Article:

Chen, X, Dyer, M orcid.org/0000-0002-2018-0374, Goldberg, LA et al. (4 more authors) (2015) The complexity of approximating conservative counting CSPs. Journal of Computer and System Sciences, 81 (1). pp. 311-329. ISSN 0022-0000

https://doi.org/10.1016/j.jcss.2014.06.006

The complexity of approximating conservative counting CSPs*

Xi Chen[†] Martin Dyer[‡] Leslie Ann Goldberg[§] Mark Jerrum[¶]
Pinyan Lu[‖] Colin McQuillan[**] David Richerby[§]

May 30, 2014

Abstract

We study the complexity of approximately solving the weighted counting constraint satisfaction problem #CSP($\mathcal{F}$). In the conservative case, where $\mathcal{F}$ contains all unary functions, there is a classification known for the case in which the domain of functions in $\mathcal{F}$ is Boolean. In this paper, we give a classification for the more general problem where functions in $\mathcal{F}$ have an arbitrary finite domain. We define the notions of *weak log-modularity* and *weak log-supermodularity*. We show that if $\mathcal{F}$ is weakly log-modular, then #CSP($\mathcal{F}$) is in FP. Otherwise, it is at least as difficult to approximate as #BIS, the problem of counting independent sets in bipartite graphs. #BIS is complete with respect to approximation-preserving reductions for a logically defined complexity class #RHΠ_1, and is believed to be intractable. We further sub-divide the #BIS-hard case. If $\mathcal{F}$ is weakly log-supermodular, then we show that #CSP($\mathcal{F}$) is as easy as a Boolean log-supermodular weighted #CSP. Otherwise, we show that it is NP-hard to approximate. Finally, we give a full trichotomy for the arity-2 case, where #CSP($\mathcal{F}$) is in FP, or is #BIS-equivalent, or is equivalent in difficulty to #SAT, the problem of approximately counting the satisfying assignments of a Boolean formula in conjunctive normal form. We also discuss the algorithmic aspects of our classification.

1 Introduction

In the weighted counting constraint satisfaction problem, there is a fixed finite domain D and a fixed finite "weighted constraint language" $\mathcal{F}$, which is a set of functions. Every

*This work was partially supported by the EPSRC Research Grant "Computational Counting".

†Department of Computer Science, Columbia University, 450 Computer Science Building, 1214 Amsterdam Avenue, Mailcode: 0401, New York, NY 10027-7003, USA. Supported by NSF Grant CCF-1139915 and start-up funds from Columbia University.

‡School of Computing, University of Leeds, Leeds, LS2 9JT, UK.

§Department of Computer Science, University of Oxford, Wolfson Building, Parks Road, Oxford, OX1 3QD, UK. The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) ERC grant agreement no. 334828. 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.

¶School of Mathematical Sciences, Queen Mary, University of London, Mile End Road, London, E1 4NS, UK.

‖Microsoft Research Asia, Microsoft Shanghai Technology Park, No 999, Zixing Road, Minhang District, Shanghai, 200241, China.

**Department of Computer Science, University of Liverpool, Liverpool, L69 3BX, UK.

function $F \in \mathcal{F}$ maps a tuple of domain elements to a value called a "weight". In the computational problem #CSP($\mathcal{F}$), an instance consists of a set $V = \{v_1, \dots, v_n\}$ of variables and a set of "weighted constraints". Each weighted constraint applies a function from $\mathcal{F}$ to an appropriate-sized tuple of variables.

For example, with the Boolean domain $D = \{0, 1\}$ we could consider the situation in which $\mathcal{F}$ consists of the single binary (arity-2) function F defined by $F(0, 0) = F(0, 1) = F(1, 0) = 1$ and $F(1, 1) = 2$. We can construct an instance with variables v_1, v_2 and v_3 and weighted constraints $F(v_1, v_2)$ and $F(v_2, v_3)$. If $\mathbf{x}$ is an assignment of domain elements to the variables then the total weight associated with $\mathbf{x}$ is the product of the weighted constraints, evaluated at $\mathbf{x}$.

For example, the assignment that maps v_1, v_2 and v_3 all to 0 has weight $F(0, 0)F(0, 0) = 1$ but the assignment that maps all of them to 1 has weight $F(1, 1)F(1, 1) = 4$. Two assignments have weight $F(1, 1)F(1, 0) = F(0, 1)F(1, 1) = 2$ and the other four assignments each have weight 1. The computational problem is to evaluate the sum of the weights of the assignments. For this instance, the solution is 13.

There has been a lot of work on classifying the computational difficulty of exactly solving #CSP($\mathcal{F}$). For some weighted constraint languages $\mathcal{F}$, this is a computationally easy task, while for others, it is intractable. We will give a brief summary of what is known. For more details, see the surveys of Chen [11] and Lu [23].

First, suppose that the domain D is Boolean (that is, suppose that $D = \{0, 1\}$). For this case, Creignou and Hermann [15] gave a dichotomy for the case in which weights are also in $\{0, 1\}$. In this case, they showed that #CSP($\mathcal{F}$) is in FP (the set of polynomial-time computable function problems) if all of the functions in $\mathcal{F}$ are affine, and that otherwise, it is #P-complete. Dyer, Goldberg, and Jerrum [17] extended this to the case in which weights are non-negative rationals. For this case, they showed that the problem is solvable in polynomial time if (1) every function in $\mathcal{F}$ is expressible as a product of unary functions, equalities and disequalities, or (2) every function in $\mathcal{F}$ is a constant multiple of an affine function. Otherwise, they showed the problem is complete in the complexity class $\mathrm{FP}^{\#\mathrm{P}}$. We will not deal with negative weights in this paper. However, it is worth mentioning that these results have been extended to the case in which weights can be negative [5], to the case in which they can be complex [21], and to the related class of Holant* problems [10]. Other dichotomies are also known for Holant problems (see [23]).

Next, consider an arbitrary finite domain D. For the case in which weights are in $\{0, 1\}$, Bulatov's breakthrough result [3] showed that #CSP($\mathcal{F}$) is always either in FP or #P-hard. A simplified version was given by Dyer and Richerby [19], who introduced a new criterion called "strong balance". The dichotomy was extended to include non-negative rational weights by Bulatov, Dyer, Goldberg, Jalsenius, Jerrum and Richerby [4] and then to include all non-negative algebraic weights by Cai, Chen and Lu [8, 9]. Cai, Chen and Lu gave a generalised notion of balance that we will use in this work. Finally, Cai and Chen [7] extended the dichotomy to include all algebraic complex weights. The criterion for the unweighted #CSP dichotomy is known to be decidable [19] and this carries through to non-negative rational weights and non-negative algebraic weights [8]. Decidability is currently open for the complex case.

Much less is known about the complexity of approximately solving #CSP($\mathcal{F}$). Before describing what is known, it helps to say a little about the complexity of approximate counting within #P. Dyer, Goldberg, Greenhill and Jerrum [16] identified three complexity classes for approximation problems within #P. These are:

1. problems that have a fully polynomial randomised approximation scheme (FPRAS),

2. a logically defined complexity class called $\#\mathrm{RH}\Pi_1$, and

3. a class of problems for which approximation is NP-hard.

A typical complete problem in the class $\#\mathrm{RH}\Pi_1$ is #BIS, the problem of approximately counting independent sets in bipartite graphs. It is known that either all complete problems in $\#\mathrm{RH}\Pi_1$ have an FPRAS, or none do; it is conjectured that none do [20]. A typical complete problem in the third class is #SAT, the problem of counting satisfying assignments of a Boolean formula in conjunctive normal form. Another concept that turns out to be important in the classification of approximate counting CSPs is log-supermodularity [6]. A function with Boolean domain is log-supermodular if its logarithm is supermodular; we give a formal definition later.

Given those rough ideas, we can now describe what is known about the complexity of approximately solving $\#\mathrm{CSP}(\mathcal{F})$ when the domain, D, is Boolean. For the case in which weights are in $\{0, 1\}$, Dyer, Goldberg and Jerrum gave a trichotomy [18]. If every function in $\mathcal{F}$ is affine, then $\#\mathrm{CSP}(\mathcal{F})$ is in FP. Otherwise, it is as hard to approximate as #BIS. The hard approximation problems are divided into #BIS-equivalent cases (which arise when the functions in $\mathcal{F}$ can be expressed using "implies" and fixing the values of certain variables) and the remaining cases, which are all shown to be #SAT-equivalent.

In the more general case where the domain D is still Boolean, but the weights can be arbitrary non-negative values, no complete classification is known. However, Bulatov, Dyer, Goldberg, Jerrum and McQuillan [6] gave a classification for the so-called "conservative" case, in which $\mathcal{F}$ contains all unary functions. Their result is reproduced as Lemma 7 below. Here is an informal description.

- If every function in $\mathcal{F}$ can be expressed in a certain simple way using disequality and unary functions, then, for any finite $\mathcal{G} \subset \mathcal{F}$, $\#\mathrm{CSP}(\mathcal{G})$ has an FPRAS.

- Otherwise,

  - there is a finite $\mathcal{G} \subset \mathcal{F}$ such that $\#\mathrm{CSP}(\mathcal{G})$ is at least as hard to approximate as #BIS and,
  - if $\mathcal{F}$ contains any function that is not log-supermodular, then there is a finite $\mathcal{G} \subset \mathcal{F}$ such that $\#\mathrm{CSP}(\mathcal{G})$ is at least as hard to approximate as #SAT.

Yamakami [29] has also given an approximation dichotomy for the case in which even more unary functions (including those with negative weights) are assumed to be part of $\mathcal{F}$. The negative weights introduce cancellation, making more weighted constraint languages $\mathcal{F}$ intractable. In this paper, we stick to the non-negative case, in which more subtle complexity classifications arise.

Prior to this paper, there were no known complexity classifications for approximately solving $\#\mathrm{CSP}(\mathcal{F})$ for the case in which the domain D is not Boolean. Thus, this is the problem that we address in this paper. Our main result (Theorem 6, below) is a complexity classification for the conservative case (where all unary weights are contained in $\mathcal{F}$). Here is an informal description of the result.

- If $\mathcal{F}$ is "weakly log-modular" (a concept we define below) then, for any finite $\mathcal{G} \subset \mathcal{F}$, $\#\mathrm{CSP}(\mathcal{G})$ is in FP.

- Otherwise, there is a finite $\mathcal{G} \subset \mathcal{F}$ such that #CSP($\mathcal{G}$) is at least as hard to approximate as #BIS. Furthermore,
  - if $\mathcal{F}$ is "weakly log-supermodular" (again, defined below) then, for any finite $\mathcal{G} \subset \mathcal{F}$, there is a finite set $\mathcal{G}'$ of log-supermodular functions on the Boolean domain such that #CSP($\mathcal{G}$) is as easy to approximate as #CSP($\mathcal{G}'$);
  - otherwise, there is a finite $\mathcal{G} \subset \mathcal{F}$ such that #CSP($\mathcal{G}$) is as hard to approximate as #SAT.

Informally, $\mathcal{F}$ is weakly log-supermodular if, for every binary function F that can be expressed using functions in $\mathcal{F}$, every projection of F onto two domain elements is log-supermodular (see Definition 4). Thus, in some sense, our result shows that all the difficulty of approximating conservative weighted constraint satisfaction problems arises in the Boolean case. Even when the domain D is larger, approximations which are #SAT-equivalent are #SAT-equivalent precisely because of intractable Boolean problems which arise as subproblems.

In addition to the complexity classifications described above (FP versus #BIS-hard and "as easy as a Boolean log-supermodular problem" versus #SAT-equivalent) we also give a full trichotomy for the binary case (i.e., where all functions in $\mathcal{F}$ have arity 1 or 2).

- If $\mathcal{F}$ is weakly log-modular then, for any finite $\mathcal{G} \subset \mathcal{F}$, #CSP($\mathcal{G}$) is in FP.
- Otherwise, if $\mathcal{F}$ is weakly log-supermodular, then
  - for every finite $\mathcal{G} \subset \mathcal{F}$, #CSP($\mathcal{G}$) is as easy to approximate as #BIS and
  - there is a finite $\mathcal{G} \subset \mathcal{F}$ such that #CSP($\mathcal{G}$) is as hard to approximate as #BIS.
- Otherwise, there is a finite $\mathcal{G} \subset \mathcal{F}$ such that #CSP($\mathcal{G}$) is as hard to approximate as #SAT.

The final section of our paper discusses the algorithmic aspects of our classification for the case in which $\mathcal{F}$ is the union of a finite, weighted constraint language $\mathcal{H}$ and the set of all unary functions. In particular, we give an algorithm that takes $\mathcal{H}$ as input and correctly makes one of the following deductions:

1. #CSP($\mathcal{G}$) is in FP for every finite $\mathcal{G} \subset \mathcal{F}$;
2. #CSP($\mathcal{G}$) is LSM-easy for every finite $\mathcal{G} \subset \mathcal{F}$ and #BIS-hard for some such $\mathcal{G}$;
3. #CSP($\mathcal{G}$) is #BIS-easy for every finite $\mathcal{G} \subset \mathcal{F}$ and #BIS-equivalent for some such $\mathcal{G}$;
4. #CSP($\mathcal{G}$) is #SAT-easy for all finite $\mathcal{G} \subset \mathcal{F}$ and #SAT-equivalent for some such $\mathcal{G}$.

Further, if every function in $\mathcal{H}$ has arity at most 2, the output is not deduction 2. The term "LSM-easy" in deduction 2 will be formally defined later. Informally, it means "as easy as a Boolean log-supermodular weighted counting CSP".

1.1 Previous work

The first contribution of our paper is to show that, if $\mathcal{F}$ is weakly log-modular then, for any finite $\mathcal{G} \subset \mathcal{F}$, #CSP($\mathcal{G}$) is in FP. Otherwise, there is a finite $\mathcal{G} \subset \mathcal{F}$ for which #CSP($\mathcal{G}$) is at least as hard to approximate as #BIS. We also show that, if $\mathcal{F}$ is not weakly log-supermodular, then there is a finite $\mathcal{G} \subset \mathcal{F}$, such that #CSP($\mathcal{G}$) is #SAT-hard. This work is presented in Sections 2 and 3 below and builds on two strands of previous work.

- The hardness results build on the approximation classification in the Boolean case [6] and, in particular, on the key role played by log-supermodular functions.
- The easiness results build on the classification of the exact evaluation of #CSP($\mathcal{F}$) in the general case [8], and in particular on the key role played by "balance".

The second contribution of our paper is to show that, if $\mathcal{F}$ is weakly log-supermodular, then, for any finite $\mathcal{G} \subset \mathcal{F}$, there is a finite set $\mathcal{G}'$ of log-supermodular functions on the Boolean domain such that #CSP($\mathcal{G}$) is as easy to approximate as #CSP($\mathcal{G}'$). This builds on three key studies of the complexity of optimisation CSPs by Takhanov [26,27], Cohen, Cooper and Jeavons [12] and Kolmogorov and Živný [22]. In all three cases, we use their arguments and ideas, and not merely their results. Thus, we delve into these three papers in some detail.

