Classes of Submodular Constraints Expressible by Graph Cuts*

Stanislav Živný Peter G. Jeavons
Computing Laboratory, University of Oxford, UK
{stanislav.zivny,peter.jeavons}@comlab.ox.ac.uk

Abstract

Submodular constraints play an important role both in theory and practice of valued constraint satisfaction problems (VCSPs). It has previously been shown, using results from the theory of combinatorial optimisation, that instances of VCSPs with submodular constraints can be minimised in polynomial time. However, the general algorithm is of order $O(n^6)$ and hence rather impractical. In this paper, by using results from the theory of pseudo-Boolean optimisation, we identify several broad classes of submodular constraints over a Boolean domain which are expressible using binary submodular constraints, and hence can be minimised in cubic time. Furthermore, we describe how our results translate to certain optimisation problems arising in computer vision.

1 Introduction

The Constraint Satisfaction Problem (CSP) is a general framework which can be used to model many different problems [12, 13, 31, 37]. However, the CSP model considers only the feasibility of satisfying a collection of simultaneous requirements (so-called *hard constraints*).

Various extensions have been proposed to this model to allow it to deal with different kinds of optimisation criteria, or preferences, between different feasible solutions (so-called *soft constraints*). Two very general extended frameworks that have been proposed are the SCSP (semi-ring CSP) framework and the VCSP (valued CSP) framework [2]. The SCSP framework is slightly more general (the main difference is that costs in VCSPs represent violation levels and have to be totally ordered, whereas costs in SCSPs represent preferences and might be ordered only partially), but the VCSP framework is sufficiently powerful to model a wide range of optimisation problems [2, 37, 40].

Informally, in the Valued Constraint Satisfaction Problem (VCSP) framework, an instance consists of a set of variables, a set of possible values, and a set of (soft) constraints. Each constraint has an associated cost function which assigns a cost (or a degree of violation) to every possible tuple of values for the variables in the scope of the constraint. The goal is to find an assignment of values to all of the variables which has the minimum total cost. We remark that

*A preliminary version of this paper appeared in *Proceedings of the 14th International Conference on Principles and Practice of Constraint Programming (CP)*, 2008, pp. 112–127.

infinite costs can be used to indicate infeasible assignments (hard constraints), and hence the VCSP framework includes the standard CSP framework as a special case and is equivalent to the Constraint Optimisation Problem (COP) framework [37], which is widely used in practice.

One significant line of research on the VCSP is to identify restrictions which ensure that instances are solvable in polynomial time. There are two main types of restrictions that have been studied in the literature. Firstly, we can limit the *structure* of the instances. We will not deal with this approach in this paper.

Secondly, we can restrict the *forms* of the valued constraints which are allowed in the problem, giving rise to so-called *language restrictions*. Several language restrictions which ensure tractability have been identified in the literature [8, 12].

One important and well-studied restriction on valued constraints is *submodularity*. The class of valued constraints with submodular cost functions is the only non-trivial tractable class of optimisation problems in the dichotomy classification of Boolean VCSPs [8], and the only tractable class in the dichotomy classification of Max-CSPs for both 3-element domains [25] and arbitrary finite domains allowing constant constraints, also known as fixed-value constraints [14]. Submodularity also plays a crucial role in the Digraph Min-Cost Homomorphism problem [20], which is a special case of the VCSP framework.

The notion of submodularity originally comes from combinatorial optimisation where submodular functions are defined on subsets of a given base set [22, 32, 42]. The time complexity of the fastest known general strongly polynomial algorithm for the problem of Submodular Function Minimisation (SFM) is $O(n^6 + n^5 L)$, where L is the look-up time (needed to evaluate the cost of an assignment to all variables), and n is the number of variables of the function to be minimised [33], see also [23]. This general algorithm works in the so-called oracle-valued model, where the function to be minimised is given by an oracle.

An important and well-studied sub-problem of the SFM is the minimisation of *locally-defined* submodular functions [10], or submodular functions with *succinct representation* [15]. We will call this problem Bounded Submodular Function Minimisation, SFM_b. In this scenario the submodular function to be minimised is defined as the sum of a collection of functions which each depend only on a bounded number of variables. Locally-defined optimisation problems occur in a variety of contexts:

- In the context of Pseudo-Boolean Optimisation, such problems involve the minimisation of Boolean polynomials of bounded *degree* [3].
- In the context of computer vision, such problems are often formulated as Gibbs Energy Minimisation problems [18] or Markov Random Fields (also Conditional Random Fields) [30]. (More on this can be found in Section 6.)
- In the context of artificial intelligence, they have been studied as Valued Constraint Satisfaction problems [37] with constraints of bounded arity.

Our primary focus in this paper is on solving submodular VCSP instances efficiently. (Note that we are considering only *exact* algorithms. See [7] for approximation algorithms for the Max-CSP, which is a special case of the VCSP,

and [15] for approximation algorithms for the SFM.) However, as submodular VCSPs are equivalent to SFM_b, and SFM_b has been studied in several other areas of computer science, we will use techniques from a number of different fields. In particular, we will use techniques from pseudo-Boolean optimisation to identify new classes of submodular VCSPs which can be solved efficiently.

Our results have direct consequences for the other formalisms and frameworks in which the SFM_b problem has been studied. For example, many of the problems that arise in computer vision can be expressed in terms of energy minimisation [28]. The problem of energy minimisation is NP-hard in general, and therefore a lot of research has been devoted to identifying instances which can be solved more efficiently. Kolmogorov and Zabih identified classes of instances for which the energy minimisation problem can be solved efficiently [28], and which are applicable to a wide variety of vision problems, including image restoration, stereo vision and motion tracking, image synthesis, image segmentation, multi-camera scene reconstruction and medical imaging. The so-called *regularity* condition, which specifies the efficiently solvable classes in [28], is equivalent to submodularity. We discuss this application further in Section 6.

Cohen et al. have shown that VCSPs with submodular constraints over an arbitrary finite domain can be reduced to the SFM problem over a special family of sets known as a ring family [8]. This problem is equivalent to the general SFM problem [41], thus giving an algorithm of order $O(n^6 + n^5 L)$, where L is the look-up time (needed to evaluate the cost of an assignment to all variables), for any VCSP instance with n variables and with submodular constraints. This tractability result has since been generalised to a wider class of valued constraints over arbitrary finite domains known as tournament-pair constraints [6]. An alternative approach to solving VCSP instances with submodular constraints, based on linear programming, can be found in [10].

A general algorithm for SFM can always be used for the more restricted SFM_b, but the special features of this more restricted problem sometimes allow more efficient special-purpose algorithms to be used. Hence some classes of SFM_b are known to be solvable more efficiently than the general SFM, see [3] for a survey.

In this paper we focus on submodular constraints over a Boolean domain $\{0, 1\}$, which correspond precisely to submodular set functions [8]. We describe an algorithm based on graph cuts which can be used to solve certain VCSPs with submodular constraints over a Boolean domain much more efficiently than the general case. Some of our results are closely related to known efficient cases of SFM_b, and other previous results from different areas of computer science, but we present them here in a unified and constraints-based framework which allows us to make the proofs more consistent and often simpler. Moreover, we identify novel classes of VCSP instances with submodular constraints of arbitrary arities that can be solved efficiently.

The paper is organised as follows. In Section 2, we define the VCSP framework and submodular constraints, and note that submodular constraints over a Boolean domain can be represented by polynomials. In Section 3, we show that the standard (s, t)-Min-Cut problem can be expressed in the VCSP framework with a restricted constraint language Γ_{cut}, and that any instance of $\mathrm{VCSP}(\Gamma_{\mathsf{cut}})$ is solvable in cubic time. Moreover, we show that Γ_{cut} can express all *binary* submodular constraints. In Section 4, we reestablish known results that any instance of the VCSP with constraints whose corresponding polynomials

have only negative coefficients for terms of degree ≥ 2 can be expressed in VCSP(Γ_{cut}). We show the same for all $\{0,1\}$-valued submodular constraints, and also for all ternary submodular constraints. In Section 5, we present a necessary condition for a quartic polynomial to be submodular. Moreover, for every $k \geq 4$, we identify new classes of k-ary submodular constraints which can be expressed over VCSP(Γ_{cut}), and thus solved efficiently. Section 6 relates our results to applications in computer vision. Finally, in Section 7, we summarise our results and discuss related work.

2 Background

2.1 Valued constraint satisfaction and expressibility

In this section we define the VALUED CONSTRAINT SATISFACTION PROBLEM (VCSP). In the original definition of this problem, given in [40], costs were allowed to lie in any positive totally ordered monoid called a *valuation structure*. For our purposes, it is sufficient to consider costs which lie in the set $\overline{\mathbb{Q}}_+$ consisting of all non-negative rational numbers together with infinity[1].

Given a fixed set D, a function from D^k to $\overline{\mathbb{Q}}_+$ will be called a *cost function*. If the range of ϕ is $\{0,\infty\}$, then ϕ is called a *crisp* cost function. Note that crisp cost functions correspond precisely to relations, so we shall use these terms interchangeably. If the range of ϕ lies entirely within $\mathbb{Q}_+$, the set of non-negative rationals, then ϕ is called a *finite-valued* cost function.

