DIMACS Technical Report 2001-33

Pseudo-Boolean Optimization

by

Endre Boros[1,2,3] Peter L. Hammer[1,2,3]

[1]Permanent Member
[2]RUTCOR, Rutgers University, 640 Bartholomew Road, Piscataway, NJ 08854-8003, USA. Email: {boros,hammer}@rutcor.rutgers.edu
[3]This research was supported in part by the Office of Naval Research (Grant N00014-92-J-1375) and by the National Science Foundation (Grant DMS 98-06389)

ABSTRACT

This survey examines the state of the art of a variety of problems related to pseudo-Boolean optimization, i.e. to the optimization of set functions represented by closed algebraic expressions. The main parts of the survey examine general pseudo-Boolean optimization, the specially important case of quadratic pseudo-Boolean optimization (to which every pseudo-Boolean optimization can be reduced), several other important special classes, and approximation algorithms.

Contents

1 Introduction

Set functions, i.e. mappings from the family of subsets of a finite ground set to the set of reals, have been present in the mathematical literature for more than a century, with a substantial development of this area starting in the early 1950's. The importance of set functions in game theory and optimization brought them to the full focus of attention of applied mathematicians, especially of those working in operations research. This increased interest in set functions is motivated by their presence in the mathematical models of a wide spectrum of problems occurring in a variety of applications.

Set functions are frequently considered as being defined by an oracle, or more specifically, by an algorithm capable of delivering their values for any subset of the given finite ground set. Any graph parameter (e.g., chromatic number, stability number, etc.) associated to the subgraphs induced by a subset of the vertices of a given graph is an example for such a set function. However, in numerous examples a set function can be defined by a closed algebraic formula, an advantageous special case of an oracle. As a trivial example, we can think of the cardinality of a subset S of the finite ground set $X = \{1, 2, ..., n\}$ being given by $|S| = x_1 + x_2 + \cdots + x_n$, where $(x_1, x_2, \ldots, x_n)$ is the characteristic vector of the subset S. Several other examples of set functions given by closed algebraic formulae will be given in Section 3.

The explicit knowledge of closed algebraic representations of set functions makes possible the application of a substantially larger set of mathematical techniques and results for their analysis and optimization. For instance, results of convex and non-convex analysis, and various other techniques of nonlinear programming can thus be directly applied to problems, which are otherwise discrete in nature. It is important to remark that every set function defined on a finite ground set admits closed form algebraic representations. However, the derivation of such a form can be computationally difficult, or even intractable in some cases.

In this paper we shall focus on those set functions which are defined on a finite ground set and are given by closed algebraic formulae, and shall pay special attention to the case of multi-linear polynomial representations. Because of the analogy with Boolean functions, these functions will be called *pseudo-Boolean*.

The natural connections between pseudo-Boolean functions and nonlinear binary optimization have motivated and strongly influenced some of the first studies in this area (see e.g. [72, 87, 88, 91]). Since then the study of pseudo-Boolean functions grew to a major area of research with hundreds of related publications in the last 30 years (see e.g. [94, 95]).

Pseudo-Boolean functions appearing in polynomial (or other algebraic) representation play a major role in optimization models in a variety of areas, including VLSI design (via minimization [17, 34]), statistical mechanics (spin glasses [17, 50]), reliability theory (fault location [137]), computer science (maximum satisfiability [114]), statistics (clustering [145] and ranking [144]), economics ([92]), finance ([105, 120, 121]), operations research (location [73, 141, 162]), management science (project selection [165]), discrete mathematics (graphs, hypergraphs and networks [61, 74, 91, 140, 156]), manufacturing (production and scheduling [9, 48, 118]).

Beside optimization problems, pseudo-Boolean functions also appear in many other models of current interest. They constitute for instance the main object of investigation in cooperative game theory, where they are viewed as characteristic functions of games with side-payments [86, 132, 134, 155]. They are used in various models of theoretical physics, where they describe the Hamiltonian energy function of spin glass systems [115, 126, 167] and of neural networks [6, 107, 138]. They occur in combinatorial theory, as rank functions of matroids [44, 164], or as functions associated with certain graph-parameters, such as stability number, chromatic number, etc. [21, 61, 133, 156]).

In this paper we present a brief overview of the theory and algorithmic aspects of pseudo-Boolean functions and their optimization. Due to the large volume of related research, our survey is far from complete. We focus on results and techniques specific to this type of representation of set functions, with particular attention being paid to special classes, including quadratic, sub- and supermodular, and hyperbolic pseudo-Boolean functions.

2 Definitions and Notations

Let us denote by $\mathbb{R}$ the set of reals, by $\mathbb{Z}$ the set of integers, and let $\mathbb{B} = \{0, 1\}$ and $\mathbb{U} = [0, 1]$. Further let n denote a positive integer, and let $\mathbf{V} = \{1, 2, ..., n\}$. For a subset $S \subseteq V$ let us denote by $\mathbf{1}^S \in \mathbb{B}^n$ its characteristic vector, i.e.

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

We shall consider functions in n binary variables, denoted by x_1, x_2, ..., x_n, and shall use $\mathbf{x} = (x_1, ..., x_n) \in \mathbb{B}^n$ to denote a binary vector, as well as the vector of these variables. In many situations the binary values 0 and 1 are used to encode unordered bivalent attributes, and play a perfectly symmetric role. It is natural to work both with the variables x_i and with their *complements* $\overline{x}_i \stackrel{\text{def}}{=} 1 - x_i$, for $i \in \mathbf{V}$, called together *literals*. Let $\mathbf{L} = \{x_1, \overline{x}_1, ..., x_n, \overline{x}_n\}$ denote the set of literals.

Mappings $f : \mathbb{B}^n \mapsto \mathbb{R}$ are called *pseudo-Boolean functions*. Since there is a one-to-one correspondence between the subsets of $\mathbf{V}$ and the set of binary vectors $\mathbb{B}^n$, these functions are in fact *set functions*, i.e. mappings which associate a real value to every subset of a finite set. In this survey we shall refer to f as a set function, whenever we want to emphasize that the values of f are given in some implicit way (e.g. via an oracle), and we shall call it a pseudo-Boolean function if it is given explicitly by an algebraic expression.

The simplest, and perhaps least efficient, way of representing a pseudo-Boolean function is by a table listing the real values $f(\mathbf{x})$ corresponding to every binary vector $\mathbf{x} \in \mathbb{B}^n$.

As we shall see later, all pseudo-Boolean functions can be uniquely represented as *multilinear polynomials*, of the form

$$f(x_1, ..., x_n) = \sum_{S \subseteq \mathbf{V}} c_S \prod_{j \in S} x_j. \tag{1}$$

By convention, we shall always assume that $\prod_{j \in \emptyset} x_j = 1$.

The size of the largest subset $S \subseteq \mathbf{V}$ for which $c_S \neq 0$ is called the *degree* of f, and is denoted by $\deg(f)$. We shall call a pseudo-Boolean function f *linear* (*quadratic*, *cubic*, etc.) if $\deg(f) \leq 1$ ($\leq 2, 3$, etc.)

The *size* of an expression (1) is the total number of variable occurrences in it, i.e.

$$\text{size}(f) \stackrel{\text{def}}{=} \sum_{S: c_S \neq 0} |S|. \tag{2}$$

Let us associate to a pseudo-Boolean function f its *i-th derivative*,

$$\begin{aligned} \Delta_i(\mathbf{x}) &\stackrel{\text{def}}{=} \tfrac{\partial f}{\partial x_i}(\mathbf{x}) \\ &= f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) - f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) \end{aligned} \tag{3}$$

and its *i-th residual*

$$\Theta_i(\mathbf{x}) \stackrel{\text{def}}{=} f(\mathbf{x}) - x_i \Delta_i(\mathbf{x}), \tag{4}$$

for all indices $i \in \mathbf{V}$. Let us notice that the functions Δ_i and Θ_i are themselves pseudo-Boolean functions, which depend on all the variables, but x_i.

Frequently, pseudo-Boolean functions are also represented as *posiforms*, i.e. polynomial expressions in terms of all the literals, having the form

$$\phi(x_1, ..., x_n) = \sum_{T \subseteq \mathbf{L}} a_T \prod_{u \in T} u, \tag{5}$$

where $a_T \geq 0$ whenever $T \neq \emptyset$. It is also customary to assume that $a_T = 0$ if $\{u, \overline{u}\} \subseteq T$ for some $u \in \mathbf{L}$, since otherwise the product $\prod_{u \in T} u$ is identically zero over $\mathbb{B}^n$.

Similarly to the case of polynomial expressions, let us call the size of the largest subset T of literals for which $a_T \neq 0$ the *degree* of the posiform ϕ, and denote it by $\deg(\phi)$; let us call a posiform ϕ *linear* (*quadratic*, *cubic*, etc.) if $\deg(\phi) \leq 1$ ($\leq 2, 3$, etc.) Furthermore, let us measure the *size* of a posiform as the total number of literal occurrences in it, i.e.

$$\text{size}(\phi) \stackrel{\text{def}}{=} \sum_{T: a_T \neq 0} |T|. \tag{6}$$

For the purpose of analyzing algorithms, the sum of the coefficients

$$A(\phi) \stackrel{\text{def}}{=} \sum_{T \neq \emptyset} a_T \tag{7}$$

will also be frequently needed.

It is obvious that a posiform (5) determines uniquely a pseudo-Boolean function. However, the reverse is not true: a pseudo-Boolean function can have many different posiforms representing it. Let us also add that while it is computationally easy to generate a posiform from a polynomial expression (1), it might be computationally difficult to generate the unique polynomial expression corresponding to a given posiform.

In the sequel, we shall use the letters x, y, and z to refer to variables, u, v, and w to refer to literals., and bold face letters $\mathbf{x}$, $\mathbf{y}$, $\mathbf{p}$, etc., to denote vectors. The letters f, g and h will usually denote pseudo-Boolean functions as well as their unique multi-linear polynomial expressions, while the greek letters ϕ and ψ will denote posiforms.

In this survey we shall consider minimization and maximization problems; whenever it is not necessary to specify which one is considered, we shall simply speak of optimization (or simply opt) to refer to any one of them.

3 Examples

There are a large number of combinatorial optimization models, which arise naturally or can be formulated easily as pseudo-Boolean optimization problems. In this section we recall some of these (mostly well-known) formulations.

One of the simplest problems of this type is well-known in algorithmic graph theory. Given a graph $G = (V, E)$ with vertex set V and edges set E, a subset $S \subseteq V$ is called *independent*, if no edge of G has both its endpoints in S. In the *maximum independent set problem* we need to find the largest cardinality independent set, the cardinality of which we shall denote by $\alpha(G)$. It is easy to show that

$$\alpha(G) = \max_{\mathbf{x}\in\mathbb{B}^V} \left(\sum_{i\in V} x_i - \sum_{(i,j)\in E} x_i x_j \right).$$

holds for every graph, furthermore that if $\mathbf{x}^* = \mathbf{1}^S$ is a maximizing binary vector of the above quadratic function, then a maximum cardinality independent set S^* of G (in fact with $S^* \subseteq S$) can be obtained in $O(n)$ time.

