The complexity of Boolean surjective general-valued CSPs*

Peter Fulla[1] and Stanislav Živný[1]

1 **Department of Computer Science, University of Oxford, UK**
{peter.fulla,standa.zivny}@cs.ox.ac.uk

Abstract

Valued constraint satisfaction problems (VCSPs) are discrete optimisation problems with a $\overline{\mathbb{Q}}$-valued objective function given as a sum of fixed-arity functions, where $\overline{\mathbb{Q}} = \mathbb{Q} \cup \{\infty\}$ is the set of extended rationals.

In Boolean surjective VCSPs variables take on labels from $D = \{0, 1\}$ and an optimal assignment is required to use both labels from D. A classic example is the *global min-cut* problem in graphs. Building on the work of Uppman, we establish a dichotomy theorem and thus give a complete complexity classification of Boolean surjective VCSPs. The newly discovered tractable case has an interesting structure related to projections of downsets and upsets. Our work generalises the dichotomy for $\{0, \infty\}$-valued constraint languages (corresponding to CSPs) obtained by Creignou and Hébrard, and the dichotomy for $\{0, 1\}$-valued constraint languages (corresponding to Min-CSPs) obtained by Uppman.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases constraint satisfaction problems, surjective CSP, valued CSP, min-cut, polymorphisms, multimorphisms

Digital Object Identifier 10.4230/LIPIcs.MFCS.2017.4

1 Introduction

The (s, t)-Min-Cut problem asks, given a digraph $G = (V, E)$ with source $s \in V$, sink $t \in V$, and edge weights $w : E \to \mathbb{Q}_{>0}$, for a subset $C \subseteq V$ with $s \in C$ and $t \notin C$ minimising $w(C) = \sum_{(u,v)\in E, u\in C, v\notin C} w(u, v)$ [22]. This fundamental problem is an example of a Boolean valued constraint satisfaction problem (VCSP).

Let D be an arbitrary finite set called the *domain*. A *valued constraint language*, or just a language, Γ is a set of weighted relations; each weighted relation $\gamma \in \Gamma$ is a function $\gamma : D^{\mathrm{ar}(\gamma)} \to \overline{\mathbb{Q}}$, where $\mathrm{ar}(\gamma) \in \mathbb{N}$ is the *arity* of γ and $\overline{\mathbb{Q}} = \mathbb{Q} \cup \{\infty\}$ is the set of extended rationals. If $|D| = 2$ then Γ is called a *Boolean* language. An *instance* $I = (V, D, \phi_I)$ of the VCSP on domain D is given by a finite set of n variables $V = \{x_1, \ldots, x_n\}$ and an objective function $\phi_I : D^n \to \overline{\mathbb{Q}}$ expressed as a weighted sum of *valued constraints* over V, i.e. $\phi_I(x_1, \ldots, x_n) = \sum_{i=1}^{q} w_i \cdot \gamma_i(\mathbf{x}_i)$, where γ_i is a weighted relation, $w_i \in \mathbb{Q}_{\geq 0}$ is the *weight* and $\mathbf{x}_i \in V^{\mathrm{ar}(\gamma_i)}$ the *scope* of the ith valued constraint. (We note that we allow zero weights

* The authors were supported by a Royal Society Research Grant. Stanislav Živný was supported by a Royal Society University Research Fellowship. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 714532). The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein.

and for $w_i = 0$ we define $w_i \cdot \infty = \infty$.) Given an instance I, the goal is to find an assignment $s : V \to D$ of domain labels to the variables that *minimises* ϕ_I. Given a language Γ, we denote by VCSP(Γ) the class of all instances I that use only weighted relations from Γ in their objective function. Valued CSPS are also called general-valued CSPs to emphasise that (decision) CSPs are a special case of valued CSPs.

To continue with the (s,t)-Min-Cut example, let $D = \{0,1\}$. We will use the following three weighted relations: $\gamma : D^2 \to \overline{\mathbb{Q}}$ is defined by $\gamma(x,y) = 1$ if $x = 0$ and $y = 1$, and $\gamma(x,y) = 0$ otherwise; $\rho_d : D \to \overline{\mathbb{Q}}$, for $d \in D$, is defined by $\rho_d(x) = 0$ if $x = d$ and $\rho_d(x) = \infty$ if $x \neq d$. Now, given an (s,t)-Min-Cut instance $G = (V,E)$, $s,t \in V = \{x_1,\ldots,x_n\}$, and $w : E \to \mathbb{Q}_{>0}$ as before, the problem of finding an optimal (s,t)-Min-Cut in G is equivalent to solving the following instance of VCSP(Γ_{cut}), where $\Gamma_{\mathsf{cut}} = \{\gamma, \rho_0, \rho_1\}$: $I = (V, D, \phi_I)$ and $\phi_I(x_1,\ldots,x_n) = \sum_{(u,v)\in E} w(u,v) \cdot \gamma(x_u, x_v) + \rho_0(s) + \rho_1(t)$.

It is well known that the (s,t)-Min-Cut problem is solvable in polynomial time. Since every instance I of VCSP(Γ_{cut}) can be reduced to an instance of the (s,t)-Min-Cut problem, I is solvable in polynomial time. Thus, Γ_{cut} is an example of a *tractable* constraint language. When VCSP(Γ) is NP-hard, we call Γ an *intractable* language.

It is natural to ask about the complexity of VCSP(Γ) in terms of Γ. For *Boolean* valued constraint languages, we have a complete answer: Cohen et al. [8] showed that every Boolean valued constraint language is either tractable or intractable, thus obtaining what is known as a dichotomy theorem. In fact, [8] identified eight different types of tractable valued constraint languages; one of these types corresponds to submodularity [22] and includes Γ_{cut}. The dichotomy theorem from [8] is an extension of Schaefer's celebrated result, which gave a dichotomy for $\{0,\infty\}$-valued constraint languages [21], and the work of Creignou [9], who gave a dichotomy theorem for $\{0,1\}$-valued constraint languages.

The (global) Min-Cut problem asks, given a graph $G = (V,E)$ and edge weights $w : E \to \mathbb{Q}_{>0}$, for a subset $C \subseteq V$ with $\emptyset \subsetneq C \subsetneq V$ minimising $w(C) = \sum_{\{u,v\}\in E, |\{u,v\}\cap C|=1} w(u,v)$ [22]. This fundamental problem is an example of a Boolean *surjective* VCSP. Given a VCSP instance $I = (V, D, \phi_I)$, in the surjective setting the goal is to find a surjective assignment $s : V \to D$ minimising ϕ_I; here s is called surjective if for every $d \in D$ there is $x \in V$ such that $s(x) = d$. For Boolean VCSPs with $D = \{0,1\}$, this simply means that the all-zero and all-one assignments are not allowed. Given a language Γ, we denote by $\mathrm{VCSP_s}(\Gamma)$ the class of all instances I of the surjective VCSP that use only weighted relations from Γ in their objective function.

Let $D = \{0,1\}$ and define $\gamma : D \to \overline{\mathbb{Q}}$ by $\gamma(x,y) = 0$ if $x = y$ and $\gamma(x,y) = 1$ if $x \neq y$. $\mathrm{VCSP_s}(\{\gamma\})$ captures Min-Cut as every instance of $\mathrm{VCSP_s}(\{\gamma\})$ is a Min-Cut instance and vice versa. Since Min-Cut is solvable in polynomial time (say, by a reduction to the (s,t)-Min-Cut problem but other algorithms exist [23]), $\{\gamma\}$ is an example of a surjectively tractable, or *s-tractable* for short, valued constraint language. As before, if $\mathrm{VCSP_s}(\Gamma)$ is NP-hard we call Γ a surjectively intractable, or *s-intractable* for short, language.

