Efficient Minimization of Higher Order Submodular Functions using Monotonic Boolean Functions

Srikumar Ramalingam[1] Chris Russell[2] L'ubor Ladický[3] Philip H.S. Torr[4]

[1]Mitsubishi Electric Research Laboratories, Cambridge, USA,
[2]Queen Mary, University of London, UK,
[3]University of Oxford, Oxford, UK,
[4]Oxford Brookes University, Oxford, UK,

Abstract. Submodular function minimization is a key problem in a wide variety of applications in machine learning, economics, game theory, computer vision and many others. The general solver has a complexity of $O(n^6 + n^5 L)$ where L is the time required to evaluate the function and n is the number of variables [33]. On the other hand, many useful applications in computer vision and machine learning are defined over special subclasses of submodular functions that can be written as the sum of many submodular cost functions defined over cliques containing few variables. In such functions, the pseudo-Boolean (or polynomial) representation [3] of these subclasses are of degree (or order, or clique size) k where $k \ll n$. In this work, we develop efficient algorithms for the minimization of this useful subclass of submodular functions. To do this, we define novel mapping that transform submodular functions of order k into quadratic ones. The underlying idea is to use auxiliary variables to model the higher order terms and the transformation is found using a carefully constructed linear program. In particular, we model the auxiliary variables as monotonic Boolean functions, allowing us to obtain a compact transformation using as few auxiliary variables as possible. The transformed quadratic function can be efficiently minimized using the standard max-flow algorithm with a time complexity of $O((n+m)^3)$ where m is the total number of auxiliary variables involved in transforming all the higher order terms to quadratic ones. Specifically, we show that our approach for fourth order function requires only 2 auxiliary variables in contrast to 30 or more variables used in existing approaches. In the general case, we give an upper bound for the number or auxiliary variables required to transform a function of order k using Dedekind number, which is substantially lower than the existing bound of 2^{2^k}.

Keywords: submodular functions, quadratic pseudo-Boolean functions, monotonic Boolean functions, Dedekind number, max-flow/mincut algorithm

1 Introduction

Many optimization problems in several domains such as operations research, computer vision, machine learning, and computational biology involve *submodular* function minimization. Submodular functions (See Definition 1) are discrete analogues of convex functions [31]. Examples of such functions include cut capacity functions, matroid rank functions and entropy functions. Submodular function minimization techniques may be broadly classified into two categories: general algorithms for submodular functions and specialized algorithms for subclasses of submodular functions. This paper falls under the latter category.

General solvers: The role of submodular functions in optimization was first discovered by Edmonds when he gave several important results on the related poly-matroids [10]. Grötschel, Lovász and Schrijver first gave a polynomial-time algorithm for minimization of submodular function using ellipsoid method

[17]. Recently several combinatoric and strongly polynomial algorithms [13, 22, 23, 41] have been developed based on the work of Cunningham [9]. The current best strongly polynomial algorithm for minimizing general submodular functions [33] has a run-time complexity of $O(n^5L + n^6)$, where L is the time taken to evaluate the function and n is the number of variables. Weakly polynomial time algorithms with a smaller dependence on n also exist. For example, to minimize the submodular function $f(x)$ the scaling algorithm of Iwata [24] has a run-time complexity of $O(n^4L + n^5) \log M$. As before, L refers to the time required to compute the function f and M refers to the maximum absolute value of the function f.

Specialized solvers: There has been much recent interest in the use of higher order submodular functions for better modeling of computer vision and machine learning problems [26, 30, 21]. Such problems typical involve millions of pixels making the use of general solvers highly infeasible. Further, each pixel may take multiple discrete values and the conversion of such a problem to a Boolean one introduces further variables. On the other hand, the cost functions for many such optimization algorithms belong to a small subclass of submodular functions. The goal of this paper is to provide an efficient approach for minimizing these subclasses of submodular functions using a max-flow algorithm.

Definition 1. *Submodular functions map $f : \mathbb{B}^{\mathbf{V}} \rightarrow \mathbb{R}$ and satisfy the following condition:*

$$f(X) + f(Y) \geq f(X \vee Y) + f(X \wedge Y) \tag{1}$$

where X and Y are elements of $\mathbb{B}^n$

In this paper, we use a pseudo-Boolean polynomial representation for denoting submodular functions.

Definition 2. *Pseudo-Boolean functions (*PBF*) take a Boolean vector as argument and return a real number, i.e. $f : \mathbb{B}^n \rightarrow \mathbb{R}$ [3]. These can be uniquely expressed as multi-linear polynomials i.e. for all f there exists a unique set of real numbers $\{a_S : S \in \mathbb{B}^N\}$:*

$$f(x_1, ..., x_n) = \sum_{S \subseteq V} a_S (\prod_{j \in S} x_j), a_S \in \mathbb{R}, \tag{2}$$

where $a_\emptyset$ is said to be the constant term.

The term *order* refers to the maximum degree of the polynomial. A submodular function of second order involving Boolean variables can be easily represented using a graph such that the minimum cut, computed using a max-flow algorithm, also efficiently minimizes the function. However, max-flow algorithms can not exactly minimize non-submodular functions or some submodular ones of an order greater than 3 [45]. There is a long history of research in solving subclasses of submodular functions both exactly and efficiently using max-flow algorithms [1, 27, 18, 44, 35]. In this paper we propose a novel linear programming formulation that works in the following manner: given any pseudo Boolean function, it can derive a quadratic submodular formulation of the same cost, should one exist, suitable for solving with graph-cuts. Where such a quadratic submodular formulation does not exist, it will find the *closest* quadratic submodular function.

Definition 3. *$\mathcal{F}^k$ denotes a class of pseudo-Boolean functions of order k such that every function $f(\mathbf{x}) \in \mathcal{F}^k$ satisfies the submodularity property given in Definition 1.*

It was first shown in [18] that any function in $\mathcal{F}^2$ can be minimized exactly using a max-flow algorithm. Billionnet and Minoux [1] showed that any function in $\mathcal{F}^3$ can be transformed into a function in $\mathcal{F}^2$ using additional variables. A conflict-graph approach was employed to perform this transformation and the approach employs one auxiliary variable to transform every third-degree term in the function. [1] also notes that the same result was independently obtained by L.A. Wolsey around the same time and it

was not published. The approach of L.A. Wolsey did not use conflict graph and it is based on an extension of Balinski-Rhys property[37]. It was shown that a special subclass of higher order functions referred to as *negative-positive* pseudo-Boolean functions (where the coefficients of the linear terms have any sign, and higher degree terms have negative coefficients) can be converted to a max-flow problem [37]. It is important to note the contribution of Kolmogorov and Zabih in first establishing the connection to submodularity concepts for vision applications [27]. Kolmogorov and Zabih [27] showed how to transform a third order submodular function involving three variables to a function in $\mathcal{F}^2$. This approach is similar to the one by L.A. Wolsey and it uses an extension of Balinski-Rhys property. In all these techniques, we transform the higher order function to a function in $\mathcal{F}^2$ using extra variables, which we refer to as *auxiliary variables* (AV). In the course of this paper, you will see that these AVs are often more difficult to handle than variables in the original function and our algorithms are driven by the quest to understand the role of these auxiliary variables and to eliminate the unnecessary ones.

Kolmogorov improved the complexity of Iwata's capacity scaling algorithm [24] for special functions which are represented as a sum of submodular terms. This is the first line of research that does not use use auxiliary variables to handle higher order terms. The formulation of Kolmogorov also closely resembles the approach of Cooper [7], who used a linear program with an exponential number of constraints for solving the minimization of the submodular function.

Recently, Zivny et al. made substantial progress in characterizing the class of functions that can be transformed to $\mathcal{F}^2$. Their most notable result is to show that not all functions in $\mathcal{F}^4$ can be transformed to a function in $\mathcal{F}^2$. This result stands in strong contrast to the third order case that was positively resolved more than two decades earlier [1]. Using Theorem 5.2 from [34] it is possible to decompose a given submodular function in $\mathcal{F}^4$ into 10 different groups $\mathcal{G}_i, i = \{1..10\}$ where each $\mathcal{G}_i$ is shown in Table 6. Zivny et al. showed that one of these groups can not be expressed using any function in $\mathcal{F}^2$ employing any number of AVs. Most of these results were obtained by mapping the problem of minimizing submodular functions to a valued constraint satisfaction problem.

1.1 Problem Statement and main contributions

Largest subclass of submodular functions We are interested in transforming a given function in $\mathcal{F}^k$ into a function in $\mathcal{F}^2$ using AVs. As such a transformation is not possible for all submodular functions of order four or more [45], our goal is to implicitly map the largest subclass $\mathcal{F}_2^k$ that can be transformed into $\mathcal{F}^2$. This distinction between the two classes $\mathcal{F}_2^k$ and $\mathcal{F}^k$ will be crucial in the remainder of the paper (see Figure 1).

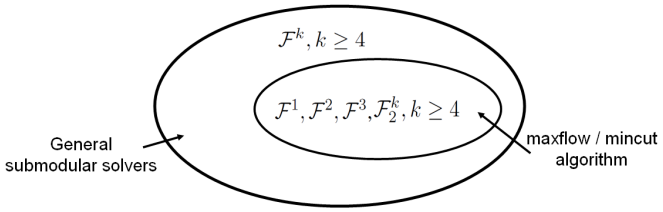


Fig. 1. *All the function in the classes $\mathcal{F}^1, \mathcal{F}^2, \mathcal{F}^3$ and $\mathcal{F}_2^k, k \geq 2$ can be transformed to functions in $\mathcal{F}^2$ and minimized using the maxflow/mincut algorithm.*

Definition 4. *The class $\mathcal{F}_2^k$ is the largest subclass of $\mathcal{F}^k$ such that every function $f(\mathbf{x}) \in \mathcal{F}_2^k$ has an equivalent quadratic function $h(\mathbf{x}, \mathbf{z}) \in \mathcal{F}^2$ using AVs $\mathbf{z} = z_1, z_2, ..., z_m \in \mathbb{B}^m$ satisfying the following*

condition:

$$f(\mathbf{x}) = \min_{\mathbf{z} \in \mathbb{B}^m} h(\mathbf{x}, \mathbf{z}), \quad \forall \mathbf{x}. \tag{3}$$

In this paper, we are interested in developing an algorithm to transform every function in this class $\mathcal{F}_2^k$ to a function in $\mathcal{F}^2$.

Efficient transformation of higher order functions: We propose a principled framework to transform higher order submodular functions to quadratic ones using a combination of monotonic Boolean functions(MBF)[8] and linear programming. This framework provides several advantages. First we show that the state of an AV in a minimum cost labeling is equivalent to an MBF defined over the original variables. This provides an upper bound on the number of AVs given by the Dedekind number [28], which is defined as the total number of MBFs over a set of n binary variables. In the case of fourth order functions, there are 168 such functions. Using the properties of MBFs and the nature of these AVs in our transformation, we prove that these 168 AVs can be replaced by two AVs.

Minimal use of AVs*:* One of our goals is to use a minimum number(m) of AVs in performing the transformation of (3). Although, given a fixed choice of $\mathcal{F}_2^k$, reducing the value of m does not change the complexity of the resulting min/cut algorithm asymptotically, it is crucial in several machine learning and computer vision problems. In general, most image based labeling problems involve millions of pixels and in typical problems, the number of fourth order priors is linearly proportional to the number of pixels. Such problems may be infeasible for large values of m. A recent work shows that the transformation of functions in $\mathcal{F}_2^4$ using about 31 additional nodes [46]. On the other hand, we show that we can transform the same class of functions using only 2 additional nodes. Note that this reduction is applicable to every fourth order term in the function. A typical vision problem may involve functions having 10000 $\mathcal{F}_2^4$ terms for an image of size 100×100. Under these parameters, our algorithm will use 20000 AVs, whereas the existing approach [46] would use as large as 310000 AVs. In several practical problems, this improvement will make a significant difference in the running time of the algorithm.

For a function in $\mathcal{F}_2^k$, the maximum number of AVs required is given by 2^{2^k}. We show that one can transform the function using substantially lower number of AVs given by Dedekind number. In section 3.1, we prove that the Dedekind number is substantially lower than 2^{2^k}. Cohen et al. [6] proposed an upper bound of 2^{2^k} for the maximal number of auxiliary variables using an LP based approach. However, the behaviour of many of the variables [1] considered is not consistent with a submodular function. In contrast, we prove an equivalence between the set of auxiliary variables and the set of monotonic functions, and show that the state of each AV must be a monotonic increasing function, and visa versa . This correspondence allows us to formulate a constructive method based on linear programming for generating the nearest function in $\mathcal{F}_2^k$ to any arbitrary function defined over $\mathbf{x}$.

We also use an LP-based approach, however the use of monotonic Boolean functions enables us to improve this bound to Dedekind number. The idea of reducing the number of AVs in an LP formulation has been done in other contexts [42], where a combinatorial structure commonly referred to as *gadgets* were computed using linear programming. This enables the transformation of constraints from one optimization problem to another. In this work, we show that we can transform a function with several AVs to a function involving much fewer AVs using a linear programming approach.