The complement $V \setminus S$ of an independent set S of G is called a *vertex cover* of the graph. Denoting by $\tau(G) = |V| - \alpha(G)$ the size of a smallest vertex cover of G, it is easy to show that

$$\tau(G) = \min_{\mathbf{x}\in\mathbb{B}^V} \left(\sum_{i\in V} x_i + \sum_{(i,j)\in E} \overline{x}_i \overline{x}_j \right).$$

The above formulations can be extended analogously to the weighted variants of these problems, as well as to hypergraphs. For instance, given a hypergraph $\mathcal{H} \subseteq 2^V$, a subset S of its vertices is called a *vertex cover* of $\mathcal{H}$ (known also as a *hitting set*) if $S \cap H \neq \emptyset$ for all hyperedges $H \in \mathcal{H}$. Denoting by $\tau(\mathcal{H})$ the size of a smallest vertex cover, it can be shown that

$$\tau(\mathcal{H}) = \min_{\mathbf{x}\in\mathbb{B}^V} \left(\sum_{i\in V} x_i + \sum_{H\in\mathcal{H}} \prod_{i\in H} \overline{x}_i \right).$$

This optimization problem can also be viewed as a pseudo-Boolean formulation of the equivalent *set covering* problem (over the transposed hypergraph.)

For a subset S of vertices of a graph $G = (V, E)$, let us denote by $\delta(S)$ the number of edges with exactly one endpoint in S. The *maximum cut* problem is to find a subset S which maximizes $\delta(S)$. Using the characteristic vector $\mathbf{x} = \mathbf{1}^S$ to represent a subset S, we can easily see that

$$\max_{S\subseteq V} \delta(S) = \max_{\mathbf{x}\in\mathbb{B}^V} \left(\sum_{(i,j)\in E} (x_i\overline{x}_j + \overline{x}_i x_j) \right).$$

A graph $G = (V, E)$ is called *signed* if the edge set is partitioned $E = E^+ \cup E^-$ into two (disjoint) subsets, called *positive*, and *negative* edges, respectively. A signed graph is called *balanced* if every cycle of it involves an even number of negative edges. The *signed graph balancing* problem consists in finding a minimum cardinality subset F of edges in a signed graph G, the removal of which makes G balanced. Denoting by $\sigma(G)$ the size of such a minimum edge set, we can immediately see that

$$\sigma(G) = \min_{\mathbf{x}\in\mathbb{B}^V} \left(\sum_{(i,j)\in E^+} (x_i\overline{x}_j + \overline{x}_i x_j) + \sum_{(i,j)\in E^-} (x_i x_j + \overline{x}_i\overline{x}_j) \right).$$

The *maximum satisfiability* problem, one of the most frequently studied problems in the recent applied mathematics/theoretical computer science literature, has also a natural pseudo-Boolean formulation. In this problem the input consists of a family $\mathcal{C}$ of subsets $C \subseteq \mathbf{L}$ of literals, called *clauses*. We say that a binary assignment $\mathbf{x} \in \mathbb{B}^V$ is *satisfying* such a clause $C \subseteq \mathbf{L}$, if the (Boolean) disjunction of the literals in C takes value 1 (true) for this assignment. The maximum satisfiability problem consists in finding a binary assignment satisfying the maximum number of clauses of $\mathcal{C}$. It is easy to see that $\mathbf{x} \in \mathbb{B}^V$ satisfies a clause C iff $\prod_{u\in C} \overline{u} = 0$. Thus, the problem is equivalent with the pseudo-Boolean maximization problem

$$\max_{\mathbf{x}\in\mathbb{B}^V} \left(\sum_{C\in\mathcal{C}} \left(1 - \prod_{u\in C} \overline{u} \right) \right).$$

In the weighted version of the problem there is also a nonnegative weight a_C associated to every clause $C \in \mathcal{C}$, and the objective is to maximize the total weight of satisfied clauses

$$\max_{\mathbf{x}\in\mathbb{B}^V} \left(\sum_{C\in\mathcal{C}} a_C \left(1 - \prod_{u\in C} \overline{u} \right) \right).$$

4 General Pseudo-Boolean Functions

4.1 Representations of pseudo-Boolean functions

In this section we review a few basic properties of multi-linear polynomial and posiform representations of a pseudo-Boolean function, starting with the non-uniqueness of posiform representations.

Example 4.1 *Let us show first on a small example that pseudo-boolean functions can have different posiform representations. Consider for this the following two posiforms:*

$$\psi_1 = 5x_1 + 4\overline{x}_1\overline{x}_2x_3 + 7x_1x_2x_4 + 9x_3\overline{x}_4, \text{ and} \tag{8}$$

$$\psi_2 = x_1 + 4x_1x_2 + 4x_1\overline{x}_2\overline{x}_3 + 7x_1x_2x_4 + 4\overline{x}_2x_3 + 9x_3\overline{x}_4. \tag{9}$$

It is easy to verify that in fact both posiforms define the same pseudo-Boolean function $g : \mathbb{B}^4 \mapsto \mathbb{R}$ given by the table:

$\mathbf{x}$				$g(\mathbf{x})$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	13
0	0	1	1	4
0	1	0	0	0
0	1	0	1	0
0	1	1	0	9
0	1	1	1	0
1	0	0	0	5
1	0	0	1	5
1	0	1	0	14
1	0	1	1	5
1	1	0	0	5
1	1	0	1	12
1	1	1	0	14
1	1	1	1	12

Let us observe next that if a pseudo-Boolean function $f : \mathbb{B}^n \mapsto \mathbb{R}$ is represented by a posiform ϕ (5), then

$$a_\emptyset \leq \min_{\mathbf{x}\in\mathbb{B}^n} f(\mathbf{x}). \tag{10}$$

This trivial observation will serve us handily, when looking for lower bounds to a minimization problem. Let us show next that in fact

Proposition 1 *Every pseudo-Boolean function $f : \mathbb{B}^n \mapsto \mathbb{R}$ can be represented by a posiform ϕ for which*

$$a_\emptyset = \min_{\mathbf{x}\in\mathbb{B}^n} f(\mathbf{x}).$$

Proof. Let us start by defining $a_\emptyset = \min_{\mathbf{x}\in\mathbb{B}^n} f(\mathbf{x})$, as in the statement. Let us further define for every binary vector $\mathbf{y} \in \mathbb{B}^n$ a corresponding set of literals $L(\mathbf{y}) \subseteq \mathbf{L}$ by setting $L(\mathbf{y}) \stackrel{\text{def}}{=} \{x_i | y_i = 1, i \in \mathbf{V}\} \cup \{\overline{x}_i | y_i = 0, i \in \mathbf{V}\}$, and let $a_{L(\mathbf{y})} = f(\mathbf{y}) - a_\emptyset$ for all $\mathbf{y} \in \mathbb{B}^n$. We claim that with this notation

$$\Phi_f(\mathbf{x}) = a_\emptyset + \sum_{\mathbf{y}\in\mathbb{B}^n} a_{L(\mathbf{y})} \prod_{u\in L(\mathbf{y})} u \tag{11}$$

is indeed a posiform representing f. To verify this claim we can notice that $a_{L(\mathbf{y})} \geq 0$ for all $\mathbf{y} \in \mathbb{B}^n$ by the above definitions, that the term $\left(\prod_{u\in L(\mathbf{y})} u\right)(\mathbf{x}) = 0$ for all binary vectors $\mathbf{x} \neq \mathbf{y}$, and that this term takes the value 1 if $\mathbf{x} = \mathbf{y}$. Hence, in every $\mathbf{x} \in \mathbb{B}^n$ $\Phi_f(\mathbf{x}) = a_\emptyset + a_{L(\mathbf{x})} = f(\mathbf{x})$. □

Let us remark that the posiform (11) is in fact uniquely defined for every pseudo Boolean function f. We shall call this unique posiform Φ_f the *min-term* representation of f.

Let us prove next that every pseudo-Boolean function has a unique multi-linear polynomial representation.

Proposition 2 ([87, 91]) *Every pseudo-Boolean function $f : \mathbb{B}^n \mapsto \mathbb{R}$ has a unique multi-linear polynomial representation of the form (1).*

Proof. We need to show that the values of f over $\mathbb{B}^n$ determine uniquely the coefficients c_S, $S \subseteq \mathbf{V}$ in (1). Let us show this by induction on the size of these subsets. Clearly, $f(0, 0, ..., 0) = c_\emptyset$, and hence $c_\emptyset$ is uniquely determined by the value $f(\mathbf{1}^\emptyset)$.

Let us assume next that the coefficients c_T are already shown to be unique for all subsets T of size less than k, and let $S \subseteq \mathbf{V}$ be a subset of size k. By observing that

$$f(\mathbf{1}^S) = \sum_{T\subseteq S} c_T,$$

we obtain

$$c_S = f(\mathbf{1}^S) - \sum_{T\subset S} c_T. \tag{12}$$

Since all terms on the right hand side have unique values, in view of (12) the same holds for c_S, thus completing the proof. □

Example 4.2 *The function g in Example 4.1 has*

$$g(x_1, x_2, x_3, x_4) = 5x_1 + 13x_3 - 4x_1x_3 - 4x_2x_3 - 9x_3x_4 + 4x_1x_2x_3 + 7x_1x_2x_4$$

as its unique multi-linear polynomial form.

It will be useful later to express the derivatives and the residuals of a pseudo-Boolean function in terms of the coefficients of its multi-linear polynomial expression.

Proposition 3 *Given a pseudo-Boolean function f by its multi-linear polynomial expression (1), we have the following equalities for its derivatives (3) and residuals (4) for all indices $i \in \mathbf{V}$:*

$$\Delta_i(\mathbf{x}) = \sum_{S \subseteq \mathbf{V} \setminus \{i\}} c_{S \cup \{i\}} \prod_{j \in S} x_j \quad \text{and} \quad \Theta_i(\mathbf{x}) = \sum_{S \subseteq \mathbf{V} \setminus \{i\}} c_S \prod_{j \in S} x_j. \tag{13}$$

Proof. Immediate from (1) by elementary calculations. □

Example 4.3 *For illustration, if $f(x_1, x_2) = -1 + 3x_1 + 4x_2 - 2x_1x_2$, then $\Delta_1(\mathbf{x}) = 3 - 2x_2$, and $\Theta_1(\mathbf{x}) = -1 + 4x_2$, etc.*

4.2 Rounding procedures and derandomization

Let us note that (1) can also be viewed as a real valued expression, which can be evaluated for any real vector $\mathbf{r} \in \mathbb{R}^n$.

Proposition 4 ([37, 148]) *Let us consider a pseudo-Boolean function f given by (1), and let $\mathbf{r} \in \mathbb{U}^n$. Then there exist binary vectors $\mathbf{x}, \mathbf{y} \in \mathbb{B}^n$ for which*

$$f(\mathbf{x}) \leq f(\mathbf{r}) \leq f(\mathbf{y}); \tag{14}$$

furthermore, such vectors can be generated in $O(\text{size}(f))$ time.

Proof. We shall give a constructive proof only for the existence of $\mathbf{x}$, since an analogous procedure will work for the existence of $\mathbf{y}$.

ROUNDDOWN(f,$\mathbf{r}$)

Initialize: Set $t = 0$ and $\mathbf{q}^0 = \mathbf{r}$.

Loop: While there exists an index $j \in \mathbf{V}$ for which $0 < q_j < 1$:

Set $t = t + 1$ and define

$$q_i^t = \begin{cases} q_i^{t-1} & \text{if } i \neq j, \\ 0 & \text{if } i = j \text{ and } \Delta_j(\mathbf{q}^{t-1}) > 0, \\ 1 & \text{otherwise.} \end{cases}$$

Output: Set $\mathbf{x} = \mathbf{q}^t$.

Let us note first that in every iteration of the above procedure one of the fractional components is changed to an integral one, and hence RoundDown terminates in at most n iterations by outputting a binary vector $\mathbf{x}$.