Surjective VCSPs

What can we say about the complexity of $\mathrm{VCSP_s}(\Gamma)$ *for arbitrary* Γ*? In particular, is every* Γ *s-tractable or s-intractable? What is the mathematical structure of s-tractable languages?*

First, observe that for a language Γ defined on D, we have $\mathrm{VCSP}(\Gamma) \leq_p \mathrm{VCSP_s}(\Gamma)$. Indeed, given an instance I of VCSP(Γ), construct a new instance I' of $\mathrm{VCSP_s}(\Gamma)$ by adding $|D|$ extra variables. Then, any solution to I can be extended to a surjective solution to I' of the same value and conversely, any (surjective) solution to I' induces a solution to I of the same value.

Second, observe that for a language Γ defined on D, we have $\mathrm{VCSP}_{\mathrm{s}}(\Gamma) \leq_p \mathrm{VCSP}(\Gamma \cup \mathcal{C}_D)$, where $\mathcal{C}_D$ is the set of constants on D; that is, $\mathcal{C}_D = \{\rho_d \mid d \in D\}$, where ρ_d is defined by $\rho_d(x) = 0$ if $x = d$ and $\rho_d(x) = \infty$ if $x \neq d$. Indeed, given an instance of $\mathrm{VCSP}_{\mathrm{s}}(\Gamma)$, constants can be used to go through all $O(n^{|D|})$ ways to assign all the labels from D to a $|D|$-subset of the n variables, each resulting in an instance of $\mathrm{VCSP}(\Gamma \cup \mathcal{C}_D)$. Consequently, a tractable language Γ defined on D with $\mathcal{C}_D \subseteq \Gamma$ is also an s-tractable language.

In this paper we deal with Boolean valued constraint languages defined on $D = \{0, 1\}$. By the two observations above, the only Boolean valued constraint languages for which tractability could be different from s-tractability are tractable languages that do not include constants.

For Boolean $\{0, \infty\}$-valued languages, Schaefer's dichotomy [21] gives six tractable cases, four of which include constants. Creignou and Hébrard showed that the remaining two cases (0-valid and 1-valid) are s-intractable, thus obtaining a dichotomy in the surjective setting [10].

For Boolean $\{0, 1\}$-valued languages, Creignou's dichotomy [9] gives three tractable cases, one of which includes constants. Uppman showed that the remaining two cases (0-valid and 1-valid) are s-tractable if they are almost-min-min or almost-max-max, respectively, and s-intractable otherwise, thus obtaining a dichotomy in the surjective setting [24].

Contributions As our main contribution we classify all Boolean valued constraint languages (i.e. $\overline{\mathbb{Q}}$-valued languages) as s-tractable or s-intractable. Our result extends the classifications from [10] and [24]. Six of the eight tractable cases identified for Boolean valued constraint languages [8] include constants and thus are also s-tractable. The remaining two cases (0-optimal and 1-optimal[1]) are s-tractable if they satisfy a certain condition. This condition, defined formally in Definition 5, says that both the feasibility and optimality relations of every weighted relation in the language have to be a projection of a downset (in the 0-optimal case), or a projection of an upset (in the 1-optimal case). This shows that, surprisingly, s-tractability of valued constraint languages (that are not covered by the tractable languages with constants) does not depend on the rational-values in the weighted relations. It is only the structure of the underlying feasibility and optimality relations that matters. (However, the running time of our algorithm depends on these values.) Identifying this condition and establishing that it captures the precise borderline of s-tractability is our main contribution.

The hardness part of our result is proved in the same spirit as for $\{0, \infty\}$-valued and $\{0, 1\}$-valued languages by carefully analysing the types of weighted relations that can be obtained in gadgets in the surjective setting, and relying on the explicit dichotomy for Boolean VCSPs [8].

While 0-optimal and 1-optimal languages are trivially tractable for VCSPs, the algorithm for surjective VCSPs over the newly identified languages is nontrivial. The s-tractability part of our result is established by a reduction from $\overline{\mathbb{Q}}$-valued $\mathrm{VCSP}_{\mathrm{s}}$ to the *Generalised Min-Cut* problem (defined in Section 3), in which we require to find in polynomial time all α-optimal solutions, where α is a constant depending on the (finite) valued constraint language. The algorithm for the Generalised Min-Cut problem is essentially the same as in [24]. We show that the algorithm works in the more general setting with one part of the objective function being a $(\mathbb{Q}_{\geq 0} \cup \{\infty\})$-valued superadditive set function given by an oracle; see Section 3 for the details. By providing a tighter analysis we are able to improve the bound on the running

[1] A weighted relation is 0-optimal (1-optimal) if the all-zero (all-one) tuple minimises it.

time from roughly $O(n^{3^{3\alpha}})$ to $O(n^{20\alpha})$, thus answering one of the open problems from [24]. We also show that the dependence of the running time on the language is unavoidable unless P = NP (cf. Example 27).

All omitted proofs are available in the full version of the paper [14].

Related work Recent years have seen some remarkable progress on the computational complexity of CSPs and VCSPs parametrised by the (valued) constraint language, see [1] for a survey. We highlight the resolution of the "bounded width conjecture" [2] and the result that a dichotomy for CSPs, conjectured in [11], implies a dichotomy for VCSPs [19, 18]. All this work is for arbitrary (and thus not necessarily Boolean) finite domains and relies on the so-called algebraic approach initiated in [6] and nicely described in a recent survey [3]. One of the important aspects of the algebraic approach is the assumption that constants are present in (valued) constraint languages. (This is without loss of generality with respect to polynomial-time solvability.) It is the lack of constants in the surjective setting that makes it difficult, if not impossible, to employ the algebraic approach in this setting. See the work of Chen [7] for an initial attempt.

For a binary (unweighted) relation γ, $\mathrm{VCSP_s}(\{\gamma\})$ has been studied under the name of surjective γ-Colouring [4, 20] and vertex-compaction [26]. We remark that our notion of surjectivity is global. For the γ-Colouring problem, a local version of surjectivity has also been studied [13, 12].

2 Preliminaries

We denote by $\leq_p$ the standard polynomial-time Turing reduction. If $A \leq_p B$ and $B \leq_p A$ we write $A \equiv_p B$.

We use the notation $[n] = \{1, \dots, n\}$. For any tuple $\mathbf{x} \in D^r$, we refer to its ith element as x_i. For $\mathbf{x}, \mathbf{y} \in D^r$, we define $\mathbf{x} \leq \mathbf{y}$ if and only if $x_i \leq y_i$ for all $i \in [r]$.

We define *relations* as a special case of weighted relations with range $\{c, \infty\}$, where value $c \in \mathbb{Q}$ is assigned to tuples that are elements of the relation in the conventional sense. We will use both views interchangeably and choose $c = 0$ unless stated otherwise. Relations are also called unweighted or *crisp*.