1.2 Limitations of Current Approaches and Open Problems

Decomposition of submodular functions: Many existing algorithms for transforming higher order functions target the minimization of a single k-variable k^{th} order function. However, the transformation frame-

[1] All those that are not monotonic increasing with respect to the original variables $\mathbf{x}$.

work is incomplete without showing that a given n-variable submodular function of k^{th} order can be decomposed into several individual k-variable k^{th} order sub-functions. Billionet proved that it is possible to decompose a function in $\mathcal{F}^3$ involving several variables into 3-variable functions in $\mathcal{F}^3$ [1]. To the best of our knowledge, the decomposition of fourth or higher order functions is still an open problem and it will remain a hard problem due to the following reasoning. In [16], it was proven that testing a membership of a function f with n variables in $\mathcal{F}^4$ is NP-complete. It is easy to test the submodularity of a fourth order function with 4 variables. Thus if a function f with n variables is decomposed into several 4-variable fourth order functions and if each of these individual 4-variable functions are submodular, then the function f is submodular. This seems to be most possible case when we know that a function is submodular. Thus it is very unlikely to know that a function is submodular and not know its decomposition. Given this, it is likely that specialized solvers based on max-flow algorithms may never solve the general class of submodular functions. However, this decomposition problem is not a critical issue in machine learning and vision problems. This is because the higher order priors from natural statistics already occur in different sub-functions of k nodes - in other words, the decomposition is known a priori. This paper only focuses on the transformation of a single k-variable function in $\mathcal{F}^k$. As mentioned above, the solution to this problem is still sufficient to solve large functions with hundreds of nodes and higher order priors in machine learning and vision applications.

Non-Boolean problems: The results in this paper are applicable only to set or pseudo-Boolean functions. Many real world problems involve variables that can take multiple discrete values. Ishikawa showed that it is possible to transform a multi-label second order function to a Boolean second order function function using Boolean variables to encode multi-label variables [20]. To denote a single multi-label variable with l labels, l Boolean variables were used. Ishikawa's method considered functions with convex priors, a class of functions that is slightly more restricted than general submodular functions. Schlesinger and Flach later showed that it is possible to transform general submodular multi-label functions of second order to Boolean second order functions [40]. This approach used $l-1$ Boolean variables to encode an l-label multi-label variable. Ramalingam et al.[28] generalized this work for transforming multi-label higher order functions to Boolean second-order functions. In [28], the transformation does not preserve submodularity for fourth or higher order functions[36]. Zivny et al.[45] recently proved that it is not possible to have a submodularity preserving transformation for fourth or higher order functions.

Excess AVs*:* The complexity of an efficient max-flow algorithm is $O((n+m)^3)$ where n is the number of variables in the original higher order function and m is the number of AVs. Typically in imaging problems, the number of higher order terms is of $O(n)$ and the order k is less than 10. Thus the minimization of the function corresponding to an entire image with $O(n)$ higher order terms will still have a complexity of $O((n+n)^3)$. However when m becomes at least quadratic in n, for example, if a higher-order term is defined over every triple of variables in V, the complexity of the max-flow algorithm will exceed that of a general solver being $O((n+n^3)^3)$. Thus in applications involving a very large number of higher order terms, a general solver may be more appropriate.

2 Notation and preliminaries

In what follows, we use a vector $\mathbf{x}$ to denote $\{x_1, x_2, x_3, ..., x_n\}$. Let $\mathbb{B}$ denote the Boolean set $\{0,1\}$ and $\mathbb{R}$ the set of reals. Let the vector $\mathbf{x} = (x_1, ..., x_n) \in \mathbb{B}^n$, and $\mathbf{V} = \{1, 2, ..., n\}$ be the set of indices of $\mathbf{x}$. Let $\mathbf{z} = (z_1, z_2, ..., z_k) \in \mathbb{B}^k$ denote the AVs. We introduce a *set representation* to denote the labellings of $\mathbf{x}$. Let $S_4 = \{1,2,3,4\}$ and let $\mathcal{P}$ be the power set of S_4. For example a labeling $\{x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 1)$ is denoted by the set $\{1,3,4\}$. For a subset $A \subseteq V$, let us denote by $\mathbf{1}^A \in \mathbb{B}^n$ its characteristic vector, i.e.

$$\mathbf{1}_j^S = \begin{cases} 1 & \text{if } j \in S \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Definition 5. *The (discrete) derivative of a function $f(x_1, \ldots, x_n)$ with respect to x_i is given by:*

$$\frac{\delta f}{\delta x_i}(x_1, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) - f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) \tag{5}$$

Definition 6. *The second discrete derivative of a function $\Delta_{i,j}(\mathbf{x})$ is given by*

$$\begin{aligned} \Delta_{i,j}(\mathbf{x}) &= \frac{\delta}{\delta x_j}\frac{\delta f}{\delta x_i}(x_1, \ldots, x_n) \\ &= \Big(f(x_1,\ldots,x_{i-1},1,x_{i+1},x_{j-1},1,x_{j+1}\ldots,x_n) - f(x_1,\ldots,x_{i-1},0,x_{i+1},x_{j-1},1,x_{j+1}\ldots,x_n) \Big) \\ &- \Big(f(x_1,\ldots,x_{i-1},1,x_{i+1},x_{j-1},0,x_{j+1}\ldots,x_n) - f(x_1,\ldots,x_{i-1},0,x_{i+1},x_{j-1},0,x_{j+1}\ldots,x_n) \Big). \end{aligned} \tag{6}$$

Note that it follows from the definition of submodular functions (1)*, that their second derivative is always non-positive for all* $\mathbf{x}$

3 Transforming functions in $\mathcal{F}_2^n$ to $\mathcal{F}^2$

Consider the following submodular function $f(\mathbf{x}) \in \mathcal{F}_2^n$ represented as a multi-linear polynomial:

$$f(\mathbf{x}) = \sum_{S \in B^n} a_S \Big(\prod_{j \in S} x_j\Big), a_S \in \mathbb{R} \tag{7}$$

Let us consider a function $h(\mathbf{x}, \mathbf{z}) \in \mathcal{F}^2$ where $\mathbf{z}$ is a set of AVs used to model functions in $\mathcal{F}_2^n$. Any general function in $\mathcal{F}^2$ can be represented as a multi-linear polynomial (consisting of linear and bi-linear terms involving all variables):

$$h(\mathbf{x}, \mathbf{z}) = \sum_i a_i\, x_i - \sum_{i,j:i>j} a_{i,j}\, x_i x_j + \sum_l a_l\, z_l - \sum_{l,m:l>m} a_{l,m}\, z_l z_m - \sum_{i,l} a_{i,l}\, x_i z_l \tag{8}$$

The negative signs in front of the bi-linear terms $(x_i x_j, z_l x_i, z_l z_m)$ emphasize that their coefficients $(-a_{ij}, -a_{il}, -a_{lm})$ must be non-positive if the function is submodular. We are seeking a function h such that:

$$f(\mathbf{x}) = \min_{\mathbf{z} \in \mathbb{B}^n} h(\mathbf{x}, \mathbf{z}), \forall \mathbf{x}. \tag{9}$$

Here the function $f(\mathbf{x})$ is known. We are interested in computing the coefficients $\mathbf{a}$, and in determining the number of auxiliary variables required to express a function as a pairwise submodular function. The problem is extremely challenging due to the inherent instability and dependencies within the problem – different choices of parameters cause auxiliary variables to take different states. To explore the space of possible solutions fully, we must characterize what states an AV takes.

3.1 Auxiliary Variables as Monotonic Boolean Functions

Definition 7. *A monotonic (increasing) Boolean function (*MBF*)* $m : \mathbb{B}^n \rightarrow \mathbb{B}$ *takes a Boolean vector as argument and returns a Boolean, s.t if* $y_i \leq x_i, \forall i \implies m(\mathbf{y}) \leq m(\mathbf{x})$

Lemma 1. *Let* $z_s(\mathbf{x})$ *be a function that takes an argument* $\mathbf{x}$ *and returns a Boolean as shown below:*

$$z_s(\mathbf{x}) = \arg\min_{z_s} \left(\min_{\mathbf{z}'} h(\mathbf{x}, \mathbf{z}', z_s) \right). \tag{10}$$

where $h(\mathbf{x}, \mathbf{z}', z_s)$ *is a submodular function defined in Equation 8. The function* $z_s(\mathbf{x})$ *that maps a Boolean vector* $\mathbf{x}$ *to the Boolean state of* z_s *is an* MBF *(See Definition 7), where* $\mathbf{z}'$ *is the set of all auxiliary variables except* z_s*.*

Proof. We consider a current labeling $\mathbf{x}$ with an induced labeling of $z_s = z_s(\mathbf{x})$. We first note

$$h'(\mathbf{x}, z_s) = \min_{\mathbf{z}'} h(\mathbf{x}, \mathbf{z}', z_s) \tag{11}$$

is a submodular function i.e. it satisfies (1). We now consider *increasing* the value of $\mathbf{x}$, that is given a current labeling $\mathbf{x}$ we consider a new labeling $\mathbf{x}^{(i)}$ such that

$$x_j^{(i)} = \begin{cases} 1 & \text{if } j = i \\ x_j & \text{otherwise.} \end{cases} \tag{12}$$

We wish to prove

$$z_s(\mathbf{x}^{(i)}) \geq z_s(\mathbf{x}) \ \forall \mathbf{x}, i. \tag{13}$$

Note that if $z_s(\mathbf{x}) = 0$ or $x_i = 1$ this result is trivial. This leaves the case: $z_s(\mathbf{x}) = 1$ and $x_i = 0$. It follows from (6) that:

$$\begin{aligned} &h'(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, 0) + h'(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, 1) \geq \\ &h'(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, 1) + h'(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, 0). \end{aligned} \tag{14}$$

As, by hypothesis, $z_s(\mathbf{x}) = 1$ and $x_i = 0$ we have:

$$h'(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, 0) \geq h'(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, 1). \tag{15}$$

Hence

$$\begin{aligned} &h'(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, 0) - h'(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, 0) \geq \\ &h'(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, 1) - h'(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, 1), \end{aligned} \tag{16}$$

and

$$h'(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, 0) \geq h'(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, 1). \tag{17}$$

Therefore $z_s(\mathbf{x}^{(i)}) = 1$. Repeated application of the statement gives $y_i \leq x_i, \forall i \implies z_s(\mathbf{y}) \leq z_s(\mathbf{x})$ as required □

Definition 8. *The Dedekind number* $M(n)$ *is the number of* MBF*s of n variables. Finding a closed-form expression for* $M(n)$ *is known as the Dedekind problem [25, 28].*

The Dedekind number of known values are shown below: $M(1) = 3$, this corresponds to the set of functions:

$$M_1(x_1) \in \{\mathbf{0}, \mathbf{1}, x_1\}, \tag{18}$$

where **0** and **1** are the functions that take any input and return 0 or 1 respectively. $M(2) = 6$ corresponding to the set of functions:

$$M_2(x_1, x_2) = \{\mathbf{0}, \mathbf{1}, x_1, x_2, x_1 \vee x_2, x_1 \wedge x_2\} \tag{19}$$

Similarly, $M(3) = 20$, $M(4) = 168$, $M(5) = 7581$, $M(6) \approx 7.8 \times 10^6$, $M(7) \approx 2.4 \times 10^{12}$, and $M(8) \approx 5.6 \times 10^{23}$.

Lemma 2. *On transforming the largest graph-representable subclass of k^{th} order function to pairwise Boolean function, the upper bound on the maximal number of required* AV*s is given by the Dedekind number $D(k)$.*

Proof. The proof is straightforward. Consider a general multinomial, of similar form to equation (7), given in Page 6 with more than $D(k)$ AVs. It follows from lemma 1 that at least 2 of the AVs must correspond to the same MBF, and always take the same values. Hence, all references to one of these AV in the pseudo-Boolean representation can be replaced with references to the other, without changing the associated costs. Repeated application of this process will leave us with a solution with at most $D(k)$ AVs. □

Although this upper bound is large for even small values of k, it is much tighter than the existing upper bound of $S(k) = 2^{2^k}$ [6] (Also see Proposition 24 in [47]).

Lemma 3. *The Dedekind number $D(k) \ll S(k)$ where $S(k) = 2^{2^k}$ for all positive values of k.*

Proof. For even small values of $k = \{3, ..., 8\}$ the upper bound using Dedekind's number is much tighter compared to $S(k)$:$(M(3) = 20, S(3) = 256)(M(4) = 168, S(4) = 65536), (M(5) = 7581, S(5) \approx 4.29 \times 10^9), (M(6) \approx 7.8 \times 10^6, S(6) \approx 1.85 \times 10^{19}), (M(7) \approx 2.4 \times 10^{12}, S(7) \approx 3.4 \times 10^{38}$ and $(M(8) \approx 5.6 \times 10^{23}, S(8) \approx 1.156 \times 10^{77})$. For $k > 8$, $D(k)$ remains unknown, and the development of a closed form solution remains an active area of research.