Let us also observe that if j is the index chosen in the t-th iteration then we have $\Delta_j(\mathbf{q}^t) = \Delta_j(\mathbf{q}^{t-1})$ and $\Theta_j(\mathbf{q}^t) = \Theta_j(\mathbf{q}^{t-1})$, since these functions depend only on the variables x_i for $i \neq j$, and since the vectors $\mathbf{q}^t$ and $\mathbf{q}^{t-1}$ differ only in their j-th components. Thus the inequality

$$f(\mathbf{q}^{t-1}) - f(\mathbf{q}^t) = (\mathbf{q}_j^{t-1} - \mathbf{q}_j^t)\Delta_j(\mathbf{q}^{t-1}) \geq 0$$

holds by the update rule in the core of the **Loop**, and hence we have

$$f(\mathbf{x}) = f(\mathbf{q}^t) \leq f(\mathbf{q}^{t-1}) \leq \cdots \leq f(\mathbf{q}^1) \leq f(\mathbf{q}^0) = f(\mathbf{r}).$$

To see the claimed complexity, we need to organize these computations carefully. First of all we can preselect the fractional components of $\mathbf{r}$ in $O(n)$ steps, and build a variable-term data structure, and pre-compute the values of the terms of (1) in $O(\text{size}(f))$ steps. After this, each selection in the **Loop** can be executed in constant time, and both the evaluation of Δ_j and the update of the values of the terms (of those which depend on the j-th component) can be executed in time proportional to the number of occurrences of x_j in (1). Hence the total time of the algorithm after the pre-computations is $O(\text{size}(f))$, thus proving our claim. $\square$

Let us remark that by changing the inequality in the core of the **Loop** of the above procedure, we obtain an increasing sequence of function values, and produce a binary vector $\mathbf{y}$. We shall refer to that version of the above procedure as RoundUp(f,$\mathbf{r}$).

The above simple properties have a number of consequences. To state them, we need a few more definitions.

Let us introduce the notation $\text{Argopt}_D(f)$ for the subset consisting of the points of D which are optimal solutions of the optimization problem $opt_{\mathbf{x}\in D} f(\mathbf{x})$. Let us further introduce the generic notations $\mathbf{r}^{min}$, $\mathbf{r}^{max}$, $\mathbf{x}^{min}$ and $\mathbf{x}^{max}$ to denote an arbitrary optimizing vector for the respective optimization problems, i.e. $\mathbf{r}^{min} \in \text{Argmin}_{\mathbb{U}^n}(f)$, $\mathbf{r}^{max} \in \text{Argmax}_{\mathbb{U}^n}(f)$, $\mathbf{x}^{min} \in \text{Argmin}_{\mathbb{B}^n}(f)$, and $\mathbf{x}^{max} \in \text{Argmax}_{\mathbb{B}^n}(f)$.

Example 4.4 *Returning to the pseudo Boolean function g in Example 4.1, we can see that*

$$\begin{aligned}
\text{Argmin}_{\mathbb{B}^4}(g) &= \{(0,0,0,0),(0,1,0,0),(0,0,0,1),(0,1,0,1),(0,1,1,1)\},\\
\text{Argmax}_{\mathbb{B}^4}(g) &= \{(1,0,1,0),(1,1,1,0)\},\\
\text{Argmin}_{\mathbb{U}^4}(g) &= \{(0,a,0,b),(0,1,c,1) | 0 \leq a,b,c \leq 1\}, \ \textit{and}\\
\text{Argmax}_{\mathbb{U}^4}(g) &= \{(1,a,1,0) | 0 \leq a \leq 1\}.
\end{aligned}$$

The first consequence we can draw easily from the above properties is that both the maximization and the minimization of a pseudo-Boolean function over $\mathbb{B}^n$ can in fact be viewed as continuous non-linear optimization problems over $\mathbb{U}^n$.

Corollary 1 *For any pseudo-Boolean function f*

$$\operatorname*{opt}_{\mathbf{x}\in\mathbb{B}^n} f(\mathbf{x}) = \operatorname*{opt}_{\mathbf{r}\in\mathbb{U}^n} f(\mathbf{r}).$$

Proof. On the one hand $\mathbb{U}^n \supset \mathbb{B}^n$ implies that

$$\min_{\mathbf{q}\in\mathbb{U}^n} f(\mathbf{q}) \leq \min_{\mathbf{x}\in\mathbb{B}^n} f(\mathbf{x}) \leq \max_{\mathbf{x}\in\mathbb{B}^n} f(\mathbf{x}) \leq \max_{\mathbf{q}\in\mathbb{U}^n} f(\mathbf{q}). \tag{15}$$

On the other hand, applying RoundDown(f,$\mathbf{r}^{min}$), we can obtain by Proposition 4 a binary vector $\mathbf{x} \in \mathbb{B}^n$ for which $f(\mathbf{x}) \leq f(\mathbf{r}^{min})$, and hence $\min_{\mathbf{x}\in\mathbb{B}^n} f(\mathbf{x}) \leq f(\mathbf{r}^{min}) = \min_{\mathbf{q}\in\mathbb{U}^n} f(\mathbf{q})$ follows, implying $\min_{\mathbf{q}\in\mathbb{U}^n} f(\mathbf{q}) \geq \min_{\mathbf{x}\in\mathbb{B}^n} f(\mathbf{x})$. This, together with (15) implies that

$$\min_{\mathbf{x}\in\mathbb{B}^n} f(\mathbf{x}) = \min_{\mathbf{q}\in\mathbb{U}^n} f(\mathbf{q}).$$

Similarly, applying RoundUp(f,$\mathbf{r}^{max}$), we can verify that

$$\max_{\mathbf{x}\in\mathbb{B}^n} f(\mathbf{x}) = \max_{\mathbf{q}\in\mathbb{U}^n} f(\mathbf{q}).$$

□

Another, algorithmic consequence of Proposition 4 is the existence of linear time "rounding" procedures, which can be viewed as efficient ways of "derandomization".

Proposition 5 *Let us assume that the binary variables x_i, $i \in \mathbf{V}$ are independent random variables with* $\text{Prob}(x_i = 1) = p_i$ *and* $\text{Prob}(x_i = 0) = q_i \stackrel{\text{def}}{=} 1 - p_i$ *for $i \in \mathbf{V}$, where $\mathbf{p} = (p_1, ..., p_n) \in \mathbb{U}^n$ is a given vector of the probabilities defining the distribution. Let further $f : \mathbb{B}^n \mapsto \mathbb{R}$ be a pseudo-Boolean function given by (1). Then the expected value of f is*

$$\text{Exp}[f] = f(\mathbf{p}).$$

Proof. Since the sum of the expectations of random variables is the same as the expectation of their sum, it is enough to show that

$$\text{Exp}[c_S \prod_{j\in S} x_j] = c_S \prod_{j\in S} p_j$$

for an arbitrary term of (1). This latter equality follows readily by the independence of the variables, since that implies

$$\text{Exp}[c_S \prod_{j\in S} x_j] = c_S \prod_{j\in S} \text{Exp}[x_j] = c_S \prod_{j\in S} p_j.$$

□

Let us note that only multi-linearity was needed for the above proof, hence the same property will also hold for any posiform.

In view of the above result, we can regard RoundDown and RoundUp as efficient derandomizations, since e.g. for a fractional vector $\mathbf{p} \in \mathbb{U}^n$ a simple probabilistic argument guarantees the existence of a binary vector $\mathbf{x} \in \mathbb{B}^n$ with $f(\mathbf{x}) \leq \mathrm{Exp}[f]$, while RoundDown provides an efficient deterministic way of generating such a vector.

This kind of probabilistic arguments are frequently used in combinatorics and in various approximation algorithms for combinatorial optimization problems (see e.g. [5, 69, 70]), and the above simple rounding procedures are special cases of the "probabilistic rounding" technique based on conditional probabilities, introduced in [37, 142, 143]. When started from $\mathbf{p} = (\frac{1}{2}, ..., \frac{1}{2})$, RoundDown is essentially equivalent with the heuristic procedure proposed in [109].

4.3 Local Optima

In this section, following the presentation of [91], we shall recall some necessary conditions of optimality. To avoid unnecessary repetitions, we shall state results only for the case of minimization problems, although similar necessary conditions can be stated for maximization problems, as well.

Let us associate to a binary vector $\mathbf{x} \in \mathbb{B}^n$ its *neighborhood* $N(\mathbf{x})$, defined as

$$N(\mathbf{x}) = \{\mathbf{y} | \rho_H(\mathbf{x}, \mathbf{y}) \leq 1\}, \tag{16}$$

where $\rho_H(\mathbf{x}, \mathbf{y})$ denotes the so called *Hamming distance* of the vectors $\mathbf{x}$ and $\mathbf{y}$, defined as the number of components in which these two vectors differ.

Given a pseudo Boolean function $f : \mathbb{B}^n \mapsto \mathbb{R}$, a binary vector $\mathbf{x} \in \mathbb{B}^n$ is called a *local minimum* of f if $f(\mathbf{y}) \geq f(\mathbf{x})$ for all neighboring vectors $\mathbf{y} \in N(\mathbf{x})$. Let us add that of course every (global) minimum of f is also a local minimum.

Proposition 6 *Given a pseudo Boolean function f, a binary vector $\mathbf{x} \in \mathbb{B}^n$ is a local minimum of f if and only if*

$$x_i = \begin{cases} 1 & \text{if} \quad \Delta_i(\mathbf{x}) < 0, \\ 0 & \text{if} \quad \Delta_i(\mathbf{x}) > 0, \end{cases} \tag{17}$$

for all $i \in \mathbf{V}$.

Proof. Let us denote by $\mathbf{y}^i$ the binary vector obtained from $\mathbf{x}$ by switching its i-th component. Then, by the above definition, $\mathbf{x}$ is a local minimum iff $f(\mathbf{y}^i) \geq f(\mathbf{x})$ for all indices $i \in \mathbf{V}$. In view of (4) and of the fact that $\mathbf{x}$ and $\mathbf{y}^i$ differ only in their i-th components, we have

$$f(\mathbf{y}^i) = y_i^i \Delta_i(\mathbf{y}^i) + \Theta_i(\mathbf{y}^i) = (1 - x_i)\Delta_i(\mathbf{x}) + \Theta_i(\mathbf{x})$$

for $i \in \mathbf{V}$. Hence
$$f(\mathbf{y}^i) - f(\mathbf{x}) = (1 - 2x_i)\Delta_i(\mathbf{x})$$
follows for every index $i \in \mathbf{V}$. Thus $f(\mathbf{y}^i) \geq f(\mathbf{x})$ implies (17), for all $i \in \mathbf{V}$. □

Let us note that, given a binary vector $\mathbf{x} \in \mathbb{B}^n$, conditions (17) are very easy to test. Even if the pseudo-Boolean function f is given only by an oracle, the derivatives can be evaluated by $n+1$ calls to this oracle (by obtaining the values for $f(\mathbf{x})$ and for $f(\mathbf{y}^i)$ for $i \in \mathbf{V}$, as in the proof above.)

The above set of conditions inspired a large number of heuristic algorithms, the so called *local search methods*. These algorithms focus on finding a local minimum, in the hope that it turns out to be a global one, as well.

LocalSearch(f,x^0)

Input: A pseudo-Boolean function f and a binary vector $\mathbf{x}^0 \in \mathbb{B}^n$. Set $k = 0$. [It is assumed that for every binary vector $\mathbf{x} \in \mathbb{B}^n$ there exists a well defined and computable subset $N(\mathbf{x}) \subseteq \mathbb{B}^n$, called the neighborhood of $\mathbf{x}$.]

