The Expressive Power of Binary Submodular Functions[☆]

Stanislav Živný[*,a], David A. Cohen[b], Peter G. Jeavons[a]

[a] *Computing Laboratory, University of Oxford, UK*
[b] *Dept. of Computer Science, Royal Holloway, University of London, UK*

Abstract

We investigate whether all Boolean submodular functions can be decomposed into a sum of binary submodular functions over a possibly larger set of variables. This question has been considered in several different contexts in computer science, including computer vision, artificial intelligence, and pseudo-Boolean optimisation. Using a connection between the expressive power of valued constraints and certain algebraic properties of functions, we answer this question negatively.

Our results have several corollaries. First, we characterise precisely which submodular polynomials of arity 4 can be expressed by binary submodular polynomials. Next, we identify a novel class of submodular functions of arbitrary arities which can be expressed by binary submodular functions, and therefore minimised efficiently using a so-called expressibility reduction to the Min-Cut problem. More importantly, our results imply limitations on this kind of reduction and establish for the first time that it cannot be used in general to minimise arbitrary submodular functions. Finally, we refute a conjecture of Promislow and Young on the structure of the extreme rays of the cone of Boolean submodular functions.

Key words: Decomposition of submodular functions, Min-Cut, Pseudo-Boolean optimisation, Submodular function minimisation.

---

1. Introduction

1.1. Background

A function $f : 2^V \to \mathbb{R}$ is called *submodular* if for all $S, T \subseteq V$,

$$f(S \cap T) + f(S \cup T) \;\le\; f(S) + f(T).$$

[☆] A preliminary version of this paper appeared in *Proceedings of the 34th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, 2009.

[*] Corresponding author. Address: Computing Laboratory, University of Oxford, Wolfson Building, Parks Road, Oxford OX1 3QD, UK. Tel: +44 (0)1865 273884

Email addresses: `stanislav.zivny@comlab.ox.ac.uk` (Stanislav Živný), `dave@cs.rhul.ac.uk` (David A. Cohen), `peter.jeavons@comlab.ox.ac.uk` (Peter G. Jeavons)

Submodular functions are a key concept in operational research and combinatorial optimisation [18, 27, 34, 39, 40, 50, 51]. Examples include cut capacity functions, matroid rank functions, and entropy functions. Submodular functions are often considered to be a discrete analogue of convex functions [37].

Both minimising and maximising submodular functions, possibly under some additional conditions, have been considered extensively in the literature. Submodular function maximisation is easily shown to be NP-hard [50] since it generalises many standard NP-hard problems such as the maximum cut problem. In contrast, the problem of *minimising* a submodular function (SFM) can be solved efficiently with only polynomially many oracle calls, using the ellipsoid algorithm [21, 22], or by using one of several combinatorial algorithms that have been obtained in the last decade [25, 26, 27, 28, 29, 42, 49]. The time complexity of the fastest known general algorithm for SFM is $O(n^6 + n^5L)$, where n is the number of variables and L is the time required to evaluate the function [42].

The minimisation of submodular functions on sets is equivalent to the minimisation of submodular functions on distributive lattices [50]. Krokhin and Larose have also studied the more general problem of minimising submodular functions on non-distributive lattices [35].

An important and well-studied sub-problem of SFM is the minimisation of submodular functions of bounded arity (SFM$_b$), also known as *locally defined* submodular functions [12], or submodular functions with *succinct representation* [16]. 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 of this kind occur in a wide variety of contexts:

- In the context of pseudo-Boolean optimisation, such problems involve the minimisation of Boolean polynomials of bounded *degree* [4].
- In the context of artificial intelligence, they have been studied as *valued constraint satisfaction problems* (VCSP) [47], also known as *soft* or *weighted* constraint satisfaction problems.
- In the context of computer vision, such problems are often formulated as *Gibbs energy minimisation* problems or *Markov Random Fields* (also known as *Conditional Random Fields*) [36, 52].

We will present our results primarily in the language of pseudo-Boolean optimisation. Hence an instance of SFM$_b$ with n variables will be represented as a polynomial in n Boolean variables, of some fixed bounded degree.

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. (Note that we are focusing on *exact* algorithms which find an optimal solution.) In particular, it has been shown that certain cases can be solved much more efficiently by reducing to the Min-Cut problem; that is, the problem of finding a minimum cut in a directed graph which includes a given source vertex and excludes a given target vertex. For example, it has been known since 1965 that the minimisation of *quadratic* submodular polynomials is equivalent to finding a minimum cut in a corresponding directed graph [4, 24]. Hence quadratic submodular polynomials can be minimised in $O(n^3)$ time, where n is the number of variables.

A Boolean polynomial in at most 2 variables has degree at most 2, so any *sum* of binary Boolean polynomials has degree at most 2; in other words, it is quadratic. It follows that an efficient algorithm, based on reduction to Min-Cut, can be used to minimise any class of functions that can be written as a sum of binary submodular polynomials. We will say that a polynomial that can be written in this way, perhaps with additional variables to be minimised over, is *expressible* by binary submodular polynomials (see Section 2). The following classes of functions have all been shown to be expressible by binary submodular polynomials in this way[1], over the past four decades:

- polynomials where all terms of degree 2 or more have negative coefficients (also known as *negative-positive* polynomials) [45];
- cubic submodular polynomials [2];
- $\{0, 1\}$-valued submodular functions (also known as 2-monotone functions) [8, 14];
- a class recently found by Živný and Jeavons [55] and independently by Zalesky [54].

All these classes of functions have been shown to be expressible by binary submodular polynomials and hence minimisable in cubic time (in the total number of variables). Moreover, several classes of submodular functions over non-Boolean domains have also been shown to be expressible by binary submodular functions and hence minimisable in cubic time [6, 7, 8].

This series of positive expressibility results naturally raises the following question:

Question 1. *Are* all *submodular polynomials expressible by binary submodular polynomials, over a possibly larger set of variables?*

Each of the above expressibility results was obtained by an ad-hoc construction, and no general technique[2] has previously been proposed which is sufficiently powerful to address Question 1.

[1]In fact, it is known that *all* Boolean polynomials (of arbitrary degree) are expressible by binary polynomials [4, 46], but the general construction does not preserve submodularity; that is, the resulting binary polynomials are not necessarily submodular.

[2]For example, standard combinatorial counting techniques cannot resolve this question because we allow arbitrary real-valued coefficients in submodular polynomials. We also allow an arbitrary number of additional variables.

1.2. Contributions

Cohen et al. recently developed a novel algebraic approach to characterising the expressive power of valued constraints in terms of certain algebraic properties of those constraints [9].

Using this systematic algebraic approach we are able to give a negative answer to Question 1: we show that there exist submodular polynomials of degree 4 that cannot be expressed by binary submodular polynomials. More precisely, we characterise exactly which submodular polynomials of arity 4 are expressible by binary submodular polynomials and which are not.

On the way to establishing these results we show that two broad families of submodular functions, known as *upper fans* and *lower fans*, are all expressible by binary submodular functions. This provides a new class of submodular polynomials of all arities which are expressible by binary submodular polynomials and hence solvable efficiently by reduction to MIN-CUT. We use the expressibility of this family, and the existence of non-expressible functions, to refute a conjecture from [43] on the structure of the extreme rays of the cone of Boolean submodular functions, and suggest a more refined conjecture of our own.