If $\mathbf{s} \in [r]^n$ is a tuple of coordinates then for any $\mathbf{x} \in D^r$ we denote its projection to $\mathbf{s}$ by $\mathrm{Pr}_{\mathbf{s}}(\mathbf{x}) = (x_{s_1}, \dots, x_{s_n}) \in D^n$. For any relation ρ, we define $\mathrm{Pr}_{\mathbf{s}}(\rho) = \{\mathrm{Pr}_{\mathbf{s}}(\mathbf{x}) \mid \mathbf{x} \in \rho\}$. Note that the coordinates in $\mathbf{s}$ may repeat.

We denote by $\rho_=$ the binary equality relation $\{(x, x) \mid x \in D\}$. Recall from Section 1 that we denote, for any $d \in D$, by ρ_d the unary relation $\{(d)\}$, i.e. $\rho_d(x) = 0$ if $x = d$ and $\rho_d(x) = \infty$ if $x \neq d$.

▶ **Definition 1.** For a weighted relation $\gamma : D^r \to \overline{\mathbb{Q}}$, we denote by
- $\mathrm{Feas}(\gamma) = \{\mathbf{x} \in D^r \mid \gamma(\mathbf{x}) < \infty\}$ the underlying *feasibility relation*; and by
- $\mathrm{Opt}(\gamma) = \{\mathbf{x} \in \mathrm{Feas}(\gamma) \mid \gamma(\mathbf{x}) \leq \gamma(\mathbf{y})$ for every $\mathbf{y} \in D^r\}$ the relation of *optimal* tuples.

We define $\mathrm{Feas}(\Gamma) = \{\mathrm{Feas}(\gamma) \mid \gamma \in \Gamma\}$ and $\mathrm{Opt}(\Gamma) = \{\mathrm{Opt}(\gamma) \mid \gamma \in \Gamma\}$.

An assignment $s : V \to D$ for a VCSP instance $I = (V, D, \phi_I)$ with $V = \{x_1, \dots, x_n\}$ is called *feasible* if $\phi_I(s(x_1), \dots, s(x_n)) < \infty$; s is called *optimal* if $\phi_I(s) \leq \phi_I(s')$ for every assignment s'.

Recall from Section 1 that any set of weighted relation Γ is called a valued constraint language. Γ is called *s-tractable* if for any finite subset $\Gamma' \subseteq \Gamma$ any instance of $\mathrm{VCSP_s}(\Gamma')$ can be solved in polynomial time. Γ is called *s-intractable* if $\mathrm{VCSP}(\Gamma')$ is NP-hard for some finite $\Gamma' \subseteq \Gamma$.

We apply a k-ary operation $h : D^k \to D$ to k r-tuples componentwise; i.e. $h(\mathbf{x}^1, \dots, \mathbf{x}^k) = (h(x_1^1, x_1^2, \dots, x_1^k), h(x_2^1, x_2^2, \dots, x_2^k), \dots, h(x_r^1, x_r^2, \dots, x_r^k))$.

The following notion is at the heart of the algebraic approach to decision CSPs [6].

▶ **Definition 2.** Let γ be a weighted relation on D. A k-ary operation $h : D^k \to D$ is a *polymorphism* of γ (and γ is *invariant under* or *admits* h) if, for every $\mathbf{x}^1, \dots, \mathbf{x}^k \in \mathrm{Feas}(\gamma)$, we have $h(\mathbf{x}^1, \dots, \mathbf{x}^k) \in \mathrm{Feas}(\gamma)$. We say that h is a polymorphism of a language Γ if it is a polymorphism of every $\gamma \in \Gamma$.

The following notion, which involves a collection of k k-ary polymorphisms, plays an important role in the complexity classification of Boolean valued constraint languages [8].

▶ **Definition 3.** Let γ be a weighted relation on D. A list $\langle h_1, \dots, h_k \rangle$ of k-ary polymorphisms of γ is a k-ary *multimorphism* of γ (and γ *admits* $\langle h_1, \dots, h_k \rangle$) if, for every $\mathbf{x}^1, \dots, \mathbf{x}^k \in \mathrm{Feas}(\gamma)$, we have

$$\sum_{i=1}^{k} \gamma(h_i(\mathbf{x}^1, \dots, \mathbf{x}^k)) \;\leq\; \sum_{i=1}^{k} \gamma(\mathbf{x}^i)\,.$$

$\langle h_1, \dots, h_k \rangle$ is a multimorphism of a language Γ if it is a multimorphism of every $\gamma \in \Gamma$.

Boolean VCSPs

For the rest of the paper let $D = \{0, 1\}$. We define some important operations on D. For any $a \in D$, c_a is the constant unary operation such that $c_a(x) = a$ for all $x \in D$. Operation $\neg$ is the unary negation, i.e. $\neg(0) = 1$ and $\neg(1) = 0$. Binary operation min (max) returns the smaller (larger) of its two arguments with respect to the order $0 < 1$. Ternary operation Mn (for minority) is the unique ternary operation on D satisfying $\mathrm{Mn}(x, x, y) = \mathrm{Mn}(x, y, x) = \mathrm{Mn}(y, x, x) = y$ for all $x, y \in D$. Ternary operation Mj (for majority) is the unique ternary operation on D satisfying $\mathrm{Mj}(x, x, y) = \mathrm{Mj}(x, y, x) = \mathrm{Mj}(y, x, x) = x$ for all $x, y \in D$.

▶ **Theorem 4** ([8])**.** *Let Γ be a Boolean valued constraint language. Then, Γ is tractable if it admits any the following eight multimorphisms $\langle c_0 \rangle$, $\langle c_1 \rangle$, $\langle \min, \min \rangle$, $\langle \max, \max \rangle$, $\langle \min, \max \rangle$, $\langle \mathrm{Mn}, \mathrm{Mn}, \mathrm{Mn} \rangle$, $\langle \mathrm{Mj}, \mathrm{Mj}, \mathrm{Mj} \rangle$, $\langle \mathrm{Mj}, \mathrm{Mj}, \mathrm{Mn} \rangle$. Otherwise, Γ is intractable.*

We note that Theorem 4 is a generalisation of Schaefer's classification of $\{0, \infty\}$-valued constraint languages [21] and Creignou's classification of $\{0, 1\}$-valued constraint languages [9].

The following definition is new in this paper and crucial for our main result.

▶ **Definition 5.** A relation ρ is a *downset (upset)* if for any $\mathbf{x}, \mathbf{y}$ such that $\mathbf{x} \geq \mathbf{y}$ ($\mathbf{x} \leq \mathbf{y}$) and $\mathbf{x} \in \rho$ it holds $\mathbf{y} \in \rho$. We will refer to relations that are a projection of a downset (upset) as *PDS (PUS)*. A weighted relation γ is called a *PDS (PUS) weighted relation* if both $\mathrm{Feas}(\gamma)$ and $\mathrm{Opt}(\gamma)$ are PDS (PUS). A language Γ is called PDS (PUS) if every weighted relation from Γ is PDS (PUS).

▶ **Example 6.** Relation $\rho = \{(0, 0), (0, 1), (1, 0)\}$ is a downset and hence also a PDS, while $\rho' = \{(0, 0, 0), (0, 1, 1), (1, 0, 0)\}$ is a PDS (as $\rho' = \mathrm{Pr}_{(1,2,2)}(\rho)$) but not a downset. Relation $\rho_=$ is a PDS relation and also a PDS weighted relation.

As one of the reviewers pointed out, PDS (PUS) relations are characterised by the binary polymorphism $x \wedge \neg y$ ($\neg x \vee y$).