Iteration: While there exists $\mathbf{y} \in N(\mathbf{x}^k)$ for which $f(\mathbf{y}) < f(\mathbf{x}^k)$, let $\mathbf{x}^{k+1} = \mathbf{y}$ and set $k = k + 1$.

Output: RETURN the local minimum $\mathbf{x}^k$ of f.

Though, these algorithms tend to work very well and fast in practice, theoretical guarantees for efficiency exist only in some special cases. Finding a local minimum remains, in general, a difficult problem, even in cases when the number of local minima is very large (see e.g. [96, 110, 157, 158]), and even for special classes of pseudo-Boolean functions [82]. A further complication is that the number of local minima can indeed be very large (e.g., exponentially large in the input size of the problem) even for quadratic pseudo Boolean functions (see e.g. [112]), and obviously, not all of those are equally good solutions. A natural idea to increase the chances that a local minimum is also a global one is to use larger neighborhoods (e.g., considering all points within Hamming distance k, for $k = 2, 3, ...$). Unfortunately, not only the computational cost of each iteration increases by this, but there are results indicating that, in a worst case sense (see e.g. [151, 161]) even the use of substantially larger neighborhoods will not necessarily yield better results either. Various techniques have been proposed in the literature for the implicit handling of much larger neighborhoods without a sharp increase in the computational cost of the iterations. Perhaps the most successful and most widely applied such method is the so called *tabu search* algorithm, see e.g. [64, 65].

Let us add that continuous local optimization techniques are efficient for the minimization of a convex (or maximization of a concave) function, since any local optimum of those is also a global one. Recognition of convexity or concavity of some continuous extensions of pseudo-Boolean functions is hence important, and considered also in the literature (see e.g.

[41, 42, 124]). Other techniques modify the function to achieve convexity (or concavity) by giving up multi-linearity of the objective function (see e.g. [90, 117]).

4.4 Reductions to Quadratic Optimization

In this section we recall from [149] that the optimization of a pseudo-Boolean function can always be reduced in polynomial time to the optimization of a quadratic pseudo-Boolean function. The basic idea in this reduction is the substitution of the product of two variables by a new one, and the addition of appropriate penalty terms having the role of forcing, at any point of optimum, the new variable to take the value of the product of the two substituted variables.

The following simple observation provides the basic tool for such a substitution.

Observation 1 *Assume that $x, y, z \in \mathbb{B}$. Then the following equivalences hold:*

$$xy = z \quad \textit{iff} \quad xy - 2xz - 2yz + 3z = 0,$$

and

$$xy \neq z \quad \textit{iff} \quad xy - 2xz - 2yz + 3z > 0.$$

These equivalences can easily be verified by trying out all 8 possible binary substitutions for the variables x, y, and z. Let us further remark that, of course, the above quadratic expression is not the only one for which such equivalences would hold.

Since a quadratic constraint can be eliminated by inserting the corresponding expression into the objective function as a penalty term (with a large multiplier), the above equivalences suggest the following method for reducing a pseudo-Boolean minimization problem to a quadratic one:

REDUCEMIN(f)

Input: A pseudo-Boolean function f given by its multi-linear polynomial form (1).

Initialize: Set $M \stackrel{\text{def}}{=} 1 + 2\sum_{S \subseteq \mathbf{V}} |c_S|$, $m = n$, and $f^n = f$.

Loop: While there exists a subset $S^* \subseteq \mathbf{V}$ for which $|S^*| > 2$ and $c_{S^*} \neq 0$ repeat:

1. Choose two elements i and j from S^* and update $c_{\{i,j\}} = c_{\{i,j\}} + M$, set $c_{\{i,m+1\}} = c_{\{j,m+1\}} = -2M$ and $c_{\{m+1\}} = 3M$, and for all subsets $S \supseteq \{i,j\}$ with $c_S \neq 0$ define $c_{(S \setminus \{i,j\}) \cup \{m+1\}} = c_S$ and set $c_S = 0$.
2. Define $f^{m+1}(x_1, ..., x_{m+1}) = \sum_{S \subseteq \mathbf{V}} c_S \prod_{k \in S} x_k$, and set $m = m + 1$.

Output: Set $g = f^m$.

Theorem 1 ([149]) REDUCEMIN(f) *terminates in polynomial time in the size of f, and produces a pseudo-Boolean function g in m variables, the size of which is polynomially bounded in* size(f), *and such that*

$$\min_{\mathbf{y} \in \mathbb{B}^m} g(\mathbf{y}) = \min_{\mathbf{x} \in \mathbb{B}^n} f(\mathbf{x}).$$

Proof. In the main loop of the above algorithm we replace each occurrence of $x_i x_j$ by x_{m+1}, and we add the expression $M(x_i x_j - 2x_i x_{m+1} - 2x_j x_{m+1} + 3x_{m+1})$ to the objective function.

Hence, by Observation 1 and by our choice of M we have $f^{m+1}(x_1, ..., x_{m+1}) = f^m(x_1, ..., x_m) \leq \max_{\mathbf{x} \in \mathbb{B}^m} f^m(\mathbf{x}) < M/2$ whenever $x_{m+1} = x_i x_j$, and $f^{m+1}(x_1, ..., x_{m+1}) \geq M/2$ whenever $x_{m+1} \neq x_i x_j$. Thus,

$$\min_{\mathbf{y} \in \mathbb{B}^{m+1}} f^{m+1}(\mathbf{y}) = \min_{\mathbf{x} \in \mathbb{B}^m} f^m(\mathbf{x}),$$

implying the claimed equality of the minima.

It also follows that the number of those terms in f^{m+1} for which $c_S \neq 0$, $|S| > 2$ is at least one less than in f^m, hence the algorithm must terminate in at most size(f) iterations, proving the claim about complexity. $\square$

Example 4.5 *Let us consider the following pseudo-Boolean function*

$$f(x_1, x_2, x_3, x_4, x_5) \stackrel{\text{def}}{=} 5x_1x_2 - 7x_1x_2x_3x_4 + 2x_1x_2x_3x_5.$$

Applying ReduceMin *to this function we get initially* $M = 1 + 5 + 7 + 2 = 15$, $m = 5$ *and* $f^5 = f$. *We can choose in the* **Loop** *first* $i = 1$ *and* $j = 2$ *and we get*

$$\begin{aligned} c_{1,2} &= c_{1,2} + M = 20, \\ c_{1,6} &= c_{2,6} = -2M = -30, \ \textit{and} \\ c_6 &= 3M = 45, \end{aligned}$$

and for the two terms containing both x_1 *and* x_2 *we get*

$$\begin{aligned} c_{3,4,6} &= c_{1,2,3,4} = -7, \\ c_{1,2,3,4} &= 0, \\ c_{3,5,6} &= c_{1,2,3,5} = 2, \ \textit{and} \\ c_{1,2,3,5} &= 0. \end{aligned}$$

Thus we obtain

$$f^6 = 45x_6 + 20x_1x_2 - 30x_1x_6 - 30x_2x_6 - 7x_3x_4x_6 + 2x_3x_5x_6.$$

Since there are still terms of degree higher than 2, we have to repeat the **Loop**, *and can choose e.g.* $i = 3$ *and* $j = 6$, *yielding*

$$\begin{aligned} c_{3,6} &= c_{3,6} + M = 15, \\ c_{3,7} &= c_{6,7} = -2M = -30, \ \textit{and} \\ c_7 &= 3M = 45, \end{aligned}$$

while for the two terms containing both x_3 *and* x_6 *we get*

$$\begin{aligned} c_{4,7} &= c_{3,4,6} = -7, \\ c_{3,4,6} &= 0, \\ c_{5,7} &= c_{3,5,6} = 2, \ \textit{and} \\ c_{3,5,6} &= 0. \end{aligned}$$

Therefore, for $m = 7$ *we obtain*

$$\begin{aligned} f^7 = 45x_6 + 45x_7 + 20x_1x_2 - 30x_1x_6 - 30x_2x_6 + 15x_3x_6 - 30x_3x_7 \\ - 7x_4x_7 + 2x_5x_7 - 30x_6x_7. \end{aligned}$$

Since there are no non-quadratic terms left, the algorithm terminates outputting $g = f^7$. *It is easy to verify that indeed,*

$$\min_{\mathbf{x} \in \mathbb{B}^5} f(\mathbf{x}) = \min_{\mathbf{y} \in \mathbb{B}^7} g(\mathbf{y}) = -2$$

and that the relations $x_6 = x_1x_2$ *and* $x_7 = x_3x_6$ *provide a one-to-one correspondence between the minima of these functions, i.e. between the sets* $\mathrm{Argmin}_{\mathbb{B}^5}(f) = \{(1,1,1,1,0)\}$, *and* $\mathrm{Argmin}_{\mathbb{B}^7}(f^7) = \{(1,1,1,1,0,1,1)\}$.

One might hope for a more efficient reduction by trying to substitute at a time 3 (or more) variables with a new one. It is easy to see however that the above procedure cannot be generalized in this way:

Observation 2 *There is no quadratic pseudo-Boolean function $f(x, y, z, u)$ (in four binary variables) for which $f(x, y, z, u) = 0$ if $u = xyz$, and $f(x, y, z, u) > 0$ whenever $u \neq xyz$.*

Let us note also that the number of (new) variables in the final output depends clearly on the selection of the pairs i, j in the **Loop** of the above algorithm. One could try to minimize the number of (new) variables by finding a better selection procedure for these pairs. However, this optimization problem is NP-hard, even for cubic inputs, as shown by the following simple observation.

Observation 3 *Let us consider a graph $G = (\mathbf{V}, E)$, and define a pseudo-Boolean function*

$$f(x_0, x_1, ..., x_n) = \sum_{(i,j)\in E} a_{i,j} x_0 x_i x_j,$$

where the coefficients $a_{i,j}$ are arbitrary reals for all $(i, j) \in E$. Let us denote by m the smallest number for which ReduceMin(f) *produces a quadratic pseudo-Boolean function g in m binary variables. Then,*

$$m = n + \tau(G),$$

where $\tau(G)$ denotes the size of a smallest vertex cover of the graph G.

Clearly, this shows that determining the smallest m is not easier then computing the vertex cover of a graph, which is known to be NP-complete (see e.g. [63]).

4.5 Persistency

Let us return now to posiforms, and review some of their specific properties. Let us first recall the trivial fact that all (non-constant) terms of (5) have nonnegative coefficients, and hence their sum can never be less than zero. This implies that, in a way, minimizing a posiform is essentially the same as trying to make as many of its terms as possible vanish. This is evident e.g. in the *maximum satisfiability* (or in short MAX-SAT) problem, which is equivalent with the minimization of a given posiform ϕ in which $a_\emptyset = 0$, with the important exception that the objective there is stated as

$$\max_{\mathbf{x}\in\mathbb{B}^n} \left(A(\phi) - \phi(\mathbf{x}) \right),$$

i.e. as the maximization of the total weight of terms which can be made to vanish simultaneously (and not as the minimization of ϕ.) Recall that $A(\phi)$ denotes the sum of the coefficients of the posiform ϕ, as defined in (7).