In what follows, we will prove that the relation $D(k) \ll S(k)$ holds true for $k > 8$. Several upper bounds have been derived for $D(k)$ and we use the following bound by Hansel [19, 25] to prove our result.

$$D(k) \leq 3^{\binom{k}{\lfloor \frac{k}{2} \rfloor}} \tag{20}$$

$$D(k) \leq 2^{\log_2(3)\binom{k}{\lfloor \frac{k}{2} \rfloor}} \tag{21}$$

In order to show $D(k) \ll 2^{2^k}$ we prove that $\log_2(3)\binom{k}{\lfloor \frac{k}{2} \rfloor} \ll 2^k$. where $\binom{k}{\lfloor \frac{k}{2} \rfloor}$ is given below using factorials:

$$\binom{k}{\lfloor \frac{k}{2} \rfloor} = \frac{k!}{\lfloor k/2 \rfloor! (k - \lfloor k/2 \rfloor)!}$$

We use the following Stirling's approximation (or the Stirling's formula) for replacing the factorials from the above equation.

$$\sqrt{2\pi} k^{k+1/2} e^{-k} \leq k! \leq e k^{k+1/2} e^{-k}$$

Therefore

$$\frac{k!}{\lfloor k/2 \rfloor!(k-\lfloor k/2 \rfloor)!} \leq \frac{e}{2\pi} \frac{k^{k+1/2}}{((k/2)^{k/2+1/2})^2} \tag{22}$$

$$\leq \frac{e}{2\pi} \exp((k+1/2)\ln k - 2(k/2+1/2)\ln k/2) \tag{23}$$

$$\leq \frac{e}{2\pi} \exp((k+1/2)\ln k - (k+1)(\ln k - \ln 2)) \tag{24}$$

$$\leq \frac{e}{2\pi} \exp((-1/2)\ln k + (k+1)\ln 2) \tag{25}$$

$$\leq \frac{e}{2\pi\sqrt{k}} exp((k+1)\ln 2) \tag{26}$$

$$\leq \frac{e}{\pi\sqrt{k}} 2^k \tag{27}$$

$$\leq \frac{e}{\sqrt{2}\pi\sqrt{\frac{k}{2}}} 2^k \tag{28}$$

Since $\frac{\sqrt{2}\pi}{e} > \log_2 3$ we have the following relation:

$$\binom{k}{\lfloor k/2 \rfloor} \log_2 3 \leq \frac{2^k}{\sqrt{\frac{k}{2}}} \tag{29}$$

This implies that

$$D(k) \leq S(k)^{\frac{2}{k}}, \quad \forall k \geq 2 \tag{30}$$

We have already shown that $D(k) \ll S(k)$ for $k <= 8$. Thus we have shown that $D(k) \ll S(k)$ for all non-negative values of k.

In [48], the problem of improving this upper bound was mentioned as an open problem. Both these upper bounds are not practically feasible for even small values of k. In section 5, we will show that fourth order functions can be transformed with only two AVs.

Note that this representation of AVs as MBF is over-complete, for example if the MBF of a auxiliary variable z_i is the constant function $z_i(\mathbf{x}) = \mathbf{1}$ we can replace $\min_{\mathbf{z},z_i} h(\mathbf{x},\mathbf{z},z_i)$ with the simpler (i.e. one containing less auxiliary variables) function $\min_{\mathbf{z}} h(\mathbf{x},\mathbf{z},1)$. Despite this, this is sufficient preliminary work for our main result:

Theorem 1. *Given any function f in $\mathcal{F}_2^k$, the equivalent pairwise form $f' \in \mathcal{F}^2$ can be found by solving a linear program.*

The construction of the linear program is given in the following section.

4 The Linear Program

A sketch of the formulation can be given as follows: In general, the presence of AVs of indeterminate state, given a labeling $\mathbf{x}$ makes the minimizing an LP non-convex and challenging to solve directly. Instead of optimizing this problem containing AVs of unspecified state, we create an auxiliary variable associated with every MBF. Hence given any labeling $\mathbf{x}$ the state of every auxiliary variable is fixed a priori, making the problem convex. We show how the constraints that a particular AV must conform to a given MBF can be formulated as linear constraints, and that consequently the problem of finding the closest member of $f' \in \mathcal{F}^2$ to any pseudo Boolean function is a linear program.

This program will make use of the max-flow linear program formulation to guarantee that the minimum cost labeling of the AVs corresponds to their MBFs. To do this we must first rewrite the cost of equation (8), given in page 6, in a slightly different form. In order to do this, we consider a directed network graph $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ where s and t refer to the source and the sink of the network N. Let the capacity of the edge $(i, j) \in \mathcal{E}$ be given be non-negative weights $c(i, j)$, which refers to the maximum amount of flow on the edge (i, j). We write:

$$\begin{aligned} f(\mathbf{x}, \mathbf{z}) = c_\emptyset + \sum_i c_{s,i}\,(1 - x_i) + \sum_i c_{i,t}\,x_i + \sum_{i,j:i>j} c_{i,j}\,x_i(1 - x_j) \\ + \sum_l c_{s,l}\,(1 - z_l) + \sum_l c_{l,t}\,z_l + \sum_{l,m:l>m} c_{l,m}\,z_l\,(1 - z_m) + \sum_{i,l} c_{i,l}\,x_i\,(1 - z_l) \end{aligned} \tag{31}$$

where $c_\emptyset$ is a constant that may be either positive or negative. By [11], this form is equivalent to that of (8), in that any function that can be written in form (8), can also be written as (31) and visa versa.

4.1 The Max-flow Linear Program

Under the assumption that $\mathbf{x}$ is fixed, we are interested in finding a minima of the equation:

$$\begin{aligned} f_{\mathbf{x}}(\mathbf{z}) &= c_\emptyset + \sum_i c_{s,i}\,(1 - x_i) + \sum_i c_{i,t}\,x_i + \sum_{i,j:i>j} c_{i,j}\,x_i(1 - x_j) \\ &+ \sum_l c_{s,l}\,(1 - z_l) + \sum_l c_{l,t}\,z_l + \sum_{l,m:l>m} c_{l,m}\,z_l\,(1 - z_m) + \sum_{i,l} c_{i,l}\,x_i\,(1 - z_l) \\ &= d_{\mathbf{x},\emptyset} + \sum_l d_{\mathbf{x},s,l}\,(1 - z_l) + \sum_l d_{\mathbf{x},l,t}\,z_l + \sum_{l,m:l>m} d_{\mathbf{x},l,m}\,z_l\,(1 - z_m) \end{aligned} \tag{32}$$

where

$$d_{\mathbf{x},\emptyset} = c_\emptyset + \sum_{i:x_i=0} c_{s,i} + \sum_{i:x_i=1} c_{i,t} + \sum_{i,j:i>j \wedge x_i=1 \wedge x_j=0} c_{i,j} \tag{33}$$

$$d_{\mathbf{x},s,l} = c_{s,l} + \sum_{i:x_i=1} c_{i,l},\ d_{\mathbf{x},l,t} = c_{l,t} \text{ and } d_{\mathbf{x},l,m} = c_{l,m}. \tag{34}$$

Then the minimum cost of equation (31), given in page 10, may be found by solving its dual max-flow program. Writing $\nabla_{\mathbf{x},s}$ for flow from source, and $\nabla_{\mathbf{x},t}$ for flow to the sink, we seek

$$\max \nabla_{\mathbf{x},s} + d_{\mathbf{x},\emptyset} \tag{35}$$

Subject to the constraints that

$$\begin{aligned} f_{\mathbf{x},ij} - d_{\mathbf{x},ij} &\leq 0 \ \forall (i,j) \in E \\ \textstyle\sum_{j:(j,i)\in E} f_{\mathbf{x},ji} - \sum_{j:(i,j)\in E} f_{\mathbf{x},ij} &\leq 0 \quad \forall i \neq s,t \\ \nabla_{\mathbf{x},s} + \textstyle\sum_{j:(s,j)\in E} f_{\mathbf{x},sj} - \sum_{j:(s,j)\in E} f_{\mathbf{x},sj} &\leq 0 \\ \nabla_{\mathbf{x},t} + \textstyle\sum_{j:(j,t)\in E} f_{\mathbf{x},jt} - \sum_{j:(j,t)\in E} f_{\mathbf{x},jt} &\leq 0 \\ f_{\mathbf{x},ij} &\geq 0 \quad (i,j) \in E \end{aligned} \tag{36}$$

where E is the set of all ordered pairs $(l, m) : \forall l > m$, $(s, l) : \forall l$ and $(l, t) : \forall t$, and $f_{\mathbf{x},i,j}$ corresponds to the flow through the edge (i, j).

We will not use this exact LP formulation, but instead rely on the fact that $f_{\mathbf{x}}(\mathbf{z})$ *is a minimal cost labeling if and only if there exists a flow satisfying constraints (36) such that*

$$f_{\mathbf{x}}(\mathbf{z}) - \nabla_{\mathbf{x},s} - d_{\mathbf{x},\emptyset} \leq 0. \tag{37}$$

4.2 Choice of MBF as a set of linear constraints

We are seeking minima of a quadratic pseudo Boolean function of the form (31), where $\mathbf{x}$ is the variables we are interested in minimizing and $\mathbf{z}$ the auxiliary variables. As previously mentioned, formulations that allow the state of the auxiliary variable to vary tend to result in non-convex optimization problems. To avoid such difficulties, we specify as the location of minima of $\mathbf{z}$ as a set of hard constraints. We want that:

$$\min_{\mathbf{z}} f_{\mathbf{x}}(\mathbf{z}) = f_{\mathbf{x}}([m_1(\mathbf{x}), m_2(\mathbf{x}), \dots m_{D(k)}(\mathbf{x})]) \ \forall \mathbf{x}. \tag{38}$$

where $f_{\mathbf{x}}$ is defined as in (32), and $m_1, \dots m_{D(k)}$ are the set of all possible MBFs defined over $\mathbf{x}$. By setting all of the capacities $d_{i,j}$ to 0, it can be seen that a solution satisfying (38) must exist. It follows from the reduction described in lemma 1, and that all functions that can be expressed in a pairwise form can also be expressed in a form that satisfies these restrictions.

We enforce condition (38) by the set of linear constraints (36) and (37) for all possible choices of $\mathbf{x}$. Formally we enforce the condition

$$f_{\mathbf{x}}([m_1(\mathbf{x}), \dots, m_{D(k)}(\mathbf{x})]) - \nabla_{\mathbf{x},s} - d_{\mathbf{x},\emptyset} \leq 0. \tag{39}$$

Substituting in (32) we have 2^k sets of conditions, namely,

$$\sum_l d_{\mathbf{x},s,l}\,(1 - m_l(\mathbf{x})) + \sum_l d_{\mathbf{x},l,t}\,m_l(\mathbf{x}) + \sum_{l,m:l>m} d_{\mathbf{x},l,m}\,m_l(\mathbf{x})\,(1 - m_m(\mathbf{x})) - \nabla_{\mathbf{x},s} \leq 0, \tag{40}$$

subject to the set of constraints (36) for all $\mathbf{x}$. Note that we make use of the max-flow formulation, and not the more obvious min-cut formulation, as this remains a linear program even if we allow the capacity of edges $d_{\mathbf{x},l,m}$ to vary.

Submodularity Constraints We further require that the quadratic function is submodular or equivalently, the capacity of all edges $c_{i,j}$ is non-negative. This can be enforced by the set of linear constraints that

$$c_{i,j} \geq 0, \ \forall i, j. \tag{41}$$

4.3 Finding the nearest submodular Quadratic Function

We now assume that we have been given an arbitrary function $g(\mathbf{x})$ to minimize, that may or may not lie in $\mathcal{F}^k$. We are interested in finding the closest possible function in $\mathcal{F}^2$ to it. To find the closest function to it (under the L_1 norm), we minimize:

$$\min_{\mathbf{c}} \sum_{\mathbf{x}\in\mathbb{B}^k} \Big| g(\mathbf{x}) - \min_{\mathbf{z}} f(\mathbf{x}, \mathbf{z}) \Big| = \tag{42}$$

$$\min_{\mathbf{c}} \sum_{\mathbf{x}\in\mathbb{B}^k} \Big| g(\mathbf{x}) - f(\mathbf{x}, \mathbf{m}(\mathbf{x})) \Big| = \tag{43}$$

$$\begin{aligned} \min_{\mathbf{c}} \sum_{\mathbf{x}\in\mathbb{B}^k} \Big| g(\mathbf{x}) - \big(& c_\emptyset + \sum_i c_{i,s}\,(1 - x_i) + \sum_i c_{t,i}\,x_i + \sum_{i,j:i>j} c_{i,j}\,x_i(1 - x_j) \\ & + \sum_l c_{s,l}\,(1 - m_l(\mathbf{x})) + \sum_l c_{l,t}\,m_l(\mathbf{x}) + \sum_{l,m:l>m} c_{l,m}\,m_l(\mathbf{x})\,(1 - m_m(\mathbf{x})) \\ & + \sum_{i,l} c_{i,l}\,x_i\,(1 - m_l(\mathbf{x})) \big) \Big| \end{aligned} \tag{44}$$

where $\mathbf{m}(\mathbf{x}) = [m_1(\mathbf{x}), \dots, m_{D(k)}(\mathbf{x})]$ is the vector of all MBFs over $\mathbf{x}$, and subject to the family of constraints set out in the previous subsection. Note that expressions of the form $\sum_i |g_i|$ can be written as $\sum_i h_i$ subject to the linear constraints $h_i > g_i$ and $h_i > -g_i$ and this is a linear program. □

4.4 Discussion

Several results follow from the linear program described in the previous section. In particular, if we consider a function g of the same form as equation (2), given in page 2, such that

$$\min_{\mathbf{c}} \sum_{\mathbf{x}\in\mathbb{B}^k} \Big| g(\mathbf{x}) - \min_{\mathbf{z}} f(\mathbf{x},\mathbf{z}) \Big| = 0 \tag{45}$$

exactly defines a linear polytope for any choice of $|\mathbf{x}| = k$, and this result holds for any choice of basis functions.