▶ **Observation 7.** *Any PDS relation can be written as a sum of a downset and binary equality relations. More formally, if $\rho : D^r \to \{0, \infty\}$ is a PDS then we can write*

$$\rho(x_1, \ldots, x_r) \;=\; \rho'(x_{\pi(1)}, \ldots, x_{\pi(r')}) + \sum_{j=r'+1}^{r} \rho_{=}(x_{\pi(j)}, x_{i_j}),$$

where $\rho' : D^{r'} \to \{0, \infty\}$ is a downset, π is a permutation of $[r]$, and $i_j \in \{\pi(1), \ldots, \pi(r')\}$ for every $r' + 1 \le j \le r$.

The following result is our main contribution.

▶ **Theorem 8 (Main).** *Let Γ be a Boolean valued constraint language. Then, Γ is s-tractable if it admits any of the following six multimorphisms $\langle \min, \min \rangle$, $\langle \max, \max \rangle$, $\langle \min, \max \rangle$, $\langle \mathrm{Mn}, \mathrm{Mn}, \mathrm{Mn} \rangle$, $\langle \mathrm{Mj}, \mathrm{Mj}, \mathrm{Mj} \rangle$, $\langle \mathrm{Mj}, \mathrm{Mj}, \mathrm{Mn} \rangle$, or Γ is PDS, or Γ is PUS. Otherwise, Γ is s-intractable.*

Theorem 8 generalises the following two previously established results.

▶ **Theorem 9** ([10])**.** *Let Γ be a Boolean $\{0, \infty\}$-valued constraint language. Then, Γ is s-tractable if it admits any of the following four polymorphisms* min*,* max*,* Mn*,* Mj*. Otherwise, Γ is s-intractable.*

▶ **Theorem 10** ([24])**.** *Let Γ be a Boolean $\{0, 1\}$-valued constraint language. Then, Γ is s-tractable if it admits the $\langle \min, \max \rangle$ multimorphism, or Γ is PDS, or Γ is PUS. Otherwise, Γ is s-intractable.*

Theorem 10 is stated differently in [24] as the definition of PDS (PUS) languages is introduced in this paper. In fact, it was not a priori clear what the right condition for tractability should be for $\overline{\mathbb{Q}}$-valued constraint languages. Note that for $\{0, 1\}$-valued languages, the condition Feas(Γ) being PDS or PUS is vacuously true. Thus, a $\{0, 1\}$-valued language Γ is PDS (PUS) if Opt(Γ) is PDS (PUS) and this is equivalent to Γ being almost-min-min (almost-max-max) [24].

Recall that $\neg$ is the unary negation operation. For a weighted relation γ, we define $\neg(\gamma)$ to be the weighted relation $\neg(\gamma)(\mathbf{x}) = \gamma(\neg(\mathbf{x}))$. For a language Γ, we define $\neg(\Gamma) = \{\neg(\gamma) \mid \gamma \in \Gamma\}$.

The following observation follows from Definition 5.

▶ **Observation 11.** *A valued constraint language Γ is PDS if and only if $\neg(\Gamma)$ is PUS.*

▶ **Lemma 12.** *For a Boolean valued constraint language Γ, $\mathrm{VCSP}_{\mathrm{s}}(\Gamma) \equiv_p \mathrm{VCSP}_{\mathrm{s}}(\neg(\Gamma))$.*

Proof of Theorem 8. The s-tractability of languages admitting any of the six multimorphisms in the statement of the theorem follows from Theorem 4 via the reduction $\mathrm{VCSP}_{\mathrm{s}}(\Gamma) \le_p \mathrm{VCSP}(\Gamma \cup \{\rho_0, \rho_1\})$ discussed in the introduction. The s-tractability of PDS languages follows from Theorem 19, proved in Section 3. The s-tractability of PUS languages is then a simple corollary of Theorem 19, Observation 11, and Lemma 12.

The s-intractability of the remaining languages is proved in the full version of the paper [14]. The key in the hardness proof is to identify certain operators on weighted relations that preserve s-tractability, polymorphisms for crisp relations and multimorphisms for weighted relations, thus allowing for the construction of hardness gadgets. These operators include scaling by a nonnegative rational, adding a rational, permutation of arguments, identification of arguments, addition, the Feas($\cdot$) and Opt($\cdot$) operators, and pinning labels to variables (assuming the corresponding crisp constant is available). ◀

3 Tractability of PDS languages

We prove that PDS languages are s-tractable by a reduction to a generalised variant of the Min-Cut problem. The problem and the reduction are stated in Subsection 3.1. The tractability of the Generalised Min-Cut problem is established in Subsection 3.2.

3.1 Reduction to the Generalised Min-Cut problem

Let V be a finite set. A *set function* on V is a function $f : 2^V \to \mathbb{Q}_{\geq 0} \cup \{\infty\}$ with $f(\emptyset) = 0$.

▶ **Definition 13.** A set function $f : 2^V \to \mathbb{Q}_{\geq 0} \cup \{\infty\}$ is *increasing* if $f(X) \leq f(Y)$ for all $X \subseteq Y \subseteq V$; it is *superadditive* if $f(X) + f(Y) \leq f(X \cup Y)$ for all disjoint $X, Y \subseteq V$; it is *posimodular* if $f(X) + f(Y) \geq f(X \setminus Y) + f(Y \setminus X)$ for all $X, Y \subseteq V$; and finally it is *submodular* if $f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y)$ for all $X, Y \subseteq V$.

It is known and easy to show that any superadditive set function is also increasing.

▶ **Example 14.** Let U be a finite set and $T \subseteq U$ a non-empty subset. We define a set function f on U by $f(X) = 1$ if $T \subseteq X$ and $f(X) = 0$ otherwise. Intuitively, this corresponds to a soft NAND constraint if we interpret T as its scope and X as the set of variables assigned *true*. The set function f is superadditive, and hence also increasing.

We now formally define the Min-Cut problem introduced in Section 1.

▶ **Definition 15.** An instance of the *Min-Cut* (MC) problem is given by a graph $G = (V, E)$ with edge weights $w : E \to \mathbb{Q}_{>0}$. The goal is to minimise the objective function g, which is a set function on V defined by $g(X) = \sum_{\{u,v\} \in E, |\{u,v\} \cap X| = 1} w(u, v)$.

Note that g from Definition 15 is posimodular.

Any set X such that $\emptyset \subsetneq X \subsetneq V$ is called a *solution* of the MC problem. Note that a cut $(X, V \setminus X)$ corresponds to two solutions, namely X and $V \setminus X$. Any solution that minimises the objective function g is called *optimal*, and any optimal solution with no proper subset being an optimal solution is called *minimal*. Note that any two different minimal optimal solutions X, Y are disjoint, as $X \setminus Y$ and $Y \setminus X$ are also optimal solutions (by the posimodularity of g).

We now define the Generalised Min-Cut problem, which is key to establishing Theorem 8.

▶ **Definition 16.** Let $G = (V, E)$ be an undirected graph with edge weights $w : E \to \mathbb{Q}_{>0}$. Let g be the objective function of the Min-Cut problem on G and f a superadditive set function on V given by an oracle. A solution to the *Generalised Min-Cut* (GMC) problem is any set X such that $\emptyset \subsetneq X \subsetneq V$, and the objective is to minimise the value of $f(X) + g(X)$.