1.3. Applications

The concept of submodularity is important in a wide variety of fields within computer science; in this paper we briefly discuss two of these: artificial intelligence and computer vision. Our results can be directly applied to both of these areas, as we show in Section 4 below.

Artificial Intelligence. A major area of investigation in artificial intelligence is the *Constraint Satisfaction* problem (CSP) [47]. A number of extensions have been added to the basic CSP framework to deal with questions of optimisation, including semi-ring CSPs, valued CSPs, soft CSPs and weighted CSPs. These extended frameworks can be used to model a wide range of discrete optimisation problems [3, 47, 48], including standard problems such as MIN-CUT, MAX-SAT, MAX-ONES SAT, MAX-CSP [11, 14], and MIN-COST HOMOMORPHISM [23].

The differences between the various frameworks are not relevant for our purposes, so we will simply focus on one very general framework, the valued constraint satisfaction problem or VCSP. Informally, in the VCSP framework, an instance consists of a set of variables, a set of possible values for those variables, and a set of constraints. Each constraint has an associated cost function which assigns a cost (or 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.

The class of constraints with submodular cost functions is the only nontrivial tractable class of optimisation problems in the dichotomy classification of the Boolean VCSP [11], and the only tractable class in the dichotomy classification of the MAX-CSP problem for both 3-element sets [31] and arbitrary finite sets allowing constant (that is, fixed-value) constraints [15].

Cohen et al. showed that VCSP instances with submodular constraints over an arbitrary finite domain can be reduced to SFM [11], and hence can be solved in polynomial time. This tractability result has since been generalised to a wider class of valued constraints over arbitrary finite domains known as tournament-pair constraints [10]. An alternative approach to solving VCSP instances with bounded-arity submodular constraints, based on linear programming, can be found in [12].

Computer Vision. Gibbs energy minimisation, Markov Random Fields and Conditional Random Fields play an important role in computer vision as they 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 [33]. Reducing energy minimisation to the Min-Cut problem has recently become a very popular approach, leading to the rediscovery of the property of submodularity [17, 33], and showing that certain special classes of functions can be minimised using graph cuts by introducing extra variables [32, 44].

Our results below characterise precisely which 4-ary submodular functions can be minimised using graph cuts in this way and which cannot. We also identify a very broad new class of submodular functions of arbitrary arity which can be minimised efficiently in this way.

2. Preliminaries

In this section, we introduce the basic definitions and the main tools used throughout the paper.

2.1. Cost functions and expressibility

We denote by $\overline{\mathbb{R}}$ the set of all real numbers together with (positive) infinity. For any fixed set D, a function ϕ from D^n to $\overline{\mathbb{R}}$ will be called a *cost function* on D of arity n. If the range of ϕ lies entirely within $\mathbb{R}$, then ϕ is called a *finite-valued* cost function. If the range of ϕ is $\{0, \infty\}$, then ϕ can be viewed as a predicate, or *relation*, allowing just those tuples $t \in D^n$ for which $\phi(t) = 0$.

Cost functions can be added and multiplied by arbitrary real values, hence for any given set of cost functions, Γ, we define the convex cone generated by Γ, as follows.

Definition 2. For any set of cost functions Γ, the *cone generated by* Γ, denoted $\mathrm{Cone}(\Gamma)$, is defined by:

$$\mathrm{Cone}(\Gamma) = \{\alpha_1\phi_1 + \cdots + \alpha_r\phi_r \mid r \geq 1;\ \phi_1, \ldots, \phi_r \in \Gamma;\ \alpha_1, \ldots, \alpha_r \geq 0\}.$$

Definition 3. A cost function ϕ of arity n is said to be *expressible* by a set of cost functions Γ if $\phi = \min_{y_1,\ldots,y_j} \phi'(x_1, \ldots, x_n, y_1, \ldots, y_j) + \kappa$, for some $\phi' \in \mathrm{Cone}(\Gamma)$ and some constant κ.

The variables $y_1, \ldots, y_j$ are called *extra* (or *hidden*) variables, and ϕ' is called a *gadget* for ϕ over Γ.

$$\begin{array}{ccccc}
\begin{array}{c} t_1 \\ t_2 \\ \vdots \\ t_k \end{array} & \begin{array}{cccc} t_1[1] & t_1[2] & \dots & t_1[n] \\ t_2[1] & t_2[2] & \dots & t_2[n] \\ & & \vdots & \\ t_k[1] & t_k[2] & \dots & t_k[n] \\ \hline \end{array} & \xrightarrow{\phi} & \left.\begin{array}{c} \phi(t_1) \\ \phi(t_2) \\ \vdots \\ \phi(t_k) \end{array}\right\} & \displaystyle\sum_{i=1}^{k} \phi(t_i) \\
 & & & & \geq \\
\begin{array}{c} t'_1 = f_1(t_1, \dots, t_k) \\ t'_2 = f_2(t_1, \dots, t_k) \\ \vdots \\ t'_k = f_k(t_1, \dots, t_k) \end{array} & \begin{array}{cccc} t'_1[1] & t'_1[2] & \dots & t'_1[n] \\ t'_2[1] & t'_2[2] & \dots & t'_2[n] \\ & & \vdots & \\ t'_k[1] & t'_k[2] & \dots & t'_k[n] \end{array} & \xrightarrow{\phi} & \left.\begin{array}{c} \phi(t'_1) \\ \phi(t'_2) \\ \vdots \\ \phi(t'_k) \end{array}\right\} & \displaystyle\sum_{i=1}^{k} \phi(t'_i)
\end{array}$$

Figure 1: Inequality establishing $\mathcal{F} = \langle f_1, \dots, f_k \rangle$ as a multimorphism of cost function ϕ (see Definition 4).

Note that in the special case of relations this notion of expressibility corresponds to the standard notion of expressibility using conjunction and existential quantification (*primitive positive formulas*) [5]. Note that the notion of expressibility has been a major tool in the complexity analysis of a wide variety of Boolean constraint satisfaction problems carried out by Creignou et al. [14], where it was referred to as *implementation*.

We denote by $\langle \Gamma \rangle$ the *expressive power* of Γ, which is the set of all cost functions expressible by Γ.

It was shown in [9] that the expressive power of a set of cost functions is characterised by certain algebraic properties of those cost functions called fractional polymorphisms. For the results of this paper, we will only need a certain subset of these algebraic properties, called *multimorphisms* [11]. These are defined in Definition 4 below (see also Figure 1).

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

Definition 4 ([11]). Let $\mathcal{F} : D^k \to D^k$ be the function whose k-tuple of output values is given by the tuple of functions $\mathcal{F} = \langle f_1, \dots, f_k \rangle$, where each $f_i : D^k \to D$.

For any n-ary cost function ϕ, we say that $\mathcal{F}$ is a k-ary *multimorphism* of ϕ if, for all $t_1, \dots, t_k \in D^n$,

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

For any set of cost functions, Γ, we will say that $\mathcal{F}$ is a multimorphism of Γ if $\mathcal{F}$ is a multimorphism of every cost function in Γ. The set of all multimorphisms of Γ will be denoted $\mathsf{Mul}(\Gamma)$.