Our final contribution is the trichotomy for the binary case. This relies additionally on work of Cohen, Cooper, Jeavons and Krokhin [13] generalising Rudolf and Woeginger's decomposition [25] of Monge matrices. The Monge property can be viewed as a generalisation of binary submodular functions to a larger domain, and the decomposition shows how to decompose such functions in a useful manner.

1.2 Preliminaries and statement of results

Let D be a finite domain with $|D| \geq 2$. It will be convenient to refer to the set $\mathrm{Func}_k(D,R)$ of all functions $D^k \to R$ for some codomain R, and the set $\mathrm{Func}(D,R) = \bigcup_{k=0}^{\infty} \mathrm{Func}_k(D,R)$. Let EQ be the binary equality function defined by $\mathrm{EQ}(x,x) = 1$ and $\mathrm{EQ}(x,y) = 0$ for $x \neq y$; let $\mathrm{NEQ}(x,y) = 1 - \mathrm{EQ}(x,y)$.

We use the following definitions from [6]. Let $\mathcal{F}$ be a subset of $\mathrm{Func}(D,R)$. Let $V = \{v_1,\dots,v_n\}$ be a set of variables. An atomic formula has the form $\varphi = G(v_{i_1},\dots,v_{i_a})$ where $G \in \mathcal{F}$, $a = a(G)$ is the arity of G, and $(v_{i_1},v_{i_2},\dots,v_{i_a}) \in V^a$ is called a "scope". Note that repeated variables are allowed. The function $F_\varphi\colon D^n \to R$ represented by the atomic formula $\varphi = G(v_{i_1},\dots,v_{i_a})$ is just $F_\varphi(\mathbf{x}) = G(\mathbf{x}(v_{i_1}),\dots,\mathbf{x}(v_{i_a}))$, where $\mathbf{x}\colon \{v_1,\dots,v_n\} \to D$ is an assignment to the variables. To simplify the notation, we write $x_j = \mathbf{x}(v_j)$ so

$$F_\varphi(\mathbf{x}) = G(x_{i_1},\dots,x_{i_a}).$$

A pps-formula ("primitive product summation formula") is a finite summation of a finite product of atomic formulas. A pps-formula ψ over $\mathcal{F}$ in variables $V' = \{v_1,\dots,v_{n+k}\}$ has the form

$$\psi = \sum_{v_{n+1},\dots,v_{n+k}} \prod_{j=1}^{m} \varphi_j,$$

where φ_j are all atomic formulas over $\mathcal{F}$ in the variables V'. (The variables V are free, and the others, $V' \setminus V$, are bound.) The formula ψ specifies a function $F_\psi\colon D^n \to R$ in the following way:

$$F_\psi(\mathbf{x}) = \sum_{\mathbf{y}\in D^k} \prod_{j=1}^{m} F_{\varphi_j}(\mathbf{x},\mathbf{y}),$$

where $\mathbf{x}$ and $\mathbf{y}$ are assignments $\mathbf{x}\colon \{v_1,\dots,v_n\} \to D$ and $\mathbf{y}\colon \{v_{n+1},\dots,v_{n+k}\} \to D$. The *functional clone* $\langle\mathcal{F}\rangle_\#$ *generated by* $\mathcal{F}$ is the set of all functions that can be represented by a pps-formula over $\mathcal{F}\cup\{\mathrm{EQ}\}$. Crucially, $\langle\langle\mathcal{F}\rangle_\#\rangle_\# = \langle\mathcal{F}\rangle_\#$ [6, Lemma 2.1]; we will rely on this transitivity property implicitly.

Definition 1. A *weighted constraint language* $\mathcal{F}$ is a subset of $\mathrm{Func}(D,\mathbb{Q}_{\geq 0})$. Functions in $\mathcal{F}$ are called *weight functions*.

In Section 4 we will introduce *valued constraint languages* and *cost functions*, which pertain to optimisation CSPs. It is important to distinguish these from the weighted version, which is used for counting.

Definition 2. A weighted constraint language $\mathcal{F}$ is *conservative* if $\mathcal{U}_D \subseteq \mathcal{F}$, where $\mathcal{U}_D = \mathrm{Func}_1(D,\mathbb{Q}_{\geq 0})$.

Definition 3. A weighted constraint language $\mathcal{F}$ is *weakly log-modular* if, for all binary functions $F\in\langle\mathcal{F}\rangle_\#$ and elements $a,b\in D$,

$$\begin{aligned} &F(a,a)F(b,b) = F(a,b)F(b,a), \text{ or}\\ &F(a,a) = F(b,b) = 0, \text{ or}\\ &F(a,b) = F(b,a) = 0. \end{aligned} \tag{1}$$

Definition 4. $\mathcal{F}$ is *weakly log-supermodular* if, for all binary functions $F\in\langle\mathcal{F}\rangle_\#$ and elements $a,b\in D$,

$$F(a,a)F(b,b) \geq F(a,b)F(b,a) \quad \text{or} \quad F(a,a) = F(b,b) = 0. \tag{2}$$

Definition 5. A function $F\in \mathrm{Func}_k(\{0,1\},\mathbb{Q}_{\geq 0})$ is *log-supermodular* if

$$F(\mathbf{x}\vee\mathbf{y})F(\mathbf{x}\wedge\mathbf{y}) \geq F(\mathbf{x})F(\mathbf{y})$$

for all $\mathbf{x},\mathbf{y}\in\{0,1\}^k$, where $\wedge$ (min) and $\vee$ (max) are applied component-wise. LSM is the set of all log-supermodular functions in $\mathrm{Func}(\{0,1\},\mathbb{Q}_{\geq 0})$.

It is known [6, Lemma 4.2] that $\langle\mathsf{LSM}\rangle_\# = \mathsf{LSM}$. Here is a precise statement of the computational task that we study. A #CSP problem is parameterised by a finite, weighted constraint language $\mathcal{F}$ as follows.

Name $\#\mathrm{CSP}(\mathcal{F})$.

Instance A pps-formula ψ consisting of a product of m atomic $\mathcal{F}$-formulas over n free variables $\mathbf{x}$. (Thus, ψ has no bound variables.)

Output The value $\sum_{\mathbf{x}\in D^n} F_\psi(\mathbf{x})$ where F_ψ is the function defined by ψ.

Where convenient, we abuse notation by writing $\#\mathrm{CSP}(F)$ to mean $\#\mathrm{CSP}(\{F\})$ and by writing $\#\mathrm{CSP}(\mathcal{F},\mathcal{F}')$ to mean $\#\mathrm{CSP}(\mathcal{F}\cup\mathcal{F}')$.

As in [6] (and other works) we take the size of a $\#\mathrm{CSP}(\mathcal{F})$ instance to be $n+m$, where n is the number of (free) variables and m is the number of weighted constraints (atomic formulas). In unweighted versions of CSP and #CSP, we can just use n as the size of an instance, since the number of constraints can be bounded by a polynomial in the number of variables. However, in weighted cases, the multiplicity of constraints matters so we cannot bound m in terms of n. We typically denote an instance of $\#\mathrm{CSP}(\mathcal{F})$ by I and the output by $Z(I)$, which is often called the "partition function".

A counting problem, for our purposes, is any function from instances (encoded as words over a finite alphabet Σ) to $\mathbb{Q}_{\geq 0}$. A *randomised approximation scheme* for a counting problem $\#X$ is a randomised algorithm that takes an instance w and returns an approximation Y to $\#X(w)$. The approximation scheme has a parameter $\varepsilon \in (0,1)$ which specifies the error tolerance. Since the algorithm is randomised, the output Y is a random variable depending on the "coin tosses" made by the algorithm. We require that, for every instance w and every $\varepsilon \in (0,1)$,

$$\Pr\left[e^{-\varepsilon}\#X(w) \leq Y \leq e^{\varepsilon}\#X(w)\right] \geq 3/4\,. \tag{3}$$

The randomised approximation scheme is said to be a *fully polynomial randomised approximation scheme*, or *FPRAS*, if it runs in time bounded by a polynomial in $|w|$ (the length of the word w) and ε^{-1}. See Mitzenmacher and Upfal [24, Definition 10.2].

Suppose that $\#X$ and $\#Y$ are two counting problems. An "approximation-preserving reduction" (AP-reduction) [16] from $\#X$ to $\#Y$ gives a way to turn an FPRAS for $\#Y$ into an FPRAS for $\#X$. Specifically, an *AP-reduction from $\#X$ to $\#Y$* is a randomised algorithm $\mathcal{A}$ for computing $\#X$ using an oracle for $\#Y$. The algorithm $\mathcal{A}$ takes as input a pair $(w,\varepsilon) \in \Sigma^* \times (0,1)$, and satisfies the following three conditions: (i) every oracle call made by $\mathcal{A}$ is of the form (v,δ), where $v \in \Sigma^*$ is an instance of $\#Y$, and $0 < \delta < 1$ is an error bound satisfying $\delta^{-1} \leq \mathrm{poly}(|w|, \varepsilon^{-1})$; (ii) the algorithm $\mathcal{A}$ meets the specification for being a randomised approximation scheme for $\#X$ (as described above) whenever the oracle meets the specification for being a randomised approximation scheme for $\#Y$; and (iii) the run-time of $\mathcal{A}$ is polynomial in $|w|$ and ε^{-1}. If an AP-reduction from $\#X$ to $\#Y$ exists we write $\#X \leq_{\mathrm{AP}} \#Y$. Note that, subsequent to [16], the notation $\leq_{\mathrm{AP}}$ has been used to denote a different type of approximation-preserving reduction which applies to optimisation problems. In this paper, our emphasis is on counting problems so we hope this will not cause confusion.

The notion of pps-definability described earlier is closely related to AP-reductions. In particular, [6, Lemma 10.1] shows that $G \in \langle\mathcal{F}\rangle_{\#}$ implies that $\#\mathrm{CSP}(\mathcal{F}, G) \leq_{\mathrm{AP}} \#\mathrm{CSP}(\mathcal{F})$. We will use this fact without comment.

As mentioned above, #BIS is the problem of counting the independent sets in a bipartite graph and #SAT is the problem of counting the solutions to a Boolean formula in conjunctive normal form. We say that a counting problem $\#X$ is $\#Y$-easy if $\#X \leq_{\mathrm{AP}} \#Y$ and that it is $\#Y$-hard if $\#Y \leq_{\mathrm{AP}} \#X$. A problem $\#X$ is LSM-*easy* if there is a finite, weighted constraint language $\mathcal{F} \subset \mathsf{LSM}$ such that $\#X \leq_{\mathrm{AP}} \#\mathrm{CSP}(\mathcal{F})$.

We now state our main theorem. Note that we have only defined the problem $\#\mathrm{CSP}(\mathcal{F})$ for finite languages whereas conservative languages are, by definition, infinite.

Theorem 6. *Let $\mathcal{F}$ be a conservative weighted constraint language taking values in $\mathbb{Q}_{\geq 0}$.*

1. *If $\mathcal{F}$ is weakly log-modular then $\#\mathrm{CSP}(\mathcal{G})$ is in* FP *for every finite $\mathcal{G} \subset \mathcal{F}$.*

2. *If $\mathcal{F}$ is weakly log-supermodular but not weakly log-modular, then $\#\mathrm{CSP}(\mathcal{G})$ is* LSM*-easy for every finite $\mathcal{G} \subset \mathcal{F}$ and* #BIS*-hard for some such $\mathcal{G}$.*

3. *If $\mathcal{F}$ is weakly log-supermodular but not weakly log-modular and consists of functions of arity at most two, then* #CSP($\mathcal{G}$) *is* #BIS-*easy for every finite $\mathcal{G} \subset \mathcal{F}$ and* #BIS-*equivalent for some such $\mathcal{G}$.*

4. *If $\mathcal{F}$ is not weakly log-supermodular, then* #CSP($\mathcal{G}$) *is* #SAT-*easy for every finite $\mathcal{G} \subset \mathcal{F}$ and* #SAT-*equivalent for some such $\mathcal{G}$.*

In particular, among conservative #CSPs, there are no new complexity classes below #BIS or above LSM; furthermore there is a trichotomy for conservative weighted constraint languages with no functions of arity greater than two.

The #BIS-hardness and #SAT-equivalence are proved in Section 2, where they are restated as Theorem 10. The membership in FP is established as Theorem 15 at the end of Section 3. LSM-easiness and #BIS-easiness are established by Theorem 47 at the end of Section 6. Algorithmic aspects are discussed in Section 7.

2 Hardness results

Our hardness results use the following result from [6].

Lemma 7. *[6, Theorem 10.2] Let $\mathcal{F}$ be a finite, weighted constraint language with $D = \{0, 1\}$.*

- *If $\mathcal{F} \subset \langle \mathrm{NEQ}, \mathcal{U}_{\{0,1\}} \rangle_\#$ then, for any finite $S \subset \mathcal{U}_{\{0,1\}}$,* #CSP($\mathcal{F} \cup S$) *has an FPRAS.*
- *If $\mathcal{F} \not\subset \langle \mathrm{NEQ}, \mathcal{U}_{\{0,1\}} \rangle_\#$, then there is a finite $S \subset \mathcal{U}_{\{0,1\}}$ such that* #CSP($\mathcal{F} \cup S$) *is* #BIS-*hard.*
- *If $\mathcal{F} \not\subset \langle \mathrm{NEQ}, \mathcal{U}_{\{0,1\}} \rangle_\#$ and $\mathcal{F} \not\subset$ LSM, then there is a finite $S \subset \mathcal{U}_{\{0,1\}}$ such that* #CSP($\mathcal{F} \cup S$) *is* #SAT-*hard.*

Remark 8. In the statement of [6, Theorem 10.2], the set $\mathcal{U}_{\{0,1\}}$ is replaced with $\mathcal{B}_1^p$, the set of all unary functions from $\{0, 1\}$ to the set of non-negative efficiently computable reals. In this paper, we restrict to rationals for simplicity. Even though the statement of [6, Theorem 10.2] does not imply Lemma 7, the proof of [6, Theorem 10.2] does establish the lemma. No functions in $\mathcal{B}_1^p$ with irrational weights are used explicitly in the proof — unary functions that are used (for example, in the proof of [6, Lemma 7.1]) are constructed by multiplying and dividing other values in the codomains of functions in $\mathcal{F}$.

In fact we will only use the following special case of Lemma 7.

Lemma 9. *[6, Theorem 10.2] Let F be a function in* $\mathrm{Func}_2(\{0, 1\}, \mathbb{Q}_{\geq 0})$.

- *If $F \notin \langle \mathrm{NEQ}, \mathcal{U}_{\{0,1\}} \rangle_\#$ then $\{F\} \cup \mathcal{U}_{\{0,1\}}$ is* #BIS-*hard.*
- *If $F \notin \langle \mathrm{NEQ}, \mathcal{U}_{\{0,1\}} \rangle_\# \cup$ LSM then $\{F\} \cup \mathcal{U}_{\{0,1\}}$ is* #SAT-*hard.*

Our hardness results now follow from Lemma 9.