We will denote this minimum by λ. A solution achieving the minimum is called *optimal*. In case of $0 < \lambda < \infty$, we will also be looking for all α-*optimal* solutions (for $\alpha \geq 1$) to the problem, i.e. solutions X such that $f(X) + g(X) \leq \alpha\lambda$. As shown in Subsection 3.2, Theorem 26, this can be done in polynomial time (for a fixed α).

Uppman [24] used the term "Generalised Min-Cut problem" for a special case of Definition 16, in which the superadditive function f is given explicitly as a weighted sum of soft NANDs (cf. Example 14).

Our reduction from the surjective VCSP over a PDS language to the GMC problem is based on the following notion.

▶ **Definition 17.** Let γ be an r-ary weighted relation on domain $D = \{0, 1\}$. We will associate any r-tuple $\mathbf{x} \in D^r$ with the set $X = \{i \in [r] \mid x_i = 1\}$ and use them interchangeably.

Let J be an instance of the GMC problem on vertices $[r]$ with $J(X)$ denoting the objective value for any $\mathbf{x} \in D^r$ (including the all-zero and all-one tuples, which correspond to non-solutions $\emptyset$ and $[r]$). For any $\alpha \geq 1$, we say that J *α-approximates* γ if $J(X) \leq \gamma(\mathbf{x}) \leq \alpha \cdot J(X)$ for all $\mathbf{x} \in D^r$.

▶ **Lemma 18.** *Let γ be a weighted relation such that* $\mathrm{Feas}(\gamma)$ *is a downset,* $\mathrm{Opt}(\gamma)$ *is a PDS, and $\gamma(\mathbf{x}) = 0$ for $\mathbf{x} \in \mathrm{Opt}(\gamma)$. There is a constant α and an instance of the GMC problem that α-approximates γ.*

Proof. We define a set function f_{Feas} on $[r]$ as $f_{\mathrm{Feas}}(X) = 0$ if $\mathbf{x} \in \mathrm{Feas}(\gamma)$ and $f_{\mathrm{Feas}}(X) = \infty$ otherwise. Because $\mathrm{Feas}(\gamma)$ is a downset, f_{Feas} is superadditive.

By Observation 7, we can write $\mathrm{Opt}(\gamma)$ as a sum of a downset ρ on coordinates $A \subseteq [r]$ and equalities $x_i = x_j$ for $(i, j) \in E$ with $|A| + |E| = r$. Let $\mathbf{x}|_A$ denote the projection of an r-tuple $\mathbf{x}$ to coordinates A in the same order as in ρ. We define a set function f_{Opt} on $[r]$ as $f_{\mathrm{Opt}}(X) = 0$ if $\mathbf{x}|_A \in \rho$ and $f_{\mathrm{Opt}}(X) = |X \cap A|$ otherwise. Because ρ is a downset, f_{Opt} is superadditive.

We define a GMC instance J' on vertices $[r]$, unit-weight edges E, and the superadditive set function $f_{\mathrm{Feas}} + f_{\mathrm{Opt}}$. By the construction, it holds

$$J'(X) = \infty \iff f_{\mathrm{Feas}}(X) = \infty \iff \mathbf{x} \notin \mathrm{Feas}(\gamma) \iff \gamma(\mathbf{x}) = \infty \tag{1}$$

and

$$\begin{aligned} J'(X) = 0 &\iff f_{\mathrm{Feas}}(X) = f_{\mathrm{Opt}}(X) = 0 \ \wedge\ |\{i, j\} \cap X| \neq 1 \text{ for all } (i, j) \in E && (2)\\ &\iff \mathbf{x} \in \mathrm{Feas}(\gamma) \ \wedge\ \mathbf{x}|_A \in \rho \ \wedge\ x_i = x_j \text{ for all } (i, j) \in E && (3)\\ &\iff \mathbf{x} \in \mathrm{Opt}(\gamma) \iff \gamma(\mathbf{x}) = 0\,. && (4) \end{aligned}$$

Moreover, for any X such that $0 < J'(X) < \infty$ it holds $1 \leq J'(X) \leq r$.

If γ is crisp then the instance J' 1-approximates γ; otherwise let $\delta_{\min}, \delta_{\max}$ denote the minimum and maximum of $\{\gamma(\mathbf{x}) \mid 0 < \gamma(\mathbf{x}) < \infty\}$. We scale the weights of the edges and the superadditive function of J' by a factor of $\delta_{\min}/r$ to obtain an instance J such that $J(X) \leq \gamma(\mathbf{x})$ for all X. Setting $\alpha = r \cdot \delta_{\max}/\delta_{\min}$ then gives $\gamma(\mathbf{x}) \leq \alpha \cdot J(X)$ for all X. ◀

Finally, we state the reduction.

▶ **Theorem 19.** *Let Γ be a Boolean valued constraint language. If Γ is PDS, then it is s-tractable.*

Proof. Let $\Gamma' \subseteq \Gamma$ be a finite language. The feasibility relation $\mathrm{Feas}(\gamma)$ for any $\gamma \in \Gamma'$ is a PDS and hence, by Observation 7, a sum of a downset and binary equality relations. A crisp equality constraint $\rho_=(x, y)$ in an instance can be omitted after identifying the variables x and y. Therefore, we will assume that $\mathrm{Feas}(\gamma)$ is a downset. Moreover, we will assume that the minimum value assigned by γ is 0, as changing values $\gamma(\mathbf{x})$ by the same constant for all $\mathbf{x} \in D^{\mathrm{ar}(\gamma)}$ affects all assignments equally.

By Lemma 18, for any $\gamma \in \Gamma'$, there is a constant α_γ and a GMC instance J_γ that α_γ-approximates γ. Let α be the smallest integer such that $\alpha \geq \alpha_\gamma$ for all $\gamma \in \Gamma'$.

Given a $\mathrm{VCSP_s}(\Gamma')$ instance I with an objective function $\phi_I(x_1, \ldots, x_n) = \sum_{i=1}^{q} w_i \cdot \gamma_i(\mathbf{x}^i)$, we construct a GMC instance J that α-approximates ϕ_I. For $i \in [q]$, we relabel the vertices of J_{γ_i} to match the variables in the scope $\mathbf{x}^i$ of the ith constraint (i.e. vertex j is relabelled

to x^i_j) and identify vertices in case of repeated variables. We also scale both the weights of the edges of J_{γ_i} and the superadditive function by w_i. The instance J is obtained by adding up the GMC instances J_{γ_i} for all $i \in [q]$.

Let $\mathbf{x} \in D^n$ denote a surjective assignment minimising ϕ_I and $\mathbf{y} \in D^n$ an optimal solution to J with $J(Y) = \lambda$. Because J α-approximates ϕ_I, it holds

$$\lambda \leq J(X) \leq \phi_I(\mathbf{x}) \leq \phi_I(\mathbf{y}) \leq \alpha \cdot J(Y) = \alpha\lambda\,, \tag{5}$$

and hence $\mathbf{x}$ is an α-optimal solution to J. By Lemma 20 in Subsection 3.2, we can determine whether $\lambda = 0$, in which case any optimal solution to J is also optimal for ϕ_I; and whether $\lambda = \infty$. If $0 < \lambda < \infty$, we find all α-optimal solutions by Theorem 26 in Subsection 3.2. ◀

3.2 Tractability of the Generalised Min-Cut problem

Proofs of Lemmas 20, 21, and 23 can be found in the full version of the paper [14].

