text{if } t = \langle 0,2\rangle, \\ 0 & \text{otherwise.} \end{cases}$$

For $c \in \{1,3,5\}$, let μ_c be a unary max-closed finite-valued cost function defined as

$$\mu_c(x) = \begin{cases} c & \text{if } x = 0, \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mathcal{P} = \langle V, D, \mathcal{C}\rangle$ where $V = \{x, y, z, u_1, u_2, u_3, v_1, v_2, v_3, v_4, v_5, v_6, w\}$ and the set of constraints $\mathcal{C}$ is shown in Figure 4.

We claim that $\langle \mathcal{P}, \langle x, y, z\rangle\rangle$ is a gadget for expressing ϕ over $\mathbf{R}^{\max}_{3,2} \cup \mathbf{F}^{\max}_{3,1}$.

If all x, y and z are non-zero, then there is an assignment of the other variables with values one and two such that the total cost is 0.

If any of x, y, z is zero, then either u_1 or u_2 is assigned zero, and for the same reason u_3 is assigned zero.

If at least two of x, y, z are zero, then at least one of the variables v_1, v_2, v_3 is assigned zero and consequently, at least one of v_4, v_5 is assigned zero. Therefore, v_6 is assigned 0.

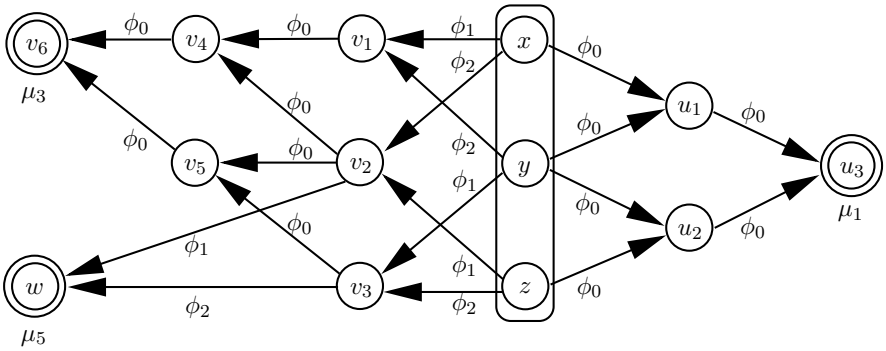


Figure 4: The gadget expressing $\phi = (\#0)^2$ from Example 6.11.

If all x, y and z are zero, then both v_2 and v_3 are assigned zero and consequently w is assigned zero.

Note that a similar gadget works for bigger domains.

7 Conclusions and Open Problems

We have investigated the expressive power of valued constraints in general and max-closed valued constraints in particular.

In the case of relations we built on previously known results about the expressibility of an arbitrary relation in terms of binary or ternary relations. We were able to prove in a similar way that an arbitrary max-closed relation can be expressed using binary or ternary max-closed relations. The results about the collapse of the set of all relations and all max-closed relations contrast sharply with the case of finite-valued cost functions, where we showed an infinite hierarchy for max-closed cost functions. This shows that the VCSP is not just a minor generalisation of the CSP – max-closed finite-valued cost functions behave very differently from max-closed crisp cost functions. We also showed the collapse of general cost functions by investigating the feasibility polymorphisms and fractional polymorphisms of general max-closed cost functions. This shows that allowing infinite costs in max-closed cost functions increases their expressive power substantially.

We remark that all of our results about max-closed cost functions obviously have equivalent versions for *min-closed* cost functions, that is, those which have the multimorphism $\langle \textsc{Min}, \textsc{Min} \rangle$. In the Boolean crisp case these are precisely the relations that can be expressed by a conjunction of **Horn** clauses.

One of the reasons why understanding the expressive power of valued constraints is important is for the investigation of **submodular functions**. A cost function ϕ is called submodular if it has the multimorphism $\langle \textsc{Min}, \textsc{Max} \rangle$. The standard problem of submodular function minimisation corresponds to solving a VCSP with submodular cost functions over the Boolean domain [6].