Definition 2.1. An instance $\mathcal{P}$ of VCSP is a triple $\langle V, D, \mathcal{C}\rangle$, where V is a finite set of *variables*, which are to be assigned values from the set D, and $\mathcal{C}$ is a set of *valued constraints*. Each $c \in \mathcal{C}$ is a pair $c = \langle \sigma, \phi\rangle$, where σ is a tuple of variables of length $|\sigma|$, called the *scope* of c, and $\phi : D^{|\sigma|} \to \overline{\mathbb{Q}}_+$ is a cost function. An *assignment* for the instance $\mathcal{P}$ is a mapping s from V to D. The *cost* of an assignment s is defined as follows:

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

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

Any set Γ of cost functions is called a *valued constraint language*. The class VCSP(Γ) is defined to be the class of all VCSP instances where the cost functions of all valued constraints lie in Γ.

Example 2.2. Let $D = \{0,1\}$. We define two unary cost functions as follows:

$$\mu_2(x) \;=\; \begin{cases} 0 & \text{if } x = 0, \\ 5 & \text{if } x = 1, \end{cases}$$

$$\mu_5(x) \;=\; \begin{cases} 4 & \text{if } x = 0, \\ 2 & \text{if } x = 1. \end{cases}$$

We also define six binary cost functions by the following table:

[1] See [9] for a discussion of why limiting ourselves to the $\overline{\mathbb{Q}}_+$ valuation structure is not a severe restriction.

	ϕ_{12}	ϕ_{14}	ϕ_{23}	ϕ_{34}	ϕ_{35}	ϕ_{45}
00	3	0	0	9	3	4
01	2	4	1	7	5	3
10	3	2	0	8	4	2
11	1	5	0	1	4	1

The set $\Gamma = \{\mu_2, \mu_5, \phi_{12}, \phi_{14}, \phi_{23}, \phi_{34}, \phi_{35}, \phi_{45}\}$ is an example of a valued constraint language. We will now give an example of a VCSP(Γ) instance. Let $V = \{x_1, x_2, x_3, x_4, x_5\}$ be a set of variables, and let $\mathcal{C}$ be a set of constraints, defined as:

$$\mathcal{C} = \{\langle\langle x_1, x_2\rangle, \phi_{12}\rangle, \langle\langle x_1, x_4\rangle, \phi_{14}\rangle, \langle\langle x_2, x_3\rangle, \phi_{23}\rangle, \\ \langle\langle x_3, x_4\rangle, \phi_{34}\rangle, \langle\langle x_3, x_5\rangle, \phi_{35}\rangle, \langle\langle x_4, x_5\rangle, \phi_{45}\rangle, \langle\langle x_2\rangle, \mu_2\rangle, \langle\langle x_5\rangle, \mu_5\rangle\}.$$

Then $\mathcal{P} = \langle V, D, \mathcal{C}\rangle$ is a VCSP(Γ) instance, illustrated in Figure 1.

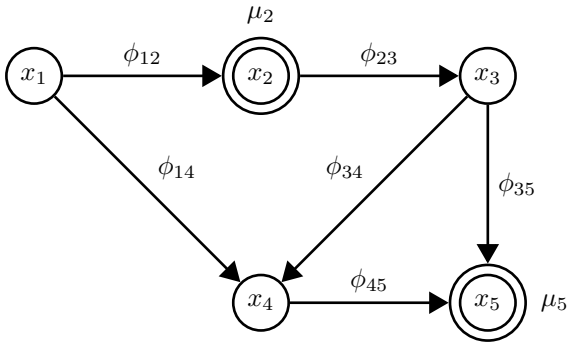

Figure 1: The instance $\mathcal{P}$ from Example 2.2.

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

Definition 2.3 ([8])**.** For any VCSP instance $\mathcal{I} = \langle V, D, \mathcal{C}\rangle$, and any list of variables of $\mathcal{I}$, $l = \langle v_1, \ldots, v_m\rangle$, the *projection* of $\mathcal{I}$ onto l, denoted $\pi_l(\mathcal{I})$, is the m-ary cost function defined as follows:

$$\pi_l(\mathcal{I})(x_1, \ldots, x_m) \;=\; \min_{\{s:V\to D \mid \langle s(v_1),\ldots,s(v_m)\rangle = \langle x_1,\ldots,x_m\rangle\}} \mathrm{Cost}_{\mathcal{I}}(s).$$

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

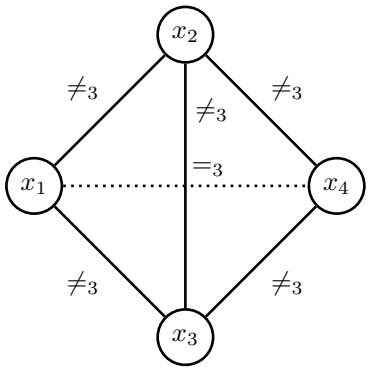

Figure 2: The gadget expressing $=_3$ over $\{\neq_3\}$, from Example 2.4.

Note that in the special case of relations (crisp cost functions) this notion of expressibility corresponds to the standard notion of expressibility using conjunction and existential quantification (*primitive positive formulas*) [4].

We denote by $\langle\Gamma\rangle$ the *expressive power* of Γ which is the set of all cost functions expressible over Γ up to additive and multiplicative constants in $\mathbb{Q}_+$.

Example 2.4. Let D be a finite set of size d. Consider a valued constraint language $\Gamma = \{\neq_d\}$ over D which consists of a binary disequality relation, $\neq_d$, given by

$$\neq_d = \{\langle a, b\rangle \in D^2 \mid a \neq b\}.$$

Consider an instance $\mathcal{P} = \{V, D, \mathcal{C}\}$ of VCSP(Γ), where $V = \{x_1, \ldots, x_{n+1}\}$, $n = d$, and

$$\mathcal{C} = \{\langle\langle x_i, x_j\rangle, \neq_d\rangle \mid i \neq j \in \{1, \ldots, n\}\} \cup \{\langle\langle x_i, x_{n+1}\rangle, \neq_d\rangle \mid i \in \{2, \ldots, n\}\}.$$

In order to satisfy all the constraints from $\mathcal{C}$, variables $x_1, \ldots, x_n$ have to be assigned different values. Moreover, the value of the variable x_{n+1} has to be different from the values of the variables $x_2, \ldots, x_n$. Hence, the only remaining value that can be assigned to the variable x_{n+1} is the value which is assigned to the variable x_1. Therefore, every solution s to $\mathcal{P}$ with minimum total cost (in this case zero) satisfies $s(x_1) = s(x_{n+1})$. Therefore, $\langle\mathcal{P}, \{x_1, x_{n+1}\}\rangle$ is a gadget for the equality relation, $=_d$, given by

$$=_d = \{\langle a, b\rangle \in D^2 \mid a = b\}.$$

In other words, the equality relation can be expressed using the disequality relation. An example of this construction for $|D| = 3$ is shown in Figure 2.

Example 2.5. Consider the VCSP instance $\mathcal{P}$ from Example 2.2. The projection of $\mathcal{P}$ onto $\langle x_2, x_4\rangle$, denoted by $\pi(\mathcal{P})_{\langle x_2, x_4\rangle}$ is a binary cost function defined by minimising over the remaining variables. The following table, which enumerates all assignments s in which x_2 and x_4 are both assigned 0, together with the cost of these assignments, shows that $\pi(\mathcal{P})_{\langle x_2, x_4\rangle}(0, 0) = 21$.

x_2	x_4	x_1	x_3	x_5	$Cost_{\mathcal{P}}(s)$
0	0	0	0	0	23
0	0	0	0	1	22
0	0	0	1	0	24
0	0	**0**	**1**	**1**	**21**
0	0	1	0	0	25
0	0	1	0	1	24
0	0	1	1	0	26
0	0	1	1	1	23

Similarly, it is straightforward to check that

$$\pi(\mathcal{P})_{\langle x_2,x_4\rangle}(x,y) = \begin{cases} 21 & \text{if } x=0 \text{ and } y=0, \\ 16 & \text{if } x=0 \text{ and } y=1, \\ 24 & \text{if } x=1 \text{ and } y=0, \\ 19 & \text{if } x=1 \text{ and } y=1. \end{cases}$$

Hence this cost function can be expressed over the valued constraint language Γ defined in Example 2.2.

Example 2.6. Consider a ternary finite-valued cost function ϕ over $D = \{0,1,2\}$ whose value on any input is the square of the number of zeros in that input. We show a gadget for expressing ϕ using only binary crisp cost functions and finite-valued unary cost functions.