▶ **Lemma 20.** *There is a polynomial-time algorithm that, given an instance of the GMC problem, either finds a solution X with $f(X) + g(X) = 0$, or determines that $\lambda = \infty$, or determines that $0 < \lambda < \infty$.*

In view of Lemma 20, we can assume that $0 < \lambda < \infty$. Our goal is to show that, for a given $\alpha \geq 1$, all α-optimal solutions to a GMC instance can be found in polynomial time. This will be proved in Theorem 26; before that we need to prove several auxiliary lemmas on properties of the MC and GMC problems.

▶ **Lemma 21.** *For any instance J of the GMC problem on a graph $G = (V, E)$ and any non-empty set $V' \subseteq V$, there is an instance J' on the induced subgraph $G[V']$ that preserves the objective value of all solutions $X \subsetneq V'$. In particular, any α-optimal solution X of J such that $X \subsetneq V'$ is α-optimal for J' as well.*

▶ **Lemma 22.** *Let X be an optimal solution to an instance of the GMC problem over vertices V with $\lambda < \infty$, and Y a minimal optimal solution to the underlying MC problem. Then $X \subseteq Y$, $X \subseteq V \setminus Y$, or X is an optimal solution to the underlying MC problem.*

Proof. Assume that $X \not\subseteq Y$ and $X \not\subseteq V \setminus Y$. If $Y \subseteq X$, we have $f(Y) \leq f(X)$ as f is increasing, and hence $f(Y) + g(Y) \leq f(X) + g(X) < \infty$. Therefore, Y is optimal for the GMC problem and X is optimal for the MC problem. In the rest, we assume that $Y \not\subseteq X$.

By the posimodularity of g we have $g(X) + g(Y) \geq g(X \setminus Y) + g(Y \setminus X)$. Because $Y \setminus X$ is a proper non-empty subset of Y, it holds $g(Y \setminus X) > g(Y)$, and hence $g(X) > g(X \setminus Y)$. But then $f(X) + g(X) > f(X \setminus Y) + g(X \setminus Y)$ as $\infty > f(X) \geq f(X \setminus Y)$. Set $X \setminus Y$ is non-empty, and therefore contradicts the optimality of X. ◀

The following lemma relates the number of optimal solutions and the number of minimal optimal solutions to the MC problem. Note that this bound is tight for (unweighted) paths and cycles with at most one path attached to each vertex.

▶ **Lemma 23.** *For any instance of the MC problem on a connected graph with n vertices and p minimal optimal solutions, there are at most $p(p-1) + 2(n-p)$ optimal solutions.*

▶ **Lemma 24.** *For any instance of the GMC problem on n vertices with $0 < \lambda < \infty$, the number of optimal solutions is at most $n(n-1)$. There is an algorithm that finds all of them in polynomial time.*

Note that the bound of $n(n-1)$ optimal solutions precisely matches the known upper bound of $\binom{n}{2}$ for the number of minimum cuts [17]; the bound is tight for cycles.

Proof. Let $t(n)$ denote the maximum number of optimal solutions for such instances on n vertices. We prove the bound by induction on n. If $n = 1$, there are no solutions and hence $t(1) = 0$. For $n \geq 2$, let $Y_1, \ldots, Y_p$ be the minimal optimal solutions to the underlying MC problem. As there exists at least one minimum cut and the minimal optimal solutions are all disjoint, it holds $2 \leq p \leq n$.

Suppose that the minimal optimal solutions cover all vertices, i.e. $\bigcup Y_i = V$. By Lemma 22, any optimal solution to the GMC problem is either a proper subset of some Y_i or an optimal solution to the underlying MC problem. Restricting solutions to a proper subset of Y_i is by Lemma 21 equivalent to considering a GMC problem instance on vertices Y_i, and hence the number of such optimal solutions is bounded by $t(|Y_i|) \leq |Y_i| \cdot (|Y_i| - 1)$. Note that the sum $\sum |Y_i| \cdot (|Y_i| - 1)$ is maximised when $p - 1$ of the sets Y_i are singletons and the size of the remaining one equals $n - p + 1$. If the graph is connected then, by Lemma 23, there are at most $p(p-1) + 2(n-p)$ optimal solutions to the underlying MC problem. Adding these upper bounds we get

$$p(p-1) + 2(n-p) + \sum_{i=1}^{p} |Y_i| \cdot (|Y_i| - 1) \tag{6}$$

$$\leq p(p-1) + 2(n-p) + (p-1) \cdot 1 \cdot 0 + (n-p+1) \cdot (n-p) \tag{7}$$

$$= n(n-1) - 2(p-2) \cdot (n-p) \tag{8}$$

$$\leq n(n-1)\,. \tag{9}$$

If the graph is disconnected, the sets $Y_1, \ldots, Y_p$ are precisely its connected components. The optimal solutions to the underlying MC problem are precisely unions of connected components (with the exception of $\emptyset$ and V), which means that there can be exponentially many of them. However, only the sets $Y_1, \ldots, Y_p$ themselves can be optimal solutions to the GMC problem: We have $0 < \lambda \leq f(Y_i) + g(Y_i) = f(Y_i)$. Because f is superadditive, it holds $f(Y_{i_1} \cup \cdots \cup Y_{i_k}) \geq f(Y_{i_1}) + \cdots + f(Y_{i_k}) \geq k\lambda$ for any distinct $i_1, \ldots, i_k$, and hence no union of two or more connected components can be an optimal solution to the GMC problem. This gives us an upper bound of $p \leq p(p-1) + 2(n-p)$, and the rest follows as in the previous case.

Finally, suppose that $\bigcup Y_i \neq V$, and hence the graph is connected. This case can be handled similarly as for $\bigcup Y_i = V$. (The full version of the paper [14] includes a complete proof.)

Using a procedure generating all minimum cuts [25], it is straightforward to turn the above proof into a recursive algorithm that finds all optimal solutions in polynomial time. ◀

▶ **Lemma 25.** *Let $\alpha, \beta \geq 1$. Let X be an α-optimal solution to an instance of the GMC problem over vertices V with $0 < \lambda < \infty$, and Y an optimal solution to the underlying MC problem. If $g(Y) < \lambda/\beta$, then*

$$(f(X \setminus Y) + g(X \setminus Y)) + (f(X \cap Y) + g(X \cap Y)) < \left(\alpha + \frac{2}{\beta}\right) \lambda\,; \tag{10}$$

if $g(Y) \geq \lambda/\beta$, then X is an $\alpha\beta$-optimal solution to the underlying MC problem.

Proof. If $g(Y) \geq \lambda/\beta$, it holds $g(X) \leq f(X) + g(X) \leq \alpha\lambda \leq \alpha\beta \cdot g(Y)$, and hence X is an $\alpha\beta$-optimal solution to the underlying MC problem. In the rest we assume that $g(Y) < \lambda/\beta$.

Because g is posimodular, we have

$$g(X) + g(Y) \geq g(X \setminus Y) + g(Y \setminus X) \tag{11}$$
$$g(Y) + g(Y \setminus X) \geq g(X \cap Y) + g(\emptyset)\,, \tag{12}$$

and hence

$$g(X) + 2g(Y) \geq g(X \setminus Y) + g(X \cap Y)\,. \tag{13}$$

By superadditivity of f, it holds $f(X) \geq f(X \setminus Y) + f(X \cap Y)$. The claim then follows from the fact that $f(X) + g(X) + 2g(Y) < (\alpha + 2/\beta)\lambda$. ◀

Finally, we prove that α-optimal solutions to the GMC problem can be found in polynomial time.