Submodular function minimisation (SFM) is a central problem in discrete optimisa-

tion, with links to many different areas [10, 17, 20]. Although it has been known for a long time that the ellipsoid algorithm can be used to solve SFM in polynomial time, this algorithm is not efficient in practice. Relatively recently, several new strongly polynomial combinatorial algorithms have been discovered for SFM [10, 11, 12]. Unfortunately, the time complexity of the fastest published algorithm for SFM is roughly of an order of $O(n^7)$ where n is the total number of variables [11].

However, for certain special cases of SFM, more efficient algorithms are known to exist. For example, the (weighted) MIN-CUT problem is a special case of SFM that can be solved in cubic time [10]. Moreover, it is known that SFM over a Boolean domain can be solved in $O(n^3)$ time when the submodular function f satisfies various extra conditions [1, 8, 18]. In particular, in the case of non-Boolean domains, a cubic-time algorithm exists for SFM when f can be expressed as a sum of *binary* submodular functions [6].

These observations naturally raise the following question: What is the most general class of submodular functions that can be minimised in cubic time (or better)? One way to tackle this question is to investigate the expressive power of particular submodular functions which are known to be solvable in cubic time. Any fixed set of functions which can be expressed using such functions will have the same complexity [4].

One intriguing result is already known for submodular *relations*. In the case of relations, having $\langle \text{MIN}, \text{MAX} \rangle$ as a multimorphism implies having both MIN and MAX as polymorphisms. The ternary MEDIAN operation can be obtained by composing the operations MAX and MIN, so all submodular relations have the MEDIAN operation as a polymorphism. It follows that submodular relations are binary decomposable [14], and hence all submodular relations are expressible using binary submodular relations.

For finite-valued and general submodular cost functions it is an important open question whether they can be expressed using submodular relations of some fixed arity. If they can, then this raises the possibility of designing new, more efficient, algorithms for submodular function minimisation.

Acknowledgments

The authors would like to thank Martin Cooper, Martin Green, Chris Jefferson and András Salamon for many helpful discussions.

References

[1] Billionet, A., Minoux, M.: Maximizing a supermodular pseudo-boolean function: a polynomial algorithm for cubic functions. Discrete Applied Mathematics **12** (1985) 1–11

[2] 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

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

[4] Cohen, D., Cooper, M., Jeavons, P.: An algebraic characterisation of complexity for valued constraints. In: Proceedings 12th International Conference on Constraint Programming—CP'06. Volume 4204 of Lecture Notes in Computer Science., Springer-Verlag (2006) 107–121

[5] Cohen, D., Cooper, M., Jeavons, P., Krokhin, A.: Soft constraints: Complexity and multimorphisms. In: Proceedings 9th International Conference on Constraint Programming—CP'03 (Kinsale, September 2003). Volume 2833 of Lecture Notes in Computer Science., Springer-Verlag (2003) 244–258

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

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

[8] 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, Philadelphia, PA. (2001)

[9] Denecke, K., Wismath, S.: Universal Algebra and Applications in Theoretical Computer Science. Chapman and Hall/CRC Press (2002)

[10] Fujishige, S.: Submodular Functions and Optimization. 2nd edn. Volume 58 of Annals of Discrete Mathematics. Elsevier (2005)

[11] Iwata, S.: A faster scaling algorithm for minimizing submodular functions. SIAM Journal on Computing **32** (2003) 833–840

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

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

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

[15] Jeavons, P., Cohen, D., Gyssens, M.: How to determine the expressive power of constraints. Constraints **4** (1999) 113–131

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

[17] Narayanan, H.: Submodular Functions and Electrical Networks. North-Holland, Amsterdam (1997)

[18] Queyranne, M.: Minimising symmetric submodular functions. Mathematical Programming **82** (1998) 3–12

[19] Rossi, F., van Beek, P., Walsh, T., eds.: The Handbook of Constraint Programming. Elsevier (2006)

[20] Topkis, D.: Supermodularity and Complementarity. Princeton University Press (1998)