Of equal note, the convex-concave procedure [43], is a generic move-making algorithm that finds local optima by successively minimizing a sequence of convex (i.e. tractable) upper-bound functions that are tight at the current location ($\mathbf{x}'$). [32] showed how this could be similarly done for quadratic Boolean functions, by decomposing them into submodular and supermodular components. The work [29] showed that any function could be decomposed into a quadratic submodular function, and an additional overestimated term. Nevertheless, this decomposition was not optimal, and they did not suggest how to find a optimal overestimation. The optimal overestimation which lies in $\mathcal{F}^2$ for a cost function defined over a clique $\mathbf{g}$ may be found by solving the above LP subject to the additional requirements:

$$g(\mathbf{x}) \leq f(\mathbf{x},\mathbf{z}) \ \forall \mathbf{x} \tag{46}$$

$$g(\mathbf{x}') \geq f(\mathbf{x}',\mathbf{z}) \tag{47}$$

Efficiency concerns As we consider larger cliques, it becomes less computationally feasible to use the techniques discussed in this section, at least without pruning the number of auxiliary variables considered. As previously mentioned, constant AVs and AVs that corresponds to that of a single variable in x i.e. $z_l = x_i$ can be safely discarded without loss of generality. In the following section, we show that a function in $\mathcal{F}_2^4$ can be represented by only two AVs, rather than 168 as suggested by the number of possible MBF. However, in the general case a minimal form representation eludes us. As a matter of pragmatism, it may be useful to attempt to solve the LP of the previous section without making use of any AV, and to successively introduce new variables, until a minimum cost solution is found.

5 Transforming functions in $\mathcal{F}_2^4$ to $\mathcal{F}^2$

Consider the following submodular function $f(x_1,x_2,x_3,x_4) \in \mathcal{F}^4$ represented as a multi-linear polynomial:

$$f(x_1,x_2,x_3,x_4) = a_0 + \sum_i a_i x_i + \sum_{i>j} a_{ij} x_i x_j + \sum_{i>j>k} a_{ijk} x_i x_j x_k + a_{1234} x_1 x_2 x_3 x_4, \quad \Delta_{ij}(\mathbf{x}) \leq 0 \tag{48}$$

where $i,j,k = S_4$ and $\Delta_{ij}(\mathbf{x})$ is the discrete second derivative of $f(\mathbf{x})$ with respect to x_i and x_j.

Consider a function $h(x_1,x_2,x_3,x_4,z_s) \in \mathcal{F}^2$ where z_s is an AV used to model functions in $\mathcal{F}^4$. Any general function in $\mathcal{F}^2$ can be represented as a multi-linear polynomial (consisting of linear and bilinear terms involving all five variables):

$$h(x_1,x_2,x_3,x_4,z_s) = b_0 + \sum_i b_i x_i - \sum_{i>j} b_{ij} x_i x_j + (g_s - \sum_{i=1}^{4} g_{s,i} x_i) z_s, \qquad b_{ij} \geq 0, g_{s,i} \geq 0, i,j \in S_4. \tag{49}$$

The negative signs in front of the bilinear terms $(x_i x_j, z_s x_i)$ emphasize that their coefficients $(-b_{ij}, -g_{s,i})$ must be non-positive to ensure submodularity. We have the following condition from equation (3), given in page 4:

$$f(x_1,x_2,x_3,x_4) = \min_{z_s \in \mathbb{B}} h(x_1,x_2,x_3,x_4,z_s), \forall \mathbf{x}. \tag{50}$$

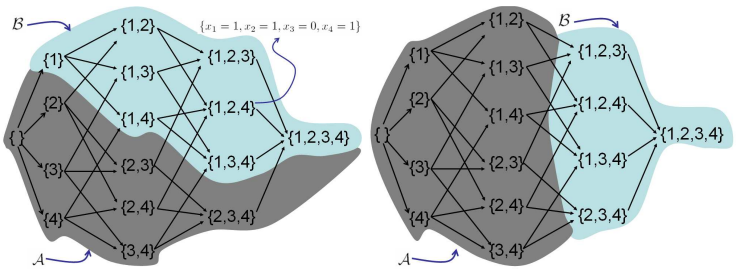


Fig. 2. We show some examples of partitions using Hasse diagrams. Here, we use set representation for denoting the labellings of (x_1, x_2, x_3, x_4). For example the set $\{1, 2, 4\}$ is equivalent to the labeling $\{x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 1\}$. In (a), $\mathcal{A} = \{\{\}, \{2\}, \{3\}, \{4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}, \{2, 3, 4\}\}$ and $\mathcal{B} = \{\{1\}, \{1, 2\}, \{1, 3\}, \{1, 4\}, \{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 4\}, S_4\}$. (a) and (b) are examples of partitions. On searching the space of all possible partitions (2^{16}) we found that only 168 partitions belong to this class. These are the only partitions which will be useful in our analysis because any arbitrary AV must be associated with one of these 168 partitions. (See text for the relation between these partitions and MBFs).

Here the coefficients $(a_i, a_{ij}, a_{ijk}, a_{ijkl})$ in the function $f(\mathbf{x})$ are known. We wish to compute the coefficients $(b_i, b_{ij}, g_s, g_{s,n})$ where $i, j \in \mathbf{V}, i \neq j, n \in S_4$. If we were given $(g_s, g_{s,i})$ then from equations (49) and (50) we would have

$$z_s = \begin{cases} 1 & \text{if } g_s - \sum_{i=1}^{4} g_{s,i} x_i < 0, \\ 0 & \text{otherwise.} \end{cases} \tag{51}$$

Our main result is to prove that any function $h \in \mathcal{F}^2$ can be transformed to a function $h'(x_1, x_2, x_3, x_4, z_f, z_b) \in \mathcal{F}^2$ involving only two auxiliary variables z_f and z_b. Using this result we can transform a given function $f(x_1, x_2, x_3, x_4) \in \mathcal{F}_2^4$, the form of which we characterize later, to a function $h'(x_1, x_2, x_3, x_4, z_f, z_b) \in \mathcal{F}^2$.

Let $\mathcal{A}$ be the family of sets corresponding to labellings of $\mathbf{x}$ such that:$z_s = 0 = \arg\min_{z_s} h(\mathbf{x}, z_s)$. In the same way let $\mathcal{B}$ be the family of sets corresponding to labellings of $\mathbf{x}$ such that:$z_s = 1 = \arg\min_{z_s} h(\mathbf{x}, z_s)$. These sets $\mathcal{A}$ and $\mathcal{B}$ partition $\mathbf{x}$, as defined below:

Definition 9. *A partition divides $\mathcal{P}$ into sets $\mathcal{A}$ and $\mathcal{B}$ such that $\mathcal{A} = \{\mathcal{S}(\mathbf{x}) : 0 = \arg\min_{z \in \mathbb{B}} h(\mathbf{x}, z), \mathbf{x} \in \mathbb{B}^4\}$ and $\mathcal{B} = \mathcal{P} \backslash \mathcal{A}$. Note that $\emptyset \in \mathcal{A}$.*

In the rest of the paper, we say that the AV z_s is associated with $[\mathcal{A}, \mathcal{B}]$ or denote it by $z_s : [\mathcal{A}, \mathcal{B}]$. We illustrate the concept of a *partition* in figure 2. For simplicity, we use the function $\kappa(s, B) = (g_s - \sum_{i=1}^{4} g_{s,i} \mathbf{1}_i^B) \leq 0$ where $B \in \mathcal{B}_s$.

From lemma 2, we could use 168 different AVs in our transformation. However, we show that the same class can be represented using only two AVs. In other words, all existing partitions could be converted to these two reference partitions represented by two AVs taking the states shown below.

If set $B \in \mathcal{B}_s$ then $\kappa(s, B) = (g_s - \sum_{i=1}^{4} g_{s,i} \mathbf{1}_i^B) \leq 0$

Definition 10. *The* **forward reference partition** $[\mathcal{A}_f, \mathcal{B}_f]$ *takes the form:*

$$B \in \mathcal{B}_f \iff |B| \geq 3, \mathcal{A}_f = \mathcal{P} \backslash \mathcal{B}_f \tag{52}$$

On the other hand, a **backward reference partition** $[\mathcal{A}_b, \mathcal{B}_b]$ *is shown below:*

$$B \in \mathcal{B}_b \iff |B| \geq 2, \mathcal{A}_b = \mathcal{P} \backslash \mathcal{B}_b \tag{53}$$

The forward and backward reference partitions are shown in figure 3.

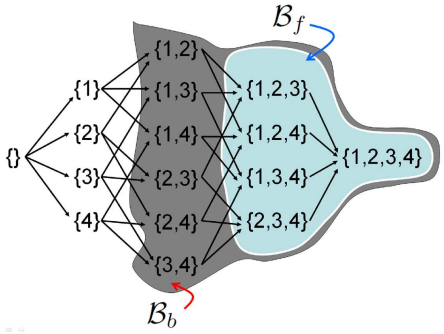


Fig. 3. *The two reference partitions used to represent all the functions in $\mathcal{F}_2^4$ are shown.*

We approach this problem by first considering the simplified case in which no interactions between AVs are allowed. This is covered in section 5.1, while section 5.2 builds on these results to handle the case of pairwise interactions between AV.

5.1 Non-interacting AVs

In this section we study the role of AV in the function $h(\mathbf{x}, \mathbf{z})$ that is linear in z_i. We do not consider energy terms having $z_i z_j$.

We show several results that allow us to replace multiple AVs to fewer AVs.

Definition 11. *We say that a function $h(\mathbf{x}, \mathbf{z})$ can be transformed to another function $h'(\mathbf{x}, \mathbf{z}')$ where $\mathbf{z} \neq \mathbf{z}'$ if the following condition is satisfied:*

$$\min_{\mathbf{z}} h(\mathbf{x}, \mathbf{z}) = \min_{\mathbf{z}'} h'(\mathbf{x}, \mathbf{z}'), \forall \mathbf{x} \tag{54}$$

where $\mathbf{z}$ and $\mathbf{z}'$ are auxiliary variables with different partitions. The cardinality of $\mathbf{z}$ need not be equal to the cardinality of $\mathbf{z}'$.

Through a sequence of transformations of the above form, we start with a general function $h(\mathbf{x}, \mathbf{z})$ and finally compute a function $h'(\mathbf{x}, z_f, z_b)$ with only two AVs in reference partitions.

Lemma 4. *Let $z_s : [\mathcal{A}_s, \mathcal{B}_s]$ be an AV in a function $h(\mathbf{x}, z_s)$ in $\mathcal{F}^2$, then h can be transformed to some function $h'(\mathbf{x}, z_t)$ in $\mathcal{F}^2$ involving $z_t : [\mathcal{A}_t, \mathcal{B}_t]$, such that for all $B \in \mathcal{B}_t$, $|B| \geq 2$.*

Proof. We say that a function h can be transformed to h' if $\min_{z_s} h(\mathbf{x}, z_s) = \min_{z_t} h'(\mathbf{x}, z_t), \forall \mathbf{x}$. It does not imply that $h(\mathbf{x}, z_s) = h'(\mathbf{x}, z_t), \forall \mathbf{x}$. We first consider the case where $\emptyset \in \mathcal{B}_s$. If this is the case, $\arg\min_{z_t} h'(\mathbf{x}, z_t) = 1 \ \forall \mathbf{x}$. Hence we can transform $h(\mathbf{x}, z_s)$ to $h'(\mathbf{x})$ and the lemma holds trivially. Next we assume that there exists a singleton $\{e\} \in \mathcal{B}_s$, i.e. $\{e\}$ is $\{1\}$,$\{2\}$,$\{3\}$ or $\{4\}$. We decompose h as:

$$\min_{z_s} h(x_1, x_2, x_3, x_4, z_s) = h_1(x_1, x_2, x_3, x_4) + \min_{z_s} \underbrace{(g_s - \sum_{i=1}^{4} g_{s,i} x_i) z_s}_{h_2}$$

where h_2 is the part of h dependent on z_s.

$$\min_{z_s} h_2 = \min_{z_s}((g_s - g_{s,e})x_e z_s + (g_s - g_s x_e - \sum_{i=S_4 \setminus e} g_{s,i}x_i)z_s).$$

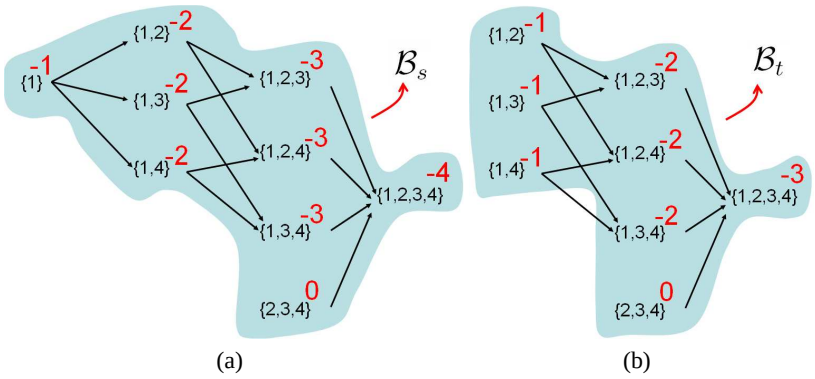
As $\{e\} \in \mathcal{B}_s$, $g_s - g_{s,e} \leq 0$. As a result, $z_s = 1$ when $x_e = 1$, i.e. $x_e \implies z_s$ or $x_e z_s = x_e$. In the above equation we replace $x_e z_s$ using simply x_e to obtain the following:

$$\min_{z_s} h_2 = \min_{z_s}((g_s - g_{s,e})x_e + (g_s - g_s x_e - \sum_{i=S_4 \setminus e} g_{s,i}x_i)z_s).$$

The decomposition of the original function can then be written, replacing z_s by z_t:

$$h' = \underbrace{h_1 + (g_s - g_{s,e})x_e)}_{h'_1} + \underbrace{(g_s - g_s x_e - \sum_{i=S_4 \setminus e} g_{s,i}x_i)z_t}_{h'_2}.$$

A sample reduction for this lemma is shown in figure 4. Note that h'_2 equals 0 for the singleton $\{e\}$. Although z_t could take either 0 or 1 for the singleton $\{e\}$ we consider $z_t = 0$. Thus $\{e\}$ is not present in $\mathcal{B}_t$. Similarly any other singleton $\{e'\}$ can also be removed from $\mathcal{B}_s$ using the same approach. After repeated application, our final partition, $\mathcal{B}_t$ does not contain any singletons. □



(a) (b)

Fig. 4. *An example of lemma 4. The* AV z_s *is replaced by* z_t *and the associated partitions* $[\mathcal{A}_s, \mathcal{B}_s]$ *and* $[\mathcal{A}_t, \mathcal{B}_t]$ *are shown in (a) and (b) respectively. The initial and the final set of parameters are given by:*$(g_s = 3, g_{s,1} = 4, g_{s,2} = 1, g_{s,3} = 1, g_{s,4} = 1), (g_t = 3, g_{t,1} = 3, g_{t,2} = 1, g_{t,3} = 1, g_{t,4} = 1)$. *In the initial partition we have the singleton* $\{1\} \in \mathcal{B}_s$. *After the transformation all the singletons* $\{e\} \in \mathcal{A}_t$.