Theorem 10. *Let $\mathcal{F}$ be a conservative weighted constraint language taking values in $\mathbb{Q}_{\geq 0}$.*

- *If $\mathcal{F}$ is not weakly log-modular, there is a finite $\mathcal{G} \subset \mathcal{F}$ such that* #CSP($\mathcal{G}$) *is* #BIS-*hard.*
- *If $\mathcal{F}$ is not weakly log-supermodular, there is a finite $\mathcal{G} \subset \mathcal{F}$ such that* #CSP($\mathcal{G}$) *is* #SAT-*hard.*

- *For all finite* $\mathcal{G} \subset \mathcal{F}$*,* #CSP($\mathcal{G}$) *is* #SAT*-easy.*

Proof. First, we establish the hardness results.

Suppose that $\mathcal{F}$ is not weakly log-modular. Let $H \in \langle \mathcal{F} \rangle_\#$ be a function violating (1) and let a and b be the relevant domain elements, which must be distinct. Let $\varphi\colon \{0,1\} \to D$ be a unary function with $\varphi(0) = a$ and $\varphi(1) = b$. Define $H_\varphi\colon \{0,1\}^2 \to \mathbb{Q}_{\geq 0}$ by $H_\varphi(x,y) = H(\varphi(x), \varphi(y))$. The following three equations must all fail to hold:

$$\begin{gathered} H_\varphi(0,0)H_\varphi(1,1) = H_\varphi(0,1)H_\varphi(1,0) \\ H_\varphi(0,0) = H_\varphi(1,1) = 0 \\ H_\varphi(0,1) = H_\varphi(1,0) = 0. \end{gathered}$$

By [6, Remark 7.3], every binary function in $\langle \mathrm{NEQ}, \mathcal{U}_{\{0,1\}} \rangle_\#$ has one of three forms: $U_1(x)U_2(y)$, $U(x)\mathrm{EQ}(x,y)$ or $U(x)\mathrm{NEQ}(x,y)$. Therefore, $H_\varphi \notin \langle \mathrm{NEQ}, \mathcal{U}_{\{0,1\}} \rangle_\#$. By Lemma 9 there is a finite set $S \subset \mathcal{U}_{\{0,1\}}$ such that #BIS $\leq_{\mathrm{AP}}$ #CSP(H_φ, S).

For each $U \in \mathcal{U}_{\{0,1\}}$, define $U_{\varphi^{-1}} \in \mathcal{U}_D$ by

$$U_{\varphi^{-1}}(x) = \begin{cases} U(0) & \text{if } x = a, \\ U(1) & \text{if } x = b, \\ 0 & \text{otherwise.} \end{cases}$$

Let $E(0) = E(1) = 1$. Let $S' = \{U_{\varphi^{-1}} \mid U \in S \cup \{E\}\}$. Note that $\{H\} \cup S' \subset \langle F, \mathcal{U}_D \rangle_\#$ is finite.

We describe a reduction from #CSP(H_φ, S) to #CSP(H, S'). Given an instance I of #CSP(H_φ, S), replace each use of H_φ by H, and each use of $U \in S$ by $U_{\varphi^{-1}} \in S'$, and introduce an atomic formula $E_{\varphi^{-1}}(v)$ for each variable v, to obtain a new instance I' of #CSP(H, S') with $Z(I) = Z(I')$. Thus #CSP(H, S') is #BIS-hard.

A similar argument shows that $\mathcal{F}$ is #SAT-hard if it is not weakly log-supermodular. In this case, we start with a function $H \in \langle \mathcal{F} \rangle_\#$ violating (2) on the elements $a, b \in D$. Defining φ and H_φ as above, we find that $H_\varphi \notin \mathsf{LSM}$. Since H also violates (1) on a, b, the argument above establishes $H_\varphi \notin \langle \mathrm{NEQ}, \mathcal{U}_{\{0,1\}} \rangle_\#$. By Lemma 9 there is a finite set $S \subset \mathcal{U}_{\{0,1\}}$ such that #SAT $\leq_{\mathrm{AP}}$ #CSP(H_φ, S). We then proceed as before.

#SAT-easiness follows from the construction in Section 3 of [16], which shows that any problem in #P is #SAT-easy. The weighted counting CSPs that we deal with here are equivalent, by [4], to unweighted ones, which are in #P. $\square$

3 Balance and weak log-modularity

In this section we show that weak log-modularity implies tractability, by showing that every weakly log-modular weighted constraint language is balanced in the following sense.

We may associate a matrix $\mathbf{M}$ with an undirected bipartite graph $G_\mathbf{M}$ whose vertex partition consists of the set of rows R and columns C of $\mathbf{M}$. A pair $(r, c) \in R \times C$ is an edge of $G_\mathbf{M}$ if, and only if, $\mathbf{M}_{rc} \neq 0$. A *block* of $\mathbf{M}$ is a submatrix whose rows and columns form a connected component in $G_\mathbf{M}$. $\mathbf{M}$ has block-rank 1 if all its blocks have rank 1.

We say that a weighted constraint language $\mathcal{F}$ is *balanced* [8] if, for every function $F(x_1, \dots, x_n) \in \langle \mathcal{F} \rangle_\#$ with arity $n \geq 2$, and every k with $0 < k < n$, the $|D|^k \times |D|^{n-k}$

matrix $F((x_1,\dots,x_k),(x_{k+1},\dots,x_n))$ has block-rank 1. (This notion reduces to Dyer and Richerby's notion of "strong balance" [19] in the unweighted case.)

A function $F\colon \{0,1\}^n \to \mathbb{R}$ is *strictly positive* if its range is contained in $\mathbb{R}_{>0}$. F has *rank* 1 if it has the form $F(x_1,\dots,x_k) = U_1(x_1)\cdots U_k(x_k)$. Given a non-singular two-by-two matrix $T \in \mathbb{R}^{2\times 2}$ we define $T^{\otimes n}F\colon \{0,1\}^n \to \mathbb{R}$ by

$$(T^{\otimes n}F)(x_1,\dots,x_n) = \sum_{y_1,\dots,y_n \in \{0,1\}} \left(\prod_{i=1}^{n} T_{x_i y_i}\right) F(y_1,\dots,y_n)$$

The rows and columns of T are considered to be indexed by $\{0,1\}$. We associate with any function $F\colon \{0,1\}^2 \to \mathbb{R}$, the matrix $M_F \in \mathbb{R}^{2\times 2}$ defined by $(M_F)_{ij} = F(i,j)$.

Lemma 11. *Let $\mathbf{M} \in \mathbb{R}^{k\times k}$. Let $F\colon \{0,1\}^k \to \mathbb{R}$. Let $T \in \mathbb{R}^{2\times 2}_{\geq 0}$ be non-singular.*

1. *If $k = 2$ then $\mathbf{M}$ has block-rank 1 if and only if it has rank 1 or it has at most two non-zero entries. F has rank 1 if and only if $\det M_F = 0$.*

2. *$\mathbf{M}$ has block-rank 1 if and only if the matrix*

$$N_{M,\mathbf{u},\mathbf{u}',\mathbf{v},\mathbf{v}'} = \begin{pmatrix} \mathbf{M}(\mathbf{u},\mathbf{v}) & \mathbf{M}(\mathbf{u},\mathbf{v}') \\ \mathbf{M}(\mathbf{u}',\mathbf{v}) & \mathbf{M}(\mathbf{u}',\mathbf{v}') \end{pmatrix}$$

has block-rank 1 for every $\mathbf{u},\mathbf{u}',\mathbf{v},\mathbf{v}'$.

3. *(Topkis's theorem) If F is strictly positive and not of rank 1, there is a function $F'\colon \{0,1\}^2 \to \mathbb{R}$ of the following form that is not of rank 1.*

$$F'(x_i,x_j) = F(c_1,\dots,c_{i-1},x_i,c_{i+1},\dots,c_{j-1},x_j,c_{j+1},\dots,c_k).$$

Here $1 \leq i < j \leq k$, and each c_ℓ is a fixed element of $\{0,1\}$.

4. *F has rank 1 if and only if $T^{\otimes k}F$ has rank 1.*

Proof. 1. A 2×2 matrix that has block-rank 1 either has rank 1 or is diagonal or anti-diagonal so has two zeroes. Conversely, a matrix that has rank 1 has no submatrix whose rank exceeds 1, so has block-rank 1. A matrix with two or more zeroes has no 2×2 block so can only have blocks of rank 1.

For the second statement, if F has rank 1 then there are unary functions U_0 and U_1 so that $F(x,y) = U_0(x)U_1(y)$, which implies that $\det M_F = 0$. Going the other way, if F is identically 0 then it has rank 1. Otherwise, suppose $F(i,j) \neq 0$. Let $U_0(x) = F(x,j)$ and $U_1(y) = F(i,y)/F(i,j)$. If $\det M_F = 0$ then $F(x,y) = U_0(x)U_1(y)$, so F has rank 1.

2. [19, Lemma 38].

3. Say that a strictly positive function F is *log-modular* if $f = \log F$ is modular: that is, $F(\mathbf{x}\vee\mathbf{y})F(\mathbf{x}\wedge\mathbf{y}) = F(\mathbf{x})F(\mathbf{y})$ for all $\mathbf{x},\mathbf{y} \in \{0,1\}^k$. If $f(\mathbf{x})$ is modular, then can be expressed as a linear sum of the x_i's (see for example [2, Proposition 24]), so a strictly positive log-modular function is a product of unary functions, so it has rank 1. The result is then Topkis's theorem [28] in the form stated in [6, Lemma 5.1].

4. If F is of the form $U_1(x_1)\cdots U_n(x_n)$ then

$$(T^{\otimes n}F)(x_1,\ldots,x_n) = (T^{\otimes 1}U_1)(x_1)\cdots(T^{\otimes 1}U_n)(x_n)$$

The reverse implication follows from $(T^{-1})^{\otimes n}T^{\otimes n}F = F$, where T^{-1} is the matrix inverse of T.

□

A function $F\colon D^n \to \mathbb{Q}_{\geq 0}$ is *essentially pseudo-Boolean* if its support (the set of vectors $\mathbf{x}$ satisfying $F(\mathbf{x}) > 0$) is contained in a set $D_1 \times \cdots \times D_n$ with $|D_1|,\ldots,|D_n| \leq 2$. The *projection* of a relation $R \subseteq D^n$ onto indices $1 \leq i < j \leq n$ is the set of pairs $(a,b) \in D^2$ such that there exists $\mathbf{x} \in R$ with $x_i = a$ and $x_j = b$. A *generalised* NEQ is a relation of the form $\{(x_i,x_j),(y_i,y_j)\} \subset D^2$ for some $x_i \neq y_i$ and $x_j \neq y_j$.

Lemma 12. *Let $F\colon D^n \to \mathbb{Q}_{\geq 0}$ be an essentially pseudo-Boolean function which is not of rank* 1, *and assume that no binary projection of the support of F is a generalised* NEQ. *Then $\{F\} \cup \mathcal{U}_D$ is not weakly log-modular.*

Proof. Let the support of F be contained in $D_1 \times \cdots \times D_n$ where $|D_i| = 2$ for all i. Choose bijections $\rho_i\colon \{0,1\} \to D_i$ for each $1 \leq i \leq n$. Define $F_\rho\colon \{0,1\}^n \to \mathbb{Q}_{\geq 0}$ by

$$F_\rho(x_1,\ldots,x_n) = F(\rho_1(x_1),\ldots,\rho_n(x_n))$$

for all $x_1,\ldots,x_n \in \{0,1\}$. Let $T = \left(\begin{smallmatrix} 2 & 1 \\ 1 & 2 \end{smallmatrix}\right)$ and note that $T^{\otimes n}F_\rho$ is strictly positive. Since F is not of rank 1, F_ρ is not of rank 1, so by Lemma 11 part (4), $T^{\otimes n}F_\rho$ is not of rank 1. By Lemma 11 part (3), there is a function $B\colon \{0,1\}^2 \to \mathbb{Q}_{\geq 0}$ of the following form that is not of rank 1.

$$B(x_i,x_j) = (T^{\otimes n}F_\rho)(c_1,\ldots,c_{i-1},x_i,c_{i+1},\ldots,c_{j-1},x_j,c_{j+1},\ldots,c_n).$$

For all indices $k \in \{1,\ldots,n\} \setminus \{i,j\}$, define $U_k \in \mathcal{U}_D$ by $U_k(\rho_k(x_k)) = T_{c_k x_k}$ for all $x_k \in \{0,1\}$, and $U_k(z) = 0$ if $z \notin D_k$. Define $G,H\colon D^2 \to \mathbb{Q}_{\geq 0}$ and $G_{\rho_i,\rho_j}, H_{\rho_i,\rho_i}\colon \{0,1\}^2 \to \mathbb{Q}_{\geq 0}$ as follows. Note in these definitions that i and j are fixed, but ρ_i and ρ_j are used as subscripts in the name of some of the functions as a reminder of the bijections that are being applied to the inputs. Thus, in H_{ρ_i,ρ_i}, the bijection ρ_i is applied to both arguments, even though the function depends on both i and j.

$$\begin{aligned}
G(y_i,y_j) &= \sum\left(\prod_{k\neq i,j} U_k(y_k)\right) F(y_1,\ldots,y_n) && \text{for all } y_i,y_j \in D \\
G_{\rho_i,\rho_j}(x_i,x_j) &= \sum\left(\prod_{k\neq i,j} T_{c_k,x_k}\right) F_\rho(x_1,\ldots,x_n) && \text{for all } x_i,x_j \in \{0,1\} \\
H(y',y'') &= \sum_{y\in D} G(y',y)G(y'',y) && \text{for all } y',y'' \in D \\
H_{\rho_i,\rho_i}(x',x'') &= \sum_{x\in\{0,1\}} G_{\rho_i,\rho_j}(x',x)G_{\rho_i,\rho_j}(x'',x) && \text{for all } x',x'' \in \{0,1\}
\end{aligned}$$

where the first sum is over all $y_1,\ldots,y_{i-1},y_{i+1},\ldots,y_{j-1},y_{j+1},\ldots,y_n \in D$ and the second sum is over all $x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_{j-1},x_{j+1},\ldots,x_n \in \{0,1\}$.

Note that $M_{H_{\rho_i,\rho_i}} = M_{G_{\rho_i,\rho_j}} M^{\mathrm{t}}_{G_{\rho_i,\rho_j}} = T^{-1} M_B (T^{-1})^{\mathrm{t}} T^{-1} M_B^t (T^{-1})^{\mathrm{t}}$ where t denotes transpose. Taking determinants and applying Lemma 11 part (1) this implies that H_{ρ_i,ρ_i} is not of rank 1. Also, since T is strictly positive, the support of G_{ρ_i,ρ_j} is the binary projection of the support of F_ρ onto i and j which, by assumption, is not NEQ or EQ. Hence H_{ρ_i,ρ_i} is strictly positive but not of rank 1. Again using Lemma 11 part (1) we see that H is a witness that $\{F\} \cup \mathcal{U}_D$ is not weakly log-modular. □

Lemma 13. *Every conservative weakly log-modular weighted constraint language is balanced.*

Proof. Let $\mathcal{F}$ be a conservative weighted constraint language that is not balanced. We will show that $\mathcal{F}$ is not weakly log-modular.