Define three binary crisp cost functions as follows:

$$\phi_0(x,y) = \begin{cases} \infty & \text{if } x=0 \text{ and } y=1, \\ \infty & \text{if } x=0 \text{ and } y=2, \\ 0 & \text{otherwise,} \end{cases}$$

$$\phi_1(x,y) = \begin{cases} \infty & \text{if } x=0 \text{ and } y=1, \\ 0 & \text{otherwise,} \end{cases}$$

and

$$\phi_2(x,y) = \begin{cases} \infty & \text{if } x=0 \text{ and } y=2, \\ 0 & \text{otherwise.} \end{cases}$$

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

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

Let $\mathcal{P} = \langle V, D, \mathcal{C}\rangle$ where $V = \{x, y, z, u_1, u_2, u_3, v_1, v_2, v_3, v_4, v_5, v_6, w\}$ and the set of constraints $\mathcal{C}$ is shown in Figure 3.

We claim that $\langle \mathcal{P}, \langle x, y, z\rangle\rangle$ is a gadget for expressing ϕ.

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

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

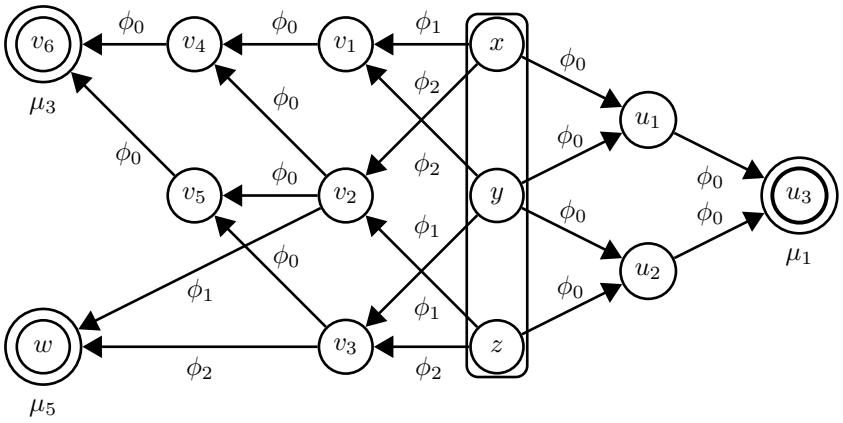

Figure 3: The gadget expressing $\phi = (\#0)^2$ from Example 2.6.

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

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

Note that a similar gadget can be constructed for bigger domains.

2.2 Submodular functions and polynomials

A function $\psi : 2^V \to \mathbb{Q}$ defined on subsets of a set V is called a *submodular function* [32] if, for all subsets S and T of V, $\psi(S \cap T) + \psi(S \cup T) \leq \psi(S) + \psi(T)$. The problem of Submodular Function Minimisation (SFM) consists in finding a subset S of V for which the value of $\psi(S)$ is minimal.

For any lattice-ordered set D, a cost function $\phi : D^k \to \overline{\mathbb{Q}}_+$ is called *submodular* if for every $u, v \in D^k$, $\phi(\min(u, v)) + \phi(\max(u, v)) \leq \phi(u) + \phi(v)$ where both min and max are applied coordinate-wise on tuples u and v. Note that expressibility preserves submodularity [8]: if every $\phi \in \Gamma$ is submodular, and $\phi' \in \langle\Gamma\rangle$, then ϕ' is also submodular.

In this paper, we restrict our attention to *finite-valued* submodular cost functions. (We discuss general submodular cost functions in Section 7.)

We also focus on problems over Boolean domains. We denote by $\Gamma_{\mathsf{sub},k}$ the set of all finite-valued submodular cost functions of arity at most k on a Boolean domain $D = \{0, 1\}$, and we set $\Gamma_{\mathsf{sub}} = \bigcup_k \Gamma_{\mathsf{sub},k}$. We will show below that $\mathrm{VCSP}(\Gamma_{\mathsf{sub},2})$ can be solved in cubic time, and hence we will be concerned with what other cost functions are expressible over $\Gamma_{\mathsf{sub},2}$, and so can also be solved efficiently.

Example 2.7. Consider the valued constraint language Γ from Example 2.2. It is straightforward to check that all of the cost functions in Γ are submodular: unary cost functions are submodular by definition; each binary $\phi \in \Gamma$ satisfies $\phi(0, 0) + \phi(1, 1) \leq \phi(0, 1) + \phi(1, 0)$. Hence the instance $\mathcal{P}$ from Example 2.2 is an instance of $\mathrm{VCSP}(\Gamma_{\mathsf{sub},2})$.

A cost function of arity k can be represented as a table of values of size D^k. Alternatively, a (finite-valued) cost function $\phi : D^k \to \mathbb{Q}_+$ on a Boolean domain $D = \{0, 1\}$ can be represented as a *polynomial* in k (Boolean) variables with coefficients from $\mathbb{Q}$ [3] (such functions are sometimes called *pseudo-Boolean functions*). Over a Boolean domain we have $x^2 = x$, so the degree of any variable in any term can be restricted to 0 or 1, and this polynomial representation is then unique. Hence, in what follows, we will often represent a finite-valued cost function on a Boolean domain by a polynomial.

Note that if Γ is a set of finite-valued cost functions on a Boolean domain, with arity at most k, then any instance of VCSP(Γ) with n variables can be uniquely represented as a polynomial p in n Boolean variables, of degree at most k. Conversely, any such polynomial represents an n-ary cost function which can be expressed over a set of cost functions on a Boolean domain, with arity at most k. As mentioned above, over a Boolean domain it holds that $x^2 = x$. Hence p has at most 2^n terms, which correspond to subsets of variables.

Example 2.8. A unary cost function ϕ on a Boolean domain can be expressed as the polynomial $p(x_1) = \phi(0) + (\phi(1) - \phi(0))x_1$. Similarly, a binary cost function ϕ can be expressed as

$$\begin{aligned} p(x_1, x_2) = \phi(0,0) \\ &+ (\phi(1,0) - \phi(0,0))x_1 \\ &+ (\phi(0,1) - \phi(0,0))x_2 \\ &+ (\phi(1,1) - \phi(0,1) - \phi(1,0) + \phi(0,0))x_1x_2. \end{aligned}$$

Consider the instance $\mathcal{P}$ from Example 2.2. The corresponding polynomial is

$$\begin{aligned} p(x_1, \ldots, x_5) = 3 + 0x_1 - x_2 - x_1x_2 \\ &+ 0 + 2x_1 + 4x_4 - x_1x_4 \\ &+ 0 + 0x_2 + x_3 - x_2x_3 \\ &+ 9 - x_3 - 2x_4 - 5x_3x_4 \\ &+ 3 + x_3 + 2x_5 - 2x_3x_5 \\ &+ 4 - 2x_4 - x_5 + 0x_4x_5 \\ &+ 0 + 5x_2 \\ &+ 4 - 2x_5, \end{aligned}$$

which can be simplified as

$$\begin{aligned} p(x_1, \ldots, x_5) = 23 + 2x_1 + 4x_2 + x_3 - x_5 \\ - x_1x_2 - x_1x_4 - x_2x_3 - 5x_3x_4 - 2x_3x_5. \end{aligned}$$

A general construction for finding the polynomial representation is given in [3].

For polynomials over Boolean variables there is a standard way to define *derivatives* of each order (see [3]). For example, the second-order derivative of a polynomial p, with respect to the first two indices, denoted $\delta_{12}(\mathbf{x})$, is defined as $p(1, 1, \mathbf{x}) - p(1, 0, \mathbf{x}) - p(0, 1, \mathbf{x}) + p(0, 0, \mathbf{x})$. Derivatives for other pairs of indices are defined analogously.

Proposition 2.9 ([3])**.** *A polynomial $p(x_1, \ldots, x_n)$ over Boolean variables $x_1, \ldots, x_n$ represents a submodular cost function if, and only if, its second-order derivatives $\delta_{ij}(\mathbf{x})$ are non-positive for all $1 \leq i < j \leq n$ and all $\mathbf{x} \in D^{n-2}$.*

Corollary 2.10. *A* quadratic *polynomial* $a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{1 \le i < j \le n} a_{ij} x_i x_j$ *over Boolean variables* $x_1, \ldots, x_k$*, represents a submodular cost function if, and only if,* $a_{ij} \le 0$ *for every* $1 \le i < j \le n$*.*

Example 2.11. Notice that the polynomial p from Example 2.8 has all quadratic coefficients non-positive, and hence it is submodular. This confirms that the instance $\mathcal{P}$ from Example 2.2 is an instance of $\mathrm{VCSP}(\Gamma_{\mathsf{sub},2})$, as shown in Example 2.7.

3 Binary submodular constraints

In this section we show that a constraint language Γ_{cut}, consisting of certain simple binary and unary cost functions over a Boolean domain, has cubic time complexity. We also show that Γ_{cut} can express any binary submodular cost function over a Boolean domain, that is, $\Gamma_{\mathsf{sub},2} \subseteq \langle \Gamma_{\mathsf{cut}} \rangle$. It follows that any instance of $\mathrm{VCSP}(\Gamma_{\mathsf{sub},2})$ can also be solved in cubic time.

For any $w \in \overline{\mathbb{Q}}_+$, we define the binary cost function χ^w as follows:

$$\chi^w(x,y) \;=\; \begin{cases} w & \text{if } x = 0 \text{ and } y = 1, \\ 0 & \text{otherwise.} \end{cases}$$

For any $d \in D$ and $c \in \overline{\mathbb{Q}}_+$, we define the unary cost function μ_d^c as follows:

$$\mu_d^c \;=\; \begin{cases} c & \text{if } x \neq d, \\ 0 & \text{if } x = d. \end{cases}$$

It is straightforward to check that all functions χ^w and μ_d^c are submodular.