Lemma 5. *Any function* $h(\mathbf{x}, z_s)$ *in* $\mathcal{F}^2$ *with* z_s *associated with the partition* $[\mathcal{A}_s, \mathcal{B}_s]$ *satisfying the condition* $\mathcal{B}_s \subseteq \mathcal{B}_f$ *can be transformed to some function* $h'(\mathbf{x}, z_f)$ *where* z_f *belongs to the forward partition. In the same manner, any function* $h(\mathbf{x}, z_s)$ *in* $\mathcal{F}^2$ *with* z_s *associated with the partition* $[\mathcal{A}_s, \mathcal{B}_s]$ *satisfying the condition* $\mathcal{A}_s \subseteq \mathcal{A}_b$ *can be transformed to some function* $h'(\mathbf{x}, z_b)$ *where* z_b *belongs to the backward partition.*

Proof. First we will prove the case when $\mathcal{B}_s \subseteq \mathcal{B}_f$. The proof is by construction. Let the parameters of the partition $[\mathcal{A}_s, \mathcal{B}_s]$ be $(g_s, g_{s,1}, g_{s,2}, g_{s,3}, g_{s,4})$. Our goal is to compute a new set of parameters

$(g_f, g_{f,1}, g_{f,2}, g_{f,3}, g_{f,4})$ corresponding to the forward reference partition such that the associated functions keep the same value at the minimum:

$$\min_{z_f} h'(\mathbf{x}, z_f) = \min_{z_s} h(\mathbf{x}, z_s), \forall \mathbf{x} \tag{55}$$

$$\min_{z_f} (h'_1(\mathbf{x}) + h'_2(\mathbf{x}, z_f)) = \min_{z_s} (h_1(\mathbf{x}) + h_2(\mathbf{x}, z_s)), \forall \mathbf{x} \tag{56}$$

$$\min_{z_f} (h'_2(\mathbf{x}, z_f)) = \min_{z_s} (h_2(\mathbf{x}, z_s)), \forall \mathbf{x} \tag{57}$$

We can rewrite h_2 and h'_2 using κ function:

$$\min_{z_f} \kappa(f, S) z_f = \min_{z_s} \kappa(s, S) z_s, \forall S \in \mathcal{P} \tag{58}$$

By substituting the values of z_s and z_f for all $S \in \mathcal{P}$ we obtain five equations with five unknowns $(g_f, g_{f,1}, g_{f,2}, g_{f,3}, g_{f,4})$. We rewrite the equations as:

$$\underbrace{\begin{pmatrix} 1 & 0 & -1 & -1 & -1 \\ 1 & -1 & 0 & -1 & -1 \\ 1 & -1 & -1 & 0 & -1 \\ 1 & -1 & -1 & -1 & 0 \\ 1 & -1 & -1 & -1 & -1 \end{pmatrix}}_{\mathcal{H}} \begin{pmatrix} g_f \\ g_{f,1} \\ g_{f,2} \\ g_{f,3} \\ g_{f,4} \end{pmatrix} = \begin{pmatrix} \min(0, \kappa(s, \{2,3,4\})) \\ \min(0, \kappa(s, \{1,3,4\})) \\ \min(0, \kappa(s, \{1,2,4\})) \\ \min(0, \kappa(s, \{1,2,3\})) \\ \min(0, \kappa(s, S_4)) \end{pmatrix} \tag{59}$$

The solution to the above linear system is unique because $\mathcal{H}$ is of rank 5. Now we show that the solution satisfies submodularity condition and corresponds to the forward reference partition. Submodularity is ensured by the constraint that the parameters $(g_{f,1}, g_{f,2}, g_{f,3}, g_{f,4})$ are all non-negative. Using equation (59) and the non-negativity of original variables $(g_{s,i})$ we obtain the following:

$$g_{f,i} = \min(0, \kappa(s, S_4 \backslash i)) - \min(0, \kappa(s, S_4)) \tag{60}$$

$$\kappa(s, S_4) \leq \kappa(s, S_4 \backslash i) \tag{61}$$

From these equations we can show that $g_{f,i}$ is always non-negative:

$$g_{f,i} = \begin{cases} 0 & \text{if } \kappa(s, S_4) \geq 0 \text{ and } \kappa(s, S_4 \backslash i) \geq 0 \\ -\kappa(s, S_4) & \text{if } \kappa(s, S_4) \leq 0 \text{ and } \kappa(s, S_4 \backslash i) = 0 \\ \kappa(s, S_4 \backslash i) - \kappa(s, S_4) & \text{if } \kappa(s, S_4) \leq 0 \text{ and } \kappa(s, S_4 \backslash i) \leq 0 \end{cases} \tag{62}$$

We now prove that the computed parameters correspond to the forward reference partition:

$$S \in \begin{cases} \mathcal{B}_f & \text{if } |S| \geq 3 \\ \mathcal{A}_f & \text{otherwise} \end{cases} \tag{63}$$

From equation (59), given in page 16, it follows that any set S, such that $|S| \geq 3$, exists in $\mathcal{B}_f$. We need to prove the remaining case where where any set S such that $|S| \leq 3$ exists in $\mathcal{A}_f$. To do this, we consider $S = \{i, j\} = S_4 \backslash \{k, l\}$ and examine its partition coefficients:

$$\kappa(f, \{i,j\}) = \kappa(f, \{i,j,k\}) + g_{f,k}$$
$$\kappa(f, \{i,j\}) = \kappa(f, \{i,j,k\}) + ((\kappa(f, \{i,j,l\}) - \kappa(f, \{i,j,k,l\})$$
$$\kappa(f, \{i,j\}) = \min(0, \kappa(s, \{i,j,k\})) + \min(0, \kappa(s, \{i,j,l\})) - \min(0, \kappa(s, \{i,j,k,l\}))$$

As in table 1, $\kappa(f,\{i,j\})$ has four possible values and $\kappa(f,\{i,j\}) \geq 0$ in all. As each set S with a cardinality of two exist in $\mathcal{A}_f$, every other set with a cardinality less than two must also exist in $\mathcal{A}_f$. The proof for the second part of the theorem when $\mathcal{A}_s \subseteq \mathcal{A}_b$ can be obtained analogously. We first prove that the set with cardinality less than two can be made to lie in $\mathcal{A}_b$. Second we can show that the set with cardinality greater than or equal to two can be made to lie in $\mathcal{B}_b$. □

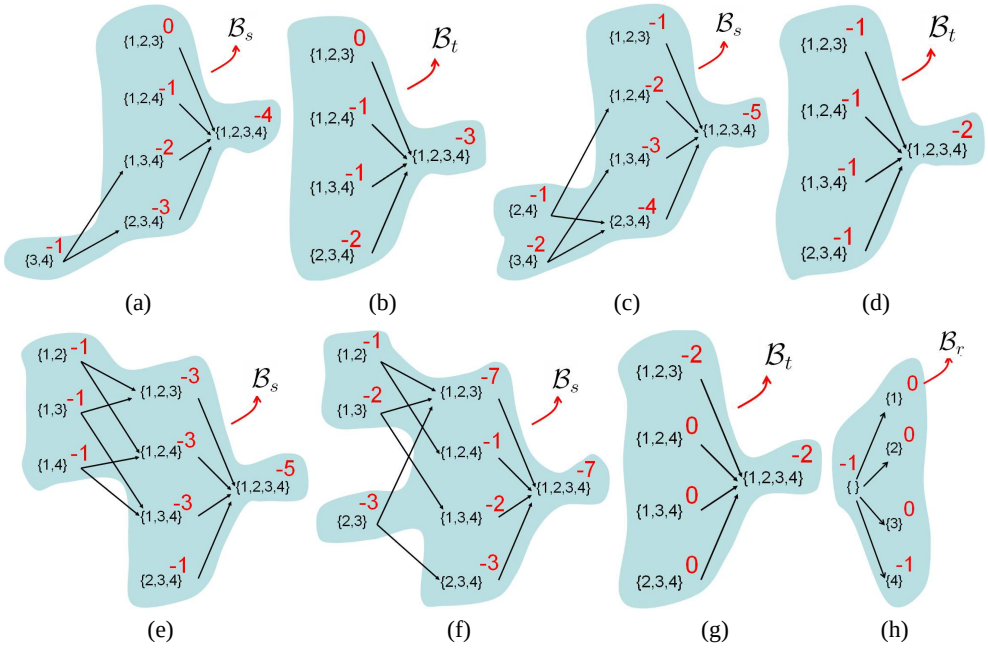


Fig. 5. *Examples for the four cases in tables 2, 3, 4 and 5. In the first case the transition in (a) is mapped to that in (b) and the associated parameters are given by:* $((g_s = 6, g_{s,1} = 1, g_{s,2} = 1, g_{s,3} = 1, g_{s,4} = 1), (g_t = 5, g_{t,1} = 1, g_{t,2} = 2, g_{t,3} = 2, g_{t,4} = 3))$. *The generated pairwise term, independent of* AVs, *is* $-x_3x_4$. *The second case is in (c) and (d) with the parameters* $((5,1,2,3,4,5),(2,1,1,1,1))$ *(shown in the same order as the earlier one) and the pairwise function is* $-x_2x_4 - 2x_3x_4$. *The third case is in (e) and (d) with the parameters* $((5,4,1,1,1),(2,1,1,1,1))$ *along with the pairwise function* $-x_1x_2 - x_1x_3 - x_1x_4$. *The final case is in (f), (g) and (h), as the final function has two* AVs z_2 *and* z_3. *The function consisting of unary and pairwise terms independent of* AVs *is given by* $1 - x_1 - x_2 - x_3 - x_1x_3 - 2x_2x_3$. *Corresponding parameters are given by:* $((g_s = 8, g_{s,1} = 4, g_{s,2} = 5, g_{s,3} = 6, g_{s,4} = 0), (g_t = 4, g_{t,1} = 2, g_{t,2} = 2, g_{t,3} = 2, g_{t,4} = 0), (g_r = 2, g_{r,1} = 1, g_{r,2} = 1, g_{r,3} = 1, g_{r,4} = 0))$

i	$\min(0,\kappa(s,\{i,j,k\}))$	$\min(0,\kappa(s,\{i,j,l\}))$	$\min(0,\kappa(s,\{i,j,k,l\}))$	$\kappa(f,\{i,j\})$
1	0	0	0	0
2	0	$\kappa(s,\{i,j,l\})$	$\kappa(s,S_4)$	$g_{s,k}$
3	$\kappa(s,\{i,j,k\})$	0	$\kappa(s,\{i,j,k,l\})$	$g_{s,l}$
4	$\kappa(s,\{i,j,k\})$	$\kappa(s,\{i,j,l\})$	$\kappa(s,\{i,j,k,l\})$	$\kappa(s,\{i,j\})$

Table 1. See lemma 5. *In all four cases* $\kappa(f,\{i,j\})$ *is non-negative. This result holds for the fourth case as* $\kappa(s,\{i,j\}) \geq 0$.