This idea also gives us the possibility to recognize whether a partial assignment of binary values to some of the variables is "optimal" in some sense. For this, let us call a binary vector $\mathbf{y} \in \mathbb{B}^S$ corresponding to a subset $S \subseteq \mathbf{V}$ a *partial assignment*. (Note that $\mathbb{B}^n$ is only a simpler and possibly more conventional notation used instead of the more accurate notation $\mathbb{B}^{\mathbf{V}}$.) Furthermore, for a subset $S \subseteq \mathbf{V}$ of indices and a vector $\mathbf{x} \in \mathbb{B}^n$, let us denote by $\mathbf{x}[S] \in \mathbb{B}^S$ the subvector corresponding to the indices in S, i.e. $\mathbf{x}[S] = (x_i | i \in S)$. For a partial assignment $\mathbf{y} \in \mathbb{B}^S$ and for a vector $\mathbf{x} \in \mathbb{B}^n$, let us define the *switch* of $\mathbf{x}$ by $\mathbf{y}$ to be the binary vector $\mathbf{z}$ defined by

$$z_j = \begin{cases} x_j & \text{if } j \notin S, \\ y_j & \text{if } j \in S, \end{cases}$$

and let us denote it by $\mathbf{z} = \mathbf{x}[S \leftarrow \mathbf{y}]$. For example, if $n = 5$, $S = \{1, 2, 5\}$ and $\mathbf{y}$ is the partial assignment $y_1 = 1$, $y_2 = 0$, and $y_5 = 1$, then the switch of $\mathbf{x} = (1, 1, 1, 0, 0)$ by $\mathbf{y}$ will be the vector $\mathbf{z} = (1, 0, 1, 0, 1)$.

Given a pseudo-Boolean function f, and a partial assignment $\mathbf{y} \in \mathbb{B}^S$ for some subset $S \subseteq \mathbf{V}$, following [79] we shall say that *strong persistency* holds for f at $\mathbf{y}$, if for all $\mathbf{x} \in \mathrm{Argmin}_{\mathbb{B}^n}(f)$ we have $\mathbf{x}[S] = \mathbf{y}$, i.e. if the restriction of all minimizing points of f to S concide with the partial assignment $\mathbf{y}$. *Weak persistency* is said to hold for f at $\mathbf{y}$ if $\mathbf{x}[S \leftarrow \mathbf{y}] \in \mathrm{Argmin}_{\mathbb{B}^n}(f)$ holds for all $\mathbf{x} \in \mathrm{Argmin}_{\mathbb{B}^n}(f)$, i.e. if a switch by the partial assignment $\mathbf{y}$ in a minimizing point always results in a minimizing point. Clearly, strong persistency implies weak persistency.

There are several examples for persistency observed in the literature, including both weak and strong persistency for quadratic pseudo-Boolean functions [79], weak persistency in an integer programming formulation of vertex covering [129], etc. Since satisfiability problems can also be viewed as testing whether a given posiform has its minimum equal to its constant term, the so called "autark" assignments introduced in [127] turn out also to be special cases of weak persistency (c.f. [32].)

Example 4.6 *Let us return again to the pseudo Boolean function g in Example 4.1, and its posiform representation ψ_1. If we let $\mathbf{y}^* = (0, 1) \in \mathbb{B}^{\{1,2\}}$ to be a partial assignment to the first two variables, then we can see that for each vector in* $\mathrm{Argmin}_{\mathbb{B}^4}(g)$ *switching the first two components to* $(0, 1)$ *yields always a vector in* $\mathrm{Argmin}_{\mathbb{B}^4}(g)$*. Hence, weak persistency holds for the pseudo-Boolean function g at $\mathbf{y}^*$. Furthermore, at the partial assignment $\mathbf{z}^* = (0) \in \mathbb{B}^{\{1\}}$, both weak and strong persistency hold for g.*

Let us add that verifying if weak or strong persistency holds for a pseudo-Boolean function f at a given partial assignment $\mathbf{y} \in \mathbb{B}^S$ is, in general, a difficult task. However, in some special cases posiforms may provide an efficient guarantee for persistency to hold.

Given a posiform (5), and a partial assignment $\mathbf{y} \in \mathbb{B}^S$ for some subset $S \subseteq \mathbf{V}$, we shall say that $\mathbf{y}$ is a *contractor* for ϕ if whenever $a_T > 0$ and there is an index $i \in S$ for which x_i or $\overline{x}_i$ is in T we have $\prod_{u \in T} u(\mathbf{y}) \equiv 0$. In other words, $\mathbf{y}$ is a contractor for ϕ if it makes all those terms of ϕ vanish which involve at least one of the variables x_i, $i \in S$, regardless of the values of the other variables x_j, $j \notin S$.

Let us note that checking if a given partial assignment is a contractor for a posiform ϕ can be done in linear $O(\text{size}(\phi))$ time.

Proposition 7 *Given a posiform ϕ as in (5), and a contractor $\mathbf{y} \in \mathbb{B}^S$ of it, we have*

$$\phi(\mathbf{x}[S \leftarrow \mathbf{y}]) \leq \phi(\mathbf{x})$$

for all binary vectors $\mathbf{x} \in \mathbb{B}^n$.

Proof. Let us denote by $L(S)$ the set of literals corresponding to variables with indices in S, i.e. $L(S) = \{x_i, \overline{x}_i | i \in S\}$. Let us further consider an arbitrary binary vector $\mathbf{x} \in \mathbb{B}^n$, and let $\mathbf{z} = \mathbf{x}[S \leftarrow \mathbf{y}]$ denote its switch by $\mathbf{y}$.

We can see that

$$\left(\sum_{T \subseteq \mathbf{L}, T \cap L(S) = \emptyset} a_T \prod_{u \in T} u \right) (\mathbf{x}) = \left(\sum_{T \subseteq \mathbf{L}, T \cap L(S) = \emptyset} a_T \prod_{u \in T} u \right) (\mathbf{z}),$$

since these terms involve only variables not belonging to S, and $\mathbf{z}[\mathbf{V} \setminus S] = \mathbf{x}[\mathbf{V} \setminus S]$ holds by the definition of $\mathbf{z}$. We can also observe that

$$\left(\sum_{T \subseteq \mathbf{L}, T \cap L(S) \neq \emptyset} a_T \prod_{u \in T} u \right) (\mathbf{z}) = 0,$$

since $\mathbf{y}$ is assumed to be a contractor for ϕ. Therefore we have

$$\begin{aligned} \phi(\mathbf{z}) &= \left(\sum_{T \subseteq \mathbf{L}, T \cap L(S) = \emptyset} a_T \prod_{u \in T} u \right) (\mathbf{z}) \\ &= \left(\sum_{T \subseteq \mathbf{L}, T \cap L(S) = \emptyset} a_T \prod_{u \in T} u \right) (\mathbf{x}) \\ &\leq \phi(\mathbf{x}), \end{aligned}$$

since the coefficients of the non-trivial terms of ϕ are all nonnegative, and the constant term, corresponding to $T = \emptyset$, is included in the summation on the previous line. □

Corollary 2 *If the pseudo-Boolean function f is given as a posiform ϕ, and the partial assignment $\mathbf{y} \in \mathbb{B}^S$, $S \subseteq \mathbf{V}$ is a contractor for ϕ, then weak persistency holds for f at $\mathbf{y}$.*

Proof. Immediate by Proposition 7. □

In fact, we shall see later that all of the cases for weak persistency cited in the literature follow from this corollary, and hence can be verified simply by exhibiting an appropriate posiform of the pseudo Boolean function in question.

Example 4.7 *Returning to the pseudo-Boolean function of Example 4.1, we can note first that the vector* $\mathbf{y}^*$*, as defined in Example 4.6, is a contractor for the posiform* ψ_1*.*

Let us further remark that not all persistencies follow easily by contractors, e.g. $\mathbf{z}^*$*, defined in the same example, is not a contractor for* ψ_1*, though the pseudo-Boolean function* g *has another posiform* ψ_2*, given in Example (4.1), for which* $\mathbf{z}^*$ *is a contractor. Of course, this is not always the case.*

As a conclusion of the discussions on persistency and contractors, we can see that posiform representations can provide not only bounds to the optimum values of a pseudo Boolean function, but also some information about the values of the variables in the optima.

4.6 Basic Algorithm

In this section, following [91], we present a general algorithm for finding the optimum of a pseudo-Boolean function, based on the necessary conditions of local optimality presented in subsection 4.3. This algorithm, the so called *basic algorithm*, was introduced first in [87, 88], and later simplified in [91]. We shall also recall some more recent results from [47], showing that for a special class of problems this algorithm can be implemented to run in polynomial time.

From a theoretical point of view, conditions (17) can be viewed as a characterization of the components of a local minimum in terms of the other components of that vector. This suggests the possibility of finding an expression for a component in terms of the other components, which would hold in all local minima, and thus in all minimum points as well. Such an expression could then be used to eliminate a variable, and substitute the minimization problem with another one having one variable less. Such eliminations schemes are well-known from linear algebra.

Let us first recall briefly how the basic algorithm works. For the sake of notational simplicity, we shall assume that variables are eliminated in the order x_n, x_{n-1}, ..., x_1.

BasicAlgorithm(f)

Input: Let n denote the number of variables. If $n = 1$ and $f(1) > f(0)$ then RETURN $x_1^* = 0$, otherwise RETURN $x_1^* = 1$. If $n > 1$ then continue.

Local optimality: Label the variables, and choose x_n to be eliminated. Determine the pseudo-Boolean function g_n defined by

$$g_n(x_1, ..., x_{n-1}) = \begin{cases} 1 & \text{if } \Delta_n(x_1, ..., x_{n-1}) < 0, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Recursion: Determine $f^{n-1}(x_1, ..., x_{n-1}) \stackrel{\text{def}}{=} f(x_1, ..., x_{n-1}, g_n(x_1, ..., x_{n-1}))$, and obtain the optimal values for x_1^*, x_2^*, ...,x_{n-1}^* by calling BasicAlgorithm(f^{n-1}).

Output: Set $x_n^* = g_n(x_1^*, ..., x_{n-1}^*)$, and RETURN the binary vector $\mathbf{x}^* = (x_1^*, ..., x_n^*)$.

Of course, the above algorithm needs the specification of what does it mean that a pseudo Boolean function is "determined", e.g. in the cases of g_n and f^{n-1} appearing in the two main steps of the above procedure. Let us say that we want all pseudo Boolean functions in the course of this procedure to be represented either by their unique multi-linear polynomial form, or by a posiform (though it might be possible to use some sort of oracle as a description for these functions).

Instead of giving a detailed proof here, for which the reader is referred to [91], we shall illustrate this algorithm on a small example.

Example 4.8 *Let us consider the following simple pseudo Boolean function f in three binary variables, defined by its unique multi-linear polynomial below:*

$$f(x_1, x_2, x_3) = 3 + 3x_1 - 2x_1x_2 - 4x_1x_3 + 5x_2x_3.$$

After calling BasicAlgorithm(f), *we get $\Delta_3 = -4x_1 + 5x_2$, and thus*

$$g_3(x_1, x_2) = \left\{ \begin{array}{ll} 1 & \textit{if } x_1 = 1 \textit{ and } x_2 = 0, \\ 0 & \textit{otherwise} \end{array} \right\} = x_1\overline{x}_2.$$

Thus, we get

$$\begin{aligned} f^2(x_1, x_2) &= f(x_1, x_2, x_1\overline{x}_2) = 3 + 3x_1 - 2x_1x_2 + x_1\overline{x}_2(5x_2 - 4x_1) \\ &= 3 - x_1 + 2x_1x_2 \end{aligned}$$

After calling BasicAlgorithm(f^2), *we get $\Delta_2 = 2x_1$, and therefore*

$$g_2(x_1) = \left\{ \begin{array}{ll} 1 & \textit{never,} \\ 0 & \textit{always} \end{array} \right\} = 0.$$

Hence,

$$f^1(x_1) = f^2(x_1, 0) = 3 - x_1.$$

After the call BasicAlgorithm(f^1)*, we have* $n = 1$*, and since* $f^1(0) > f^1(1)$*,* $x_1^* = 1$ *is returned.*

Then $x_2^* = g_2(x_1^*) = g_2(1) = 0$ *is computed and* $(x_1^*, x_2^*) = (1, 0)$ *is returned.*

Finally, $x_3^* = g_3(1, 0) = 1$ *is computed, and* $\mathbf{x}^* = (1, 0, 1)$ *is returned as a minimizer of* f*.*

Computationally, the above procedure can be very expensive, since both determining g_k and computing f^{k-1} can be intractable for inputs of realistic size (in a worst case, the size of both of these functions can be exponential in the input size).