▶ **Theorem 26.** *For any instance of the GMC problem on n vertices with $0 < \lambda < \infty$ and $\alpha \in \mathbb{Z}_{\geq 1}$, the number of α-optimal solutions is at most $n^{20\alpha-15}$. There is an algorithm that finds all of them in polynomial time.*

Note that for a cycle on n vertices, the number of α-optimal solutions to the MC problem is $\Theta(n^{2\alpha})$, and thus the exponent in our bound is asymptotically tight in α.

Proof. Let $\beta \in \mathbb{Z}_{\geq 3}$ be a parameter. Throughout the proof, we relax the integrality restriction on α and require only that $\alpha\beta$ is an integer. For $\alpha = 1$, the claim follows from Lemma 24, therefore we assume $\alpha \geq 1 + 1/\beta$ in the rest of the proof.

Define $\ell(x) = \frac{2(\beta+1)}{\beta-2} \cdot (\beta x - 3)$. We will prove that the number of α-optimal solutions is at most $n^{\ell(\alpha)}$; taking $\beta = 4$ then gives the claimed bound. Function ℓ was chosen as a slowest growing function satisfying the following properties required in this proof: It holds $\ell(x) + \ell(y) \leq \ell(x + y - 3/\beta)$ for any x, y, and $\ell(x) \geq 2\beta x$ for any $x \geq 1 + 1/\beta$.

We prove the bound by induction on $n + \alpha\beta$. As it trivially holds for $n \leq 2$, we will assume $n \geq 3$ in the rest of the proof. Let Y be an optimal solution to the underlying MC problem with $k = |Y| \leq n/2$. If $g(Y) \geq \lambda/\beta$ then, by Lemma 25, any α-optimal solution to the GMC problem is an $\alpha\beta$-optimal solution to the underlying MC problem. Because $g(Y) \geq \lambda/\beta > 0$, the graph is connected, and hence there are at most $2^{2\alpha\beta}\binom{n}{2\alpha\beta} \leq n^{2\alpha\beta} \leq n^{\ell(\alpha)}$ such solutions by [17]. (In detail, [17, Theorem 6.2] shows that the number of $\alpha\beta$-optimal cuts in an n-vertex graph is $2^{2\alpha\beta-1}\binom{n}{2\alpha\beta}$, and every cut corresponds to two solutions.)

From now on we assume that $g(Y) < \lambda/\beta$ and hence inequality (10) holds. Upper bounds in this case may be quite loose; in particular, we will use the following inequalities:

$$(k/n)^{\ell(\alpha)} \leq (k/n)^{\ell(1+1/\beta)} = (k/n)^{2(\beta+1)} \leq (k/n)^8 \leq (k/n)(1/2)^7 = k/128n \tag{14}$$
$$(1/n)^{2\beta} \leq (1/n)^6 \leq (1/n)(1/3)^5 < 1/128n\,. \tag{15}$$

Consider any α-optimal solution to the GMC problem X.

If $X \subsetneq Y$ then, by Lemma 21, X is an α-optimal solution to an instance on vertices Y. By the induction hypothesis, there are at most $k^{\ell(\alpha)} \leq (k/128n) \cdot n^{\ell(\alpha)}$ such solutions.

Similarly, if $X \subsetneq V \setminus Y$, then X is an α-optimal solution to an instance on vertices $V \setminus Y$, and there are at most $(n-k)^{\ell(\alpha)} = (1-k/n)^{\ell(\alpha)} \cdot n^{\ell(\alpha)} \leq (1-k/n) \cdot n^{\ell(\alpha)}$ such solutions.

If $Y \subsetneq X$, then $X \setminus Y$ is an $(\alpha - 1 + 2/\beta)$-optimal solution on vertices $V \setminus Y$ by (10) and the fact that $f(X \cap Y) + g(X \cap Y) \geq \lambda$. Similarly, if $V \setminus Y \subsetneq X$, then $X \cap Y$ is an $(\alpha - 1 + 2/\beta)$-optimal solution on vertices Y. In either case, we bound the number of such solutions depending on the value of α: For $\alpha < 2 - 2/\beta$, there are trivially none; for

$\alpha = 2 - 2/\beta$, Lemma 24 gives a bound of $n(n-1) \leq n^{\ell(\alpha)-2\beta}$; and for $\alpha > 2 - 2/\beta$ we get an upper bound of $n^{\ell(\alpha-1+2/\beta)} \leq n^{\ell(\alpha)-2\beta}$ by the induction hypothesis. The number of solutions is thus at most $n^{\ell(\alpha)-2\beta} \leq (1/128n) \cdot n^{\ell(\alpha)}$ for any α.

Finally, we consider X such that $\emptyset \subsetneq X \setminus Y \subsetneq V \setminus Y$ and $\emptyset \subsetneq X \cap Y \subsetneq Y$, i.e. $X \setminus Y$ and $X \cap Y$ are solutions on vertices $V \setminus Y$ and Y respectively. Let i be the integer for which

$$\left(1 + \frac{i}{\beta}\right)\lambda \leq f(X \cap Y) + g(X \cap Y) < \left(1 + \frac{i+1}{\beta}\right)\lambda\,. \tag{16}$$

Then, by (10), it holds $f(X \setminus Y) + g(X \setminus Y) < (\alpha - 1 - (i-2)/\beta)\lambda$. Therefore, $X \cap Y$ is a $(1 + (i+1)/\beta)$-optimal solution on vertices Y and $X \setminus Y$ is an $(\alpha - 1 - (i-2)/\beta)$-optimal solution on vertices $V \setminus Y$. Because $0 \leq i \leq (\alpha - 2)\beta + 1$, we can bound the number of such solutions by the induction hypothesis as at most

$$k^{\ell\left(1+\frac{i+1}{\beta}\right)} \cdot (n-k)^{\ell\left(\alpha-1-\frac{i-2}{\beta}\right)} \leq \left(\frac{k}{n}\right)^{\ell\left(1+\frac{i+1}{\beta}\right)} \cdot n^{\ell\left(1+\frac{i+1}{\beta}\right)+\ell\left(\alpha-1-\frac{i-2}{\beta}\right)} \tag{17}$$

$$\leq \left(\frac{k}{n}\right)^{2(\beta+1)} \cdot \frac{1}{2^i} \cdot n^{\ell(\alpha)}\,, \tag{18}$$

which is at most $2 \cdot (k/128n) \cdot n^{\ell(\alpha)}$ in total for all i.

By adding up the bounds we get that the number of α-optimal solutions is at most $n^{\ell(\alpha)}$. A polynomial-time algorithm that finds the α-optimal solutions follows from the above proof using a procedure generating all $\alpha\beta$-optimal cuts [25]. ◀

4 Conclusions

While the complexity of (valued) constraint languages is, as in this paper, studied mostly for finite languages, all known results also hold for languages of infinite size. We now show that this is *not* the case for Boolean surjective VCSPs.

▶ **Example 27.** We give an example of an infinite Boolean valued constraint language Γ that is a PDS language but $\mathrm{VCSP}_{\mathrm{s}}(\Gamma)$ is NP-hard.