Note that multimorphisms are preserved under expressibility. In other words, if $\mathcal{F} \in \mathsf{Mul}(\Gamma)$, and $\phi \in \langle\Gamma\rangle$, then $\mathcal{F} \in \mathsf{Mul}(\{\phi\})$ [9, 11]. This has two important corollaries. First, if $\langle\Gamma_1\rangle = \langle\Gamma_2\rangle$, then $\mathsf{Mul}(\Gamma_1) = \mathsf{Mul}(\Gamma_2)$. Second, if there exists $\mathcal{F} \in \mathsf{Mul}(\Gamma)$ such that $\mathcal{F} \notin \mathsf{Mul}(\{\phi\})$, then ϕ is not expressible over Γ, that is, $\phi \notin \langle\Gamma\rangle$.

2.2. Lattices and submodularity

Recall that L is a *lattice* if L is a partially ordered set in which every pair of elements (a, b) has a unique supremum and a unique infimum. For a finite lattice L and a pair of elements (a, b), we will denote the unique supremum of a and b by $a \vee b$, and the unique infimum of a and b by $a \wedge b$.

For any finite lattice-ordered set D, a cost function $\phi : D^n \to \overline{\mathbb{R}}$ is called *submodular* if for every $u, v \in D^n$, $\phi(u \wedge v) + \phi(u \vee v) \leq \phi(u) + \phi(v)$ where both $\wedge$ and $\vee$ are applied coordinate-wise on tuples u and v [40]. This standard definition can be reformulated very simply in terms of multimorphisms: ϕ is submodular if $\langle\wedge, \vee\rangle \in \mathsf{Mul}(\{\phi\})$.

Using results from [11] and [50], it can be shown that any submodular cost function ϕ can be expressed as the sum of a finite-valued submodular cost function ϕ_{fin}, and a submodular decomposable (that is, equal to the sum of their binary projections) [30], and hence expressible using only binary submodular relations. Therefore, when considering which cost functions are expressible by binary submodular cost functions, we can restrict our attention to *finite-valued* cost functions without any loss of generality.

Next we define some particular families of submodular cost functions, first described in [43], which will turn out to play a central role in our analysis.

Definition 5. Let L be a lattice. We define the following cost functions on L:

- For any set A of pairwise incomparable elements $\{a_1, \ldots, a_m\} \subseteq L$, such that each pair of distinct elements (a_i, a_j) has the same least upper bound, $\bigvee A$, the following cost function is called an *upper fan*:

$$\phi_A(x) = \begin{cases} -2 & \text{if } x \geq \bigvee A, \\ -1 & \text{if } x \ngeq \bigvee A, \text{ but } x \geq a_i \text{ for some } i, \\ 0 & \text{otherwise.} \end{cases}$$

- For any set B of pairwise incomparable elements $\{a_1, \ldots, a_m\} \subseteq L$, such that each pair of distinct elements (a_i, a_j) has the same greatest lower bound, $\bigwedge B$, the following cost function is called a *lower fan*:

$$\phi_B(x) = \begin{cases} -2 & \text{if } x \leq \bigwedge B, \\ -1 & \text{if } x \nleq \bigwedge B, \text{ but } x \leq a_i \text{ for some } i, \\ 0 & \text{otherwise.} \end{cases}$$

We call a cost function a *fan* if it is either an upper fan or a lower fan. Note that our definition of fans is slightly more general than the definition in [43]. In particular, we allow the set A to be empty, in which case the corresponding upper fan ϕ_A is a constant function. It is not hard to show that all fans are submodular [43].

2.3. Boolean cost functions and polynomials

In this paper we will focus on problems over Boolean domains, that is, where $D = \{0, 1\}$.

Any cost function of arity n can be represented as a table of values of size D^n. Moreover, a finite-valued cost function $\phi : D^n \to \mathbb{R}$ on a Boolean domain $D = \{0, 1\}$ can also be represented as a unique *polynomial* in n (Boolean) variables with coefficients from $\mathbb{R}$ (such functions are sometimes called *pseudo-Boolean functions* [4]). Hence, in what follows, we will often refer to a finite-valued cost function on a Boolean domain and its corresponding polynomial interchangeably.

For polynomials over Boolean variables there is a standard way to define *derivatives* of each order (see [4]). For example, the second-order derivative of a polynomial p, with respect to the first two indices, denoted $\delta_{1,2}(\mathbf{x})$, 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. It was shown in [41] that a polynomial $p(x_1, \dots, x_n)$ over Boolean variables $x_1, \dots, x_n$ represents a submodular cost function if, and only if, its second-order derivatives $\delta_{i,j}(\mathbf{x})$ are non-positive for all $1 \le i < j \le n$ and all $\mathbf{x} \in D^{n-2}$. An immediate corollary is that a quadratic polynomial represents a submodular cost function if, and only if, the coefficients of all quadratic terms are non-positive.

Note that a cost function is called *supermodular* if all its second-order derivatives are non-negative. Clearly, f is submodular if, and only if, $-f$ is supermodular, so it is straightforward to translate results about supermodular functions, such as those given in [8] and [43], into similar results for submodular functions, and we will use this observation several times below. Cost functions that are both submodular and supermodular (in other words, where all second-order derivatives are equal to zero) are called *modular*, and polynomials corresponding to modular cost functions are linear [4].

Example 6. For any set of indices $I = \{i_1, \dots, i_m\} \subseteq \{1, \dots, n\}$ we can define a cost function ϕ_I in n variables as follows:

$$\phi_I(x_1, \dots, x_n) \;=\; \begin{cases} -1 & \text{if } (\forall i \in I)(x_i = 1), \\ 0 & \text{otherwise.} \end{cases}$$

The polynomial representation of ϕ_I is $p(x_1, \dots, x_n) = -x_{i_1} \dots x_{i_m}$, which is a polynomial of degree m. Note that it is straightforward to verify that ϕ_I is submodular by checking the second-order derivatives of p.

However, the function ϕ_I is also expressible by *binary* submodular polynomials, using a single extra variable, y, as follows:

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

We remark that this is a special case of the expressibility result for negative-positive polynomials first obtained in [45].

Note that when $D=\{0,1\}$, the set D^n with the product ordering is isomorphic to the lattice of all subsets of an n-element set ordered by inclusion. Hence, a cost function on a Boolean domain can be viewed as a cost function defined on a lattice of subsets, and we can apply Definition 5 to identify certain Boolean functions as upper fans or lower fans, as the following example indicates.

Example 7. Let $A=\{I_1,\ldots,I_r\}$ be a set of subsets of $\{1,2,\ldots,n\}$ such that for all $i\neq j$ we have $I_i \not\subseteq I_j$ and $I_i\cup I_j=\bigcup A$.

By Definition 5, the corresponding upper fan function ϕ_A has the following polynomial representation:

$$p(x_1,\ldots,x_n) = (r-2)\prod_{i\in\bigcup A}x_i - \prod_{i\in I_1}x_i - \cdots - \prod_{i\in I_r}x_i.$$

We remark that any permutation of a set D gives rise to an automorphism of cost functions over D. In particular, for any cost function f on a Boolean domain D, the *dual* of f is the corresponding cost function which results from exchanging the values 0 and 1 for all variables. In other words, if p is the polynomial representation of f, then the dual of f is the cost function whose polynomial representation is obtained from p by replacing all variables x with $1-x$. Observe that, due to symmetry, taking the dual preserves submodularity and expressibility by binary submodular cost functions.

It is not hard to see that upper fans are duals of lower fans and vice versa.