There are however some special cases, when both the size of these functions, and the computing time for these steps can be controlled, and therefore the algorithm can be executed in polynomial time. Such a case was presented and analyzed in [47].

If a pseudo Boolean function $f : \mathbb{B}^n \mapsto \mathbb{R}$ is given by its unique multi-linear polynomial (1), let us associate to it a graph $G_f = (\mathbf{V}, E)$, called its *co-occurrence* graph, in which $(i, j) \in E$ (for $i, j \in \mathbf{V}$, $i \neq j$) iff f has a term for which $S \supseteq \{i, j\}$ and $c_S \neq 0$. We shall say that G_f is a *partial* k*-tree*, if there exists a supergraph $G^* = (\mathbf{V}, E^*)$, $E^* \supseteq E$ and a permutation $\pi \in \mathbb{S}_n$ of the indices in $\mathbf{V}$ such that the set $P^i \overset{\text{def}}{=} \{\pi_j \mid \pi_j < \pi_i,\ (j, i) \in E^*\} \subseteq \mathbf{V}$ is a clique in G^* and $|P^i| \leq k$, for every $i \in \mathbf{V}$.

Theorem 2 ([47]) *If for a pseudo Boolean function* f *its co-occurrence graph* G_f *is a partial* k*-tree, then* BasicAlgorithm *can be implemented to run in polynomial time in the input size* $\mathrm{size}(f)$ *and in* 2^k*.*

Proof. We shall present only a sketch of the proof here. To simplify notation, let us assume w.l.o.g. that the permutation π in the definition of a partial k-tree is $\pi = (1, 2, 3, ..., n)$.

Let us observe first that if $G_f = G_{f^n}$ is a partial k-tree, then Δ_n and thus g_n depend only on at most k of the variables, and hence their multi-linear polynomials have at most 2^k nonzero coefficients, which can be computed recursively by (12), in time polynomial in 2^k. The same applies to the computational needs of f^{n-1}.

Let us observe next, that if the substitution of g_n into $f = f^n$ (when computing f^{n-1}) generates new terms, and changes thus the co-occurrence graph, then for a new edge $(i, j) \in E(G_{f^{n-1}}) \setminus E(G_{f^n})$ we must have $\{i, j\} \subseteq P^n$, and hence the subgraph G^*_{n-1} of G^*, induced by $\mathbf{V} \setminus \{n\}$ is still a supergraph of $G_{f^{n-1}}$, i.e. the input to the next level is also a partial k-tree.

Since there are at most n recursive calls in the algorithm, the claimed complexity follows. $\square$

4.7 Posiform Maximization

Several of the results and methods we cited so far concern the minimization of posiforms, or simply the minimization (or maximization) of multi-linear polynomial expressions. While the minimization and the maximization of a polynomial expression are quite obviously equivalent problems (the simple change of sign providing the equivalence), there is no such simple connection between the minimization and maximization of posiforms. There is, of course, a linear transformation between these two problems (via the recursive substitutions $u = 1 - \overline{u}$ for certain literals u), however, this transformation typically changes the structure of the expressions, which is particularly damaging if one tries to translate approximation algorithms.

Let us further remark that posiforms are usually not only more concise representations of pseudo-Boolean functions than multi-linear polynomial expressions, but are also specially suggestive, since both the minimization and the maximization of posiforms have natural and intuitive interpretations.

Let us recall that for a given a graph $G = (V, E)$ its *stability number* $\alpha(G)$ is defined as the maximum size of a maximal independent vertex set (stable set). Furthermore, if there are weights $w : V \mapsto \mathbb{R}_+$ associated to the vertices, then the *weighted stability number* $\alpha_w(G)$ of G is defined as the maximum weight of a maximal independent vertex set, where the weight of a vertex set is the sum of the weights of the vertices in the set.

Given a posiform ϕ as in (5), let us associate to it a (weighted) graph $G_\phi = (\mathcal{T}, E)$, called its *conflict* graph. Vertices of G_ϕ correspond to the non-trivial terms of ϕ, i.e. $\mathcal{T} = \{T \subseteq \mathbf{L} | T \neq \emptyset,\ a_T > 0\}$. To a vertex $T \in \mathcal{T}$ we shall associate a_T as its weight. We shall say that two such terms $T, T' \in \mathcal{T}$ are in *conflict*, if there is a literal $u \in T$ for which $\overline{u} \in T'$. The edges of G_ϕ correspond to the conflicting pairs of terms, i.e. $E = \{(T, T') | T, T' \in \mathcal{T},\ T \text{ and } T' \text{ are in conflict}\}$.

Conflict graphs were introduced in [75] and the following interesting connection was shown:

Theorem 3 *For any posiform ϕ,*

$$\max_{\mathbf{x} \in \mathbb{B}^n} \phi(\mathbf{x}) = a_\emptyset + \alpha_a(G_\phi). \tag{18}$$

Proof. Given a binary vector $\mathbf{x} \in \mathbb{B}^n$, let us observe first that the set $\mathcal{S}_\mathbf{x} \stackrel{\text{def}}{=} \{T \in \mathcal{T} | T(\mathbf{x}) = 1\}$ consisting of the terms which do not vanish at $\mathbf{x}$ is in fact a stable set of the graph G_ϕ. This is clear, because no two of these terms can have a conflicting literal, since otherwise at least one of them would vanish in the point $\mathbf{x} \in \mathbb{B}^n$. Hence,

$$\phi(\mathbf{x}) = a_\emptyset + \sum_{T \in \mathcal{S}_\mathbf{x}} a_T \leq a_\emptyset + \alpha_a(G_\phi)$$

follows for all $\mathbf{x} \in \mathbb{B}^n$, by the definition of the weighted stability number.

Conversely, if $\mathcal{S} \subseteq \mathcal{T}$ is a stable set of G_ϕ, then the terms in $\mathcal{S}$ have no conflicting literals, and thus all literals appearing in these terms can be made simultaneously equal to 1. In other words, for any stable set $\mathcal{S} \subseteq \mathcal{T}$ there exists a binary vector $\mathbf{x}_\mathcal{S} \in \mathbb{B}^n$ such that $T(\mathbf{x}_\mathcal{S}) = 1$ for all $T \in \mathcal{S}$. Applying this observation to a maximum weight stable set $\mathcal{S}^*$, we get

$$a_\emptyset + \alpha_a(G_\phi) = a_\emptyset + \sum_{T \in \mathcal{S}^*} a_T = a_\emptyset + \sum_{T \in \mathcal{S}^*} a_T T(\mathbf{x}_{\mathcal{S}^*}) \leq \phi(\mathbf{x}_{\mathcal{S}^*}) \leq \max_{\mathbf{x} \in \mathbb{B}^n} \phi(\mathbf{x}).$$

□

Furthermore, it can be shown that, conversely, every weighted graph stability problem corresponds in this way to a posiform maximization.

Theorem 4 ([75]) *Given a graph $G = (V, E)$ and nonnegative weights $a_i \geq 0$ associated to the vertices $i \in V$, there exists a posiform ϕ_G in $n' < |V|$ variables, consisting of $|V|$ terms, and such that*

$$\alpha_a(G) = \max_{\mathbf{x} \in \mathbb{B}^{n'}} \phi_G(\mathbf{x}). \tag{19}$$

Proof. Let us consider a covering of the edges of G by (not necessarily induced) complete bipartite subgraphs of G, i.e. let $\mathcal{B} \stackrel{\text{def}}{=} \{(A_j, B_j) | j = 1, ..., n'\}$ be a family of pairs of subsets of the vertices, such that (i) $Aj \cap B_j = \emptyset$, (ii) $A_j \times B_j \subseteq E$ hold for all $j = 1, ..., n'$, and (iii) $E = \bigcup_{j=1}^{n'} A_j \times B_j$. Let us associate to such a covering $\mathcal{B}$ of the edges a posiform

$$\phi_\mathcal{B}(\mathbf{x}) \stackrel{\text{def}}{=} \sum_{i \in V} a_i \prod_{j: i \in A_j} x_j \prod_{j: i \in B_j} \overline{x}_j. \tag{20}$$

Let us remark that we use the convention of $\prod_{j \in \emptyset} x_1 = 1$, implying that the constant term of $\phi_\mathcal{B}$ is the sum of the weights of the isolated vertices of G. It is easy to verify that for any covering $\mathcal{B}$ of the edges by complete bipartite subgraphs, G is the conflict graph of the posiform $\phi_\mathcal{B}$.

Since all edges can be covered by at most $|V| - 1$ stars in any graph, the statement follows from Theorem 3. □

The two theorems above show that there is a very natural and strong equivalence between weighted graph stability and posiform maximization, even extending to approximative solutions and approximation algorithms. This connection shows also that posiform maximization, just like graph stability, is in all likelihood a difficult problem even if one needs only a good approximation (see e.g. [57, 136].)

On the other hand, this connection raises the problem of characterizing graph stability problems for which the corresponding posiform maximization problem has special characteristics, e.g., it involves a posiform of bounded degree. For instance the so called *quadratic graphs*, i.e. those for which there exists a family of complete bipartite subgraphs, covering all edges, such that the corresponding posiform (20) is quadratic, were studied in [22, 23, 43, 93]. Clearly, this condition means simply that the edge set of these graphs can be covered with a family $\mathcal{B}$ of complete bipartite subgraphs in such a way that each vertex belongs to at most 2 of the complete bipartite subgraphs in $\mathcal{B}$. In spite of their close formal resemblance with line graphs, no good characterization or recognition algorithm for quadratic graphs is known.

In a similar way in the reverse direction, this connection raises the problem of characterizing those posiforms for which the corresponding conflict graph has some special characteristics, e.g., bipartite, chordless, perfect, etc. For instance, a family of posiforms, having bipartite conflict graphs, was used in [24] to characterize supermodular cubic posiforms.

Let us remark finally that the above connection between graph stability and posiform maximization raises the possibility of applying algebraic manipulations to graph stability problems. Let us demonstrate this idea on a small example.

Example 4.9 *Let us consider the graph $G_1 = (V, E)$ given in Figure 1(a) and let us cover its edges with the family $\mathcal{B}_1 = \{(A_1, B_1), (A_2, B_2), (A_3, B_3), (A_4, B_4)\}$ of complete bipartite subgraphs consisting of 4 stars; here $A_i = \{i\}$ for $i = 1, ..4$, while $B_1 = \{2, 3, 7\}$, $B_2 = \{4, 5\}$, $B_3 = \{5, 6, 7\}$, and $B_4 = \{5, 6\}$. The corresponding posiform is*

$$\phi_1 = x_1 + \overline{x}_1 x_2 + \overline{x}_1 x_3 + \overline{x}_2 x_4 + \overline{x}_2 \overline{x}_3 \overline{x}_4 + \overline{x}_3 \overline{x}_4 + \overline{x}_1 \overline{x}_3.$$

Since in this example all complete bipartite subgraphs will actually be stars, we indicated these stars in the figures with small arcs along the edges pointing out form the center of the stars.