We define the constraint language Γ_{cut} to be the set of all cost functions χ^w and μ_d^c over a Boolean domain, for $c, w \in \overline{\mathbb{Q}}_+$ and $d \in \{0,1\}$.

Theorem 3.1. *The problems* (s,t)-Min-Cut *and* $\mathrm{VCSP}(\Gamma_{\mathsf{cut}})$ *are linear-time equivalent.*

Proof. Consider any instance of (s,t)-Min-Cut with (directed) graph $G = \langle V, E \rangle$ and weight function $w : E \to \overline{\mathbb{Q}}_+$. Define a corresponding instance $\mathcal{I}$ of $\mathrm{VCSP}(\Gamma_{\mathsf{cut}})$ as follows:

$$\mathcal{I} = \langle V, \{0,1\}, \{\langle \langle i,j \rangle, \chi^{w(i,j)} \rangle \mid \langle i,j \rangle \in E\} \cup \{\langle s, \mu_0^\infty \rangle, \langle t, \mu_1^\infty \rangle\} \rangle.$$

Note that in any solution to $\mathcal{I}$ the source and target nodes, s and t, must take the values 0 and 1, respectively. Moreover, the weight of any cut containing s and not containing t is equal to the cost of the corresponding assignment to $\mathcal{I}$. Hence we have shown that (s,t)-Min-Cut can be reduced to $\mathrm{VCSP}(\Gamma_{\mathsf{cut}})$ in linear time.

On the other hand, given an instance $\mathcal{I} = \langle V, D, \mathcal{C} \rangle$ of $\mathrm{VCSP}(\Gamma_{\mathsf{cut}})$, construct a graph on $V \cup \{s,t\}$ as follows: any unary constraint on variable v with cost function μ_0^c (respectively μ_1^c) is represented by an edge of weight c from the source node s to node v (respectively, from node v to the target node t). Any binary constraint on variables $\langle v_1, v_2 \rangle$ with cost function χ^w is represented by an edge of weight w from node v_1 to v_2. It is straightforward to check that a solution to $\mathcal{I}$ corresponds to a minimum (s,t)-cut of this graph. □

Corollary 3.2. VCSP(Γ_{cut}) *can be solved in* $O(n^3)$ *time, where* n *is the number of variables.*

Proof. By Theorem 3.1, VCSP(Γ_{cut}) has the same time complexity as (s,t)-Min-Cut, which is known to be solvable in cubic time [19]. □

Using a standard reduction [21], see also [3], we now show that all *binary* submodular cost functions over a Boolean domain can be expressed over Γ_{cut}.

Theorem 3.3. $\Gamma_{\mathsf{sub},2} \subseteq \langle \Gamma_{\mathsf{cut}} \rangle$.

Proof. By Corollary 2.10, any cost function from $\Gamma_{\mathsf{sub},2}$ can be represented by a quadratic Boolean polynomial $p(x_1, x_2) = a_0 + a_1 x_1 + a_2 x_2 + a_{12} x_1 x_2$ where $a_{12} \leq 0$. This can then be re-written as

$$p(x_1, x_2) = a'_0 + \sum_{i \in P} a'_i x_i + \sum_{j \in N} a'_j (1 - x_j) + a'_{12}(1 - x_1) x_2,$$

where $P \cap N = \emptyset$, $P \cup N = \{1, 2\}$, $a'_{12} = -a_{12}$, and $a'_i, a'_j, a'_{12} \geq 0$. (This is known as the *posiform* representation [3].)

Hence p can be expressed over Γ_{cut} (up to the constant a'_0) by the gadget $\langle \mathcal{I}, \langle x_1, x_2 \rangle \rangle$, where $\mathcal{I}$ is the instance $\langle \{x_1, x_2, s, t\}, \{0, 1\}, \mathcal{C} \rangle$ of VCSP(Γ_{cut}) and

$$\mathcal{C} = \{ \langle \langle s, x_i \rangle, \chi^{a'_i} \rangle \mid i \in P \} \cup \{ \langle \langle x_j, t \rangle, \chi^{a'_j} \rangle \mid j \in N \} \\ \cup \{ \langle \langle s \rangle, \mu_0^\infty \rangle, \langle \langle t \rangle, \mu_1^\infty \rangle, \langle \langle x_1, x_2 \rangle, \chi^{a'_{12}} \rangle \}.$$

□

Corollary 3.4. VCSP($\Gamma_{\mathsf{sub},2}$) *can be solved in* $O(n^3)$ *time, where* n *is the number of variables.*

Proof. By Theorem 3.3, any instance of VCSP($\Gamma_{\mathsf{sub},2}$) can be reduced to VCSP(Γ_{cut}) in linear time by replacing each constraint with a suitable gadget of fixed size. The result then follows from Corollary 3.2. (Note that we can use the same vertices s and t for all constraints.) □

Example 3.5. Consider the polynomial p from Example 2.8,

$$p(x_1, \ldots, x_5) = 23 + 2x_1 + 4x_2 + x_3 - x_5 \\ - x_1 x_2 - x_1 x_4 - x_2 x_3 - 5 x_3 x_4 - 2 x_3 x_5,$$

which represents the instance $\mathcal{P}$ from Example 2.2. We can rewrite p as in the proof of Theorem 3.1 as follows:

$$\begin{aligned} p(x_1, \ldots, x_5) &= 23 + 2x_1 + 4x_2 + x_3 - x_5 \\ &\quad + (1 - x_1) x_2 - x_2 + (1 - x_1) x_4 - x_4 + (1 - x_2) x_3 - x_3 \\ &\quad + 5(1 - x_3) x_4 - 5 x_4 + 2(1 - x_3) x_5 - 2 x_5 \\ &= 23 + 2x_1 + 3x_2 - 6x_4 - 3x_5 \\ &\quad + (1 - x_1) x_2 + (1 - x_1) x_4 + (1 - x_2) x_3 + 5(1 - x_3) x_4 + 2(1 - x_3) x_5 \\ &= 14 + 2x_1 + 3x_2 + 6(1 - x_4) + 3(1 - x_5) \\ &\quad + (1 - x_1) x_2 + (1 - x_1) x_4 + (1 - x_2) x_3 + 5(1 - x_3) x_4 + 2(1 - x_3) x_5. \end{aligned}$$

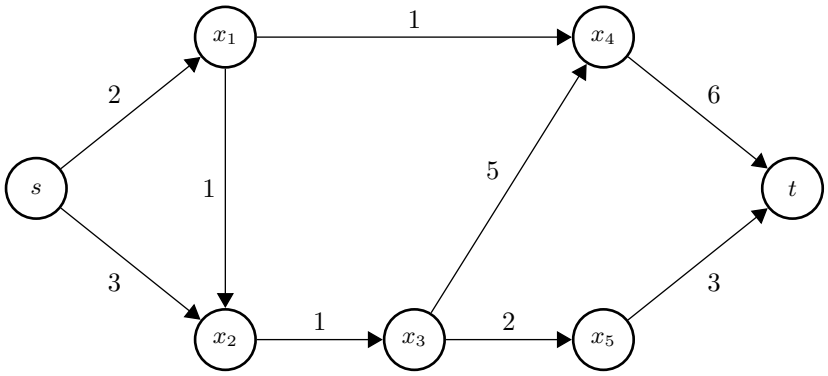

Figure 4: Graph G corresponding to polynomial p from Example 2.8

We can now build a graph G with 5 vertices corresponding to variables x_1 through x_5 and two extra vertices s and t and add edges accordingly, see Figure 4.

Now for every assignment o of values 0 and 1 to variables x_1 through x_5, $p(o(x_1), \ldots, o(x_5))$ is equal to the size of the (s,t)-cut in G given by o plus 14 (for the constant term in the posiform representation of p). The minimum cut in G, with value 2, is the set $\{s, x_1, x_2\}$. Therefore, the assignment $x_1 = x_2 = 0$ and $x_3 = x_4 = x_5 = 1$ minimises the polynomial p with the total value 16.

4 Known classes of expressible constraints

In this section, using results from pseudo-Boolean optimisation, we extend the results from Section 3 to three further classes of constraints over a Boolean domain: submodular constraints whose corresponding polynomials have negative coefficients for all terms of degree ≥ 2; $\{0,1\}$-valued submodular constraints; and ternary submodular constraints. We show that the cost functions for these three classes of submodular constraints can all be expressed over $\Gamma_{\mathsf{sub},2}$, and hence can be minimised in cubic time in the number of variables plus the number of higher-order (non-binary) constraints.

Define $\Gamma_{\mathsf{neg},k}$ to be the set of all cost functions over a Boolean domain, of arity at most k, whose corresponding polynomials have negative coefficients for all terms of degree greater than or equal to 2. It is not hard to show that these cost functions, sometimes called *negative-positive*, are submodular as any second-order derivative of the corresponding polynomial contains only terms with negative coefficients. Set $\Gamma_{\mathsf{neg}} = \bigcup_k \Gamma_{\mathsf{neg},k}$. The minimisation of cost functions chosen from Γ_{neg} using min-cuts was first studied in [36].