3. Results

In this section, we present our main results. First, we show that fans of all arities are expressible by binary submodular cost functions. Next, we characterise the multimorphisms of binary submodular cost functions. Combining these results, we then characterise precisely which 4-ary submodular cost functions are expressible by binary submodular cost functions. More importantly, we show that some submodular cost functions are *not* expressible by binary submodular cost functions, and therefore cannot be minimised using the Min-Cut problem via an expressibility reduction. Finally, we consider the complexity of recognising which cost functions are expressible by binary submodular cost functions.

3.1. Expressibility of upper fans and lower fans

We denote by $\Gamma_{\mathsf{sub},n}$ the set of all finite-valued submodular cost functions of arity at most n on a Boolean domain D, and we set $\Gamma_{\mathsf{sub}} = \bigcup_n \Gamma_{\mathsf{sub},n}$.

We denote by $\Gamma_{\mathsf{fans},n}$ the set of all fans of arity at most n on a Boolean domain D, and we set $\Gamma_{\mathsf{fans}} = \bigcup_n \Gamma_{\mathsf{fans},n}$.

Our next result shows that $\Gamma_{\mathsf{fans}} \subseteq \langle \Gamma_{\mathsf{sub},2} \rangle$.

Theorem 8. *Any fan on a Boolean domain D is expressible by binary submodular functions on D using at most $1 + \lfloor m/2 \rfloor$ extra variables, where m is the degree of its polynomial representation.*

Proof. Since upper fans are dual to lower fans, it is sufficient to establish the result for upper fans only.

Let $A = \{I_1, \ldots, I_r\}$ be a set of subsets of $\{1, 2, \ldots, n\}$ such that for all $i \neq j$ we have $I_i \not\subseteq I_j$ and $I_i \cup I_j = \bigcup A$, and let ϕ_A be the corresponding upper fan, as specified by Definition 5. The polynomial representation of ϕ_A, $p(x_1, \ldots, x_n)$, is given in Example 7.

The degree of p is equal to the total number of variables occurring in it, which will be denoted m. Note that $m = |\bigcup A|$.

If $r = 0$, then ϕ_A is constant, so the result holds trivially. If $r = 1$, we have $A = \{I\}$, where $I = \{i_1, \ldots, i_m\}$ and the polynomial representation of ϕ_A is $-2x_{i_1}x_{i_2}\cdots x_{i_m}$. In this case, it was shown in Example 6 that ϕ_A can be expressed by quadratic functions using one extra variable, as follows:

$$-2x_{i_1}x_{i_2}\cdots x_{i_m} = \min_{y \in \{0,1\}} \{2y((m-1) - \sum_{i \in I} x_i)\}.$$

For the case when $r > 1$, we first note that any $i \in \bigcup A$ must belong to all the elements of A except for at most one (otherwise there would be two elements of A, say I_i and I_j, such that $I_i \cup I_j \neq \bigcup A$, which contradicts the choice of A).

We will say that two elements of $\bigcup A$ are *equivalent* if they occur in exactly the same elements of A; that is, $i_1, i_2 \in \bigcup A$ are equivalent if $i_1 \in I_j \Leftrightarrow i_2 \in I_j$ for all $j \in \{i, \ldots, r\}$. Equivalent elements i_1 and i_2 of $\bigcup A$ can be merged by replacing them with a single new element. In the polynomial representation of ϕ_A this corresponds to replacing the variables x_{i_1} and x_{i_2} with a single new variable, z, corresponding to their product. Note that the number of equivalence classes of size two or greater is at most $\lfloor m/2 \rfloor$.

After completing all such merging, we obtain a new set $A' = \{I'_1, \ldots, I'_{r'}\}$ with the property that $|I'_i| = m' - 1$ for every i, where $m' = |\bigcup A'|$ is the size of the common join of any $I'_i, I'_j \in A'$. This set has a corresponding new upper fan, $\phi_{A'}$, over the new merged variables.

To complete the proof we will construct a simple gadget for expressing $\phi_{A'}$, and show how to use this to obtain a gadget for expressing the original upper fan ϕ_A.

Note that the sets I'_i are subsets of $\bigcup A'$, each of size $m' - 1$. Any such subset is uniquely determined by its single missing element. We denote by K the set of elements occurring in *all* sets I'_i and by L the set of elements which are missing from one of these subsets. Clearly, $|K| + |L| = m'$. We claim that the following polynomial is a gadget for expressing ϕ'_A:

$$p'(z_1, \ldots, z_{m'}) = \min_{y \in \{0,1\}} \{y(2(m'-1) - |L| - \sum_{i \in L} z_i - 2\sum_{i \in K} z_i)\}.$$

To establish this claim, we will compute the value of p', for each possible assignment to the variables $z_1, \ldots, z_{m'}$. Denote by k_0 the number of 0s assigned to variables in K, and by l_0 the number of 0s assigned to variables in L. Then we have:

$$\begin{aligned} p'(z_1, \ldots, z_{m'}) &= \min_{y \in \{0,1\}} y(2m' - 2 - |L| - \sum_{i \in L} z_i - 2\sum_{i \in K} z_i) \\ &= \min_{y \in \{0,1\}} y(2m' - 2 - |L| - (|L| - l_0) - 2(m' - |L| - k_0)) \\ &= \min_{y \in \{0,1\}} y(2m' - 2 - 2|L| + l_0 - 2m' + 2|L| + 2k_0) \\ &= \min_{y \in \{0,1\}} y(-2 + 2k_0 + l_0). \end{aligned}$$

Hence if $k_0 = l_0 = 0$, then p' takes the value -2. If $k_0 = 0$ and $l_0 = 1$, then p' takes the value -1. In all other cases (that is, $k_0 > 0$ or $l_0 > 1$), p' takes the value 0. By Definition 5, this means that p' is the (unique) polynomial representation for $\phi_{A'}$. Note that p' uses just one extra variable, y.

Finally, we show how to obtain a gadget for the original upper fan ϕ_A, from the polynomial p'. Each variable in p' represents an equivalence class of elements of $\bigcup A$, so it can be replaced by a term consisting of the product of the variables in this equivalence class. In this way we obtain a new polynomial over the original variables containing linear and negative quadratic terms together with negative higher order terms (cubic or above) corresponding to every equivalence class with 2 or more elements. However, each of these higher order terms can itself be expressed by a quadratic submodular polynomial, by introducing a single extra variable, as shown in the case when $r = 1$, above. Therefore, combining each of these polynomials, the total number of new variables introduced is at most $1 + \lfloor m/2 \rfloor$.

Many of the earlier expressibility results mentioned in Section 1.1 can be obtained as simple corollaries of Theorem 8, as the following examples indicate.

Example 9. Any negative monomial $-x_1x_2\cdots x_m$ is a positive multiple of an upper fan, and the positive linear monomial x_1 is equal to $-(1-x_1)+1$, so it is a positive multiple of a lower fan, plus a constant. Hence all negative-positive submodular polynomials are contained in Cone(Γ_{fans}), and by Theorem 8, they are expressible by binary submodular polynomials, as originally shown in [45].