Let $D = \{0,1\}$. For any $w \in \mathbb{Z}_{\geq 1}$, we define $\gamma_w : D^3 \to \overline{\mathbb{Q}}$ by $\gamma(0,0,0) = 0$, $\gamma(\cdot,\cdot,0) = w$, $\gamma(x,y,1) = 2$ if $x = y$ and $\gamma(x,y,1) = 1$ if $x \neq y$. Note that $\mathrm{Feas}(\gamma_w) = D^3$ and $\mathrm{Opt}(\gamma_w) = \{(0,0,0)\}$ are PDS relations. Let $\Gamma = \{\gamma_w \mid w \in \mathbb{Z}_{\geq 1}\}$.

Given an instance $G = (V,E)$ of the Max-Cut problem with $V = \{x_1, \ldots, x_n\}$ and no isolated vertices, we choose a value $w > 2|E|$ and construct a $\mathrm{VCSP}_{\mathrm{s}}(\Gamma)$ instance $I = (V \cup \{z\}, D, \phi_I)$ with $\phi_I(x_1, \ldots, x_n, z) = \sum_{\{i,j\} \in E, i<j} \gamma_w(x_i, x_j, z)$. Cuts in G are in one-to-one correspondence with assignments to I satisfying $z = 1$. In particular, a cut of value k corresponds to an assignment to I of value $k + 2(|E| - k) = 2|E| - k$. Moreover, any surjective assignment that assigns label 0 to variable z is of value at least $w > 2|E| \geq 2|E| - k$. Thus, solving I amounts to solving Max-Cut in G.

An obvious open problem is to consider surjective VCSPs on a three-element domain. A complexity classification is known for $\{0,\infty\}$-valued languages [5] and $\mathbb{Q}$-valued languages [15] (the latter generalises the $\{0,1\}$-valued case obtained in [16]). In fact [18] implies a dichotomy for $\overline{\mathbb{Q}}$-valued languages on a three-element domain. However, all these results depend on the notion of core and the presence of constants in the language, and thus it is unclear how to use them to obtain a complexity classification in the surjective setting. Moreover, one special case of the CSP on a three-element domain is the *3-No-Rainbow-Colouring* problem [4], whose complexity status is open.

References

1. Libor Barto. Constraint satisfaction problem and universal algebra. *ACM SIGLOG News*, 1(2):14–24, 2014. doi:10.1145/2677161.2677165.
2. Libor Barto and Marcin Kozik. Constraint Satisfaction Problems Solvable by Local Consistency Methods. *Journal of the ACM*, 61(1), 2014. Article No. 3. doi:10.1145/2556646.
3. Libor Barto, Andrei Krokhin, and Ross Willard. Polymorphisms, and how to use them. In Andrei Krokhin and Stanislav Živný, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 1–44. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017. doi:10.4230/DFU.Vol7.15301.1.
4. Manuel Bodirsky, Jan Kára, and Barnaby Martin. The complexity of surjective homomorphism problems - a survey. *Discrete Applied Mathematics*, 160(12):1680–1690, 2012. doi:10.1016/j.dam.2012.03.029.
5. Andrei Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *Journal of the ACM*, 53(1):66–120, 2006. doi:10.1145/1120582.1120584.
6. Andrei Bulatov, Andrei Krokhin, and Peter Jeavons. Classifying the Complexity of Constraints using Finite Algebras. *SIAM Journal on Computing*, 34(3):720–742, 2005. doi:10.1137/S0097539700376676.
7. Hubie Chen. An algebraic hardness criterion for surjective constraint satisfaction. *Algebra universalis*, 72(4):393–401, 2014. doi:10.1007/s00012-014-0308-x.
8. David A. Cohen, Martin C. Cooper, Peter G. Jeavons, and Andrei A. Krokhin. The Complexity of Soft Constraint Satisfaction. *Artificial Intelligence*, 170(11):983–1016, 2006. doi:10.1016/j.artint.2006.04.002.
9. Nadia Creignou. A dichotomy theorem for maximum generalized satisfiability problems. *Journal of Computer and System Sciences*, 51(3):511–522, 1995. doi:10.1006/jcss.1995.1087.
10. Nadia Creignou and Jean-Jacques Hébrard. On generating all solutions of generalized satisfiability problems. *ITA*, 31(6):499–511, 1997.
11. Tomás Feder and Moshe Y. Vardi. The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: A Study through Datalog and Group Theory. *SIAM Journal on Computing*, 28(1):57–104, 1998. doi:10.1137/S0097539794266766.
12. Jiří Fiala and Jan Kratochvíl. Locally constrained graph homomorphisms - structure, complexity, and applications. *Computer Science Review*, 2(2):97–111, 2008. doi:10.1016/j.cosrev.2008.06.001.
13. Jiří Fiala and Daniël Paulusma. A complete complexity classification of the role assignment problem. *Theoretical Computer Science*, 349(1):67–81, 2005. doi:10.1016/j.tcs.2005.09.029.
14. Peter Fulla and Stanislav Živný. The complexity of Boolean surjective general-valued CSPs. *CoRR*, abs/1702.04679, 2017. URL: http://arxiv.org/abs/1702.04679.
15. Anna Huber, Andrei Krokhin, and Robert Powell. Skew bisubmodularity and valued CSPs. *SIAM Journal on Computing*, 43(3):1064–1084, 2014. doi:10.1137/120893549.
16. Peter Jonsson, Mikael Klasson, and Andrei A. Krokhin. The approximability of three-valued MAX CSP. *SIAM Journal on Computing*, 35(6):1329–1349, 2006. doi:10.1137/S009753970444644X.
17. David R. Karger. Global Min-cuts in RNC, and Other Ramifications of a Simple Min-Cut Algorithm. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'93)*, pages 21–30, 1993.
18. Vladimir Kolmogorov, Andrei A. Krokhin, and Michal Rolínek. The complexity of general-valued CSPs. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS'15)*. IEEE Computer Society, 2015.

19 Marcin Kozik and Joanna Ochremiak. Algebraic properties of valued constraint satisfaction problem. In *Proceedings of the 42nd International Colloquium on Automata, Languages and Programming (ICALP'15)*, volume 9134 of *Lecture Notes in Computer Science*, pages 846–858. Springer, 2015. doi:10.1007/978-3-662-47672-7_69.

20 Barnaby Martin and Daniël Paulusma. The computational complexity of disconnected cut and $2K_2$-partition. *Journal of Combinatorial Theory, Series B*, 111:17–37, 2015. doi:10.1016/j.jctb.2014.09.002.

21 Thomas J. Schaefer. The Complexity of Satisfiability Problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC'78)*, pages 216–226. ACM, 1978. doi:10.1145/800133.804350.

22 Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer, 2003.

23 Mechthild Stoer and Frank Wagner. A simple min-cut algorithm. *Journal of the ACM*, 44(4):585–591, 1997. doi:10.1145/263867.263872.

24 Hannes Uppman. Max-Sur-CSP on Two Elements. In *Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming (CP'12)*, volume 7514 of *Lecture Notes in Computer Science*, pages 38–54. Springer, 2012. doi:10.1007/978-3-642-33558-7_6.

25 Vijay V. Vazirani and Mihalis Yannakakis. Suboptimal Cuts: Their Enumeration, Weight and Number (Extended Abstract). In *Proceedings of the 19th International Colloquium on Automata, Languages and Programming (ICALP'92)*, pages 366–377. Springer-Verlag, 1992. URL: http://dl.acm.org/citation.cfm?id=646246.684856.

26 Narayan Vikas. Algorithms for partition of some class of graphs under compaction and vertex-compaction. *Algorithmica*, 67(2):180–206, 2013. doi:10.1007/s00453-012-9720-9.