Theorem 4.1 ([36])**.** $\Gamma_{\mathsf{neg}} \subseteq \langle \Gamma_{\mathsf{sub},2} \rangle$.

Proof. Consider the following polynomial:

$$p_0(x_1, \ldots, x_n) = \min_{y \in \{0,1\}} \{-y + y \sum_{i \in A} (1 - x_i)\}.$$

It is straightforward to check that for a given $A \subseteq \{1, \ldots, n\}$, $p_0(\mathbf{x}) = -1$ if $A \subseteq \mathbf{x}$ and $p_0(\mathbf{x}) = 0$ otherwise (where $A \subseteq \mathbf{x}$ means $\forall i \in A, x_i = 1$).

Given any polynomial p representing a cost function from Γ_{neg}, we can replace each term $-ax_{i_1} \ldots x_{i_k}$ of p of degree k, where $a > 0$, with the gadget given above for p_0 and get a quadratic polynomial with negative quadratic coefficients, introducing a new variable y for every term of degree ≥ 3. Such a quadratic polynomial can be expressed over $\Gamma_{\mathsf{sub},2}$, by Corollary 2.10. □

Corollary 4.2. *For any fixed k, $\mathrm{VCSP}(\Gamma_{\mathsf{neg},k})$ can be solved in $O((n+r)^3)$ time, where n is the number of variables and r is the number of constraints of arity 3 or greater.*

Proof. By Theorem 4.1, any instance of $\mathrm{VCSP}(\Gamma_{\mathsf{neg},k})$ can be reduced to $\mathrm{VCSP}(\Gamma_{\mathsf{sub},2})$ in linear time by replacing each constraint with a suitable gadget. For any fixed k, the number of terms in the corresponding polynomial, and hence the number of new variables introduced by these gadgets is bounded by a constant. The result then follows from Corollary 3.4. □

Next we consider the class of submodular constraints over a Boolean domain which take only the cost values 0 and 1. (Such constraints can be used to model optimisation problems such as Max-CSP, see [7].) Define $\Gamma_{\{0,1\},\mathsf{k}}$ to be the set of all $\{0,1\}$-valued submodular cost functions over a Boolean domain, of arity at most k, and set $\Gamma_{\{0,1\}} = \bigcup_k \Gamma_{\{0,1\},\mathsf{k}}$. The minimisation of submodular cost functions from $\Gamma_{\{0,1\}}$ was studied in [12], where they were called *2-monotone* functions. The equivalence of 2-monotone and submodular cost functions and a generalisation of 2-monotone functions to non-Boolean domains was shown in [7].

Definition 4.3. A cost function ϕ is called *2-monotone* if there exist two sets $A, B \subseteq \{1, \ldots, n\}$ such that $\phi(\mathbf{x}) = 0$ if $A \subseteq \mathbf{x}$ or $\mathbf{x} \subseteq B$ and $\phi(\mathbf{x}) = 1$ otherwise (where $A \subseteq \mathbf{x}$ means $\forall i \in A, x_i = 1$ and $\mathbf{x} \subseteq B$ means $\forall i \notin B, x_i = 0$).

Theorem 4.4 ([12])**.** $\Gamma_{\{0,1\}} \subseteq \langle \Gamma_{\mathsf{sub},2} \rangle$.

Proof. Any 2-monotone cost function ϕ can be expressed over $\Gamma_{\mathsf{sub},2}$ using 2 extra variables, y_1, y_2:

$$\phi(\mathbf{x}) = \min_{y_1, y_2 \in \{0,1\}} \{(1-y_1)y_2 + y_1 \sum_{i \in A} (1-x_i) + (1-y_2) \sum_{i \notin B} x_i\}.$$

□

Corollary 4.5. $\mathrm{VCSP}(\Gamma_{\{0,1\}})$ *can be solved in $O((n+r)^3)$ time, where n is the number of variables and r is the number of constraints of arity 3 or greater.*

Finally, we consider the class $\Gamma_{\mathsf{sub},3}$ of ternary submodular cost functions over a Boolean domain. This class was studied in [1], from where we obtain the following useful characterisation of cubic submodular polynomials.

Lemma 4.6 ([1]). *A cubic polynomial $p(x_1, \ldots, x_n)$ over Boolean variables represents a submodular cost function if, and only if, it can be written as*

$$p(x_1, \ldots, x_n) = a_0 + \sum_{\{i\} \in C_1^+} a_i x_i - \sum_{\{i\} \in C_1^-} a_i x_i - \sum_{\{i,j\} \in C_2} a_{ij} x_i x_j + \sum_{\{i,j,k\} \in C_3^+} a_{ijk} x_i x_j x_k - \sum_{\{i,j,k\} \in C_3^-} a_{ijk} x_i x_j x_k,$$

where C_2 denotes the set of quadratic terms, and C_i^+ (C_i^-) denotes the set of terms of degree i with positive (negative) coefficients, for $i = 1, 3$, and

1. $a_i, a_{ij}, a_{ijk} \geq 0$ ($\{i\} \in C_1^+ \cup C_1^-, \{i,j\} \in C_2, \{i,j,k\} \in C_3^+ \cup C_3^-$),

2. $\forall \{i,j\} \in C_2$, $a_{ij} + \sum_{k | \{i,j,k\} \in C_3^+} a_{ijk} \leq 0$.

Theorem 4.7 ([1]). $\Gamma_{\mathsf{sub},3} \subseteq \langle \Gamma_{\mathsf{sub},2} \rangle$.

Proof. Let p be a polynomial representing an arbitrary cost function in $\Gamma_{\mathsf{sub},3}$. By Lemma 4.6, the quadratic terms in p are non-positive. We already know how to express a negative cubic term using a gadget over $\Gamma_{\mathsf{sub},2}$ (Theorem 4.1). To express a positive cubic term, consider the following identity:

$$x_i x_j x_k - x_i x_j - x_i x_k - x_j x_k = \min_{y \in \{0,1\}} \{(1 - x_i - x_j - x_k) y\}.$$

We can replace a positive cubic term $a_{ijk} x_i x_j x_k$ in p with

$$\min_{y \in \{0,1\}} \{a_{ijk}(1 - x_i - x_j - x_k) y + a_{ijk}(x_i x_j + x_i x_k + x_j x_k)\}.$$

It remains to check that all quadratic coefficients of the resulting polynomial are non-positive. However, this is ensured by the second condition from Lemma 4.6. □

Corollary 4.8. $\mathrm{VCSP}(\Gamma_{\mathsf{sub},3})$ *can be solved in $O((n+r)^3)$ time, where n is the number of variables and r is the number of ternary constraints .*

We remark that the proof of Corollary 4.8 given in [1] was obtained using a different approach based on the so-called conflict graphs of a supermodular polynomial (see [3]). Such graphs have been shown to be bipartite, and therefore the problem of finding a maximum weight stable set can be reduced to a flow problem. However, the resulting time complexity is the same.

5 New classes of expressible constraints

In this section we investigate the question of which submodular constraints of arity 4 or higher can be expressed by binary submodular constraints. We derive a necessary condition for a 4-ary constraint over a Boolean domain to be submodular. We also present some sufficient conditions, which give rise to new classes of submodular constraints which can be expressed over $\Gamma_{\mathsf{sub},2}$, and hence minimised efficiently. We prove the sufficient conditions first for 4-ary submodular cost functions and then generalise them to k-ary submodular cost functions for every $k \geq 4$.

5.1 4-ary constraints

One might hope to obtain a simple characterisation of 4-ary submodular cost functions over a Boolean domain similar to Lemma 4.6. However, it has been shown that testing whether a given quartic Boolean polynomial is submodular is co-NP-complete [11, 17]. Hence, one is unlikely to find a polynomial-time checkable characterisation, as this would prove that P=NP. However, we obtain the following necessary condition.

Lemma 5.1. *If a quartic polynomial $p(x_1, \ldots, x_n)$ over Boolean variables represents a submodular cost function, then it can be written such that, for all $\{i, j\} \in C_2$:*

1. $a_{ij} \leq 0$, *and*
2. $a_{ij} + \sum_{k|\{i,j,k\}\in C_3^+} a_{ijk} + \sum_{k,l|\{i,j,k,l\}\in C_4^+} a_{ijkl} + F_{ij} \leq 0$, *where*

$$F_{ij} = \sum_{k|\{i,j,k\}\in C_3^- \wedge \{i,j,k,.\}\in C_4^+} a_{ijk} + \sum_{k,l|\{i,j,k,l\}\in C_4^- \wedge \{i,j,k,.\},\{i,j,l,.\}\in C_4^+} a_{ijkl},$$

C_2 denotes the set of quadratic terms, and C_i^+ (C_i^-) denotes the set of terms of degree i with positive (negative) coefficients, for $i = 3, 4$.