Example 10. A polynomial is called *homogeneous* [2] or *polar* [13] if it can be expressed as a sum of terms of the form $ax_1x_2\ldots x_k$ or $a(1-x_1)(1-x_2)\ldots(1-x_k)$ with positive coefficients a, together with a constant term. It was observed in [2] that all polar polynomials are supermodular, so all negated polar polynomials are submodular. As every negated term $-ax_1x_2\dots x_k$, is a positive multiple of an upper fan, and every negated term $-a(1-x_1)(1-x_2)\dots(1-x_k)$, is a positive multiple of a lower fan, by Theorem 8, all cost functions which are the negations of polar polynomials are expressible by binary submodular polynomials, and solvable by reduction to MIN-CUT, as originally shown in [2].

Example 11. Any cubic submodular polynomial can be expressed as a positive sum of upper fans [43]. Hence, by Theorem 8, all cubic submodular polynomials are expressible by binary submodular polynomials, as originally shown in [2].

Example 12. A Boolean cost function ϕ is called *2-monotone* [14] if there exist two sets $R, S \subseteq \{1, \dots, n\}$ such that $\phi(\mathbf{x}) = 0$ if $R \subseteq \mathbf{x}$ or $\mathbf{x} \subseteq S$ and $\phi(\mathbf{x}) = 1$ otherwise (where $R \subseteq \mathbf{x}$ means $\forall i \in R, x[i] = 1$ and $\mathbf{x} \subseteq S$ means $\forall i \notin S, x[i] = 0$). It was shown in [8, Proposition 2.9] that a 2-valued Boolean cost function is 2-monotone if, and only if, it is submodular.

For any 2-monotone cost function defined by the sets of indices R and S, it is straightforward to check that $\phi = \min_{y \in \{0,1\}} y(1+\phi_A/2) + (1-y)(1+\phi_B/2)$ where ϕ_A is the upper fan defined by $A = \{R\}$ and ϕ_B is the lower fan defined by $B = \{\overline{S}\}$. Note that the function $y\phi_A$ is an upper fan, and the function $(1-y)\phi_B$ is a lower fan. Hence, by Theorem 8, all 2-monotone polynomials are expressible by binary submodular polynomials, and solvable by reduction to MIN-CUT, as originally shown in [14].

However, Theorem 8 also provides many new functions of all arities which have not previously been shown to be expressible by binary submodular functions, as the following example indicates.

Example 13. The function $2x_1x_2x_3x_4 - x_1x_2x_3 - x_1x_2x_4 - x_1x_3x_4 - x_2x_3x_4$ belongs to $\Gamma_{\mathsf{fans},4}$, but does not belong to any class of submodular functions which has previously been shown to be expressible by binary submodular functions. In particular, it does not belong to the class Γ_{new} identified in [54, 55].

3.2. Characterisation of $\mathsf{Mul}(\Gamma_{\mathsf{sub},2})$

Since we have seen that a cost function can only be expressed by a given set of cost functions if it has the same multimorphisms, we now investigate the multimorphisms of $\Gamma_{\mathsf{sub},2}$.

A function $\mathcal{F} : D^k \to D^k$ is called *conservative* if, for each possible choice of $x_1, \dots, x_k$, the tuple $\mathcal{F}(x_1, \dots, x_k)$ is a permutation of $x_1, \dots, x_k$ (though different inputs may be permuted in different ways).

For any two tuples $\mathbf{x} = \langle x_1, \dots, x_k \rangle$ and $\mathbf{y} = \langle y_1, \dots, y_k \rangle$ over D, we denote by $H(\mathbf{x}, \mathbf{y})$ the *Hamming distance* between $\mathbf{x}$ and $\mathbf{y}$, which is the number of positions at which the corresponding values are different.

Theorem 14. *For any Boolean domain D, and any $\mathcal{F} : D^k \to D^k$, the following are equivalent:*

1. $\mathcal{F} \in \mathsf{Mul}(\Gamma_{\mathsf{sub},2})$.
2. $\mathcal{F} \in \mathsf{Mul}(\Gamma^{\infty}_{\mathsf{sub},2})$, *where* $\Gamma^{\infty}_{\mathsf{sub},2}$ *denotes the set of binary submodular cost functions taking finite or infinite values.*
3. $\mathcal{F}$ *is conservative and Hamming distance non-increasing.*

Proof. First we consider unary cost functions. All unary cost functions on a Boolean domain are easily shown to be submodular. Also, any conservative function $\mathcal{F} : D^k \to D^k$ is clearly a multimorphism of any unary cost function, since it merely permutes its arguments.

For any $d \in D$, define the unary cost function μ_d as follows:

$$\mu_d(x) \;=\; \begin{cases} 1 & \text{if } x = d, \\ 0 & \text{if } x \neq d. \end{cases}$$

Let $\mathcal{F} : D^k \to D^k$ be a non-conservative function. In that case, there are $u_1, \ldots, u_k, v_1, \ldots, v_k \in D$ such that $\mathcal{F}(u_1, \ldots, u_k) = \langle v_1, \ldots, v_k \rangle$ and there is i such that v_i occurs more often in $\langle v_1, \ldots, v_k \rangle$ than in $\langle u_1, \ldots, u_k \rangle$. It is simple to check that $\mathcal{F}$ is not a multimorphism of the unary cost function μ_{v_i}. Hence any $\mathcal{F} \in \mathsf{Mul}(\Gamma_{\mathsf{sub},2})$ must be conservative.

By the same argument, any $\mathcal{F} \in \mathsf{Mul}(\Gamma^{\infty}_{\mathsf{sub},2})$ must be conservative.

For any $c \in \overline{\mathbb{R}}$, define the binary cost functions λ_c and χ_c as follows:

$$\lambda_c(x, y) \;=\; \begin{cases} c & \text{if } x = 0 \text{ and } y = 1, \\ 0 & \text{otherwise.} \end{cases} \qquad \chi_c(x, y) \;=\; \begin{cases} c & \text{if } x \neq y, \\ 0 & \text{otherwise.} \end{cases}$$

Note that $\chi_c(x, y) = \lambda_c(x, y) + \lambda_c(y, x)$.

By a simple case analysis, it is straightforward to check that any binary submodular cost function on a Boolean domain can be expressed by binary functions of the form λ_c, with $c > 0$ together with unary cost functions of the form μ_d.

We observe that when $c < \infty$, $\lambda_c(x, y) = (\chi_c(x, y) + c\mu_0(x) + c\mu_1(y) - c)/2$, so λ_c can be expressed by functions of the form χ_c together with unary cost functions of the form μ_d. Hence, since expressibility preserves multimorphisms, $\mathsf{Mul}(\Gamma_{\mathsf{sub},2}) = \mathsf{Mul}(\{\chi_c \mid c \in \mathbb{R}, c > 0\}) \cap \mathsf{Mul}(\{\mu_d \mid d \in D\})$.

Now let $\mathbf{u}, \mathbf{v} \in D^k$, and consider the multimorphism inequality, as given in Definition 4, for the case where $t_i = \langle \mathbf{u}[i], \mathbf{v}[i] \rangle$, for $i = 1, \ldots, k$. By Definition 4, for any $c > 0$, $\mathcal{F}$ is a multimorphism of χ_c if, and only if, the following holds for all choices of $\mathbf{u}$ and $\mathbf{v}$:

$$H(\mathbf{u}, \mathbf{v}) \geq H(\mathcal{F}(\mathbf{u}), \mathcal{F}(\mathbf{v})).$$

This proves that the multimorphisms of $\Gamma_{\mathsf{sub},2}$ are precisely the conservative functions which are also Hamming distance non-increasing.