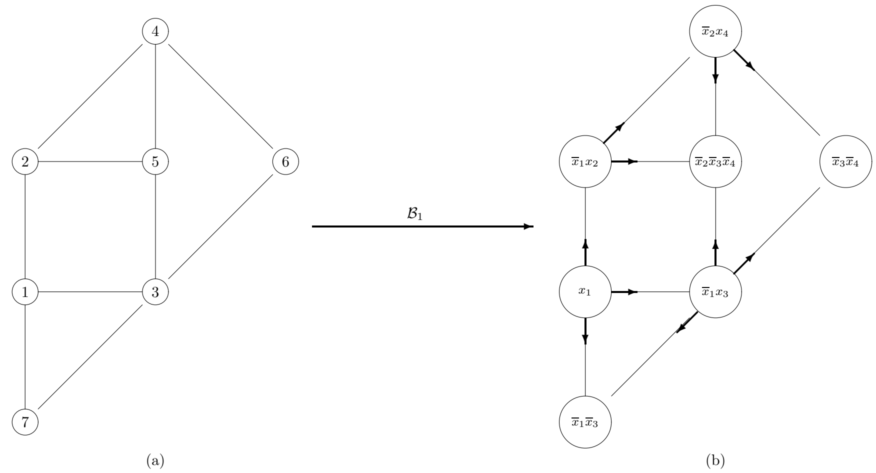


Figure 1: Graph G_1 of Example 4.9 and its conflict representation using the complete bipartite subgraph cover $\mathcal{B}_1$.

Applying the identities $\overline{x}_1 x_3 + \overline{x}_1 \overline{x}_3 = \overline{x}_1$, and $x_1 + \overline{x}_1 = 1$ we find that $\phi_1 = 1 + \phi_2$, where

$$\phi_2 = \overline{x}_2 x_4 + \overline{x}_1 x_2 + \overline{x}_2 \overline{x}_3 \overline{x}_4 + \overline{x}_3 \overline{x}_4.$$

The conflict graph $G_2 = G_{\phi_2}$ *is shown in Figure 2, together with another conflict representation of it, corresponding to the following complete bipartite subgraph covering of its edge set:* $\mathcal{B}_2 = \{(A_1, B_1),\ (A_2, B_2)\}$, *where* $A_1 = \{1\}$, $B_1 = \{2, 3, 4\}$, $A_2 = \{2\}$, *and* $B_2 = \{3\}$. *This representation yields the posiform*

$$\phi_3 = y_1 + \overline{y}_1 + \overline{y}_1 y_2 + \overline{y}_1 \overline{y}_2.$$

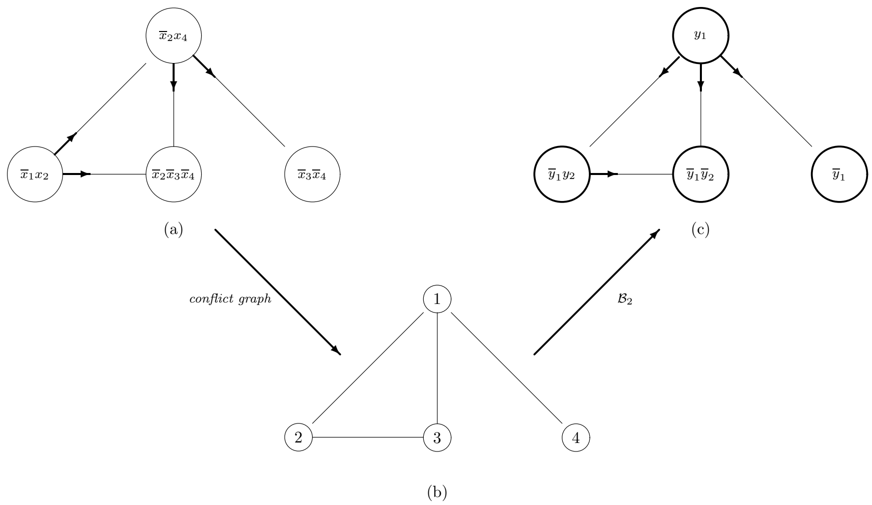


Figure 2: The conflict graph G_2 of the posiform ϕ_2 in Example 4.9, and its conflict representation using the complete bipartite subgraph cover $\mathcal{B}_2$.

Applying here again the trivial identities $\overline{y}_1 y_2 + \overline{y}_1 \overline{y}_2 = \overline{y}_1$ *and* $y_1 + \overline{y}_1 = 1$, *we obtain*

$$\phi_3 = 1 + \overline{y}_1.$$

Summarizing the above, we have

$$\begin{aligned}\alpha(G_1) &= \max_{\mathbf{x}\in\mathbb{B}^4} \phi_1(\mathbf{x}) \\ &= 1 + \max_{\mathbf{x}\in\mathbb{B}^4} \phi_2(\mathbf{x}) = 1 + \alpha(G_2) = 1 + \max_{\mathbf{y}\in\mathbb{B}^2} \phi_3(\mathbf{y}) \\ &\qquad\qquad\qquad\qquad\qquad\qquad = 2 + \max_{y_1\in\mathbb{B}} \overline{y}_1 = 3.\end{aligned}$$

Of course, not all graphs can be handled with the same efficiency. An algorithm, based on a systematic way of applying this type of algebraic manipulations, was proposed in [54], and studied further in e.g. [51, 83, 84, 102, 103, 104, 106]. Using algebraic manipulations of the type indicated above, the so called *struction algorithm* transforms a graph G into another graph G', such that

$$\alpha(G) = 1 + \alpha(G')$$

holds. Repeating this transformation, one can, in principle, arrive to a trivial graph, and hence can determine the stability number of G.

In reality, the sequence of graphs produced in this way may have an exponential growth in size [106], although several classes of graphs are known [83, 84, 102, 103, 104] where this difficulty does not arise, and the stability number of which can therefore be found in polynomial time by the repeated application of struction. In a recent study [4] it was observed that the application of a single step of the struction algorithm leads to a graph for which the gap between the upper and lower bounds of the stability number (obtained by using some of the usual techniques) is considerably smaller than the corresponding gap for the original graph.

The basic idea of the struction method is to select an arbitrary vertex, say v_0, of a graph G as the "center" of the transformation, impose an arbitrary order, say v_1, v_2, ..., v_k among the vertices in its neighborhood $N(v_0) = \{v \in V(G) | (v_0, v) \in E(G)\}$, use a family of stars with centers v_i, $i = 1, ..., k$ to cover all the edges which have at least one endpoint in the closed neighborhood $N[v_0] = N(v_0) \cup \{v_0\}$, and cover the remaining edges of the graph by an arbitrary family of other complete bipartite subgraphs. Let ϕ denote the posiform corresponding to this edge covering with complete bipartite subgraphs. It was shown in [54] that this posiform can always be written after some simple algebraic transformations as

$$\phi = 1 + \psi$$

where ψ is also a posiform, obtainable from ϕ in polynomial time. If G' is the conflict graph of ψ, then we have the equation

$$\alpha(G) = \max_{\mathbf{x} \in \mathbb{B}^n} \phi(\mathbf{x}) = 1 + \max_{\mathbf{x} \in \mathbb{B}^n} \psi(\mathbf{x}) = 1 + \alpha(G').$$

Let us illustrate this on a small example $G = (V, E)$ given in Figure 3, in which all vertex weights are equal to 1.

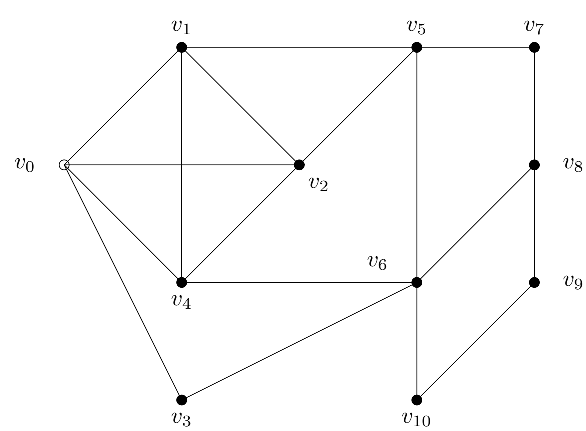


Figure 3:

For example, if we choose vertex v_0 as the center of struction, if the stars used in the transformation are $S_1 = \{\{v_1\}, \{v_0, v_2, v_4, v_5\}\}$, $S_1 = \{\{v_2\}, \{v_0, v_4, v_5\}\}$, $S_3 = \{\{v_3\}, \{v_0, v_6\}\}$, $S_4 = \{\{v_4\}, \{v_0, v_6\}\}$, if we cover the rest of the edges with an arbitrary family $\mathcal{B}$ of bipartite subgraphs, and we denote the variable associated to the star S_i, by x_i, $i = 1, ..., 4$, then the posiform corresponding to this edge covering will be

$$\phi(\mathbf{x}) = \overline{x}_1\overline{x}_2\overline{x}_3\overline{x}_4 + x_1 + \overline{x}_1 x_2 + x_3 + \overline{x}_1\overline{x}_2 x_4 + \psi(\mathbf{x})$$

where ψ is the posiform corresponding to $\mathcal{B}$. It is easy to see that

$$\overline{x}_1\overline{x}_2\overline{x}_3\overline{x}_4 + x_1 + \overline{x}_1 x_2 + x_3 + \overline{x}_1\overline{x}_2 x_4 = 1 + x_1 x_3 + \overline{x}_1 x_2 x_3 + \overline{x}_1\overline{x}_2 x_3 x_4$$

and hence

$$\phi(\mathbf{x}) = 1 + x_1 x_3 + \overline{x}_1 x_2 x_3 + \overline{x}_1\overline{x}_2 x_3 x_4 + \psi(\mathbf{x}).$$

In fact it was shown in [54] that a similar transformation which allows to write the original posiform as the sum of a positive constant and another posiform can always be carried out. It was also shown that this procedure has a direct and easy graph theoretic interpretation, describing directly the construction of the conflict graph of the new posiform, and by-passing all the algebraic manipulations.

Returning to the example, notice that each of the three new non-constant terms on the right hand side above contains exactly two non-complemented variables, and that a 1-to-1 correspondence can be established between them and the non-edges of the subgraph induced by $N(v_0)$. The conflict graph of the posiform $\phi(\mathbf{x}) - 1$ on the right hand side is shown in Figure 4 (where we have denoted by $v_{i,j}$ the vertex associated to the non-edge (v_i, v_j) in the subgraph induced by $N(v_0)$).