By the definition of balance, there is a function $F \in \langle \mathcal{F} \rangle_\#$ of arity n and a partition $\mathbf{x} = (\mathbf{u}, \mathbf{v})$ of its n variables, such that the matrix $F(\mathbf{u}, \mathbf{v})$ is not of block-rank 1. By Lemma 11 part (2) there is a two-by-two submatrix $N = N_{F,\mathbf{u},\mathbf{u}',\mathbf{v},\mathbf{v}'}$ that is not of block-rank 1.

Construct an essentially pseudo-Boolean function G from F as follows. For all $1 \le i \le n$ let $U_i \in \langle \mathcal{U}_D \rangle_\# \subseteq \langle \mathcal{F} \rangle_\#$ be the indicator function of $D^{i-1} \times D_i \times D^{n-i}$, where $D_i = \{u_i, u_i'\}$ for all $1 \le i \le k$ and $D_i = \{v_i, v_i'\}$ for all $k < i \le n$. Let $G = F \prod_{i=1}^n U_i$. Then $N_{G,\mathbf{u},\mathbf{u}',\mathbf{v},\mathbf{v}'} = N$ is not of block-rank 1, and G is essentially pseudo-Boolean.

If the binary projection of the support of G onto two indices i, j is a generalised NEQ $\{(x_i, x_j), (y_i, y_j)\}$, construct the tuple $(G', \rho(\mathbf{u}), \rho(\mathbf{u}'), \rho(\mathbf{v}), \rho(\mathbf{v}'))$ from $(G, \mathbf{u}, \mathbf{u}', \mathbf{v}, \mathbf{v}')$ as follows. Let $\rho \colon D^n \to D^{n-1}$ be the projection operator sending $\mathbf{x}$ to $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$ and let $G'(\mathbf{x}) = \sum_{\rho(\mathbf{x}')=\mathbf{x}} G(\mathbf{x}')$ for all $\mathbf{x} \in D^{n-1}$. Note that, for all $\mathbf{x} \in D^n$, $G(\mathbf{x}) \ne G'(\rho(\mathbf{x}))$ implies that $G(\mathbf{x}) = 0$ because $G(\mathbf{x}) = 0$ unless $x_i \ne x_j$. Note that N has at least three non-zero entries by Lemma 11 part (1). So the corresponding three pairs out of $((\mathbf{u},\mathbf{v})_i, (\mathbf{u},\mathbf{v})_j)$, $((\mathbf{u},\mathbf{v}')_i, (\mathbf{u},\mathbf{v}')_j)$, $((\mathbf{u}',\mathbf{v})_i, (\mathbf{u}',\mathbf{v})_j)$, and $((\mathbf{u}',\mathbf{v}')_i, (\mathbf{u}',\mathbf{v}')_j)$ must each be either (x_i, x_j) or (y_i, y_j). But then the fourth pair must also be (x_i, x_j) or (y_i, y_j), which implies that $N_{G',\rho(\mathbf{u}),\rho(\mathbf{u}'),\rho(\mathbf{v}),\rho(\mathbf{v}')} = N$. Also, G' is essentially pseudo-Boolean, and G' is obtained by summing the i'th variable, so $G' \in \langle G \rangle_\#$.

Repeating this process if necessary, we obtain $(G', \mathbf{x}, \mathbf{x}', \mathbf{y}, \mathbf{y}')$ such that G' is an essentially pseudo-Boolean function in $\langle \mathcal{F}, \mathcal{U}_D \rangle_\# = \langle \mathcal{F} \rangle_\#$ and none of the binary projections of the support of G' is a generalised NEQ, and $N_{G',\mathbf{x},\mathbf{x}',\mathbf{y},\mathbf{y}'}$ is not of block-rank 1. So, in particular, G' is not of rank 1. By Lemma 12, $\{G'\} \cup \mathcal{U}_D$ is not weakly log-modular, so $\langle \mathcal{F} \rangle_\#$ is not weakly log-modular. □

We now return to Theorem 6 and prove the tractable case. The proof relies on an important theorem of Cai, Chen and Lu about the complexity of exact evaluation.

Lemma 14. *[8] Let $\mathcal{F}$ be a finite, weighted constraint language taking non-negative algebraic real values. If $\mathcal{F}$ is balanced, then* #CSP($\mathcal{F}$) *is in* FP*, and otherwise* #CSP($\mathcal{F}$) *is* #P*-hard.*

Theorem 15. *Let $\mathcal{F}$ be a conservative weighted constraint language taking values in $\mathbb{Q}_{\ge 0}$. If $\mathcal{F}$ is weakly log-modular then, for any finite $\mathcal{G} \subset \mathcal{F}$,* #CSP($\mathcal{G}$) $\in$ FP.

Proof. By Lemma 13, $\mathcal{F}$ is balanced. Hence, every finite $\mathcal{G} \subset \mathcal{F}$ is balanced, which implies that #CSP($\mathcal{G}$) is in FP by Lemma 14. □

4 Valued clones, valued CSPs and relational clones

To define valued clones, we use the same set-up as Section 1.2 except that summation is replaced by minimisation and product is replaced by sum. Let D be a finite domain with $|D| \geq 2$ and let R be a codomain with $\{0, \infty\} \subseteq R$, where ∞ obeys the following rules for all $x \in R$: $x + \infty = \infty$, $x \leq \infty$ and $\min\{x, \infty\} = x$. Let Φ be a subset of $\mathrm{Func}(D, R)$ and let $V = \{v_1, \ldots, v_n\}$ be a set of variables. For each atomic formula $\varphi = G(v_{i_1}, \ldots, v_{i_a})$ we use the notation f_φ to denote the function represented by φ, so $f_\varphi(\mathbf{x}) = G(x_{i_1}, \ldots, x_{i_a})$.

A psm-formula ("primitive sum minimisation formula") is a minimisation of a sum of atomic formulas. A psm-formula ψ over Φ in variables $V' = \{v_1, \ldots, v_{n+k}\}$ has the form

$$\psi = \min_{v_{n+1}, \ldots, v_{n+k}} \sum_{j=1}^{m} \varphi_j, \tag{4}$$

where φ_j are all atomic formulas over Φ in the variables V'. The formula ψ specifies a function $f_\psi \colon D^n \to R$ in the following way:

$$f_\psi(\mathbf{x}) = \min_{\mathbf{y} \in D^k} \sum_{j=1}^{m} f_{\varphi_j}(\mathbf{x}, \mathbf{y}), \tag{5}$$

where $\mathbf{x}$ and $\mathbf{y}$ are assignments $\mathbf{x} \colon \{v_1, \ldots, v_n\} \to D$ and $\mathbf{y} \colon \{v_{n+1}, \ldots, v_{n+k}\} \to D$.

The *valued clone* $\langle \Phi \rangle_V$ *generated by* Φ is the set of all functions that can be represented by a psm-formula over $\Phi \cup \{\mathrm{eq}\}$, where eq is the binary equality function on D given by $\mathrm{eq}(x, x) = 0$ and $\mathrm{eq}(x, y) = \infty$ for $x \neq y$.

We next introduce valued constraint satisfaction problems (VCSPs), which are optimisation problems. In the work of Kolmogorov and Živný [22], the codomain is $R = \mathbb{Q}_{\geq 0} \cup \{\infty\}$. For reasons which will be clear below, it is useful for us to extend the codomain to include irrational numbers. This will not cause problems because, with the exception of Theorem 37 we use only formal calculations from their papers, not complexity results. For Theorem 37, we avoid irrational numbers and, in fact, restrict to cost functions taking values in $\{0, \infty\} \subset R$. Furthermore, all the real numbers we use are either rationals or their logarithms so are efficiently computable.

Let $\overline{\mathbb{R}}_{\geq 0} = \mathbb{R}_{\geq 0} \cup \{\infty\}$ be the set of non-negative real numbers together with ∞.

Definition 16. A *cost function* is a function $D^k \to \overline{\mathbb{R}}_{\geq 0}$. A *valued constraint language* is a set of cost functions $\Phi \subseteq \mathrm{Func}(D, \overline{\mathbb{R}}_{\geq 0})$.

Given a valued constraint language Φ, $\mathrm{VCSP}(\Phi)$ is the problem of taking an instance ψ, a psm-formula consisting of a sum of m atomic Φ-formulas over n free variables $\mathbf{x}$ and computing the value

$$\mathrm{minCost}(\psi) = \min_{\mathbf{x} \in D^n} f_\psi(\mathbf{x}),$$

where f_ψ is the function defined by ψ.

We typically use the notation of Kolmogorov and Živný. An instance is usually denoted by the letter I. In this case, we use f_I to denote the function specified by the psm-formula corresponding to instance I, so the value of the instance is denoted by $\mathrm{minCost}(I)$. The psm-formula corresponding to I is a sum of atomic formulas (since all of the variables are free variables). We refer to each of these atomic formulas as a *valued constraint* and we represent these by the multiset T of all valued constraints in the instance I. For each valued constraint $t \in T$ we use k_t to denote its arity, f_t to denote the function represented by the corresponding atomic formula, and σ_t to denote its scope, which is given as a tuple $(i(t,1), \ldots, i(t,k_t)) \in \{1, \ldots, n\}^{k_t}$ containing the indices of the variables in the scope. Thus,

$$f_I(\mathbf{x}) = \sum_{t \in T} f_t(x_{i(t,1)}, \ldots, x_{i(t,k_t)}) \,. \tag{6}$$

For convenience, we use $\mathbf{x}[\sigma_t]$ as an abbreviation for the tuple $(x_{i(t,1)}, \ldots, x_{i(t,k_t)})$. In this abbreviated notation, the function defined by instance I may be written $f_I(\mathbf{x}) = \sum_{t \in T} f_t(\mathbf{x}[\sigma_t])$.

Now, let $[0,1]_{\mathbb{Q}} = [0,1] \cap \mathbb{Q}$. For reasons which will be clear below, it will be useful to work with weight functions in $\mathrm{Func}(D, [0,1]_{\mathbb{Q}})$. For such a weight function F, let the cost function $\ell(F) \in \mathrm{Func}(D, \overline{\mathbb{R}}_{\geq 0})$ be the function defined by

$$(\ell(F))(\mathbf{x}) = \begin{cases} -\ln F(\mathbf{x}) & \text{if } F(\mathbf{x}) > 0 \\ \infty & \text{if } F(\mathbf{x}) = 0. \end{cases}$$

For example, $\ell(\mathrm{EQ}) = \mathrm{eq}$, where EQ and eq are the functions defined earlier. (Often, as here, we use a lower-case name like eq and an upper case name like EQ to indicate such a relationship.) For a weighted constraint language $\mathcal{F} \subseteq \mathrm{Func}(D, [0,1]_{\mathbb{Q}})$, let $\ell(\mathcal{F})$ be the valued constraint language defined by $\ell(\mathcal{F}) = \{\ell(F) \mid F \in \mathcal{F}\}$.

There is a natural bijection between instances of $\#\mathrm{CSP}(\mathcal{F})$ and $\mathrm{VCSP}(\ell(\mathcal{F}))$, obtained by replacing each function F_t in the former by the function $f_t = \ell(F_t)$ in the latter, keeping the scopes unchanged. Note that $f_I(\mathbf{x}) = -\ln F_I(\mathbf{x})$, for any assignment $\mathbf{x}$, with the convention $-\ln 0 = \infty$.

Definition 17. A valued constraint language is *conservative* if it contains all arity-1 cost functions $D \to \overline{\mathbb{R}}_{\geq 0}$.

The mapping $F \mapsto \ell(F)$ from $\mathrm{Func}(D, [0,1]_{\mathbb{Q}})$ to $\mathrm{Func}(D, \overline{\mathbb{R}}_{\geq 0})$ is not surjective because there are real numbers that are not the logarithm of any rational. For the same reason, the valued constraint language $\ell(\mathcal{F})$ is not conservative (for any weighted constraint language $\mathcal{F}$). Finally, note that we have only defined $\ell(F)$ for $F \in \mathrm{Func}(D, [0,1]_{\mathbb{Q}})$. The obvious extension to $F \in \mathrm{Func}(D, \mathbb{Q}_{\geq 0})$ would produce negative-valued cost functions and we wish to avoid this since Kolmogorov and Živný [22] do not allow it.

Definition 18. A cost function is *crisp* [14] if $f(\mathbf{x}) \in \{0, \infty\}$ for all $\mathbf{x}$.

Definition 19. For any cost function f, let $\mathrm{Feas}(f)$ be the relation defined by $\mathrm{Feas}(f) = \{\mathbf{x} \mid f(\mathbf{x}) < \infty\}$.

Thus, any cost function f can be associated with its underlying relation. Similarly, we can represent any relation by a crisp cost function f for which $f(\mathbf{x}) = 0$ if and only if $\mathbf{x}$ is in the relation. A *crisp constraint language* is a set of relations, which we always represent as crisp cost functions, not as functions with codomain $\{0, 1\}$. For a valued constraint language Φ, the crisp constraint language $\mathrm{Feas}(\Phi)$ is given by $\mathrm{Feas}(\Phi) = \{\mathrm{Feas}(f) \mid f \in \Phi\}$.

Definition 20. A crisp constraint language is *conservative* if it includes all arity-1 relations.

A *relational clone* is simply a crisp constraint language $\mathrm{Feas}(\langle\Phi\rangle_V)$ for a valued constraint language Φ.

Lemma 21. *Suppose* $\Phi \subseteq \mathrm{Func}(D, \overline{\mathbb{R}}_{\geq 0})$. *Then* $\langle \mathrm{Feas}(\Phi)\rangle_V = \mathrm{Feas}(\langle\Phi\rangle_V)$.

Proof. The mapping $\rho\colon \overline{\mathbb{R}}_{\geq 0} \to \{0, \infty\}$ defined by $\rho(\infty) = \infty$ and $\rho(x) = 0$, for all $x < \infty$, is a homomorphism of semirings, from $(\overline{\mathbb{R}}_{\geq 0}, \min, +)$ to $(\{0, \infty\}, \min, +)$. □

5 STP/MJN multimorphisms and weak log-supermodularity

In [22, Corollary 3.5], Kolmogorov and Živný give a tractability criterion for conservative VCSPs. In particular, they show that the VCSP associated with a conservative valued constraint language Φ is tractable iff Φ has an STP/MJN multimorphism.

We define STP/MJN multimorphisms below. In this section, we show (see Theorem 34 below) that, if a weighted constraint language $\mathcal{F} \in \mathrm{Func}(D, [0,1]_{\mathbb{Q}})$ is weakly log-supermodular, then the corresponding valued constraint language $\ell(\mathcal{F})$ has an STP/MJN multimorphism. In Section 6, this will enable us to use such a multimorphism (via the work of Kolmogorov and Živný [22] and Cohen, Cooper and Jeavons [12]) to prove #BIS-easiness and LSM-easiness of the weighted counting CSP.

Our proof of Theorem 34 relies on work by Kolmogorov and Živný [22] and Takhanov [26]. We start with some general definitions. Most of these are from [22], but some care is required since some of the definitions in [22] differ from those in [12].