Since $\Gamma_{\mathsf{sub},2} \subseteq \Gamma^{\infty}_{\mathsf{sub},2}$, we know that $\mathsf{Mul}(\Gamma^{\infty}_{\mathsf{sub},2}) \subseteq \mathsf{Mul}(\Gamma_{\mathsf{sub},2})$. Therefore, in order to complete the proof it is enough to show that every conservative and Hamming distance non-increasing function $\mathcal{F}$ is a multimorphism of λ_∞.

For any $\mathbf{u}, \mathbf{v} \in \{0,1\}^k$, the Hamming distance $H(\mathbf{u}, \mathbf{v})$ is equal to the symmetric difference of the sets of positions where $\mathbf{u}$ and $\mathbf{v}$ take the value 1. Hence, for tuples $\mathbf{u}$ and $\mathbf{v}$ containing some fixed number of 1s, the minimum Hamming distance occurs precisely when one of these sets of positions is contained in the other.

Now consider again the multimorphism inequality, as given in Definition 4, for the case where $t_i = \langle \mathbf{u}[i], \mathbf{v}[i] \rangle$, for $i = 1, \ldots, k$. If there is any position i where $\mathbf{u}[i] = 0$ and $\mathbf{v}[i] = 1$, then $\lambda_\infty(t_i) = \infty$, so the multimorphism inequality is trivially satisfied. If there is no such position, then the set of positions where $\mathbf{v}$ takes the value 1 is contained in the set of positions where $\mathbf{u}$ takes the value 1, so $H(\mathbf{u}, \mathbf{v})$ takes its minimum possible value over all reorderings of $\mathbf{u}$ and $\mathbf{v}$. Hence if $\mathcal{F}$ is conservative, then $H(\mathbf{u}, \mathbf{v}) \leq H(\mathcal{F}(\mathbf{u}), \mathcal{F}(\mathbf{v}))$, and if $\mathcal{F}$ is Hamming distance non-increasing, we have $H(\mathbf{u}, \mathbf{v}) = H(\mathcal{F}(\mathbf{u}), \mathcal{F}(\mathbf{v}))$. But this implies that the set of positions where $\mathcal{F}(\mathbf{v})$ takes the value 1 is contained in the set of positions where $\mathcal{F}(\mathbf{u})$ takes the value 1. By definition of λ_∞, this implies that both sides of the multimorphism inequality are zero, so $\mathcal{F}$ is a multimorphism of λ_∞.

3.3. Non-expressibility of Γ_{sub} *over* $\Gamma_{\mathsf{sub},2}$

Theorem 14 characterises the multimorphisms of $\Gamma_{\mathsf{sub},2}$, and hence enables us to systematically search (for example, using Mathematica) for multimorphisms of $\Gamma_{\mathsf{sub},2}$ which are not multimorphisms of Γ_{sub}. In this way, we have identified the function $\mathcal{F}_{sep} : \{0,1\}^5 \rightarrow \{0,1\}^5$ defined in Figure 2. We will show in this section that this function can be used to characterise all the submodular functions of arity 4 which are expressible by binary submodular functions on a Boolean domain. Using this result, we show that some submodular functions are *not* expressible in this way.

	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
$\mathbf{x}$	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
$\mathcal{F}_{sep}(\mathbf{x})$	0	0	0	0	0	0	1	1	0	0	0	1	0	0	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
	0	0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1

Figure 2: Definition of $\mathcal{F}_{sep}$.

Proposition 15. *$\mathcal{F}_{sep}$ is conservative and Hamming distance non-increasing.*

Proof. Straightforward exhaustive verification.

Theorem 16. *For any function $f \in \Gamma_{\mathsf{sub},4}$ the following are equivalent:*

1. $f \in \langle \Gamma_{\mathsf{sub},2} \rangle$.
2. $\mathcal{F}_{sep} \in \mathsf{Mul}(\{f\})$.
3. $f \in \mathrm{Cone}(\Gamma_{\mathsf{fans},4})$.

Proof. First, we show (1) $\Rightarrow$ (2). Proposition 15 and Theorem 14 imply that $\mathcal{F}_{sep}$ is a multimorphism of any binary submodular function on a Boolean domain. Hence having $\mathcal{F}_{sep}$ as a multimorphism is a necessary condition for any submodular cost function on a Boolean domain to be expressible by binary submodular cost functions.

Next, we show (2) $\Rightarrow$ (3). Consider the complete set of inequalities on the values of a 4-ary cost function resulting from having the multimorphism $\mathcal{F}_{sep}$, as specified in Definition 4. A routine calculation in Mathematica shows that, out of 16^5 such inequalities, there are 4635 which are distinct. After removing from these all those which are equal to the sum of two others, we obtain a system of just 30 inequalities which must be satisfied by any 4-ary submodular cost function which has the multimorphism $\mathcal{F}_{sep}$. Using the double description method [38], we obtain from these 30 inequalities an equivalent set of 31 extreme rays which generate the same polyhedral cone of cost functions. These extreme rays all correspond to fans or sums of fans.

Finally, we show (3) $\Rightarrow$ (1). By Theorem 8, all fans are expressible over $\Gamma_{\mathsf{sub},2}$. It follows that any cost function in this cone of functions is also expressible over $\Gamma_{\mathsf{sub},2}$.

Next we show that there are indeed 4-ary submodular cost functions which do not have $\mathcal{F}_{sep}$ as a multimorphism and therefore are not expressible by binary submodular cost functions.

Definition 17. For any Boolean tuple t of arity 4 containing exactly 2 ones and 2 zeros, we define the 4-ary cost function θ_t as follows:

$$\theta_t(x_1,x_2,x_3,x_4) \;=\; \begin{cases} -1 & \text{if } (x_1,x_2,x_3,x_4) = (1,1,1,1) \text{ or } (0,0,0,0), \\ 1 & \text{if } (x_1,x_2,x_3,x_4) = t, \\ 0 & \text{otherwise.} \end{cases}$$

Cost functions of the form θ_t were introduced in [43], where they are called *quasi-indecomposable* functions. We denote by Γ_{qin} the set of all (six) quasi-indecomposable cost functions of arity 4. It is straightforward to check that they are submodular, but the next result shows that they are *not* expressible by binary submodular functions.

Proposition 18. *For all $\theta \in \Gamma_{\mathsf{qin}}$, $\mathcal{F}_{sep} \notin \mathsf{Mul}(\{\theta\})$.*

Proof. The table in Figure 3 shows that $\mathcal{F}_{sep} \notin \mathsf{Mul}(\{\theta_{(1,1,0,0)}\})$. Permuting the columns appropriately establishes the result for all other $\theta \in \Gamma_{\mathsf{qin}}$.

Corollary 19. *For all $\theta \in \Gamma_{\mathsf{qin}}$, $\theta \notin \langle \Gamma_{\mathsf{sub},2} \rangle$.*

$$\begin{array}{ccc}
\begin{array}{c}
\\ \\ \\ \\ \\ \\ \\ \mathcal{F}_{sep} \\ \\ \\
\end{array}
&
\begin{array}{cccc}
1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 \\
\hline
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 \\
0 & 1 & 1 & 1
\end{array}
&
\begin{array}{c}
\xrightarrow{\theta_{(1,1,0,0)}} \left.\begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{array}\right\} \sum = 0 \\
\xrightarrow{\theta_{(1,1,0,0)}} \left.\begin{array}{c} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{array}\right\} \sum = 1
\end{array}
\end{array}$$

Figure 3: $\mathcal{F}_{sep} \notin \mathsf{Mul}(\{\theta_{(1,1,0,0)}\})$.