Lemma 6. *Let $P = \{i, j, k, l\} = S_4$ and let z_s be the auxiliary variable in $h(\mathbf{x}, z_s)$ associated with the partition $[\mathcal{A}_s, \mathcal{B}_s]$. If both A and $B = P \backslash A$ are elements of $\mathcal{B}_s$, then it is not possible to have both C and $D = P \backslash C$ in $\mathcal{A}_s$.*

Proof. The statement follows by contradiction. Let $\{A, B\}$, where $B = P \backslash A$, exist in $\mathcal{B}_s$. The partition coefficients of A and B with respect to z_1 are shown below:

$$\kappa(s, A) = g_s - \sum_{i=1}^{4} \mathbf{1}_i^A \leq 0$$

$$\kappa(s, B) = g_s - \sum_{i=1}^{4} \mathbf{1}_i^B \leq 0$$

Note that $A \cup B = \{i, j, k, l\}$ and $A \cap B = \emptyset$. Hence by summing the above equations we get the following:

$$2g_s - g_{s,i} - g_{s,j} - g_{s,k} - g_{s,l} \leq 0 \tag{64}$$

Assume now that a different pair $\{C, D\}$, where $D = P \backslash C$ exist in $\mathcal{A}_s$. By summing their corresponding partition coefficients we get the following equation:

$$2g_s - g_{s,i} - g_{s,j} - g_{s,k} - g_{s,l} \geq 0, \tag{65}$$

Equations 64 and 65 lead to a contradiction, therefore the lemma holds . □

Theorem 2. *Any function $h(\mathbf{x}, z_s)$ in $\mathcal{F}^2$ with z_s associated with $[\mathcal{A}_s, \mathcal{B}_s]$, such that $\forall B \in \mathcal{B}_s$, $|B| \geq 2$, can be transformed to another function $h''(\mathbf{x}, z_f, z_b)$ in $\mathcal{F}^2$ without any $z_f z_b$ terms, where z_f and z_b are* AV *correspond to the forward and backward reference partitions respectively.*

Proof. Our proof by construction takes the form of a two-step procedure. In the first stage every function $h(\mathbf{x}, z_s)$ is transformed to $h'(\mathbf{x}, z_t, z_r)$ where z_t and z_r are associated with the partition $[\mathcal{A}_t, \mathcal{B}_t]$ and the partition $[\mathcal{A}_r, \mathcal{B}_r]$ respectively and satisfy the conditions $\mathcal{B}_t \subseteq \mathcal{B}_f$ and $\mathcal{B}_r \subseteq \mathcal{B}_b$. In the second step we use lemma 5 to transform $h'(\mathbf{x}, z_t, z_s)$ to $h''(\mathbf{x}, z_f, z_b)$. In most cases only one partition, either the forward or the backward, is used.

$$\min_{z_s} h_2(\mathbf{x}, z_s) = \min_{z_s} \kappa(s, S) z_s, \forall S \in \mathcal{P} \tag{66}$$

$$\min_{z_s} h(\mathbf{x}, z_s) = \sum_{i=1}^{4} a_i x_i + \sum_{i=1}^{4} \sum_{j, i \neq j}^{4} a_{i,j} x_i x_j + \min_{z_t} \kappa(t, S) z_t + \min_{z_r} \kappa(r, S) z_r, \forall S \in \mathcal{P} \tag{67}$$

The key idea is to decompose h_2 into functions of unary and pairwise terms involving only $\mathbf{x}$ and functions involving new auxiliary variables z_t and z_r. Consider the condition $|B| \geq 2$. If $|B| \geq 3$; here we can directly use lemma 5 to obtain our desired result. We now consider the cases where at least one set $S \in \mathcal{B}_s$ has cardinality two and show a transformation similar to the general one of (67). Tables 2, 3, 4 and 5 contain details of the decomposition.

After the decomposition the new partitions $[\mathcal{A}_t, \mathcal{B}_t]$ and $[\mathcal{A}_r, \mathcal{B}_r]$ satisfy the conditions $\mathcal{B}_t \subseteq \mathcal{B}_f$ and $\mathcal{B}_r \subseteq \mathcal{B}_b$. To show this, we first consider the case where exactly one set $S \in \mathcal{B}_s$ has a cardinality of 2. There are six such occurrences, and all of them are symmetrical. The transformation for this case is in table 2.

Next, consider the case where exactly two sets of cardinality two exist in $\mathcal{B}_s$. Although there are 15 ($\binom{6}{2}$) possible cases, they must all be of the form $\{\{i, j\}, \{k, l\}\}$ or $\{\{i, j\}, \{j, k\}\}$. The first sub-case is prohibited because the presence of the mutually exclusive pair $\{\{i, j\}, \{k, l\}\}$ would not permit any other

mutually exclusive pair $\{\{i,k\},\{j,l\}\}$ to exist in $\mathcal{A}_s$ as per lemma 6. The transformation for the latter case is in table 3.

Finally, consider the case where exactly three sets of cardinality two exist in $\mathcal{B}_s$. The 20 different occurrences ($\binom{6}{3}$) can be expanded to three different scenarios:$\{\{i,j\},\{i,k\},\{i,l\}\}$, $\{\{i,j\},\{k,l\},\{i,k\}\}$ and $\{\{i,j\},\{j,k\},\{i,k\}\}$. Again, lemma 6 prevents the second scenario $\{\{i,j\},\{k,l\},\{i,k\}\}$ from occurring. The transformations of the first and the third cases are in table 4 and 5. Example transformations are shown in figure 5. □

Case 1:($\{i,j\} \in \mathcal{B}_s$.		
$h_2 = \kappa(s,\{i,j\})x_ix_j + (\underbrace{(2*g_s - g_{s,i} - g_{t,j})}_{g_t} - \underbrace{(g_s - g_{s,i})}_{g_{t,j}}x_i - \underbrace{(g_s - g_{s,j})}_{g_{t,i}}x_j - \underbrace{g_{s,k}}_{g_{t,k}}x_k - \underbrace{g_{s,l}}_{g_{t,l}}x_l)z_t$		
S	$\kappa(t,S)$	$S \in \mathcal{A}_t$ or $S \in \mathcal{B}_t$
$\{i,j\}$	0	$S \in \mathcal{A}_t$
$\{i,k\}$	$\kappa(s,\{j,k\})$	$S \in \mathcal{A}_t$ since $\{j,k\} \in \mathcal{A}_s$
$\{k,l\}$	$\kappa(s,\{i,k\}) + \kappa(s,\{j,l\})$	$S \in \mathcal{A}_t$ since $\{i,k\},\{j,l\} \in \mathcal{A}_s$
$\{i,j,k\}$	$-g_{s,k}$	$S \in \mathcal{B}_t$
$\{i,k,l\}$	$\kappa(s,\{j,k,l\})$	$S \in \mathcal{B}_t$ since $\{j,k,l\} \in \mathcal{B}_s$

Table 2. See theorem 2. *Case 1: The details of the transformation (similar to one in equation (67), given in page 18) are shown for a scenario where exactly one set ($\{i,j\}$) with cardinality two exist in $\mathcal{B}_s$. We prove that after the transformation all the sets S with $|S| = 2$ exist in $\mathcal{A}_t$ and $|S| \geq 3$ exist in $\mathcal{B}_t$. Although the reduction is illustrated for only a few cases, they are representative of the remainder.*

Case 2:$\{i,j\},\{j,k\} \in \mathcal{B}_s$.		
$h_2 = \kappa(s,\{i,j\})x_ix_j + \kappa(s,\{j,k\})x_jx_k + (\underbrace{3g_s - 2g_{s,j} - g_{s,i} - g_{s,k}}_{g_t} - \underbrace{(g_s - g_{s,j})}_{g_{t,i}}x_i - \underbrace{(2g_s - g_{s,i} - g_{s,j} - g_{s,k})}_{g_{t,j}}x_j - \underbrace{(g_s - g_{s,j)}}_{g_{t,k}}x_k - \underbrace{(g_{s,l}}_{g_{t,l}}x_l)z_t$		
S	$\kappa(t,S)$	$S \in \mathcal{A}_t$ or $S \in \mathcal{B}_t$
$\{i,j\}$	0	$S \in \mathcal{A}_t$
$\{i,l\}$	$\kappa(s,\{i,k\}) + \kappa(s,\{j,l\})$	$S \in \mathcal{A}_t$ since $\{i,k\},\{j,l\} \in \mathcal{A}_s$
$\{j,l\}$	$\kappa(s,\{j\}) + g_{s,l}$	$S \in \mathcal{A}_t$ since $\{j\} \in \mathcal{A}_s$ and $g_{s,l} \geq 0$
$\{i,j,k\}$	$-\kappa(s,\{j\})$	$S \in \mathcal{B}_t$ since $\{j\} \in \mathcal{A}_s$
$\{i,k,l\}$	$\kappa(s,\{i,k,l\})$	$S \in \mathcal{B}_t$ if $\{i,k,l\} \in \mathcal{B}_s$
$\{i,j,l\}$	$-g_{s,l}$	$S \in \mathcal{B}_t$

Table 3. See theorem 2.*Case 2: We study the scenario where exactly two sets with cardinality two $\{\{i,j\},\{j,k\})$ occur in $\mathcal{B}_s$. Note that all other cases either can not happen (according to lemma 6) or similar to the ones shown in this table. We also prove that after the transformation all the sets S with $|S| = 2$ exist in $\mathcal{A}_t$ and $|S| \geq 3$ exist in $\mathcal{B}_t$.*

Theorem 3. *Any function $h(\mathbf{x}, z_1, z_2, ...z_k)$ in $\mathcal{F}^2$ that is linear in $\mathbf{z}$ can be transformed to some function $h'(\mathbf{x}, z_f, z_b)$ in $\mathcal{F}^2$ where z_f and z_b correspond to the forward and backward reference partitions respectively.*

Case 3: $\{i,j\},\{i,k\},\{i,l\} \in \mathcal{B}_s$.

$$h_2 = \kappa(s,\{i,j\})x_ix_j + \kappa(s,\{i,k\})x_ix_k + \kappa(s,\{i,l\})x_ix_l + (\underbrace{\min(0,\kappa(s,\{j,k,l\}) + 3(g_s - g_{s,i})}_{g_t} - \underbrace{\min(0,\kappa(s,\{j,k,l\})) + 2(g_s - g_{s,i})}_{g_{t,i}})x_i - \underbrace{(g_s - g_{s,i})}_{g_{t,j}}x_j - \underbrace{(g_s - g_{s,i})}_{g_{t,k}}x_k - \underbrace{(g_s - g_{s,i})}_{g_{t,l}}x_l)z_t$$

S	$\kappa(t,S)$	$S \in \mathcal{A}_t$ or $S \in \mathcal{B}_t$
$\{i,j\}$	0	$S \in \mathcal{A}_t$
$\{j,k\}$	$\min(0,\kappa(s,\{j,k,l\})) + (g_s - g_{s,i})$	$S \in \mathcal{A}_t$ since $\{i\} \in \mathcal{A}_s$
$\{i,j,k\}$	$-\kappa(s,\{j\})$	$S \in \mathcal{B}_t$ since $\{j\} \in \mathcal{A}_s$
$\{i,k,l\}$	$\kappa(s,\{i,k,l\})$	$S \in \mathcal{B}_t$ if $\{i,k,l\} \in \mathcal{B}_s$
$\{i,j,l\}$	$-g_{s,l}$	$S \in \mathcal{B}_t$

Table 4. See theorem 2. *Case 3: Here we study the scenario where exactly three sets with cardinality two $\{\{i,j\},\{i,k\},\{i,l\})$ exist in $\mathcal{B}_s$. The only other case where three sets can exist is shown in table 5. The shown cases are generalizations of all the possible cases that can occur without violating lemma (6). We prove that after the transformation all the sets S with $|S| = 2$ exist in $\mathcal{A}_t$ and $|S| \geq 3$ exist in $\mathcal{B}_t$.*

Case 4: $\{i,j\},\{i,k\},\{i,l\} \in \mathcal{B}_s$.

$$h_2 = \kappa(s,\{i,j\})(1 - x_i - x_j - x_k) - (g_{s,k} - g_{s,j})x_ix_k - (g_{s,k} - g_{s,i})x_jx_k + (\underbrace{2(g_s - g_{s,k})}_{g_t} - \underbrace{(g_s - g_{s,k})}_{g_{t,i}}x_i - \underbrace{(g_s - g_{s,k})}_{g_{t,j}}x_j - \underbrace{(g_s - g_{s,k})}_{g_{t,k}}x_k - \underbrace{g_{s,l}}_{g_{t,l}}x_l)z_t + (\underbrace{-2\kappa(s,\{i,j\})}_{g_r} - \underbrace{(-\kappa(s,\{i,j\}))}_{g_{r,i}}(1 - x_i) - \underbrace{(-\kappa(s,\{i,j\}))}_{g_{r,j}}(1 - x_j) - \underbrace{(-\kappa(s,\{i,j\}))}_{g_{r,k}}(1 - x_k) - \underbrace{0}_{g_{r,l}}(1 - x_l))z_r$$

S	$\kappa(t,S)$	$S \in \mathcal{A}_t$ or $S \in \mathcal{B}_t$
$\{i,j\}$	0	$S \in \mathcal{A}_t \kappa = 0$
$\{i,l\}$	$\kappa(s,\{k,l\})$	$S \in \mathcal{A}_t$ since $\{k,l\} \in \mathcal{A}_s$
$\{i,j,k\}$	$-\kappa(s,\{k\})$	$S \in \mathcal{B}_t$ since $\{k\} \in \mathcal{A}_s$
$\{i,j,l\}$	$-g_{s,l}$	$S \in \mathcal{B}_t$ since $g_{s,l} \geq 0$
S	$\kappa(r,S)$	$S \in \mathcal{A}_r$ or $S \in \mathcal{B}_r$
$\{i,l\}$	0	$S \in \mathcal{A}_r$
$\{i,j\}$	$-\kappa(s,\{i,j\})$	$S \in \mathcal{A}_r$ since $\{i,j\} \in \mathcal{B}_s$
$\{i\}$	0	$S \in \mathcal{B}_r$
$\{l\}$	$\kappa(s,\{i,j\})$	$S \in \mathcal{B}_r$ since $\{i,j\} \in \mathcal{B}_s$

Table 5. See theorem 2.*Case 4: We consider three sets $\{i,j\},\{i,k\},\{j,k\} \in \mathcal{B}_s$ which involve only three elements and all three repeating in more than one set. Without loss of generality, we assume that $\kappa(s,\{i,j\}) \geq \kappa(s,\{i,k\})$ and $\kappa(s,\{i,j\}) \geq \kappa(s,\{j,k\})$. In this case we replace the* AV *z_s using two variables z_t and z_r.*

Proof. Every z_i is independent of every other z_j due to the absence of bilinear terms $z_i z_j$. Hence, the minimization under $\mathbf{z}$ can be carried out in any order.

$$\min_{z_i,z_j} h(\mathbf{x}, z_i, z_j) = \min_{z_i} \min_{z_j} h(\mathbf{x}, z_i, z_j) = \min_{z_j} \min_{z_i} h(\mathbf{x}, z_i, z_j) \tag{68}$$

Applying lemma 4, followed by theorem 2, for every AV, the function $h(\mathbf{x}, z_1, z_2, z_3, ..., z_k)$ can be transformed into $\hat{h}(\mathbf{x}, \hat{z}_1, \hat{z}'_1, ..., \hat{z}_k, \hat{z}'_k)$ where $\hat{z}_i$ and $\hat{z}'_k$ correspond to the forward and backward reference partitions respectively. In other words, every z_i in the original function is replaced by $\hat{z}_i$ and $\hat{z}'_i$. All the AVs corresponding to forward(or backward) partition can be replaced by a single variable. Thus we can transform a function $\hat{h}$ to $h'(x_1, x_2, x_3, x_4, z_f, z_b)$. □

5.2 Interacting AVs

In this section, we consider functions h having bilinear terms involving AVs of the form $z_i z_j$.