Definition 22. A *k-ary operation on D* is a function from D^k to D. An *operation on D* is a k-ary operation, for some k.

We drop the "on D" when the domain D is clear from the context.

Definition 23. A k-tuple $\langle\rho_1, \dots, \rho_k\rangle$ of k-ary operations $\rho_1, \dots, \rho_k$ is *conservative* if, for every tuple $\mathbf{x} = (x_1, \dots, x_k) \in D^k$, the multisets $\{\{x_1, \dots, x_k\}\}$ and $\{\{\rho_1(\mathbf{x}), \dots, \rho_k(\mathbf{x})\}\}$ are equal.

Note that we have now defined conservative operations and conservative constraint languages (weighted, valued and crisp). There are connections between these notions of "conservative" but we do not need these here.

Definition 24. $\langle\rho_1, \dots, \rho_k\rangle$ is a *multimorphism* of an arity-r cost function f if, for all $\mathbf{x}^1, \dots, \mathbf{x}^k \in D^r$, we have:

$$\sum_{i=1}^{k} f(\rho_i(x_1^1, \dots, x_1^k), \dots, \rho_i(x_r^1, \dots, x_r^k)) \leq \sum_{i=1}^{k} f(\mathbf{x}^i).$$

Definition 25. $\langle\rho_1, \dots, \rho_k\rangle$ is a multimorphism of a valued constraint language Φ if it is a multimorphism of every $f \in \Phi$.

These definitions imply the following.

Observation 26. If $\langle\rho_1, \dots, \rho_k\rangle$ is conservative, then it is a multimorphism of every unary cost function f.

Definition 27. Suppose $M \subseteq D^2$. A pair $\langle\sqcap, \sqcup\rangle$ of binary operations is a *symmetric tournament pair* (STP) on M if it is conservative and both operations are commutative on M. We say that it is an STP if it is an STP on D^2.

Definition 28. Suppose $M \subseteq D^2$. A triple $\langle \texttt{Mj1}, \texttt{Mj2}, \texttt{Mn3} \rangle$ of ternary operations is an *MJN on* M if it is conservative and, for all triples $(a, b, c) \in D^3$ with $\{\{a, b, c\}\} = \{\{x, x, y\}\}$ where x and y are distinct and $(x, y) \in M$, we have $\texttt{Mj1}(a, b, c) = \texttt{Mj2}(a, b, c) = x$ and $\texttt{Mn3}(a, b, c) = y$.

The reason that Definition 28 only deals with the case in which x and y are distinct is that any conservative triple $\langle \texttt{Mj1}, \texttt{Mj2}, \texttt{Mn3} \rangle$ satisfies $\texttt{Mj1}(x, x, x) = \texttt{Mj2}(x, x, x) = \texttt{Mn3}(x, x, x) = x$.

Definition 29. An *STP/MJN multimorphism* of a valued constraint language Φ consists of a pair of operations $\langle \sqcap, \sqcup \rangle$ and a triple of operations $\langle \texttt{Mj1}, \texttt{Mj2}, \texttt{Mn3} \rangle$, both of which are multimorphisms of Φ, for which, for some symmetric subset M of D^2, $\langle \sqcap, \sqcup \rangle$ is an STP on M and $\langle \texttt{Mj1}, \texttt{Mj2}, \texttt{Mn3} \rangle$ is an MJN on $\{(a, b) \in D^2 \setminus M \mid a \neq b\}$.

Definition 30. $\Phi \subseteq \mathrm{Func}(D, \overline{\mathbb{R}}_{\geq 0})$ is *weakly submodular* if, for all binary functions $f \in \langle \Phi \rangle_V$ and elements $a, b \in D$,

$$f(a, a) + f(b, b) \leq f(a, b) + f(b, a) \quad \text{or} \quad f(a, a) = f(b, b) = \infty. \tag{7}$$

Note that the definition of weak submodularity for cost functions is a restatement of Kolmogorov and Živný's "Assumption 3". It is not trivial that weak log-supermodularity for $\mathcal{F}$ is related to weak submodularity for $\ell(\mathcal{F})$. Expressibility for VCSP is different from expressibility for #CSP and, specifically, we cannot expect $\langle \ell(\mathcal{F}) \rangle_V = \ell(\langle \mathcal{F} \rangle_\#)$ to hold in general. However, the following is suitable for our purposes.

Lemma 31. *Suppose* $\mathcal{F} \subseteq \mathrm{Func}(D, [0, 1]_\mathbb{Q})$ *and let* $\Phi = \ell(\mathcal{F})$. *If* $\mathcal{F}$ *is weakly log-supermodular then* Φ *is weakly submodular.*

Proof. We prove the contrapositive. Suppose $f \in \langle \Phi \rangle_V$ is a binary function that witnesses the fact that Φ is not weakly submodular according to Definition 30, specifically,

$$f(a, a) + f(b, b) > f(a, b) + f(b, a) \quad \text{and} \quad \min\{f(a, a), f(b, b)\} < \infty.$$

Since $f \in \langle \Phi \rangle_V$, we may express f in the form

$$f(\mathbf{x}) = \min_{\mathbf{y}} g(\mathbf{x}, \mathbf{y}) = \min_{\mathbf{y}} \sum_{i=1}^{m} g_i(\mathbf{x}, \mathbf{y}),$$

where the $g_i \in \Phi$ are atomic. For $k \in \mathbb{N}$, define

$$F^{(k)}(\mathbf{x}) = \sum_{\mathbf{y}} \prod_{i=1}^{m} G_i(\mathbf{x}, \mathbf{y})^k,$$

where each G_i is such that $g_i = \ell(G_i)$. Note that $F^{(k)} \in \langle \mathcal{F} \rangle_\#$, and

$$F^{(k)}(\mathbf{x})^{1/k} \to \max_{\mathbf{y}} \prod_{i=1}^{m} G_i(\mathbf{x}, \mathbf{y}), \quad \text{as } k \to \infty.$$

Now

$$\begin{aligned} \max_{\mathbf{y}} \prod_{i=1}^{m} G_i(\mathbf{x}, \mathbf{y}) &= \max_{\mathbf{y}} \exp\Big(-\sum_{i=1}^{m} g_i(\mathbf{x}, \mathbf{y}) \Big) \\ &= \exp\Big(-\min_{\mathbf{y}} \sum_{i=1}^{m} g_i(\mathbf{x}, \mathbf{y}) \Big) \\ &= \exp(-f(\mathbf{x})) \end{aligned}$$

and

$$\exp(-f(a,a))\exp(-f(b,b)) < \exp(-f(a,b))\exp(-f(b,a)).$$

Thus $F(a,a)F(b,b) < F(a,b)F(b,a)$ where $F = F^{(k)}$ for some sufficiently large k. Also, $\min\{f(a,a), f(b,b)\} < \infty$ implies that $\max\{F(a,a), F(b,b)\} > 0$. These properties of F imply that $\mathcal{F}$ is not weakly log-supermodular, according to Definition 4. $\square$

Let Γ be a crisp constraint language. A *majority polymorphism* of Γ is a ternary operation ρ such that $\rho(a,a,b) = \rho(a,b,a) = \rho(b,a,a) = a$ for all $a,b \in D$ and for all arity-k relations $R \in \Gamma$ we have

$$\mathbf{x}, \mathbf{y}, \mathbf{z} \in R \implies (\rho(x_1,y_1,z_1), \ldots, \rho(x_k,y_k,z_k)) \in R.$$

Let $N(a,b,c,d)$ be the relation $\{(a,c),(b,c),(a,d)\}$. The existence of such a relation in $\langle\Gamma\rangle_V$ indicates that Γ is not "strongly balanced" in the terminology of [19]. Note that, on the Boolean domain, $N(0,1,0,1)$ is the "NAND" relation.

Theorem 32. *(Takhanov) Let Γ be a conservative relational clone with domain D. At least one of the following holds.*

- *There are distinct $a,b \in D$ such that $N(a,b,a,b) \in \Gamma$.*
- *There are distinct $a,b \in D$ such that $\{(a,a,a),(a,b,b),(b,a,b),(b,b,a)\} \in \Gamma$.*
- *For some $k \geq 1$, there are $a_0,\ldots,a_{2k}, b_0,\ldots,b_{2k} \in D$ such that, for each $0 \leq i \leq 2k$, $a_i \neq b_i$ and, for each $0 \leq i \leq 2k-1$,*

$$N(a_i,b_i,a_{i+1},b_{i+1}) \in \Gamma \text{ and } N(a_{2k},b_{2k},a_0,b_0) \in \Gamma.$$

- *Γ has a majority polymorphism.*

Proof. This formulation is essentially [26, Theorem 9.1] except for the last bullet point. As stated in the proof of that theorem, the first two conditions both fail if and only if the "necessary local conditions" [26, Definition 3.5] hold. Unfortunately for us, Takhanov uses the term "functional clone" differently to how we use it, so the reader will need to take this into account to understand the local conditions. However, we do not need the detail, here. Takhanov's proof of the NP-hard case of his Theorem 3.7 (at the end of his Section 4) shows the following: Given the necessary local conditions, the third condition fails only if a certain graph T_F is bipartite. If T_F is bipartite then [26, Theorem 5.5] establishes a majority polymorphism. $\square$

Lemma 33. *If $\Phi \subseteq \mathrm{Func}(D, \overline{\mathbb{R}}_{\geq 0})$ is conservative and weakly submodular, $\Gamma = \langle \mathrm{Feas}(\Phi)\rangle_V$ has a majority polymorphism.*

Proof. Since Φ is conservative (Definition 17), so is Γ (Definition 20). We will now show that the first three bullets of Theorem 32 contradict the premise of the lemma, so the fourth must hold.

The first bullet-point is easily ruled out. Suppose the given relation is in Γ. By Lemma 21, there is a binary function $f \in \langle\Phi\rangle_V$ such that $\mathrm{Feas}(f) = N(a,b,a,b)$. This function has $f(b,b) = \infty$ and $f(a,a), f(a,b), f(b,a) < \infty$, and hence violates (7).

For the second bullet-point, by Lemma 21 we have an arity-3 function $g \in \langle\Phi\rangle_V$ which is finite precisely on $\{(a,a,a),(a,b,b),(b,a,b),(b,b,a)\}$. Now let M be a sufficiently large constant and let u be the unary function defined by

$$u(z) = \begin{cases} M & \text{if } z = a, \\ 0 & \text{if } z = b, \\ \infty & \text{otherwise.} \end{cases}$$

Let

$$f(x,y) = \min_{z \in D}\{g(x,y,z) + u(z)\}.$$

Then $f(a,a) = M + g(a,a,a)$, $f(b,b) = M + g(b,b,a)$, $f(a,b) = g(a,b,b)$ and $f(b,a) = g(b,a,b)$. Clearly, f violates (7) for sufficiently large M.

Finally, let us consider the third bullet-point. By Lemma 21 we have binary functions $g_0, g_1, \dots, g_{2k} \in \langle\Phi\rangle_V$ where the underlying relation of g_i is $N(a_i, b_i, a_{i+1}, b_{i+1})$ for $0 \le i < 2k$ and the underlying relation of g_{2k} is $N(a_{2k}, b_{2k}, a_0, b_0)$. Define

$$\begin{aligned} f(x,y) = \min\big\{ & g_0(x,z_0) + u_0(z_0) + g_1(z_0,z_1) + u_1(z_1) + \\ & \cdots + u_{2k-1}(z_{2k-1}) + g_{2k}(z_{2k-1},y) \mid (z_0,\dots,z_{2k-1}) \in D^{2k}\big\}, \end{aligned}$$

where $u_i(a_i) = M$, $u_i(b_i) = 0$, and $u_i(z) = \infty$ if $z \notin \{a_i, b_i\}$. Note that $f(a_0,a_0) \ge kM$, $f(b_0,b_0) \ge (k+1)M$ and $f(a_0,b_0), f(b_0,a_0) \le kM + (2k+1)m$, where m is the largest finite value taken by any of $g_0, \dots, g_{2k}$. So f violates (7) for sufficiently large M.

So we are left with the remaining possibility that Γ has a majority polymorphism. $\square$

We can now prove the main result of this section.

Theorem 34. *Let $\mathcal{F}$ be a weighted constraint language such that* $\mathrm{Func}_1(D,[0,1]_{\mathbb{Q}}) \subseteq \mathcal{F} \subseteq \mathrm{Func}(D,[0,1]_{\mathbb{Q}})$. *If $\mathcal{F}$ is weakly log-supermodular then $\ell(\mathcal{F})$ has an STP/MJN multimorphism.*

Proof. Let $\Phi = \ell(\mathcal{F}) \cup \mathrm{Func}_1(D, \overline{\mathbb{R}}_{\ge 0})$. We will show that Φ has an STP/MJN multimorphism. By Definitions 29 and 25, this is also an STP/MJN multimorphism of the subset $\ell(\mathcal{F})$.

By Lemma 31, $\ell(\mathcal{F})$ is weakly submodular. Now, $\ell(\mathcal{F})$ contains $\ell(\mathrm{Func}_1(D,[0,1]_{\mathbb{Q}}))$. Thus, for every unary function $u \in \mathrm{Func}_1(D, \overline{\mathbb{R}}_{\ge 0})$ and every $\varepsilon \in (0,1)$, there is a unary function $u_\varepsilon \in \ell(\mathcal{F})$ such that, for all $x \in \{0,1\}$, $|u(x) - u_\varepsilon(x)| < \varepsilon$. From the definition of valued clones, and continuity, we deduce that, for every binary function $f \in \langle\Phi\rangle_V$ and every $\varepsilon > (0,1)$, there is an $f_\varepsilon \in \langle\ell(\mathcal{F})\rangle_V$ such that, for all $x,y \in \{0,1\}$, $|f(x,y) - f_\varepsilon(x,y)| < \varepsilon$. Since $\ell(\mathcal{F})$ is weakly submodular, we conclude from the definition of weak submodularity (Definition 30) that Φ is weakly submodular.

In [22, §6.1–6.4], Kolmogorov and Živný show how to construct an STP/MJN multimorphism of Φ under "Assumptions 1–3". Assumption 1 is that Φ is conservative, which is true by construction. Assumption 3 is that Φ is weakly submodular. This is given as a premise of our lemma. Assumption 2 is that $\Gamma = \mathrm{Feas}(\Phi)$ has a majority polymorphism, which follows from Assumptions 1 and 3 by Lemma 33. (Assumption 2 states that Φ has a majority polymorphism. In our terminology, this means that $\mathrm{Feas}(\Phi)$ has a majority polymorphism.) $\square$

6 LSM-easiness and #BIS-easiness

Our aim is to show that if $\ell(\mathcal{F})$ has an STP/MJN multimorphism then $\mathcal{F}$ is LSM-easy. This will involve using the arguments of [12] and [22], but we try, as much as possible, to avoid going into the details of their proofs. We start by generalising the notion of an STP multimorphism.

Definition 35. Let f be an arity-k cost function. A *multisorted multimorphism of f* is a pair $\langle \sqcap, \sqcup \rangle$, defined as follows. For $1 \le i \le k$, $\sqcap_i$ and $\sqcup_i$ are operations on the set $D_i = \{a \in D \mid \exists \mathbf{x} : x_i = a \text{ and } f(\mathbf{x}) < \infty\}$, and $\langle \sqcap_i, \sqcup_i \rangle$ is an STP of $\{f\}$.