Proof. Let p be a quartic submodular polynomial and let i and j be given, then $\delta_{ij}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n)$, the second-order derivative of p with respect to the ith and jth variable, is equal to

$$\delta_{ij} = a_{ij} + \sum_{k|\{i,j,k\}\in C_3^+} a_{ijk}x_k + \sum_{k,l|\{i,j,k,l\}\in C_4^+} a_{ijkl}x_k x_l - \sum_{k|\{i,j,k\}\in C_3^-} a_{ijk}x_k - \sum_{k,l|\{i,j,k,l\}\in C_4^-} a_{ijkl}x_k x_l.$$

Consider an assignment which sets $x_k = 1$ if $k = i$ or $k = j$, and $x_k = 0$ otherwise. By Proposition 2.9, $a_{ij} \leq 0$, which proves the first condition. By setting $x_k = 1$ for all k such that $\{i, j, k\} \in C_3^+$ and $x_k = x_l = 1$ for all k, l such that $\{i, j, k, l\} \in C_4^+$, we get the second condition. We set to 1 all variables which occur in some positive cubic or quartic term. The second condition then says that the sum of all these positive coefficients minus those which are forced, by our setting of variables, to be 1 (F_{ij}), is at most 0. (Note that this also proves Lemma 4.6.) □

Next we show a useful example of a 4-ary submodular cost function which can be expressed over the binary submodular cost functions using one extra variable.

Example 5.2. Let ϕ be the 4-ary cost function defined as follows: $\phi(\mathbf{x}) = \min\{2k, 5\}$, where k is the number of 0s in $\mathbf{x} \in \{0, 1\}^4$. The corresponding quartic polynomial representing ϕ is

$$p(x_1, x_2, x_3, x_4) = 5 + x_1x_2x_3x_4 - x_1x_2 - x_1x_3 - x_1x_4 - x_2x_3 - x_2x_4 - x_3x_4.$$

By considering second-order derivatives of p, it can be checked that p is submodular. For instance, $\delta_{12}(x_3, x_4)$, the second-order derivative of p with respect

to the first two variables, is equal to $x_1x_2x_3x_4 - 1$. Clearly, $\delta_{12}(x_3, x_4) \leq 0$. It can be shown by simple case analysis that p cannot be expressed as a quadratic polynomial with non-positive quadratic coefficients (from the definition of p, the polynomial would have to be $5 - x_1x_2 - x_1x_3 - x_1x_4 - x_2x_3 - x_2x_4 - x_3x_4$ which is not equal to p on $x_1 = x_2 = x_3 = x_4 = 1$).

However, p can be expressed over $\Gamma_{\mathsf{sub},2}$ using just one extra variable, via the following gadget:

$$p(x_1, x_2, x_3, x_4) = \min_{y \in \{0,1\}} \{5 + (3 - 2x_1 - 2x_2 - 2x_3 - 2x_4)y\}.$$

Using the same notation as in Lemma 5.1, define $\Gamma_{\mathsf{suff},4}$ to be the set of all 4-ary submodular cost functions over a Boolean domain whose corresponding quartic polynomials satisfy, for every $i < j$,

$$a_{ij} + \sum_{k \mid \{i,j,k\} \in C_3^+} a_{ijk} + \sum_{k,l \mid \{i,j,k,l\} \in C_4^+} a_{ijkl} \quad \leq \quad 0. \tag{1}$$

Theorem 5.3. $\Gamma_{\mathsf{suff},4} \subseteq \langle \Gamma_{\mathsf{sub},2} \rangle$.

Proof. Let $\phi \in \Gamma_{\mathsf{suff},4}$ and let p be the corresponding polynomial which represents ϕ. First, replace all negative cubic and quartic terms using the construction in Theorem 4.1. As in the proof of Theorem 4.7, replace every positive cubic term $a_{ijk}x_ix_jx_k$ in p with

$$\min_{y \in \{0,1\}} \{a_{ijk}(1 - x_i - x_j - x_k)y + a_{ijk}(x_ix_j + x_ix_k + x_jx_k)\}.$$

Using the same construction as in Example 5.2, replace every positive quartic term $a_{ijkl}x_ix_jx_kx_l$ with

$$\min_{y \in \{0,1\}} \{a_{ijkl}(3 - 2x_i - 2x_j - 2x_k - 2x_l)y + a_{ijkl}(x_ix_j + x_ix_k + x_ix_l + x_jx_k + x_jx_l + x_kx_l)\}.$$

It only remains to check that all quadratic coefficients in the resulting polynomial are non-positive. However, this is ensured by the definition of $\Gamma_{\mathsf{suff},4}$ and by the choice of the gadgets. □

Corollary 5.4. VCSP($\Gamma_{\mathsf{suff},4}$) *can be solved in* $O((n + r)^3)$ *time, where* n *is the number of variables and* r *is the number of constraints of arity* 3 *or greater.*

5.2 The general case

We now generalise the result from the previous section to subclasses of submodular constraints of arbitrary arities. For every $k \geq 4$, we define $\Gamma_{\mathsf{suff},k}$ to be the set of all k-ary submodular cost functions over a Boolean domain whose corresponding polynomials satisfy, for every $1 \leq i < j \leq k$,

$$a_{ij} + \sum_{s=1}^{k-2} \sum_{\{i,j,i_1,\ldots,i_s\} \in C_{s+2}^+} a_{i,j,i_1,\ldots,i_s} \quad \leq \quad 0,$$

where C_i^+ denotes the set of terms of degree i with positive coefficients. In other words, for any $1 \leq i < j \leq k$, the sum of a_{ij} and all positive coefficients of cubic and higher-degree terms which include x_i and x_j is non-positive.

We then set $\Gamma_{\mathsf{suff}} = \bigcup_k \Gamma_{\mathsf{suff},k}$.

Theorem 5.5. $\Gamma_{\mathsf{suff}} \subseteq \langle \Gamma_{\mathsf{sub},2} \rangle$.

Proof. First we show how to uniformly generate gadgets over $\Gamma_{\mathsf{sub},2}$ for polynomials of the following type:

$$p_k(x_1, \ldots, x_k) = \prod_{i=1}^{k} x_i - \sum_{1 \leq i < j \leq k} x_i x_j.$$

Note that $p_k(\mathbf{x}) = -\binom{m}{2}$, where m is the number of 1s in $\mathbf{x}$, and $\binom{0}{2} = \binom{1}{2} = 0$, unless $m = k$ ($\mathbf{x}$ consists of 1s only), in which case $p_k(\mathbf{x}) = -\binom{m}{2} + 1$.

We claim, that for any $k \geq 4$, the following, denoted by $\mathcal{P}_k$, is a gadget for p_k:

$$p_k(x_1, \ldots, x_k) = \min_{y_0, \ldots, y_{k-4} \in \{0,1\}} \{y_0(3 - 2\sum_{i=1}^{k} x_i) + \sum_{j=1}^{k-4} y_j(2 + j - \sum_{i=1}^{k} x_i)\}.$$

Notice that in the case of $k = 4$, the gadget corresponds to the gadget used in the proof of Theorem 5.3, and therefore the base case is proved. We proceed by induction on k. Assume that $\mathcal{P}_i$ is a gadget for p_i for every $i \leq k$. We prove that $\mathcal{P}_{k+1}$ is a gadget for p_{k+1}.

Firstly, take the gadget $\mathcal{P}_k$ for p_k, and replace every sum $\sum_{i=1}^{k} x_i$ with $\sum_{i=1}^{k+1} x_i$. We denote the new gadget $\mathcal{P}'$. By the inductive hypothesis, it is not difficult to see that $\mathcal{P}'$ is a valid gadget for p_{k+1} on all assignments with at most $k-1$ 1s. Also, on any assignment with exactly k 1s, $\mathcal{P}'$ returns $-\binom{k}{2} + 1$. On the assignment with $k+1$ 1s, $\mathcal{P}'$ returns: $-\binom{k}{2} + 1 - 2 - 1(k-4)$. This can be simplified as follows: $-\binom{k}{2} + 1 - 2 - k + 4 = -\binom{k}{2} + 1 - k + 2 = -(\binom{k}{2} + \binom{k}{1}) + 1 + 2 = -\binom{k+1}{2} + 1 + 2$. Hence $\mathcal{P}'$ is *almost* a gadget for p_{k+1}: we only need to subtract 1 on an assignment which has exactly k 1s, and subtract 2 on the assignment consisting of 1s only. But this is exactly what $\min_{y_{k-3} \in \{0,1\}} \{y_{k-3}(2 + (k-3) - \sum_{i=1}^{k+1} x_i)\}$ does. Therefore, we have established that $\mathcal{P}_{k+1}$ is a gadget for p_{k+1} over $\Gamma_{\mathsf{sub},2}$ with $k-3$ extra variables.