Proof. By Theorem 16 and Proposition 18.

Are there any other 4-ary submodular cost functions which are not expressible over $\Gamma_{\mathsf{sub},2}$? Promislow and Young characterised the extreme rays of the cone of all 4-ary submodular cost functions and established that $\Gamma_{\mathsf{sub},4} = \mathrm{Cone}(\Gamma_{\mathsf{fans},4} \cup \Gamma_{\mathsf{qin}})$ – see Theorem 5.2 of [43]. Hence the results in this section characterise the expressibility of all 4-ary submodular functions.

Promislow and Young conjectured that for $k \neq 4$, all extreme rays of $\Gamma_{\mathsf{sub},k}$ are fans [43]; that is, they conjectured that for all $k \neq 4$, $\Gamma_{\mathsf{sub},k} = \mathrm{Cone}(\Gamma_{\mathsf{fans},k})$. However, if this conjecture were true it would imply that all submodular functions of arity 5 and above were expressible by binary submodular functions, by Theorem 8. This is clearly not the case, because inexpressible cost functions such as those identified in Corollary 19 can be extended to larger arities (for instance, by adding dummy arguments) and remain inexpressible. Hence our results refute this conjecture for all $k \geq 5$. However, we suggest that this conjecture can be refined to a similar statement concerning just those submodular functions which are expressible by binary submodular functions, as follows:

Conjecture 20. *For all k,* $\Gamma_{\mathsf{sub},k} \cap \langle \Gamma_{\mathsf{sub},2} \rangle = \mathrm{Cone}(\Gamma_{\mathsf{fans},k})$.

This conjecture was previously known to be true for $k \leq 3$ [43]; Theorem 8 shows that $\mathrm{Cone}(\Gamma_{\mathsf{fans},k}) \subseteq \Gamma_{\mathsf{sub},k} \cap \langle \Gamma_{\mathsf{sub},2} \rangle$ for all k, and Theorem 16 confirms that equality holds for $k = 4$.

3.4. The complexity of recognising expressible functions

Finally, we show that we can test efficiently whether a submodular polynomial of arity 4 is expressible by binary submodular polynomials.

Definition 21. Let $p(x_1, x_2, x_3, x_4)$ be the polynomial representation of a 4-ary submodular cost function f. We denote by a_I the coefficient of the term $\prod_{i \in I} x_i$. We say that f satisfies condition **Sep** if for each $\{i,j\}, \{k,\ell\} \subset \{1,2,3,4\}$, with i,j,k,ℓ distinct, we have $a_{\{i,j\}} + a_{\{k,\ell\}} + a_{\{i,j,k\}} + a_{\{i,j,\ell\}} \leq 0$.

Theorem 22. *For any $f \in \Gamma_{\mathsf{sub},4}$, the following are equivalent:*

1. $f \in \langle \Gamma_{\mathsf{sub},2} \rangle$.
2. *f satisfies condition* **Sep**.

Proof. As in the proof of Theorem 16, we construct a set of 30 inequalities corresponding to the multimorphism $\mathcal{F}_{sep}$. Each of these inequalities on the values of a cost function can be translated into inequalities on the coefficients of the corresponding polynomial representation by a straightforward linear transformation. This calculation shows that 24 of the resulting inequalities impose the condition of submodularity, and the remaining 6 impose condition **Sep**. Hence a submodular cost function of arity 4 has the multimorphism $\mathcal{F}_{sep}$ if, and only if, its polynomial representation satisfies condition **Sep**. The result then follows from Theorem 16. □

Using Theorem 22, we can test whether optimisation problems given as a sum of submodular functions of arity 4 can be reduced to the Min-Cut problem via the expressibility reduction. These problems arise in Computer Vision and in Valued Constraint Satisfaction Problems.

Furthermore, by Theorem 8, the number of extra variables needed in this reduction is rather small compared to the theoretical upper bound given in [9].

It is known that the problem of recognising whether an arbitrary degree-4 polynomial is submodular is co-NP-complete [13, 19]. One might hope that the more restricted class of submodular polynomials expressible by binary submodular polynomials would be recognisable in polynomial time. At the moment, the complexity of the recognition problem for submodular polynomials of degree 4 that are expressible by binary submodular polynomials is open.

4. Applications

In this section we discuss the application of our results to two specific application areas: artificial intelligence and computer vision.

As indicated in the previous Section, in general testing for submodularity is co-NP-complete even for polynomials of degree 4 [19]. However, for many optimisation problems arising in practice, testing for submodularity is not an issue because the function to be minimised is presented as a sum of functions of bounded arity. In such cases, each of the bounded-arity sub-functions can be tested for submodularity in constant time. For example, in valued constraint satisfaction problems and energy minimisation problems in computer vision, each instance is specified as a sum of bounded-arity functions. The recognition of submodularity only becomes co-NP-complete when a function is presented without a fixed decomposition into sub-functions of this kind.

4.1. Artificial Intelligence

First we formally define the valued constraint satisfaction problem [3, 47, 48].

Definition 23. Let Γ be a set of cost functions over a set D. An instance $\mathcal{P}$ of VCSP(Γ) 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 $\phi : D^{|\sigma|} \to \overline{\mathbb{R}}$ is a cost function from Γ. 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.

Now we show how our results can be applied in this framework.

Corollary 24 (of Theorem 8). VCSP(Γ_{fans}) *is solvable in* $O((n+k)^3)$ *time, where* n *is the number of variables and* k *is the number of constraints of arity 3 or higher.*

Moreover, as shown above, VCSP($\Gamma_{\mathsf{fans},4}$) is the *maximal* class in VCSP($\Gamma_{\mathsf{sub},4}$) which can be solved by reduction to Min-Cut in this way.

Cohen et al. [9] showed that if a cost function ϕ of arity k is expressible by some set of cost functions over Γ, then ϕ is expressible by Γ using at most 2^{2^k} extra variables. Our results show that only $O(k)$ extra variables are needed to express any cost function from $\Gamma_{\mathsf{fans},k}$ by $\Gamma_{\mathsf{sub},2}$. Therefore, an instance of VCSP(Γ_{fans}) needs only linearly many (in the number of constraints of arity 3 or higher) extra variables, where the linear factor is proportional to the maximum arity of the constraints. In particular, an instance of VCSP($\Gamma_{\mathsf{sub},4}$) is either reducible to Min-Cut with only linearly many extra variables,[3] or is not reducible in this way at all.

4.2. Computer Vision

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:

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

Set V usually corresponds to pixels, x_v denotes the label of 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 also called

[3]Optimal (in the number of extra variables) gadgets for cost functions from $\Gamma_{\mathsf{fans},4}$ are given in [56].

higher-order cliques. This energy is often derived in the context of *Markov Random Fields* (also known as *Conditional Random Fields*) [1, 20]: a minimum of E corresponds to a *maximum a posteriori* (MAP) labelling $\mathbf{x}$ [36, 52].