The operation $\sqcap$ is the binary operation on $D_1 \times \cdots \times D_k$ defined by applying $\sqcap_1, \ldots, \sqcap_k$ component-wise. Similarly, $\sqcup$ is defined by applying $\sqcup_1, \ldots, \sqcup_k$ component-wise. We require that, for all $\mathbf{x}, \mathbf{y} \in D^k$, $f(\sqcup(\mathbf{x}, \mathbf{y})) + f(\sqcap(\mathbf{x}, \mathbf{y})) \le f(\mathbf{x}) + f(\mathbf{y})$. Equivalently, we require

$$f(\sqcup_1(x_1, y_1), \ldots, \sqcup_k(x_k, y_k)) + f(\sqcap_1(x_1, y_1), \ldots, \sqcap_k(x_k, y_k)) \le f(\mathbf{x}) + f(\mathbf{y}).$$

Kolmogorov and Živný [22, Equation (35)] use a slightly more general definition, where D_i can be any superset of $\{a \in D \mid \exists \mathbf{x} : x_i = a \text{ and } f(\mathbf{x}) < \infty\}$. But it does no harm to be more specific.

Where it is clearer, we use infix notation for operations such as $\sqcap$ and $\sqcup$.

Theorem 36 (Kolmogorov and Živný)**.** *Suppose Φ_0 is a finite, valued constraint language which has an STP/MJN multimorphism. Then there is a polynomial-time algorithm that takes an instance I of* $\mathrm{VCSP}(\Phi_0)$ *and returns a multisorted multimorphism $\langle \sqcap, \sqcup \rangle$ of f_I. The pair $\langle \sqcap, \sqcup \rangle$ depends only on the STP/MJN multimorphism of Φ_0 and on the relation* $\mathrm{Feas}(f_I)$ *underlying f_I. It does not depend in any other way on I.*

Proof. This is proved by Stages 1 and 2 of the proof of Theorem 3.4 in [22, §7], in which Kolmogorov and Živný establish the existence of the pair $\langle \sqcap, \sqcup \rangle$ that we require.

Note that Kolmogorov and Živný restrict to rationals, whereas we allow real numbers, but this is not a problem. Their proof constructs $\langle \sqcap, \sqcup \rangle$ using an algorithm but this algorithm does not require access to the functions in Φ_0 themselves. Instead, it only requires access to the relations in $\mathrm{Feas}(\Phi_0)$ and to the STP/MJN multimorphism that Φ_0 satisfies. These are both finite amounts of data, which can be hardwired into the algorithm, whose input is just the instance I, which is a symbolic expression. ◻

We will also use the following algorithmic consequence of [22, Theorem 3.4]. We restrict to crisp cost functions because this is all that we use and we wish to avoid issues with number systems.

Theorem 37 (Kolmogorov and Živný)**.** *Suppose that Φ_0 is a finite, crisp constraint language that has an STP/MJN multimorphism. Then there is a polynomial-time algorithm for* $\mathrm{VCSP}(\Phi_0)$.

For our eventual construction, we would like $\langle \sqcap, \sqcup \rangle$ to induce a multisorted multimorphism of f_t for each individual valued constraint t in the instance. We do not know whether this is true of the multisorted multimorphism provided by Kolmogorov and Živný's algorithm, but something sufficiently close to this is true.

Definition 38. For an instance I, a valued constraint t and a length-k_t vector $\mathbf{a}$, define

$$R_{I,t}(\mathbf{a}) = \begin{cases} 0, & \text{if there exists } \mathbf{x} \text{ with } \mathbf{x}[\sigma_t] = \mathbf{a} \text{ and } f_I(\mathbf{x}) < \infty; \\ \infty, & \text{otherwise,} \end{cases}$$

and define $f'_t = f_t + R_{I,t}$.

Thus, f'_t is a "trimmed" version of f_t whose domain is precisely the k_t-tuples of values that can actually arise in feasible solutions to instance I. We will see that if the scope σ_t contains variables with indices $i(t,1),\dots,i(t,k_t)$, then

$$\big\langle \sqcap[\sigma_t], \sqcup[\sigma_t] \big\rangle = \big\langle (\sqcap_{i(t,1)},\dots,\sqcap_{i(t,k_t)}), (\sqcup_{i(t,1)},\dots,\sqcup_{i(t,k_t)}) \big\rangle$$

is a multisorted multimorphism of f'_t, even though it might not necessarily be a multisorted multimorphism of f_t.

Note that Theorem 37 has the following consequence.

Corollary 39. *Let Φ_0 be a finite, valued constraint language that has an STP/MJN multimorphism. There is a polynomial-time algorithm that takes an instance I of* VCSP(Φ_0)*, a valued constraint t and returns a truth table for f'_t.*

Proof. By Observation 26, any STP/MJN multimorphism of Φ_0 is also an STP/MJN multimorphism of $\Phi'_0 = \Phi_0 \cup \mathrm{Func}_1(D, \{0,\infty\})$. Now, for each vector $\mathbf{a} \in D^{k_t}$ in turn, we can determine the value of $f'_t(\mathbf{a})$ as follows. Let $I_{\mathbf{a}}$ be the VCSP(Φ'_0) instance that results from adding to I the set of k_t crisp, unary, valued constraints that force the tuple of variables $\mathbf{x}[\sigma_t]$ to take value $\mathbf{a}$. By Theorem 37, we can compute in polynomial time whether $f_{I_{\mathbf{a}}}(\mathbf{a}) < \infty$ and, thus, determine the value of $f'_t(\mathbf{a})$. $\square$

The truth table produced by the algorithm of Corollary 39 is finite since all valued constraints in Φ_0 are finite.

Theorem 40 (An extension to Theorem 36)**.** *Suppose Φ_0 is a finite, valued constraint language which has an STP/MJN multimorphism. Consider the algorithm from Theorem 36 which takes an instance I of* VCSP(Φ_0) *(in the form (6)) and returns a multisorted multimorphism $\langle \sqcap, \sqcup \rangle$ of f_I. Then, for all $t \in T$, $\big\langle \sqcap[\sigma_t], \sqcup[\sigma_t] \big\rangle$ is a multisorted multimorphism of f'_t.*

Proof. Focus on a particular valued constraint t of I. Let $k = k_t$ be the arity of f_t, and for brevity denote $\sqcap[\sigma_t]$ and $\sqcup[\sigma_t]$ by $\sqcap'$ and $\sqcup'$, respectively. Without loss of generality assume $\sigma_t = (1,2,\dots,k)$. We wish to show that

$$f'_t(\mathbf{a} \sqcap' \mathbf{b}) + f'_t(\mathbf{a} \sqcup' \mathbf{b}) \le f'_t(\mathbf{a}) + f'_t(\mathbf{b}) \tag{8}$$

for all $\mathbf{a}, \mathbf{b} \in D_1 \times \cdots \times D_k$. If either $f'_t(\mathbf{a}) = \infty$ or $f'_t(\mathbf{b}) = \infty$, then we are done. Otherwise, by construction of f'_t, there exist $\mathbf{x}$ and $\mathbf{y}$ such that $\mathbf{a} = \mathbf{x}[\sigma_t]$, $\mathbf{b} = \mathbf{y}[\sigma_t]$, and $f_I(\mathbf{x}), f_I(\mathbf{y}) < \infty$. Notice that $f'_t(\mathbf{a}) = f_t(\mathbf{a}) < \infty$ and $f'_t(\mathbf{b}) = f_t(\mathbf{b}) < \infty$, also by construction of f'_t. Now consider the augmented instance I_N of I with N extra copies of the valued constraint t. We have

$$\begin{aligned} f_{I_N}(\mathbf{x}) &= f_I(\mathbf{x}) + N f_t(\mathbf{a}) \\ f_{I_N}(\mathbf{y}) &= f_I(\mathbf{y}) + N f_t(\mathbf{b}) \\ f_{I_N}(\mathbf{x} \sqcap \mathbf{y}) &= f_I(\mathbf{x} \sqcap \mathbf{y}) + N f_t(\mathbf{a} \sqcap' \mathbf{b}) \\ f_{I_N}(\mathbf{x} \sqcup \mathbf{y}) &= f_I(\mathbf{x} \sqcup \mathbf{y}) + N f_t(\mathbf{a} \sqcup' \mathbf{b}). \end{aligned} \tag{9}$$

Since $\mathrm{Feas}(f_{I_N}) = \mathrm{Feas}(f_I)$, from Theorem 36, $\langle \sqcap, \sqcup \rangle$ is also a multisorted multimorphism of f_{I_N}, i.e.,

$$f_{I_N}(\mathbf{x} \sqcap \mathbf{y}) + f_{I_N}(\mathbf{x} \sqcup \mathbf{y}) \le f_{I_N}(\mathbf{x}) + f_{I_N}(\mathbf{y}).$$

Combining this with (9), we obtain

$$f_t(\mathbf{a} \sqcap' \mathbf{b}) + f_t(\mathbf{a} \sqcup' \mathbf{b}) + O(N^{-1}) \le f_t(\mathbf{a}) + f_t(\mathbf{b}) + O(N^{-1}). \tag{10}$$

Since (10) remains true as $N \to \infty$ but f_t is independent of N, we conclude that

$$f_t(\mathbf{a} \sqcap' \mathbf{b}) + f_t(\mathbf{a} \sqcup' \mathbf{b}) \le f_t(\mathbf{a}) + f_t(\mathbf{b}).$$

Since $\mathbf{a} \sqcap' \mathbf{b}$ and $\mathbf{a} \sqcup' \mathbf{b}$ extend to feasible solutions $\mathbf{x} \sqcap \mathbf{y}$ and $\mathbf{x} \sqcup \mathbf{y}$, it follows that $f'_t(\mathbf{a} \sqcap' \mathbf{b}) = f_t(\mathbf{a} \sqcap' \mathbf{b})$ and $f'_t(\mathbf{a} \sqcup' \mathbf{b}) = f_t(\mathbf{a} \sqcup' \mathbf{b})$. The required inequality (8) follows immediately. ◻

To make use of Theorem 40, we will use the following definitions.

Definition 41. Given a finite, valued constraint language $\Phi_0 \subset \mathrm{Func}(D, \overline{\mathbb{R}}_{\ge 0})$, let Φ'_0 be the set of functions of the form $f + R$, for $f \in \Phi_0 \cap \mathrm{Func}_k(D, \overline{\mathbb{R}}_{\ge 0})$, $R \in \mathrm{Func}_k(D, \{0, \infty\})$ and $k \in \mathbb{N}$.

Note that Φ'_0 is finite because $\mathrm{Func}_k(D, \{0, \infty\})$ is finite for any finite k.

Definition 42. Suppose that I is an n-variable instance of $\mathrm{VCSP}(\Phi)$ with domain D and I' is an n'-variable instance of $\mathrm{VCSP}(\Phi')$ with domain D'. We say that I and I' are *equivalent* if there is a bijection π from $\{\mathbf{x} \in D^n \mid f_I(\mathbf{x}) < \infty\}$ to $\{\mathbf{x}' \in D'^{n'} \mid f_{I'}(\mathbf{x}') < \infty\}$ such that, for all $\mathbf{x}$ in the domain of π, $f_I(\mathbf{x}) = f_{I'}(\pi(\mathbf{x}))$.

Lemma 44 below will construct equivalent instances I and I' in a setting where $n = n'$ and $D = D'$. In this case, π will be the identity bijection from D^n to itself. Later, we will consider equivalences of instances with different domains.

Definition 43. [12] A function $f \colon D_1 \times \cdots \times D_r \to \overline{\mathbb{R}}_{\ge 0}$ is *domain-reduced* if, for each $i \in \{1, \ldots, r\}$, and for each $a \in D_i$, there is an $\mathbf{x} \in D^n$ such that $x_i = a$ and $f(\mathbf{x}) < \infty$.

Lemma 44. *Suppose Φ_0 is a finite, valued constraint language which has an STP/MJN multimorphism. Consider an instance I of* $\mathrm{VCSP}(\Phi_0)$. *There is an equivalent instance I' of* $\mathrm{VCSP}(\Phi'_0)$ *and a multisorted multimorphism $\langle \sqcap, \sqcup \rangle$ of $f_{I'}$ which induces a multisorted multimorphism of f_t for each valued constraint t of I'. Both I' and $\langle \sqcap, \sqcup \rangle$ are polynomial-time computable (given I). Moreover, each operation $\sqcap_i$ and $\sqcup_i$ induces a total order.*

Proof. We first show how to construct an equivalent instance I' and a multisorted multimorphism of $f_{I'}$ which induces multisorted multimorphisms on the valued constraints. To obtain I', start from the instance I and use Corollary 39 to replace each valued constraint $f_t(\mathbf{x}[\sigma_t])$ with $f'_t(\mathbf{x}[\sigma_t])$. This operation clearly preserves the set of feasible solutions and their costs. Then use the algorithm from Theorem 40 to construct the multisorted multimorphism $\langle \sqcap, \sqcup \rangle$.

In the remainder of the proof, we construct a new multisorted multimorphism by modifying $\langle \sqcap, \sqcup \rangle$ to ensure that its components induce total orders, as required. Consider the following claim.

Claim: Suppose that D is a domain. Given a set of functions $\Phi \subseteq \mathrm{Func}(D, \overline{\mathbb{R}}_{\ge 0})$, *let $\mathcal{P}$ be an instance of* $\mathrm{VCSP}(\Phi)$ *with variable set $\{v_1, \ldots, v_n\}$. Let $D_i = \{a \in D \mid \exists \mathbf{x} : x_i = a$ and $f_{\mathcal{P}}(\mathbf{x}) < \infty\}$. Suppose that $\langle \sqcap, \sqcup \rangle$ is a multisorted multimorphism of $\mathcal{P}$. Then there is a multisorted multimorphism $\langle \sqcap', \sqcup' \rangle$ of $\mathcal{P}$ in which each $\sqcap'_i$ induces a total order on D_i (hence $\sqcup'_i$ induces the reversal of this total order). Furthermore, for any set $J = \{i_1, \ldots, i_j\} \subseteq \{1, \ldots, n\}$ and any domain-reduced function $\phi \colon D_{i_1} \times \cdots \times D_{i_j} \to \overline{\mathbb{R}}_{\ge 0}$ for which $\langle \sqcap_J, \sqcup_J \rangle$*

is a multimorphism, $\langle \sqcap'_J, \sqcup'_J \rangle$ is also a multimorphism of ϕ. The multimorphism $\langle \sqcap', \sqcup' \rangle$ is polynomial-time computable.

This claim is proved (but not explicitly stated) in the proof of [12, Theorem 8.2].[1] The basic method is as follows. $\mathcal{P}$ is augmented with extra (redundant) valued constraints using unary and binary crisp cost functions. The binary crisp cost functions are used to enforce consistency so that when $\sqcap_i$ is modified to induce a total order on D_i, a compatible modification is made to each other $\sqcap_j$. Once $\sqcap$ and $\sqcup$ are constructed, it is proved by induction that every relevant function ϕ has the property specified in the claim. The induction is on the arity of ϕ.