Theorem 4. *Any function $h(\mathbf{x}, z_1, z_2, ...z_k)$ in $\mathcal{F}^2$ that has bilinear terms $z_i z_j$ can be transformed to some function $h'(\mathbf{x}, z_f, z'_s)$ in $\mathcal{F}^2$ where z_f correspond to forward partition and z_s corresponds to the partition $[\mathcal{A}_s, \mathcal{B}_s]$ where $|B| \geq 2, \forall B \in \mathcal{B}_s$ and $|A| \leq 2, \forall A \in \mathcal{A}_s$.*

Proof. Using Theorem 5.2 from [34] we can decompose a given submodular function in $\mathcal{F}^4$ into 10 different groups $\mathcal{G}_i, i = \{1..10\}$ where each $\mathcal{G}_i$ is shown in Table 6. As shown in [46] the functions in $\mathcal{G}_{10}$ does to belong to $\mathcal{F}^4_2$. It was also shown that any submodular function that has any subfunctions from $\mathcal{G}_{10}$ does not belong to $\mathcal{F}^4_2$ according to Theorem 16(3) in [46]. Thus all the functions in $\mathcal{F}^k_2$ should be composed of functions in the groups $\mathcal{G}_i, i \in \{1, ..., 9\}$.

The number of distinct terms in each group is given by $\mathcal{G}_i$. Overall, there is a total of 31 distinct functions in the groups $\mathcal{G}_i, i \in \{1, ..., 9\}$. The terms in the first group $\mathcal{G}_1$ has only second degree terms. Hence, the functions in this group does not require any AVs. The terms in the next 7 groups $\mathcal{G}_i, i \in \{2, ..., 8\}$ can each be represented by single auxiliary variable that is either z_f of z_b, where z_f and z_b denote AVs in the forward and backward partitions respectively. Hence any submodular function that can be decomposed into terms in the first 8 groups can be expressed with two AVs z_f and z_b. As shown in Table 6, the terms in $\mathcal{G}_9$ can be represented using two AVs z_f and z_s where z_s corresponds to the partition $[\mathcal{A}_s, \mathcal{B}_s]$ where $|B| \geq 2, \forall B \in \mathcal{B}_s$ and $|A| \leq 2, \forall A \in \mathcal{A}_s$.

Any function $h(\mathbf{x}, z_1, z_2)$ where $z_1, z_2 \in \{z_f, z_b, z_s\}$ can be transformed to a function $h'(\mathbf{x}, z_f, z_s)$ in the following manner:

$$\min_{z_1,z_2} h(\mathbf{x}, z_1, z_2) = \min_{z_f,z_s} h'(\mathbf{x}, z_f, z_s) + \sum_{i,j=\{1,2,3,4\}, i \neq j} \alpha_{ij} x_i x_j, \quad \alpha_{ij} \leq 0 \tag{69}$$

By using the above technique sequentially, we can represent any function that can be decomposed into several functions in groups $\mathcal{G}_i, i = \{1, ..., 9\}$ using only AVs z_f and z_s. Note that z_b is a special case of z_s. As a result of lemma 6, there are only 22 partitions where $z_s \neq z_b$. □

6 Linear Programming solution

For a given function $f(x_1, x_2, x_3, x_4)$ in $\mathcal{F}^4_2$, our goal is to compute a function $h(\mathbf{x}, \mathbf{z})$ in $\mathcal{F}^2$. Theorem 4 shows that we need only two AVs (z_f, z_s). We will only show the linear programming algorithm when $z_s = z_b$. The cases where $z_s \neq z_b$ can also be solved in the same manner. The required function $h(\mathbf{x}, \mathbf{z})$ is:

$$h(\mathbf{x}, z_f, z_b) = b_0 + \sum_i b_i x_i - \sum_{i>j} b_{ij} x_i x_j + (g_f - \sum_{i=1}^{4} g_{f,i} x_i) z_f + (g_b - \sum_{i=1}^{4} g_{b,i} x_i) z_b - j_{fb} z_f z_b. \tag{70}$$

Group $\mathcal{G}_i$	$\lvert\mathcal{G}_i\rvert$	$f(\mathbf{x})$	$\min_{z_1,z_2} h(\mathbf{x}, z_1, z_2)$ where $h \in \mathcal{F}^2$
$\mathcal{G}_1$	6	$-x_i x_j$	$-x_i x_j$
$\mathcal{G}_2$	4	$-x_i x_j x_k$	$\min_{z_f}(2 - x_i - x_j - x_k)z_f$
$\mathcal{G}_3$	1	$-x_1 x_2 x_3 x_4$	$\min_{z_f}(3 - x_1 - x_2 - x_3 - x_4)z_f$
$\mathcal{G}_4$	1	$-x_1x_2x_3x_4 + x_1x_2x_3 + x_1x_2x_4 + x_1x_3x_4 + x_2x_3x_4 - x_1x_2 - x_1x_3 - x_1x_4 - x_2x_3 - x_2x_4 - x_3x_4$	$\min_{z_b}(1 - x_1 - x_2 - x_3 - x_4)z_b$
$\mathcal{G}_5$	4	$x_ix_jx_kx_l - x_ix_jx_k - x_ix_l - x_jx_l - x_kx_l$	$\min_{z_b}(2 - x_i - x_j - x_k - 2x_l)z_b$
$\mathcal{G}_6$	4	$x_ix_jx_k - x_ix_j - x_ix_k - x_jx_k$	$\min_{z_b}(1 - x_i - x_j - x_k)z_b$
$\mathcal{G}_7$	4	$x_ix_jx_kx_l - x_ix_jx_k - x_ix_jx_l - x_ix_kx_l$	$\min_{z_f}(3 - 2x_i - x_j - x_k - x_l)z_f$
$\mathcal{G}_8$	1	$2x_1x_2x_3x_4 - x_1x_2x_3 - x_1x_2x_4 - x_1x_3x_4 - x_2x_3x_4$	$\min_{z_f}(2 - x_1 - x_2 - x_3 - x_4)z_f$
$\mathcal{G}_9$	6	$x_ix_jx_kx_l - x_ix_j - x_ix_kx_l - x_jx_kx_l$	$\min_{z_f,z_s}(z_s + 2z_f - z_f z_s - z_s x_i - z_s x_j - z_f x_k - z_f x_l)$
$\mathcal{G}_{10}$	6	$-x_ix_jx_kx_l + x_ix_kx_l + x_jx_kx_l - x_ix_k - x_ix_l - x_jx_k - x_jx_l - x_kx_l$	$f(\mathbf{x}) \notin \mathcal{F}_2^4$ as shown in [46]

Table 6. *The above table is adapted from Figure 2 of [47] where $\{i, j, k, l\} = S_4$. Each group $\mathcal{G}_i$ has several terms depending on the values of $\{i, j, k, l\}$. The number of distinct terms in each group is given by $\mathcal{G}_i$. Since the groups $\mathcal{G}_4$ and $\mathcal{G}_8$ involve all four variables and are symmetric, they contain one function each. z_f and z_b correspond to* AVs *for forward and backward partitions. z_s corresponds to the partition $[\mathcal{A}_s, \mathcal{B}_s]$ where $|B| \geq 2, \forall B \in \mathcal{B}_s$ and $|A| \leq 2, \forall A \in \mathcal{A}_s$*

such that $b_{ij}, g_{f,i}, g_{b,i}, j_{fb} \geq 0$ and $i, j \in S_4$. As we know the partition of (z_f, z_b) we know their Boolean values for all labellings of $\mathbf{x}$. We need the coefficients $(b_i, b_{ij}, j_{12}, g_f, g_b, g_{f,i}, g_{b,i}), i = S_4$ to compute $h(x_1, x_2, x_3, x_4, z_f, z_b)$. These coefficients satisfy both submodularity constraints (that the coefficients of all bilinear terms $(x_ix_j, x_iz_f, x_jz_b, z_fz_b)$ are less than or equal to zero) and those imposed by the reference partitions. First we list these conditions below:

$$\underbrace{\begin{pmatrix} b_{ij} \\ g_{f,i} \\ g_{b,i} \\ j_{fb} \end{pmatrix}^T}_{\mathcal{S}_p} \geq \mathbf{0}, i, j = S_4, i \neq j \tag{71}$$

where $\mathbf{0}$ refers of a vector composed 0's of appropriate length. Next we list the conditions which guarantee $f(\mathbf{x}) = \min_{z_f,z_b} h(\mathbf{x}, z_f, z_b)$ for all $\mathbf{x}$. Let $\eta(S)$ be the value of $z_f z_b$ for $S \in \mathcal{P}$.

$$\eta(S) = \begin{cases} 1 & \text{if } |S| \geq 3 \\ 0 & \text{otherwise.} \end{cases} \tag{72}$$

Let $\mathcal{G}$ and $\mathcal{H}$ denote values of functions f and h respectively:

$$\mathcal{G} = f(\mathbf{1}_1^S, \mathbf{1}_2^S, \mathbf{1}_3^S, \mathbf{1}_4^S) \tag{73}$$

$$\mathcal{H} = h(\mathbf{1}_1^S, \mathbf{1}_2^S, \mathbf{1}_3^S, \mathbf{1}_4^S, 0, 0) - (g_f - \sum_{i=1}^{4} g_{f,i}\mathbf{1}_i^S) - (g_b - \sum_{i=1}^{4} g_{b,i}\mathbf{1}_i^S) - j_{fb}\eta(S) \tag{74}$$

As a result we have the following 16 linear equations (N.B. there are $2^4(16)$ different S):

$$\mathcal{G} = \mathcal{H}, \forall S \in \mathcal{P} \tag{75}$$

Note that as in section 5 we do not make use of either auxiliary variables or the $\min$ operator over $\mathcal{H}$. Again, this is because we already know the partition of (z_f, z_b) and their appropriate values a priori. The following constraints ensure that z_f and z_b behave as reference partitions.

$$\underbrace{\begin{pmatrix} g_f - \sum_{i=1}^{4} g_{f,i}\mathbf{1}_i^S \\ g_b - \sum_{i=1}^{4} g_{b,i}\mathbf{1}_i^D \end{pmatrix}}_{\mathcal{G}_g} \geq \mathbf{0}, S \in \mathcal{A}_f, D \in \mathcal{A}_b$$

$$\underbrace{\begin{pmatrix} g_f - \sum_{i=1}^{4} g_{f,i}\mathbf{1}_i^S \\ g_b - \sum_{i=1}^{4} g_{b,i}\mathbf{1}_i^D \end{pmatrix}}_{\mathcal{G}_l} \leq \mathbf{0}, S \in \mathcal{B}_f, D \in \mathcal{B}_b.$$

Essentially we need to compute the coefficients $(b_{ij}, g_f, g_{f,i}, g_b, g_{b,i}, j_{fb})$ that satisfy the equations (71,75,76) This is equivalent to finding a feasible point in a linear programming problem:

$$\min\ const \tag{76}$$

$$s.t\ \mathcal{S}_p \geq \mathbf{0},\ \mathcal{G} = \mathcal{H},\ \mathcal{G}_g \geq \mathbf{0},\ \mathcal{G}_l \leq \mathbf{0} \tag{77}$$

As discussed in section 4, by using a different objective function we can formulate a problem to to compute a function in $\mathcal{F}^2$ closest to a given arbitrary fourth-order function.