Given a cost function $\phi \in \Gamma_{\mathsf{suff},k}$, let p be the corresponding polynomial which represents ϕ. By the construction in Theorem 4.1, we can replace all negative terms of degree ≥ 3. By the constructions in Theorem 4.7 and Theorem 5.3, we can replace all positive cubic and quartic terms. Now for any positive term of degree d, $5 \leq d \leq k$, we replace the term with the gadget $\mathcal{P}_d$ and add $\sum_{1 \leq i < j \leq k} x_i x_j$ back in. This construction works if all quadratic coefficients of the resulting polynomial are non-positive. However, this is ensured by the definition of $\Gamma_{\mathsf{suff},k}$ and by the choice of the gadgets. □

Corollary 5.6. *For any fixed $k \geq 4$, $\mathrm{VCSP}(\Gamma_{\mathsf{suff},k})$ can be solved in $O((n+r)^3)$ time, where n is the number of variables and r is the number of constraints of arity 3 or greater.*

6 Applications to Computer Vision

In this section we relate our results to certain optimisation problems arising in computer vision. In fact, certain optimisation problems studied in computer vision are equivalent to problems described in the VCSP framework.

In computer vision, many problems can be naturally formulated in terms of energy minimisation where the energy function, over a set of variables $\{x_v\}_{v \in V}$, has the following form:

$$E(\mathbf{x}) \;=\; c_0 + \sum_{v \in V} c_v(x_v) + \sum_{\langle u,v \rangle \in V \times V} c_{uv}(x_u, x_v) + \dots$$

Set V usually corresponds to pixels, x_v denotes the label of pixel $v \in V$ which must belong to a finite domain D. The constant term of the energy is c_0, the unary terms $c_v(\cdot)$ encode data penalty functions, the pairwise terms $c_{uv}(\cdot,\cdot)$ are interaction potentials, and so on. Functions of arity 3 and above are known as higher-order energy functions, or higher-order cliques. This energy is often derived in the context of Markov Random Fields (also Conditional Random Fields) [18]: a minimum of E corresponds to a *maximum a-posteriori* (MAP) labelling $\mathbf{x}$ [30].

As discussed in Section 2.2, there is a direct translation between polynomials over Boolean variables and VCSP instances. Hence it is clear that this framework of energy minimisation is equivalent to VCSP. (See [43] for a survey on the connection between computer vision and constraint satisfaction problems, although with a strong emphasis on a linear programming approach.) Therefore, for energy minimisation over Boolean variables we get the following:

Corollary 6.1 (of Theorem 5.5)**.** *Energy minimisation, where the energy function can be expressed as a sum of functions from $\Gamma_{\mathsf{suff},k}$, for some fixed k, is solvable in $O((n+r)^3)$ time, where n is the number of variables (pixels), and r is the number of functions in this sum of arity 3 or greater.*

Corollary 6.1 generalises results of Kolmogorov and Zabih, who provided alternative proofs of results on $\Gamma_{\mathsf{sub},3}$ [28], and results of Freedman and Drineas, who provided alternative proofs of results on $\Gamma_{\mathsf{sub},3}$ and Γ_{neg} [16]. Corollary 6.1 provides a strictly larger class of higher-order energy functions which can be minimised efficiently.

Higher-order energy functions have the ability to encode high level structural dependencies between pixels, which have been shown to be extremely powerful for image labeling problems. They have long been used to model image textures [29, 34, 38], image denoising and restoration [38], and texture segmentation [26]. Their use, however, is severely hampered in practice by the intractable complexity of representing and minimising such functions [39]. Our results enlarge the class of higher-order energy functions which can be (exactly) minimised efficiently using graphs cuts. Hence functions from $\Gamma_{\mathsf{suff},k}$ could be used, for instance, in image processing for efficient recognition of images or Bayesian estimation.

Many applications in computer vision deal with non-Boolean domains. It is clear that any variable over a non-Boolean domain $D = \{0, 1, \dots, d-1\}$ of size d can be encoded by $d-1$ Boolean variables. This process is known as *Booleanisation*. One such encoding is the following: $en(i) = 0^{d-i-1}1^i$. We replace each

variable with $d-1$ new Boolean variables and impose a (submodular) relation on these new variables which ensures that they only take values in the range of the encoding function en. Note that $en(\max(a,b)) = \max(en(a), en(b))$ and $en(\min(a,b)) = \min(en(a), en(b))$, so this encoding preserves submodularity.

Is is easy to observe that in any submodularity-preserving encoding of a non-Boolean variable by Boolean variables each variable over a d-element domain needs to be replaced with at least d variables. However, for practical purposes, subclasses of non-Boolean submodular functions which can be encoded by Boolean submodular functions with fewer variables have been studied [35] as well as approximation algorithms for these problems [27]. Moreover, certain higher-order functions contained in $\Gamma_{\mathsf{suff},k}$ have been used for the single view reconstruction problem in [35].

7 Conclusion and related work

Conclusion In this paper we first considered binary submodular constraints over a Boolean domain, and showed that they can be minimised in cubic time via a reduction to the minimum cut problem for graphs. We then investigated which other submodular constraints are *expressible* using binary submodular constraints over a Boolean domain, and hence can also be minimised efficiently using minimum cuts.

Using known results from combinatorial optimisation, we identified several such classes of constraints, including all ternary submodular constraints, and all $\{0,1\}$-valued submodular constraints of any arity. By constructing suitable gadgets, we identified certain new classes of k-ary submodular constraints, where $k \geq 4$, which can also be expressed by binary submodular constraints.

Using results from [8] and [42], it can be shown that any (general) submodular cost function ϕ can be expressed as the sum of a finite-valued submodular cost function ϕ_{fin}, and a submodular relation ϕ_{rel}, that is, $\phi = \phi_{fin} + \phi_{rel}$. Moreover, it is known that all submodular *relations* are binary decomposable [24], and hence expressible using only binary submodular relations.[2] Hence, our expressibility results for certain finite-valued submodular cost functions can be combined with expressibility results for crisp submodular cost functions to obtain expressibility results for general submodular cost functions taking both finite and infinite values.

Related work In the proof of Theorem 5.5, we proved that, for any $k \geq 4$ and $\phi \in \Gamma_{\mathsf{suff},k}$, ϕ can be expressed over $\Gamma_{\mathsf{sub},2}$ with $k-3$ extra variables per term. Independently of our work, Zalesky has shown that ϕ can be expressed over $\Gamma_{\mathsf{sub},2}$ with $\lfloor \frac{k-1}{2} \rfloor$ extra variables per term (see the manuscript [44]). This result yields asymptotically the same cubic time complexity for $\mathrm{VCSP}(\Gamma_{\mathsf{suff},k})$.

The class Γ_{suff} is very general; for example, it properly extends the class Γ_{neg}. However, our next example shows that it does not include all submodular functions which are expressible by binary submodular functions.

[2] If a relation R is submodular, then R admits both MIN and MAX as polymorphisms. This implies that R also admits MEDIAN as a polymorphism. As MEDIAN is a ternary near unanimity operation, it follows from [24] that R is binary decomposable.

Example 7.1. Define a 4-ary submodular cost function ϕ as follows: $\phi(\mathbf{x}) = \min(3k, 7) + 2y + z$, where k is the number of 0s in $\mathbf{x} \in \{0,1\}^4$, $y = 1$ if $\mathbf{x} = \langle 1,1,1,0\rangle$ (and 0 otherwise), and $z = 1$ if $\mathbf{x} = \langle 1,1,0,0\rangle$ (and 0 otherwise). The corresponding polynomial representing ϕ is

$$p(x_1,x_2,x_3,x_4) = 7 + 2x_1x_2x_3x_4 - 2x_1x_2x_4 - x_1x_3x_4 - x_2x_3x_4 \\ - x_1x_3 - x_1x_4 - x_2x_3 - x_2x_4 - x_3x_4.$$

By considering the second-order derivatives of p, it can easily be checked that ϕ is submodular. However, $\phi \notin \Gamma_{\mathsf{suff},4}$: for $i = 1$ and $j = 2$, the expression in Equation 1 on page 16 gives 2. Hence Theorem 5.3 does not apply to ϕ.

By a case analysis (system of equations), it can be shown that ϕ cannot be expressed over $\Gamma_{\mathsf{sub},2}$ without extra variables or with just one extra variable. However, the following gadget shows that ϕ is in fact expressible over $\Gamma_{\mathsf{sub},2}$ using just two extra variables:

$$p(x_1,x_2,x_3,x_4) = 7 - x_1x_4 - x_2x_4 - x_3x_4 \\ + \min_{y_1,y_2\in\{0,1\}} \{2y_1 + 3y_2 - y_1y_2 - y_1(x_1 + x_2 + 2x_3) - y_2(x_1 + x_2 + 2x_4)\}.$$

The main open problem in this area when we started this investigation was whether *all* bounded-arity submodular constraints over a Boolean domain can be expressed by binary submodular constraints, and hence solved in cubic time. In terms of polynomials, this is equivalent to the following problem: can any polynomial over Boolean variables with non-positive second-order derivatives be expressed as the projection of a quadratic polynomial with non-positive quadratic coefficients?

In recent work, we have shown that the answer to this question is negative; that is, there are submodular constraints which *cannot* be expressed using binary submodular constraints [45]. In fact, we have obtained a precise classification of all 4-ary submodular constraints with respect to this question [45].