To prove the lemma, we use the claim with $\Phi = \Phi'_0$, $\mathcal{P} = I'$ and, for each valued constraint t of I', $\phi = f'_t$. $\square$

Lemma 45. *If $\mathcal{F} \subseteq \mathrm{Func}(D, [0,1]_{\mathbb{Q}})$ and $\ell(\mathcal{F})$ has an STP/MJN multimorphism, then $\#\mathrm{CSP}(\mathcal{G})$ is* LSM*-easy for every finite $\mathcal{G} \subset \mathcal{F}$.*

Proof. Let $\mathcal{G}$ be a finite subset of $\mathcal{F}$. To any instance $I_{\#}$ of $\#\mathrm{CSP}(\mathcal{G})$ there corresponds an instance $I = \ell(I_{\#})$ of $\mathrm{VCSP}(\Phi_0)$, where $\Phi_0 = \ell(\mathcal{G})$: for each weighted constraint t, the function F_t is mapped to $f_t = \ell(F_t)$ while the scope σ_t remains unchanged. Using Lemma 44, we may construct an equivalent instance I' of $\mathrm{VCSP}(\Phi'_0)$ on the domain $D_1 \times \cdots \times D_n$ and a multisorted multimorphism $\langle \sqcap, \sqcup \rangle$ of that instance, where each $\sqcap_i$ induces a total order. $\langle \sqcap, \sqcup \rangle$ induces a multisorted multimorphism of each f_t.

We now construct an instance I'' over the Boolean domain that is equivalent to I' and hence to I. For each i, $1 \le i \le n$, introduce a set of $|D_i| + 1$ Boolean variables $V_i = \{z_{i,a} \mid a \in D_i^+\}$, where $D_i^+ = D_i \cup \{\bot\}$. Extend the total order on D_i^+ by placing $\bot$ below all elements of D_i. Define a nested sequence of subsets of D_i^+ by $U_{i,a} = \{b \in D_i^+ \mid b < a\}$. The idea is that the bijection π that establishes the equivalence between I' and I'' maps each domain element $a \in D_i$ to the sequence of Boolean values that assigns 1 to all variables in $U_{i,a}$, and 0 to the others in V_i. Consider the constraint asserting that only these $|D_i|$ particular assignments to V_i are allowed. This constraint can be represented by the crisp cost function f that assigns $f(\mathbf{x}) = 0$ to these assignments and $f(\mathbf{x}) = \infty$ to all others. Note that $F(x) = \exp(-f(x))$ is log-supermodular.

Note that we can use the same relation for any pair of sets D_i and D_j with $|D_i| = |D_j|$ — if D_i and D_j have different total orders then the relation is applied to the variables in D_i^+ in a different order than to the variables in D_j^+. If we add these crisp valued constraints then there is a natural bijection π between $D_1 \times \cdots \times D_n$ and feasible assignments to Boolean variables $V_1 \cup \cdots \cup V_n$. The variable $z_{i,a}$ where a is the smallest (respectively largest) element of D_i^+ always takes on the value 1 (respectively 0), and so these variables are redundant. However, their introduction simplifies the description of some constructions later in the proof.

Consider a valued constraint in I' of arity k that imposes the function $f' \in \Phi'_0$, and, without loss of generality, assume that its scope is the first k variables $x_1, \dots, x_k$. Add a corresponding valued constraint f'' to I'' with $f'' \colon 2^{V_1 \cup \cdots \cup V_k} \to \overline{\mathbb{R}}_{\ge 0}$ defined as follows, where for convenience we are viewing f'' as a function on subsets of $V_1 \cup \cdots \cup V_k$ rather than as a function of $|V_1| + \cdots + |V_k|$ Boolean variables:

$$f''(A) = \begin{cases} f'(a_1, \dots, a_k), & \text{if } A = U_{1,a_1} \cup \cdots \cup U_{k,a_k} \text{ for some } (a_1, \dots, a_k); \\ \infty, & \text{otherwise.} \end{cases}$$

[1] [12] uses somewhat different notation to ours: our $\sqcap$ and $\sqcup$ are their f and g, respectively; in [12], $\langle f, g \rangle$ denotes an ordered pair, whereas we use that notation to denote a clone.

We claim f'' is submodular, i.e., $f''(A\cap B)+f''(A\cup B) \le f''(A)+f''(B)$. If either $f''(A) = \infty$ or $f''(B) = \infty$ there is nothing to prove. So $A = U_{1,a_1} \cup \cdots \cup U_{k,a_k}$ and $B = U_{1,b_1} \cup \cdots \cup U_{k,b_k}$ for some $(a_1, \dots, a_k), (b_1, \dots, b_k) \in D_1 \times \cdots \times D_k$. Then

$$\begin{aligned}
&f''(A \cap B) + f''(A \cup B) \\
&= f''\big((U_{1,a_1} \cap U_{1,b_1}) \cup \cdots \cup (U_{k,a_k} \cap U_{k,b_k})\big) \\
&\qquad + f''\big((U_{1,a_1} \cup U_{1,b_1}) \cup \cdots \cup (U_{k,a_k} \cup U_{k,b_k})\big) \\
&= f''(U_{1,a_1 \sqcap_1 b_1} \cup \cdots \cup U_{k,a_k \sqcap_k b_k}) \\
&\qquad + f(U_{1,a_1 \sqcup_1 b_1} \cup \cdots \cup U_{k,a_k \sqcup_k b_k}) \\
&= f'(a_1 \sqcap_1 b_1, \dots, a_k \sqcap_k b_k) + f'(a_1 \sqcup_1 b_1, \dots, a_k \sqcup_k b_k) \\
&\le f'(a_1, \dots, a_k) + f'(b_1, \dots, b_k) \\
&= f''(A) + f''(B).
\end{aligned}$$

Now take stock. We have an instance I'' of Boolean VCSP, which is equivalent to I' and hence to I. It has at most $n(|D| + 1)$ Boolean variables and it has n more valued constraints than I. The number of distinct valued constraints in Φ''_0 is $|\Phi''_0| \le |\Phi'_0| + |D|$; note that these come from a fixed set of cost functions independent of the instance I and hence of $I_\#$ itself.

Now map the VCSP instance I'' back to #CSP to yield an instance $I''_\#$ over the Boolean domain in which every valued constraint comes from a certain fixed set of cost functions $\mathcal{F}''_0 \subset \mathsf{LSM}$. Specifically, $I'' = \ell(I''_\#)$ and $\Phi''_0 = \ell(\mathcal{F}''_0)$. Since I'' is equivalent to I, there is a bijection between the non-zero terms of $Z(I_\#)$ and $Z(I''_\#)$ that preserves weights, and hence $Z(I_\#) = Z(I''_\#)$. □

Lemma 45 shows that, if $\ell(\mathcal{F})$ has an STP/MJN multimorphism, then $\#\mathrm{CSP}(\mathcal{G})$ is LSM-easy for every finite $\mathcal{G} \subset \mathcal{F}$. Lemma 46 below strengthens the result by showing that $\#\mathrm{CSP}(\mathcal{G})$ is #BIS-easy. The strengthening applies when the weight functions in $\mathcal{F}$ have arity at most two.

In order to do the strengthening, we need to generalise the notion of a binary submodular function to cover binary functions over larger domains. Let D and D' be ordered sets. Following [13], we say that a function $f : D \times D' \to \overline{\mathbb{R}}_{\ge 0}$ is *submodular* if, for all $r, s \in D$ and all $r', s' \in D'$,

$$f(\min(r, r'), \min(s, s')) + f(\max(r, r'), \max(s, s')) \le f(r, s) + f(r', s').$$

To apply this concept here, suppose that f is a function with domain $D_i \times D_j$. Given orders on D_i and D_j, let $D_i(\ell)$ and $D_j(\ell)$ denote the ℓ'th element of D_i and D_j, respectively. Submodularity of f is equivalent to saying that the $|D_i| \times |D_j|$ matrix M_f satisfies the Monge property [25] where, as in Section 3, $(M_f)_{k\ell} = f(D_i(k), D_j(\ell))$. We extend Definition 5 by saying that a function $F : D_i \times D_j \to [0, 1]_{\mathbb{Q}}$ is log-supermodular (with respect to the given orders) if the function $\ell(F)$ is sub-modular (with respect to the same orders).

Lemma 46. *If $\mathcal{F} \subseteq \mathrm{Func}(D, [0, 1]_{\mathbb{Q}})$ is a weighted constraint language whose weight functions have arity at most two and $\ell(\mathcal{F})$ has an STP/MJN multimorphism, then $\#\mathrm{CSP}(\mathcal{G})$ is #BIS-easy for every finite $\mathcal{G} \subset \mathcal{F}$.*

Proof. Let $\mathcal{G}$ be a finite subset of $\mathcal{F}$. We use exactly the same construction as in the previous lemma, but go further and show that every weight function F'' appearing in instance $I''_\#$

is expressible in terms of unary weight functions in $\mathcal{U}_{\{0,1\}}$, and the binary weight function IMP defined by $\mathrm{IMP}(0,0) = \mathrm{IMP}(0,1) = \mathrm{IMP}(1,1) = 1$ and $\mathrm{IMP}(1,0) = 0$. Moreover, unary weight functions in $\mathcal{U}_{\{0,1\}}$ (even those taking irrational values) can be approximated sufficiently closely by polynomial-sized pps-formulas using IMP [6, Lemma 13.1]. This will complete the proof, since $\#\mathrm{CSP}(\mathrm{IMP}) \leq_{\mathrm{AP}} \#\mathrm{BIS}$ by [17, Theorem 5].

The task then, is to show that every weight function F'' in instance $I''_{\#}$ is expressible in terms of unary weight functions in $\mathcal{U}_{\{0,1\}}$ and IMP. We do this by considering, in turn, the different types of weight functions arising in $I''_{\#}$. The n relations (crisp cost functions) that were introduced in I'' to impose a total order on the variables in the sets V_i are clearly implementable in terms of $\mathrm{imp} = \ell(\mathrm{IMP})$.

Every other weight function F'' is associated with a cost function f'' in I'' that is an implementation over the Boolean domain of a cost function f' from I'. Since $f' \in \Phi_0$, it has arity at most 2. Our goal is to show that the function $F'(\mathbf{x}) = \ell^{-1}(f'(\mathbf{x})) = \exp(-f'(\mathbf{x}))$ is expressible in terms of unary weight functions in $\mathcal{U}_{\{0,1\}}$ and IMP. If f' is unary, this is immediate, so suppose f' is binary.

To fix the notation, suppose that f' is a function $f' \colon D_i \times D_j \to \overline{\mathbb{R}}_{\geq 0}$. We can assume without loss of generality that D_i and D_j are disjoint (otherwise, rename some elements). Also, D_i and D_j are ordered according to the linear order induced by $\sqcap$. Since $\langle \sqcap, \sqcup \rangle$ induces a multisorted multimorphism of f' (see Definition 35), the function f' is submodular (with respect to this order).

Building on the work of Rudolf and Woeginger [25], Cohen, Cooper, Jeavons and Krokhin [13, Lemma 4.5] have shown that every binary submodular function is expressible as a positive linear combination of certain simple binary submodular functions. Translated to our setting by applying ℓ^{-1}, this says that F' is expressible as a product of certain simple basis functions, namely the binary functions

$$B^{\alpha}_{a,b}(x,y) = \begin{cases} \alpha, & \text{if } x \geq a \text{ and } y \leq b; \\ 1, & \text{otherwise,} \end{cases}$$

for all $(a,b) \in (D_i, D_j)$ (with a similar set of binary functions defined by replacing $x \geq a$ and $y \leq b$ by $x \leq a$ and $y \geq b$), where α is an arbitrary constant in the range $[0,1]$.

$B^{0}_{a,b}(x,y)$ may be implemented as $\mathrm{IMP}(z_{i,a^-}, z_{j,b})$, where a^- is the element immediately below a in the total order on D_i. (To see this, note that the constraint rules out the possibility that $z_{i,a^-} = 1$, which corresponds to $x \geq a$ together with $z_{j,b} = 0$, which corresponds to $y \leq b$.) Let

$$U_\beta(z) = \begin{cases} \beta, & \text{if } z = 0; \\ 1, & \text{if } z = 1. \end{cases}$$

Then for $\alpha > 0$, the basis function $B^{\alpha}_{a,b}(x,y)$ may be implemented as

$$\mathrm{IMP}(z_{i,a^-}, w)\, \mathrm{IMP}(z_{j,b}, w) U_\alpha(z_{j,b})\, U_{1/\alpha - 1}(w),$$

where w is a new variable. □

To use Lemmas 45 and 46, we need to perform some scaling. For any k-ary weight function in $F \in \mathcal{F}$, let $m_F = \max\{f(\mathbf{x}) \mid \mathbf{x} \in D^k\}$. Let

$$\Lambda(F) = \begin{cases} F/m_F & \text{if } m_F > 1 \\ F & \text{otherwise} \end{cases}$$

and let $\Lambda(\mathcal{F}) = \{\Lambda(F) \mid F \in \mathcal{F}\}$. Note that $\Lambda(F)$ always takes values in $[0,1]_{\mathbb{Q}}$ and that, since $\mathcal{F}$ is conservative, $\mathrm{Func}_1(D, [0,1]_{\mathbb{Q}}) \subseteq \Lambda(\mathcal{F})$.

We return, once more, to the proof of Theorem 6.

Theorem 47. *Let $\mathcal{F}$ be a weakly log-supermodular, conservative weighted constraint language taking values in $\mathbb{Q}_{\geq 0}$.*

- *For any finite $\mathcal{G} \subset \mathcal{F}$, there is a finite $\mathcal{G}' \subset \mathsf{LSM}$ such that $\#\mathrm{CSP}(\mathcal{G}) \leq_{\mathrm{AP}} \#\mathrm{CSP}(\mathcal{G}')$.*
- *If $\mathcal{F}$ consists of functions of arity at most two, then $\#\mathrm{CSP}(\mathcal{G})$ is #BIS-easy for any finite $\mathcal{G} \subset \mathcal{F}$.*

Proof. By Theorem 34, $\ell(\Lambda(\mathcal{F}))$ has an STP/MJN multimorphism. The result follows from Lemmas 45 and 46 and the fact that $\#\mathrm{CSP}(\mathcal{F}) \leq_{\mathrm{AP}} \#\mathrm{CSP}(\Lambda(\mathcal{F}))$. □

Theorem 6, our classification of the complexity of approximating $\#\mathrm{CSP}(\mathcal{F})$, now follows from Theorems 10, 15 and 47.

7 Algorithmic aspects

Finally, we consider the algorithmic aspects of the classification of Theorem 6. Intuitively, there is an algorithm that determines the complexity of #CSP with constraints from a finite language $\mathcal{H}$ plus unary weights because weak log-modularity is essentially equivalent to balance and weak log-supermodularity is essentially equivalent to the existence of a STP/MJN multimorphism. As we will show below, balance and the existence of STP/MJN multimorphisms depend only on certain finite parts of the weighted constraint language so balance is decidable by [8] and the existence of STP/MJN multimorphisms can be determined by brute force, or by using more sophisticated methods from [22].