7 Experiments

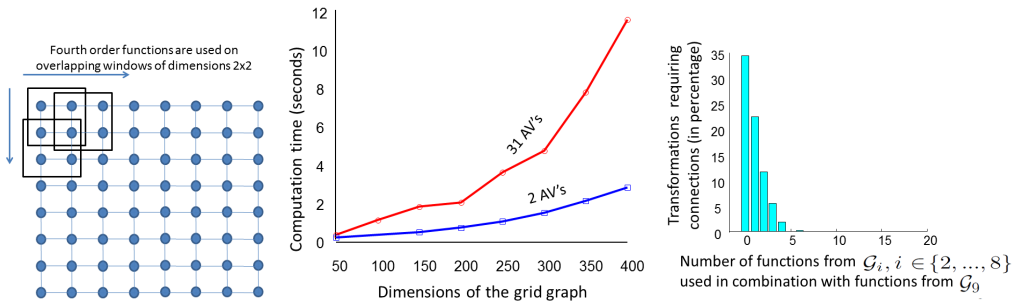
We performed two experiments to demonstrate the computational benefits of the proposed technique and to understand the role of connections between the AVs while expressing submodular functions.

7.1 Computational Benefits

In Matlab, the linear program shown in section 6 takes around 0.03 seconds and it can be further improved using efficient C++ implementation. We want to emphasize that the existing results [45, 34] does not provide a method to transform a given fourth order function to a function in $\mathcal{F}^2$ using 31 AVs. Such a transformation would more likely involve a constrained optimization algorithm that is at least as computationally expensive as linear programming with a lot more variables and constraints. Hence, we state that it is very unlikely that there may exist an alternate transformation that uses less time than our transformation for even approaches employing all 31 AVs.

In many vision applications, the parameters of the higher order prior models are known apriori. Hence it is possible to compute the transformation for fourth order functions to second order ones before minimizing the final quadratic functions using algorithms like graph cuts. Note that even functions employing iterative approaches, it is possible to precompute the parameters of the AVs for all transformations from possible fourth order to quadratic functions based on the label-space.

In Figure 6(a), we show a grid graph that is commonly employed in computer vision problems where the nodes correspond to pixels in the graph. The higher order priors are applied on pixels inside overlapping windows. In our experiments we considered windows of dimensions 2×2 involving four nodes each. We generate a fourth order function using non-negative weighted sum of functions in groups $\mathcal{G}_i, i = \{1, ..., 9\}$. In Figure 6(b), we compared the time taken for the standard maxflow algorithm on two different quadratic functions where one is obtained using 31 AVs and the other obtained using 2 AVs. As expected, we observe a significant improvement in computational speed by using 2 AVs compared to 31 AVs as the grid size increases.



(a) (b) (c)

Fig. 6. *(a) We show a grid graph with overlapping windows of dimensions* 2×2*. A fourth order submodular function is generated using the functions from the groups* $\mathcal{G}_i, i = \{2, ..., 9\}$ *and used for all the over-lapping windows. (b) We show the time requirements for standard maxflow algorithm in minimizing functions that uses 31* AVs *verses our approach that uses only 2* AVs*. (c) We generated submodular functions using non-weighted sum of functions from* $\mathcal{G}_9$ *and a subset of functions from groups* $\mathcal{G}_i, i = \{2, ..., 8\}$*. The x-axis denotes the number of functions chosen from* $\mathcal{G}_i, i = \{2, ..., 8\}$ *in generating the submodular functions and the y-axis gives the percentage of transformations requiring connection between* AVs*.*

7.2 Interaction between AVs

Note that there is no unique transformation from a function in $\mathcal{F}_2^4$ to a function in $\mathcal{F}^2$. The linear programming approach only finds one of the many feasible solutions. Any function in $\mathcal{F}_2^4$ can be expressed by a non-negative weighted sum of functions from groups $\mathcal{G}_i, i = \{2, ..., 9\}$. It was shown that only functions in $\mathcal{G}_9$ require the connection between AVs. We observed that although a single function in $\mathcal{G}_9$ requires interacting AVs, sum or two or more functions may not require the interaction. For example, let us consider two functions in $\mathcal{G}_9$ as shown below:

$$f_1(\mathbf{x}) = x_1x_2x_3x_4 - x_1x_2 - x_1x_3x_4 - x_2x_3x_4 \tag{78}$$
$$f_2(\mathbf{x}) = x_1x_2x_3x_4 - x_3x_4 - x_1x_2x_3 - x_1x_2x_4 \tag{79}$$

We show their sum $f(\mathbf{x}) = f_1(\mathbf{x}) + f_2(\mathbf{x})$ below:

$$f(\mathbf{x}) = -x_1x_2 - x_3x_4 - (2x_1x_2x_3x_4 - x_1x_2x_3 - x_1x_2x_4 - x_1x_3x_4 - x_2x_3x_4) \tag{80}$$
$$f(\mathbf{x}) = -x_1x_2 - x_3x_4 - f'(\mathbf{x}) \tag{81}$$

In the above equation the function $f'(\mathbf{x})$ is a function in $\mathcal{G}_8$ and it does not require interaction between AVs.

In our experiments, we generated submodular functions using non-weighted sum of functions from $\mathcal{G}_9$ and a subset of functions from groups $\mathcal{G}_i, i = \{2, ..., 8\}$. The number of functions $n_{\mathcal{G}}$ used from groups $\mathcal{G}_i, i = \{2, ..., 8\}$ is increased from 0 to 19. When $n_{\mathcal{G}} = 0$, the function is generated using non-negative weighted sum of all 6 functions from $\mathcal{G}_9$. When $n_{\mathcal{G}} > 0$, we randomly chose $n_{\mathcal{G}}$ functions from $\mathcal{G}_i, i = \{2, ..., 8\}$. For each value of $n_{\mathcal{G}}$, we generated 1000 functions and the non-negative weights are randomly generated in the interval $[0, 1]$. When $n_{\mathcal{G}}$ is small, the transformation used connection between the AVs. Empirically when $n_{\mathcal{G}}$ is greater than 5, the transformations seldom used connection between the AVs.

8 Discussion and open problems

There has been an increased interest in developing principled algorithms for reducing higher order functions to quadratic ones. These techniques can be broadly classified into two types: submodularity-preserving [1, 27, 18, 44, 35, 14, 37, 46] and general techniques [38, 12, 15, 2, 21]. This paper belongs to the submodular-preserving class of algorithms. The general techniques are usually employed in association with roof-duality approaches for minimizing non-submodular functions [4, 3, 5, 39]. These methods also employ AVs, but they need not be MBFs. The existing upper bound for such techniques is given by $G(k) = 2^{k-2}(k-3)+1$ for a kth order function[21]. We show the comparison between the AVs used in general and submodularity-preserving methods in Table 7.

Type	Degree	3	4	5	6	7	8
General	Ishikawa [21]	1	5	17	49	129	321
Submodularity-preserving	Dedekind [25]	20	168	7581	$\approx 7.8 \times 10^{6}$	$\approx 2.4 \times 10^{12}$	$\approx 5.6 \times 10^{23}$

Table 7. Comparison of the number of AVs used for general verses submodularity-preserving techniques.

Note that the upper bound for the number of AVs required for submodularity-preserving transformation is much higher than for general reduction techniques. By establishing the connection between AVs and MBFs, we have improved the upper bound for general functions from 2^{2^k} to Dedekind number $D(k)$. In the case of fourth order functions we have improved the upper bound from 168 to 2. We believe that this analysis will prove more useful in developing general reduction techniques where the submodularity constraints in the linear-programming no longer exist.

It is important to note that finding the minimum number of AVs required for a particular function is NP-hard and it can be reduced from a minimum size vertex cover problem [3]. This however does not mean that we can not make any progress on reducing the upper bound. Empirically, many vision problems used only polynomial number of AVs rather than exponential number [12]. Despite the significant progress in operations research and vision communities, several open questions continue to remain unsolved in this domain.

References

1. A. Billionnet and M. Minoux. Maximizing a supermodular pseudo-boolean function: a polynomial algorithm for supermodular cubic functions. *Discrete Appl. Math.*, 1985.
2. E. Boros and A. Gruber. On quadratization of pseudo-boolean functions. In *ISAIM*, 2012.
3. E. Boros and P. L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 2002.
4. E. Boros and P.L. Hammer. A max-flow approach to improved roof-duality in quadratic 0-1 minimization. *Technical Report RRR 15-1989, RUTCOR, Rutgers University*, 1989.
5. E. Boros, P.L. Hammer, R. Sun, and G. Tavares. A max-flow approach to improved lower bounds for quadratic unconstrained binary optimization (qubo). *Discrete Optimization*, 2008.
6. D.A. Cohen, M.C Cooper, and P.G. Jeavons. An algebraic characterisation of complexity for valued constraints. In *CP*, 2006.
7. M.C. Cooper. Minimization of locally defined submodular functions by optimal soft arc consistency. *Constraints*, 2008.
8. Y. Crama and P.L. Hammer. *Boolean Functions: Theory, Algorithms and Applications*. Cambridge University Press, 2011.
9. W.H. Cunningham. On submodular function minimization. *Combinatorica*, 1985.
10. J. Edmonds. Submodular functions, matroids and certain polyhedra. *Calgary International Conference on Combinatorial Structures and their applications*, 1969.

11. M.L. Fisher, G.L. Nemhauser, and L.A. Wolsey. An analysis of approximation for maximizing submodular setfunctions-i. *Mathematical Programming*, 1978.
12. A. Fix, A. Gruber, E. Boros, and R. Zabih. A graph cut algorithm for higher-order markov random fields. In *ICCV*, 2011.
13. L. Fleischer and S. Iwata. A push-relabel framework for submodular function minimization and applications to parametric optimization. *Discrete Applied Mathematics*, 2001.
14. D. Freedman and P. Drineas. Energy minimization via graph cuts: Settling what is possible. In *CVPR*, 2005.
15. A.C. Gallagher, D. Batra, and D. Parikh. Inference for order reduction in markov random fields. In *CVPR*, 2011.
16. G. Gallo and B. Simeone. On the supermodular knapsack problem. *Mathematical Programming: Series A and B*, 1989.
17. M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1981.
18. P. L. Hammer. Some network flow problems solved with pseudo-boolean programming. *Operations Research*, 1965.
19. G. Hansel. Sur le nombre des fonctions booleennes monotones de n variables. *C.R. Acad. Sci. Paris*, 1966.
20. H. Ishikawa. Exact optimization for Markov random fields with convex priors. *PAMI*, 2003.
21. H. Ishikawa. Transformation of general binary mrf minimization to the first-order case. *PAMI*, 2011.
22. S. Iwata. A fully combinatorial algorithm for submodular function minimization. *J. Comb. Theory Ser. B*, 2000.
23. S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *J. ACM*, 2001.
24. Sataru Iwata. A faster scaling algorithm for minimizing submodular functions. *SIAM J. Computing*, 2003.
25. D. Kleitman. On dedekind's problem: The number of monotone boolean functions. *Proc. Amer. Math Soc.*, 1969.
26. P. Kohli, M. P. Kumar, and P. H. S. Torr. P^3 & beyond: Solving energies with higher order cliques. In *CVPR*, 2007.
27. V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *PAMI*, 2004.
28. A.D. Korshunov. The number of monotone boolean functions. *Problemy Kibernet*, 1981.
29. L. Ladicky, C. Russell, P. Kohli, and P. Torr. Graph cut based inference with co-occurrence statistics. In *European Conference on Computer Vision*. springer, 2010.
30. X. Lan, S. Roth, D. P. Huttenlocher, and M. J. Black. Efficient belief propagation with learned higher-order Markov random fields. In *ECCV*, 2006.
31. L. Lovasz. Submodular functions and convexity. *Mathematical Programming - The State of the Art*, 1983.
32. M. Narasimhan and J.A. Bilmes. A submodular-supermodular procedure with applications to discriminative structure learning. In *Uncertainty in Artificial Intelligence*, 2005.
33. J.B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, 2009.
34. S. Promislow and V. Young. Supermodular functions on finite lattices. *Order 22(4)*, 2005.
35. M. Queyranne. Minimizing symmetric submodular functions. *SODA*, 1995.
36. S. Ramalingam, P. Kohli, K. Alahari, and P.H.S. Torr. Exact inference in multi-label crfs with higher order cliques. In *CVPR*, 2008.
37. J.M.W. Rhys. A selection problem of shared fixed costs and network flows. *Management Science*, 1970.
38. I. G. Rosenberg. Reduction of bivalent maximization to the quadratic case. *Cahiers du Centre d'Etudes de Recherche Operationnelle*, 1975.
39. C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer. Optimizing binary MRFs via extended roof duality. In *CVPR*, 2007.
40. D. Schlesinger and B. Flach. Transforming an arbitrary minsum problem into a binary one. Technical Report TUD-FI06-01, Dresden University of Technology, 2006.
41. A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *J. Combin. Theory*, 1999.
42. L. Trevisan G.B. Sorkin, M. Sudan, and D.P. Williamson. Gadgets, approximation, and linear programming. *SIAM Journal of Computing*, 2000.
43. A.L. Yuille and A. Rangarajan. The concave-convex procedure (cccp. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.

44. B. Zalesky. Efficient determination of gibbs estimators with submodular energy functions. *http://arxiv.org/abs/math/0304041v1*, 2003.
45. S. Zivny, D.A.Cohen, and P.G. Jeavons. The expressive power of binary submodular functions. *Discrete Applied Mathematics*, 2009.
46. S. Zivny and P.G. Jeavons. Classes of submodular constraints expressible by graph cuts. *Proceedings of CP*, 2008.
47. S. Zivny and P.G. Jeavons. Which submodular functions are expressible using binary submodular functions? *Oxford University Computing Laboratory Researc Report CS-RR-08-08*, 2008.
48. S. Zivny and P.G. Jeavons. Classes of submodular constraints expressible by graph cuts. *Constraints*, 2010.