Open problems There are still several interesting open questions regarding expressibility. We denote by $\langle\Gamma_{\mathsf{sub},2}\rangle^m$ the set of all (submodular) cost functions expressible over $\Gamma_{\mathsf{sub},2}$ with at most m extra variables. Clearly, $\langle\Gamma_{\mathsf{sub},2}\rangle^m \subseteq \langle\Gamma_{\mathsf{sub},2}\rangle^{m+1}$ for every $m \geq 0$. We have seen in Section 5 that $\langle\Gamma_{\mathsf{sub},2}\rangle^1$ is strictly larger than $\langle\Gamma_{\mathsf{sub},2}\rangle^0$ (see Example 5.2). In other words, allowing a single hidden variable strictly increases the expressive power of $\Gamma_{\mathsf{sub},2}$.

Also, as mentioned in Example 7.1, we know that $\langle\Gamma_{\mathsf{sub},2}\rangle^1 \subsetneq \langle\Gamma_{\mathsf{sub},2}\rangle^2$. On the other hand, we do not know whether in general allowing further hidden variables increases the expressive power any further. In other words, it is an open question whether $\langle\Gamma_{\mathsf{sub},2}\rangle^m \subsetneq \langle\Gamma_{\mathsf{sub},2}\rangle^{m+1}$ for any $m \geq 2$. However, we do know that there is a limit to the additional expressive power that can be gained by allowing an arbitrary number of hidden variables. This is a consequence of the following result, which is a general result about expressibility, and not specific to submodular constraints or Boolean domains.

Proposition 7.2 ([5])**.** *If a cost function $\phi : D^k \to \overline{\mathbb{Q}}_+$ is expressible over Γ, then ϕ is expressible over Γ using at most $|D|^{|D|^k}$ hidden variables.*

Hence, if a k-ary Boolean submodular constraint is expressible over $\Gamma_{\mathsf{sub},2}$, then it is also expressible over $\langle\Gamma_{\mathsf{sub},2}\rangle^m$, where $m = 2^{2^k}$. In the case of 4-ary

submodular constraints, we know that any 4-ary submodular cost function ϕ is either expressible with fewer than m variables, or not expressible at all [45]. Hence the theoretical upper bound 2^{16} on the number of extra variables is not obtained. We do not know whether the upper bound on the number of extra variables can be improved, or whether it is necessary for certain submodular constraints.

Acknowledgements The authors would like to thank Dave Cohen and Martin Cooper for fruitful discussions on submodular constraints, Chris Jefferson for help with using the constraint-solver MINION,[3] which was useful for simplifying some of the gadgets presented in this paper, and the anonymous reviewers for useful comments on an earlier version of this paper. Stanislav Živný gratefully acknowledges the support of EPSRC grant EP/F01161X/1.

References

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

[2] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, G. Verfaillie, Semiring-based CSPs and Valued CSPs: Frameworks, Properties, and Comparison, Constraints 4 (1999) 199–240.

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

[4] A. A. Bulatov, A. A. Krokhin, P. Jeavons, Classifying the Complexity of Constraints using Finite Algebras, SIAM Journal on Computing 34 (3) (2005) 720–742.

[5] D. Cohen, M. Cooper, P. Jeavons, An Algebraic Characterisation of Complexity for Valued Constraints, in: Proceedings of the 12th International Conference on Principles and Practice of Contraint Programming (CP), vol. 4204 of Lecture Notes in Computer Science, Springer, 2006.

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

[7] D. Cohen, M. Cooper, P. Jeavons, A. Krokhin, Supermodular Functions and the Complexity of MAX-CSP, Discrete Applied Mathematics 149 (2005) 53–72.

[8] D. Cohen, M. Cooper, P. Jeavons, A. Krokhin, The Complexity of Soft Constraint Satisfaction, Artificial Intelligence 170 (2006) 983–1016.

[9] M. Cooper, High-order Consistency in Valued Constraint Satisfaction, Constraints 10 (2005) 283–305.

[3] Available from: `http://minion.sourceforge.net/`

[10] M. C. Cooper, Minimization of Locally Defined Submodular Functions by Optimal Soft Arc Consistency, Constraints 13 (4) (2008) 437–458.

[11] Y. Crama, Recognition problems for special classes of polynomials in 0-1 variables, Mathematical Programming 44 (1989) 139–155.

[12] N. Creignou, S. Khanna, M. Sudan, Complexity Classification of Boolean Constraint Satisfaction Problems, vol. 7 of SIAM Monographs on Discrete Mathematics and Applications, SIAM, 2001.

[13] R. Dechter, Constraint Processing, Morgan Kaufmann, 2003.

[14] V. Deineko, P. Jonsson, M. Klasson, A. Krokhin, The approximability of Max CSP with fixed-value constraints, Journal of the ACM 55 (4).

[15] U. Feige, V. S. Mirrokni, J. Vondrák, Maximizing non-monotone submodular functions, in: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07), IEEE Computer Society, 2007.

[16] D. Freedman, P. Drineas, Energy Minimization via Graph Cuts: Settling What is Possible, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, 2005.

[17] G. Gallo, B. Simeone, On the supermodular knapsack problem, Mathematical Programming 45 (1988) 295–309.

[18] S. Geman, D. Geman, Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images, IEEE Transactions on Pattern Analysis and Machine Intelligence 6 (1984) 721–741.

[19] A. Goldberg, R. Tarjan, A New Approach to the Maximum Flow Problem, Journal of the ACM 35 (1988) 921–940.

[20] G. Gutin, A. Rafiey, A. Yeo, M. Tso, Level of Repair Analysis and Minimum Cost Homomorphisms of Graphs, Discrete Applied Mathematics 154 (2006) 881–889.

[21] P. L. Hammer, Some network flow problems solved with pseudo-Boolean programming, Operations Research 13 (1965) 388–399.

[22] S. Iwata, Submodular Function Minimization, Mathematical Programming 112 (2008) 45–64.

[23] S. Iwata, J. B. Orlin, A Simple Combinatorial Algorithm for Submodular Function Minimization, Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2009.

[24] P. Jeavons, D. Cohen, M. Cooper, Constraints, Consistency and Closure, Artificial Intelligence 101 (1–2) (1998) 251–265.

[25] P. Jonsson, M. Klasson, A. Krokhin, The Approximability of Three-valued MAX CSP, SIAM Journal on Computing 35 (6) (2006) 1329–1349.

[26] P. Kohli, M. P. Kumar, P. H. S. Torr, P3 & Beyond: Solving Energies with Higher Order Cliques, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, 2007.

[27] P. Kohli, L. Ladický, P. Torr, Robust Higher Order Potentials for Enforcing Label Consistency, International Journal of Computer Vision 82 (3) (2009) 302–324.

[28] V. Kolmogorov, R. Zabih, What Energy Functions Can Be Minimized via Graph Cuts?, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (2) (2004) 147–159.

[29] X. Lan, S. Roth, D. P. Huttenlocher, M. J. Black, Efficient Belief Propagation with Learned Higher-Order Markov Random Fields, in: Proceedings of the 9th European Conference on Computer Vision (ECCV), Part II, vol. 3952 of Lecture Notes in Computer Science, Springer, 2006.

[30] S. L. Lauritzen, Graphical Models, Oxford University Press, 1996.

[31] U. Montanari, Networks of Constraints: Fundamental properties and applications to picture processing, Information Sciences 7 (1974) 95–132.

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

[33] J. B. Orlin, A faster strongly polynomial time algorithm for submodular function minimization., Mathematical Programming 118 (2009) 237–251.

[34] R. Paget, I. D. Longstaff, Texture synthesis via a noncausal nonparametric multiscale Markov random field, IEEE Transactions on Image Processing 7 (6) (1998) 925–931.

[35] S. Ramalingam, P. Kohli, K. Alahari, P. Torr, Exact Inference in Multi-label CRFs with Higher Order Cliques, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, 2008.

[36] J. Rhys, A selection problem of shared fixed costs and network flows, Management Science 17 (3) (1970) 200–207.

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

[38] S. Roth, M. J. Black, Fields of experts: A framework for learning image priors, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, 2005.

[39] C. Rother, P. Kohli, W. Feng, J. Jia, Minimizing Sparse Higher Order Energy Functions of Discrete Variables, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, 2009.

[40] T. Schiex, H. Fargier, G. Verfaillie, Valued Constraint Satisfaction Problems: Hard and Easy Problems, in: Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI), 1995.

[41] A. Schrijver, A Combinatorial Algorithm Minimizing Submodular Functions in Strongly Polynomial Time, Journal of Combinatorial Theory, Series B 80 (2000) 346–355.

[42] A. Schrijver, Combinatorial Optimization: Polyhedra and Efficiency, vol. 24 of Algorithms and Combinatorics, Springer, 2003.

[43] T. Werner, A Linear Programming Approach to Max-Sum Problem: A Review, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (7) (2007) 1165–1179.

[44] B. Zalesky, Efficient Determination of Gibbs Estimators with Submodular Energy Functions, arXiv:math/0304041v1 (2003), version February 2008.

[45] S. Živný, D. A. Cohen, P. G. Jeavons, The Expressive Power of Binary Submodular Functions, Discrete Applied Mathematics, 2009. 157 (15) (2009) 3347–3358.