It is straightforward to verify that this formulation is equivalent to the VCSP. See [53] for a survey on the connection between computer vision and constraint satisfaction problems. Therefore, for energy minimisation over Boolean variables we get the following:

Corollary 25 (of Theorem 8). *Energy minimisation, where each term of the energy function belongs to Γ_{fans}, is solvable in $O((n+k)^3)$ time, where where n is the number of variables (pixels) and k is the number of higher-order (ternary and above) terms in the energy function.*

Note that any variable over a non-Boolean domain $D = \{0, 1, \ldots, d-1\}$ of size d can be encoded by $d-1$ Boolean variables. One such encoding is the following: define $en(i) = 0^{d-i-1}1^i$. 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. To convert a non-Boolean energy minimisation problem into a Boolean problem 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. Other forms of encoding, which are suitable for certain subclasses of submodular functions, and require fewer Boolean variables, have also been studied [32, 44].

Acknowledgements

The authors would like to thank Martin Cooper for fruitful discussions on submodular functions and, in particular, for help with the proof of Theorem 8. Stanislav Živný would like to thank Philip Torr and Tomáš Werner for useful discussions on the connection between constraint satisfaction problems and computer vision. Stanislav Živný was supported by EPSRC grant EP/F01161X/1.

References

[1] J. Besag, On the Statistical Analysis of Dirty Pictures, Journal of the Royal Statistical Society, Series B 48 (3) (1986) 259–302.

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

[3] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, G. Verfaillie, Semiring-based CSPs and Valued CSPs: Frameworks, Properties, and Comparison, Constraints 4 (3) (1999) 199–240.

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

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

[6] R. Burkard, B. Klinz, R. Rudolf, Perspectives of Monge Properties in Optimization, Discrete Applied Mathematics 70 (2) (1996) 95–161.

[7] D. Cohen, M. Cooper, P. Jeavons, A. Krokhin, A Maximal Tractable Class of Soft Constraints, Journal of Artificial Intelligence Research 22 (2004) 1–22.

[8] D. Cohen, M. Cooper, P. Jeavons, A. Krokhin, Supermodular Functions and the Complexity of MAX-CSP, Discrete Applied Mathematics 149 (1-3) (2005) 53–72.

[9] D. A. Cohen, M. C. Cooper, P. G. Jeavons, An Algebraic Characterisation of Complexity for Valued Constraints, in: Proceedings of the 12th International Conference on Principles and Practice of Contraint Programming (CP'06), vol. 4204 of Lecture Notes in Computer Science, Springer, 2006.

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

[11] D. A. Cohen, M. C. Cooper, P. G. Jeavons, A. A. Krokhin, The Complexity of Soft Constraint Satisfaction, Artificial Intelligence 170 (11) (2006) 983–1016.

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

[13] Y. Crama, Recognition problems for special classes of polynomials in 0-1 variables, Mathematical Programming 44 (1-3) (1989) 139–155.

[14] 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.

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

[16] 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.

[17] 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'05), IEEE Computer Society, 2005.

[18] S. Fujishige, Submodular Functions and Optimization, vol. 58 of Annals of Discrete Mathematics, 2nd ed., North-Holland, Amsterdam, 2005.

[19] G. Gallo, B. Simeone, On the supermodular knapsack problem, Mathematical Programming 45 (1-3) (1988) 295–309.

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

[21] M. Grötschel, L. Lovasz, A. Schrijver, The ellipsoid method and its consequences in combinatorial optimization, Combinatorica 1 (2) (1981) 169–198.

[22] M. Grötschel, L. Lovasz, A. Schrijver, Geometric Algorithms and Combinatorial Optimization, vol. 2 of Algorithms and Combinatorics, Springer, 1988.

[23] G. Gutin, A. Rafiey, A. Yeo, M. Tso, Level of Repair Analysis and Minimum Cost Homomorphisms of Graphs, Discrete Applied Mathematics 154 (6) (2006) 881–889.

[24] P. L. Hammer, Some network flow problems solved with pseudo-Boolean programming, Operations Research 13 (3) (1965) 388–399.

[25] S. Iwata, A fully combinatorial algorithm for submodular function minimization, Journal of Combinatorial Theory, Series B 84 (2) (2002) 203–212.

[26] S. Iwata, A faster scaling algorithm for minimizing submodular functions, SIAM Journal on Computing 32 (4) (2003) 833–840.

[27] S. Iwata, Submodular Function Minimization, Mathematical Programming 112 (1) (2008) 45–64.

[28] S. Iwata, L. Fleischer, S. Fujishige, A combinatorial strongly polynomial algorithm for minimizing submodular functions, Journal of the ACM 48 (4) (2001) 761–777.

[29] S. Iwata, J. B. Orlin, A Simple Combinatorial Algorithm for Submodular Function Minimization, in: Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'09), 2009.

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

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

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

[33] 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.

[34] B. Korte, J. Vygen, Combinatorial Optimization, vol. 21 of Algorithms and Combinatorics, 4th ed., Springer, 2007.

[35] A. Krokhin, B. Larose, Maximizing Supermodular Functions on Product Lattices, with Application to Maximum Constraint Satisfaction, SIAM Journal on Discrete Mathematics 22 (1) (2008) 312–328.

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

[37] L. Lovász, Submodular Functions and Convexity, in: A. Bachem, M. Grötschel, B. Korte (eds.), Mathematical Programming – The State of the Art, Springer, Berlin, 1983.

[38] T. Motzkin, H. Raiffa, G. Thompson, R. Thrall, The double description method, in: H. W. Kuhn, A. W. Tucker (eds.), Contributions to the Theory of Games, vol. 2, Princeton University Press, 1953, pp. 51–73.

[39] H. Narayanan, Submodular Functions and Electrical Networks, North-Holland, Amsterdam, 1997.

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

[41] G. Nemhauser, L. Wolsey, M. Fisher, An Analysis of Approximations for Maximizing Submodular Set Functions-I, Mathematical Programming 14 (1) (1978) 265–294.

[42] J. B. Orlin, A faster strongly polynomial time algorithm for submodular function minimization., Mathematical Programming 118 (2) (2009) 237–251.

[43] S. Promislow, V. Young, Supermodular Functions on Finite Lattices, Order 22 (4) (2005) 389–413.

[44] 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'08), IEEE Computer Society, 2008.

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

[46] I. Rosenberg, Reduction of bivalent maximization to the quadratic case, Cahier du Centre dEtudes de Recherche Oprationnelle 17 (1975) 71–74.

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

[48] 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'95), 1995.

[49] A. Schrijver, A Combinatorial Algorithm Minimizing Submodular Functions in Strongly Polynomial Time, Journal of Combinatorial Theory, Series B 80 (2) (2000) 346–355.

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

[51] D. Topkis, Supermodularity and Complementarity, Princeton University Press, 1998.

[52] M. J. Wainwright, M. I. Jordan, Graphical models, exponential families, and variational inference, Foundations and Trends in Machine Learning 1 (1-2) (2008) 1–305.

[53] 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.

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

[55] S. Živný, P. G. Jeavons, Classes of Submodular Constraints Expressible by Graph Cuts, in: Proceedings of the 14th International Conference on Principles and Practice of Constraint Programming (CP'08), vol. 5202 of Lecture Notes in Computer Science, Springer, 2008.

[56] S. Živný, P. G. Jeavons, Which submodular functions are expressible using binary submodular functions?, Research Report CS-RR-08-08, Computing Laboratory, University of Oxford, Oxford, UK (June 2008).