We need to determine whether the infinite language $\mathcal{H} \cup \mathcal{U}_D$ is balanced. Fortunately, it suffices to check whether $\mathcal{H} \cup \mathcal{U}'_D$ is balanced, where $\mathcal{U}'_D = \mathrm{Func}_1(D, \{1,2\})$, which is finite. (Note that it is not enough to test whether $\mathcal{H}$ is balanced; also, there is nothing special about 1 and 2: any pair of distinct, positive rationals would do. In fact, $|\mathcal{U}'_D| = 2^{|D|}$ and there are sets of size $|D|$ which would suffice, but we do not need this here.)

Lemma 48. *Let $\mathcal{H}$ be a finite, weighted constraint language taking values in $\mathbb{Q}_{\geq 0}$. The following are equivalent: (1) $\mathcal{H} \cup \mathcal{U}'_D$ is balanced; (2) every finite subset of $\mathcal{H} \cup \mathcal{U}_D$ is balanced; and (3) $\mathcal{H} \cup \mathcal{U}_D$ is balanced.*

Proof. (2) and (3) are equivalent because any pps-formula contains only a finite number of atomic formulas. (2) trivially implies (1), since $\mathcal{U}'_D$ is finite. It remains to show that (1) implies (2) so, towards this goal, suppose that $\mathcal{H} \cup \mathcal{U}'_D$ is balanced. We must show that every finite subset of $\mathcal{H} \cup \mathcal{U}_D$ is balanced. Suppose that such a subset contains r functions in $\mathcal{U}_D \setminus \mathcal{H}$.

Let $\{F_1, \ldots, F_r\}$ be unary functions such that $F_i(d) = a_{i,d}$ ($i \in \{1, \ldots, r\}$, $d \in D$) and let $\mathcal{G} = \mathcal{H} \cup \{F_1, \ldots, F_r\}$. We may consider the $a_{i,d}$ as formal variables and treat a function $G \in \langle \mathcal{G} \rangle_{\#}$ with free variables $\mathbf{x}$ as a function of both $\mathbf{x}$ and the $a_{i,d}$. We will show that, for any function G and any interpretation of the $a_{i,d}$ (i.e., any instantiation of the function symbols F_i as concrete functions $D \to \mathbb{Q}_{\geq 0}$), the matrices associated with G have block-rank 1, thus establishing that $\mathcal{G}$ is balanced.

So, consider any $G \in \langle \mathcal{G} \rangle_\#$ with arity $n \geq 2$ and choose any k with $1 \leq k < n$. We will show that the $D^k \times D^{n-k}$ matrix $M_G(\mathbf{x}, \mathbf{y})$ has block-rank 1 for any value of the $a_{i,d}$. By Lemma 11 part (2), it suffices to show that every 2×2 submatrix induced by rows $\mathbf{x}, \mathbf{x}'$ and columns $\mathbf{y}, \mathbf{y}'$ has block-rank 1. By Lemma 11 part (1), this happens if, and only if, every such submatrix has rank 1 or at least two zero entries, which happens if, and only if, the multivariate polynomial

$$p = G(\mathbf{x}, \mathbf{y}')G(\mathbf{x}', \mathbf{y})G(\mathbf{x}', \mathbf{y}')\big[G(\mathbf{x}, \mathbf{y})G(\mathbf{x}', \mathbf{y}') - G(\mathbf{x}', \mathbf{y})G(\mathbf{x}, \mathbf{y}')\big]$$

is zero for all values of $\mathbf{x}, \mathbf{x}', \mathbf{y}, \mathbf{y}'$ and for all values of the $a_{i,d}$. (Note that, if the submatrix defined by a pair of rows and columns does not have block-rank 1 but has exactly one zero, then only one of the four possible choices for $\mathbf{x}, \mathbf{x}', \mathbf{y}, \mathbf{y}'$ will make p non-zero.)

We now fix $\mathbf{x}, \mathbf{x}', \mathbf{y}, \mathbf{y}'$ and consider p as a function of just the $a_{i,d}$. Our goal is to show that (for every choice of $\mathbf{x}, \mathbf{x}', \mathbf{y}, \mathbf{y}'$), p is identically 0.

Consider first the case where every $a_{i,d}$ is a power of two. Here, every atomic formula $F_i(z)$ defines the same function as some product $U_1(z) \cdots U_\ell(z)$ of atomic formulas from $\mathcal{U}'_D$ so G is equivalent to some function in $\langle \mathcal{H} \cup \mathcal{U}'_D \rangle_\#$. But $\mathcal{H} \cup \mathcal{U}'_D$ is balanced by assumption, so $p = 0$ whenever every $a_{i,d}$ is a power of two. Therefore, $p = 0$ over a space that is a product of infinite sets. It follows from the Schwartz–Zippel lemma or from [1, Theorem 1.2] that the only polynomial with this property is the zero polynomial, so p is the zero polynomial and $\mathcal{H} \cup \{F_1, \ldots, F_r\}$ is balanced for any set $\{F_1, \ldots, F_r\}$ of unary weights. ☐

Theorem 49. *There is an algorithm that, given a finite, weighted constraint language $\mathcal{H}$ taking values in $\mathbb{Q}_{\geq 0}$, correctly makes one of the following deductions, where $\mathcal{F} = \mathcal{H} \cup \mathcal{U}_D$:*

1. #CSP($\mathcal{G}$) *is in* FP *for every finite $\mathcal{G} \subset \mathcal{F}$;*
2. #CSP($\mathcal{G}$) *is* LSM*-easy for every finite $\mathcal{G} \subset \mathcal{F}$ and* #BIS*-hard for some such $\mathcal{G}$;*
3. #CSP($\mathcal{G}$) *is* #BIS*-easy for every finite $\mathcal{G} \subset \mathcal{F}$ and* #BIS*-equivalent for some such $\mathcal{G}$;*
4. #CSP($\mathcal{G}$) *is* #SAT*-easy for every finite $\mathcal{G} \subset \mathcal{F}$ and* #SAT*-equivalent for some such $\mathcal{G}$.*

If every function in $\mathcal{H}$ has arity at most 2, the output is not deduction 2.

Proof. We reduce the problem to determining whether $\mathcal{H} \cup \mathcal{U}'_D$ is balanced, whether $\ell(\mathcal{H})$ has an STP/MJN multimorphism and whether $\mathcal{H}$ contains only functions of arity at most 2. Balance of finite languages is decidable [8]. An STP/MJN multimorphism consists of two operations $D^2 \to D$ and three operations $D^3 \to D$, which must have certain easily checked properties with respect to each of the functions in $\ell(\mathcal{H})$. Thus, we can determine the existence of an STP/MJN multimorphism by brute force, checking each possible collection of five operations, or by using the methods of Kolmogorov and Živný [22]. It is clearly decidable whether $\mathcal{H}$ contains a function of arity greater than 2.

By Lemma 48, if $\mathcal{H} \cup \mathcal{U}'_D$ is balanced, then so is any finite $\mathcal{G} \subset \mathcal{H} \cup \mathcal{U}_D$. Therefore, by Lemma 14, #CSP($\mathcal{G}$) can be solved exactly in FP so we output deduction 1. From this point, we assume that $\mathcal{H} \cup \mathcal{U}'_D$ is not balanced.

Since $\mathcal{H} \cup \mathcal{U}'_D$ is not balanced, nor is $\mathcal{H} \cup \mathcal{U}_D$ (Lemma 48). Therefore, $\mathcal{H} \cup \mathcal{U}_D$ is not weakly log-modular (Lemma 13) so there is a finite $\mathcal{G} \subset \mathcal{H} \cup \mathcal{U}_D$ such that #CSP($\mathcal{G}$) is #BIS-hard (Theorem 10).

$\ell(\Lambda(\mathcal{H} \cup \mathcal{U}_D))$ has an STP-MJN multimorphism if, and only if, $\ell(\Lambda(\mathcal{H}))$ does (Observation 26), and $\ell(\Lambda(\mathcal{H}))$ is a finite language so we can determine whether it has an STP-MJN multimorphism by exhaustive search. If $\ell(\Lambda(\mathcal{H} \cup \mathcal{U}_D))$ has an STP-MJN multimorphism, then, for all finite $\mathcal{G} \subset \Lambda(\mathcal{H} \cup \mathcal{U}_D)$, #CSP($\mathcal{G}$) is LSM-easy (Lemma 45). Since any function in $\Lambda(\mathcal{H} \cup \mathcal{U}_D)$ is a scalar multiple of some function in $\mathcal{H} \cup \mathcal{U}_D$, #CSP($\mathcal{G}$) is also LSM-easy for all finite $\mathcal{G} \subset \mathcal{H} \cup \mathcal{U}_D$. We output deduction 2, unless every function in $\mathcal{H}$ has arity at most 2, in which case #CSP($\mathcal{G}$) is #BIS-easy for all finite $\mathcal{G} \subset \mathcal{H} \cup \mathcal{U}_D$ (Lemma 46) and we output deduction 3.

On the other hand, if $\ell(\Lambda(\mathcal{H} \cup \mathcal{U}_D))$ has no STP-MJN multimorphism, then $\Lambda(\mathcal{H} \cup \mathcal{U}_D)$ is not weakly log-supermodular (Theorem 34). Because Λ is just a rescaling, $\mathcal{H} \cup \mathcal{U}_D$ is also not weakly log-supermodular. Therefore, there is a finite $\mathcal{G} \subset \mathcal{H} \cup \mathcal{U}_D$ such that #CSP($\mathcal{G}$) is #SAT-equivalent (Theorem 10 again). We output deduction 4. □

8 Acknowledgement

We thank the referees for their useful suggestions, which improved the presentation of the paper.

References

[1] N. Alon. Combinatorial Nullstellensatz. *Combin. Probab. Comput.*, 8(1-2):7–29, 1999. Recent trends in combinatorics (Mátraháza, 1995).

[2] E. Boros and P. L. Hammer. Pseudo-Boolean optimization. *Discrete Applied Mathematics*, 123(1–3):155–225, 2002.

[3] A. A. Bulatov. The complexity of the counting constraint satisfaction problem. In *Proceedings of 35th International Colloquium on Automata, Languages and Programming (ICALP) (Part I)*, volume 5125 of *LNCS*, pages 646–661. Springer, 2008.

[4] A. A. Bulatov, M. E. Dyer, L. A. Goldberg, M. Jalsenius, M. R. Jerrum, and D. M. Richerby. The complexity of weighted and unweighted #CSP. *Journal of Computer and System Sciences*, 78(2):681–688, 2012.

[5] A. A. Bulatov, M. E. Dyer, L. A. Goldberg, M. Jalsenius, and D. M. Richerby. The complexity of weighted Boolean #CSP with mixed signs. *Theoretical Computer Science*, 410(38–40):3949–3961, 2009.

[6] A. A. Bulatov, M. E. Dyer, L. A. Goldberg, M. R. Jerrum, and C. McQuillan. The expressibility of functions on the Boolean domain, with applications to counting CSPs. *J. ACM*, 60(5):32, 2013.

[7] J.-Y. Cai and X. Chen. Complexity of counting CSP with complex weights. In *Proceedings of 44th ACM Symposium on Theory of Computing (STOC)*, pages 909–920. ACM, 2012.

[8] J.-Y. Cai, X. Chen, and P. Lu. Non-negative weighted #CSPs: An effective complexity dichotomy. *CoRR*, abs/1012.5659, 2010. Although this is unpublished, a short version appeared in [9].

[9] J.-Y. Cai, X. Chen, and P. Lu. Non-negatively weighted #CSP: An effective complexity dichotomy. In *Proceedings of IEEE Conference on Computational Complexity*, pages 45–54, 2011.

[10] J.-Y. Cai, P. Lu, and M. Xia. Dichotomy for Holant* problems of Boolean domain. In *Proceedings of 22nd ACM–SIAM Symposium on Discrete Algorithms (SODA)*, pages 1714–1728, 2011.

[11] X. Chen. Guest column: Complexity dichotomies of counting problems. *SIGACT News*, 42(4):54–76, 2011.

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

[13] D. A. Cohen, M. C. Cooper, P. G. Jeavons, and A. A. Krokhin. A maximal tractable class of soft constraints. *J. Artif. Intell. Res.*, 22:1–22, 2004.

[14] D. A. Cohen, M. C. Cooper, P. G. Jeavons, and A. A. Krokhin. The complexity of soft constraint satisfaction. *Artif. Intell.*, 170(11):983–1016, 2006.

[15] N. Creignou and M. Hermann. Complexity of generalized satisfiability counting problems. *Information and Computation*, 125(1):1–12, 1996.

[16] M. E. Dyer, L. A. Goldberg, C. Greenhill, and M. R. Jerrum. The relative complexity of approximate counting problems. *Algorithmica*, 38(3):471–500, 2004.

[17] M. E. Dyer, L. A. Goldberg, and M. R. Jerrum. The complexity of weighted Boolean CSP. *SIAM Journal on Computing*, 38(5):1970–1986, 2009.

[18] M. E. Dyer, L. A. Goldberg, and M. R. Jerrum. An approximation trichotomy for Boolean #CSP. *Journal of Computer and System Sciences*, 76(3–4):267–277, 2010.

[19] M. E. Dyer and D. M. Richerby. An effective dichotomy for the counting constraint satisfaction problem. *SIAM J. Comput*, 43(3):1245–1274, 2013.

[20] L. A. Goldberg and M. R. Jerrum. Approximating the partition function of the ferromagnetic Potts model. *J. ACM*, 59(5):25, 2012.

[21] J.-Y.Cai, P. Lu, and M. Xia. Holant problems and counting CSP. In *Proceedings of 41st ACM Symposium on Theory of Computing (STOC)*, pages 715–724. ACM, 2009.

[22] V. Kolmogorov and S. Živný. The complexity of conservative valued CSPs. *Journal of the ACM*, 60(2), 2013.

[23] P. Lu. Complexity dichotomies of counting problems. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:93, 2011.

[24] M. Mitzenmacher and E. Upfal. *Probability and Computing.* Cambridge University Press, 2005.

[25] R. Rudolf and G. J. Woeginger. The cone of Monge matrices: extremal rays and applications. *Mathematical Methods of Operations Research*, 42(2):161–168, 1995.

[26] R. Takhanov. A dichotomy theorem for the general minimum cost homomorphism problem. *CoRR*, abs/0708.3226, 2007. Although this is unpublished, a short version appeared in [27].

[27] R. Takhanov. A dichotomy theorem for the general minimum cost homomorphism problem. In *Proceedings of 27th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 5 of *LIPIcs*, pages 657–668. Liebniz-Zentrum für Informatik, 2010.

[28] D. M. Topkis. Minimizing a submodular function on a lattice. *Operations Research*, 26:305–321, 1978.

[29] T. Yamakami. Approximate counting for complex-weighted Boolean constraint satisfaction problems. In *Proceedings of 8th Workshop on Approximation and Online Algorithms*, volume 6534 of *LNCS*, pages 261–272. Springer, 2010.