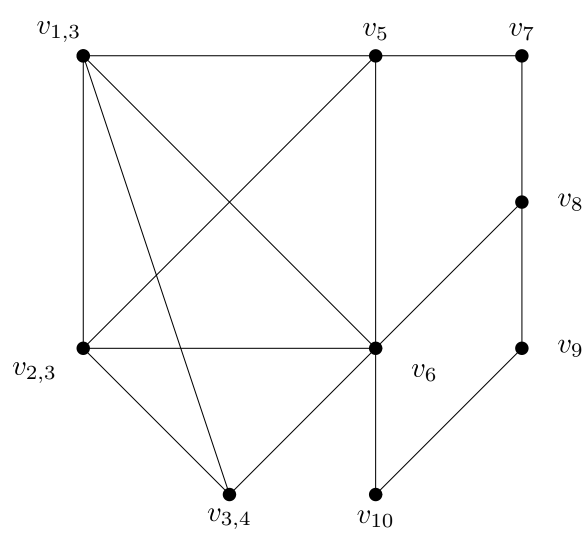


Figure 4:

Clearly, the stability number of this graph is exactly one unit smaller than that of the original graph. Repeating now the same transformation to the new graph and taking $v_{1,3}$ as the center of the struction, we find the graph in Figure 5, whose stability number is exactly 2 units smaller than that of the original graph. By executing two additional struction steps,

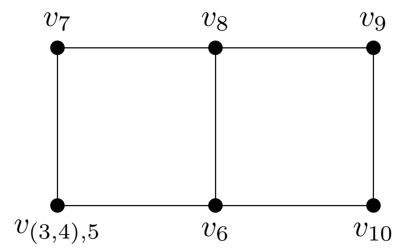
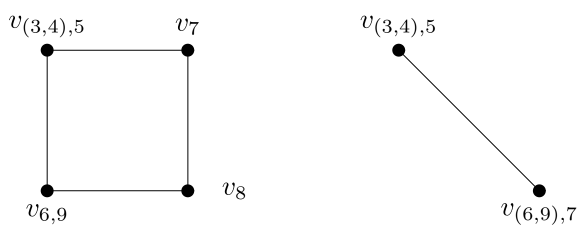


Figure 5:

using first v_{10} as center, and then v_8 as center, produces in turn the graphs in Figures 6 (a) and (b), showing that the stability number of the original graph, being 4 units higher than that of the last graph produced, was equal to 5.



(a) (b)

Figure 6:

4.8 l_2-Approximations and Applications to Game Theory

In this section, following [80], we shall consider lower degree l_2-approximations of pseudo-Boolean functions.

Let us denote by $\mathcal{F}_k$ the family of pseudo-Boolean functions of degree at most k ($k = 0, 1, ..., n$). Clearly, $\mathcal{F}_0 \subset \mathcal{F}_1 \subset \cdots \subset \mathcal{F}_n$.

Given a pseudo-Boolean function f and a fixed integer $k \geq 0$, let us consider the following approximation problem:

$$\min_{g \in \mathcal{F}_k} \sum_{\mathbf{x} \in \mathbb{B}^n} [g(\mathbf{x}) - f(\mathbf{x})]^2. \tag{21}$$

Proposition 8 *Problem (21) has a unique solution.*

Proof. Let us consider pseudo-Boolean functions in n variables represented by their table form, or equivalently, as vectors in $\mathbb{R}^{2^n}$. Then,

$$W_k \stackrel{\text{def}}{=} \{(g(\mathbf{x})|\mathbf{x} \in \mathbb{B}^n) \mid g \in \mathcal{F}_k\} \tag{22}$$

is clearly a linear subspace of $\mathbb{R}^{2^n}$, for $k = 0, 1, ..., n$. It is immediate to see then that Problem (21) is equivalent to computing the orthogonal projection of $(f(\mathbf{x})|\mathbf{x} \in \mathbb{B}^n)$ onto W_k, and hence the solution exists and is unique. $\square$

Let us denote in the sequel by $A_k[f] \in \mathcal{F}_k$ the unique solution of Problem (21), which provides the best l_2-approximation of f.

Given a subset $S \subseteq \mathbb{B}^n$, let us denote the average value of the function f over the subset S by $\text{Ave}_S[f]$, i.e.

$$\text{Ave}_S[f] = \frac{1}{|S|} \sum_{\mathbf{x} \in S} f(\mathbf{x}). \tag{23}$$

Given a term $t(\mathbf{x}) = \prod_{u \in T} u$ (where $T \subseteq \mathbf{L}$), let us associate to it the Boolean subcube $\mathbb{B}_T$ defined by

$$\mathbb{B}_T = \{\mathbf{x} \in \mathbb{B}^n | t(\mathbf{x}) = 1\}. \tag{24}$$

Furthermore, let us associate to T the half-integral vector $\mathbf{z}^T \in \{0, \frac{1}{2}, 1\}^n$, defined by

$$(\mathbf{z}^T)_j = \begin{cases} 1 & \text{if } x_j \in T, \\ 0 & \text{if } \overline{x}_j \in T, and \\ \frac{1}{2} & \text{otherwise.} \end{cases}$$

Before showing a characterizing property of l_2-approximations, we need to state two simple lemmas.

Lemma 1 *For a pseudo-Boolean function f, and a subset $T \subseteq \mathbf{L}$ we have*

$$\text{Ave}_T[f] = f(\mathbf{z}^T).$$

Proof. Follows from Proposition 5. $\square$

Lemma 2 *Given a sequence of m reals, a_1, a_2, ..., a_m, their average $\gamma = \text{Ave}[a_1, a_2, ..., a_m]$ is the unique minimizing point of the expression*

$$\sum_{i=1}^{m} (\gamma - a_i)^2.$$

Proof. This well-known fact follows immediately from the equation

$$0 = \frac{d}{d\gamma} \sum_{i=1}^{m} (\gamma - a_i)^2 = 2 \sum_{i=1}^{m} (\gamma - a_i),$$

since $\sum_{i=1}^{m} (\gamma - a_i)^2$ is a strictly convex function of γ. □

We are now ready to state a characterizing property of l_2-approximations.

Theorem 5 *Given a pseudo-Boolean function f in n variables, a function $g \in \mathcal{F}_k$ is equal to $A_k[f]$ if and only if*

$$\mathrm{Ave}_{\mathbb{B}_T}[f] = \mathrm{Ave}_{\mathbb{B}_T}[g] \tag{25}$$

for all subsets of literals $T \subseteq \mathbf{L}$ with $|T| \leq k$.

Proof. Let us assume first that $g = A_k[f]$. Choosing an arbitrary subset $T \subseteq \mathbf{L}$ for which $|T| \leq k$, let $t(\mathbf{x}) = \prod_{u \in T} u$, and let us consider the optimization problem

$$\min_{\gamma \in \mathbb{R}} \sum_{\mathbf{x} \in \mathbb{B}^n} \left[g(\mathbf{x}) + \gamma t(\mathbf{x}) - f(\mathbf{x}) \right]^2 .$$

On the one hand, we can see immediately that this problem has $\gamma = 0$ as its unique minimum, since $g' = g + \gamma t$ is also a function belonging to $\mathcal{F}_k$ (because of $|T| \leq k$), and $g = A_k[f]$ is a unique minimizer of $\sum_{\mathbf{x} \in \mathbb{B}^n} [g'(\mathbf{x}) - f(\mathbf{x})]^2$ in $\mathcal{F}_k$.

On the other hand, we can observe that only those terms can influence the above minimization problem in which $t(\mathbf{x}) \neq 0$, i.e. that the problem can be written equivalently as

$$\min_{\gamma \in \mathbb{R}} \sum_{\mathbf{x} \in \mathbb{B}_T} \left[g(\mathbf{x}) + \gamma - f(\mathbf{x}) \right]^2 = \min_{\gamma \in \mathbb{R}} \sum_{\mathbf{x} \in \mathbb{B}_T} \left[\gamma - (f(\mathbf{x}) - g(\mathbf{x})) \right]^2 .$$

Applying Lemma 2 to the sequence $\{(f(\mathbf{x}) - g(\mathbf{x})) | \mathbf{x} \in \mathbb{B}_T\}$, we obtain

$$0 = \mathrm{Ave}_{\mathbb{B}_T}[f - g] = \mathrm{Ave}_{\mathbb{B}_T}[f] - \mathrm{Ave}_{\mathbb{B}_T}[g], \tag{26}$$

thus proving (25).

For the converse direction, let us assume that (25) holds for all subsets $T \subseteq \mathbf{L}$ with $|T| \leq k$. We shall show that these equations determine $g \in \mathcal{F}_k$ uniquely, implying then the statement, since we have just shown above that the same equations hold for $A_k[f] \in \mathcal{F}_k$, too.

For this end, let us rewrite equations (25) with the help of Lemma 1 as

$$g(\mathbf{z}^T) = f(\mathbf{z}^T) \text{ for all } T \subseteq \mathbf{L},\ |T| \leq k, \tag{27}$$

and let us assume that both $g \in \mathcal{F}_k$ and $g' \in \mathcal{F}_k$ satisfy all these equalities. Then,

$$g(\mathbf{z}^T) - g'(\mathbf{z}^T) = 0$$

must also hold for all subsets $T \subseteq \mathbf{L}$, $|T| \leq k$. Let us represent $g - g' \in \mathcal{F}_k$ by its unique multilinear polynomial form

$$g(\mathbf{x}) - g'(\mathbf{x}) = \sum_{S \subseteq \mathbf{V}, |S| \leq k} \hat{c}_S \prod_{j \in S} x_j,$$

assume that $g \neq g'$, and denote by S a smallest set for which $\hat{c}_S \neq 0$. Then,

$$0 = g(\mathbf{z}^S) - g'(\mathbf{z}^S) = \sum_{S' \subseteq S} \hat{c}_{S'} = \hat{c}_S$$

since $\hat{c}_{S'} = 0$ for all proper subsets S' of S. Thus we get $\hat{c}_S = 0$, contradicting the choice of S, and hence our assumption that $g \neq g'$, thus proving the uniqueness of g. □

Let us observe that in fact we have shown somewhat more than the claim of Theorem 5. Namely, if equations (27) hold for two pseudo-Boolean functions, f and g, then g and $A_k[f]$ have the same coefficients in their multilinear polynomials for all terms of degree at most k.

Let us also observe that the equations (27), which form a system of linear equations in the coefficients of the multilinear polynomial representing $g = A_k[f]$, provide a simple and direct way of computing the best l_2-approximation $A_k[f]$ of a given pseudo-Boolean function f.

Proposition 9 *Given a pseudo-Boolean function f, the multilinear polynomial expression for $A_k[f]$ can be computed in time polynomial in $N_k^n = \sum_{i=0}^{k} \binom{n}{i}$, by evaluating f in N_k^n points.*

Proof. It can easily be seen that writing the equations (27) e.g. for all sets $T \subseteq \mathbf{L}^+$, which do not involve complemented literals and for which $|T| \leq k$, we obtain a full rank system of linear equations in the N_k^n coefficients of the multilinear polynomial expression of $A_k[f]$. □

Since A_k is an additive operator, perhaps the simplest way to compute the best l_2 approximations is to compute separately the sums of the l_2 approximations of simple terms, e.g.

$$A_k \left[\sum_{S \subseteq \mathbf{V}} c_S \prod_{j \in S} x_j \right] = \sum_{S \subseteq \mathbf{V}} c_S A_k \left[\prod_{j \in S} x_j \right].$$

For instance, the best l_2 approximations were computed in this way in [80] for some low values of k.

Proposition 10 (see [80]) *For $S \subseteq \mathbf{V}$ we have*

$$\begin{aligned} A_1 \left[\prod_{j \in S} x_j \right] &= -\frac{|S| - 1}{2^{|S|}} + \frac{1}{2^{|S|-1}} \sum_{j \in S} x_j, \ \textit{and} \\ A_2 \left[\prod_{j \in S} x_j \right] &= \frac{(|S| - 1)(|S| - 2)}{2^{|S|+1}} - \frac{|S| - 2}{2^{|S|-1}} \sum_{j \in S} x_j + \frac{1}{2^{|S|-2}} \sum_{\substack{i,j \in S \\ i < j}} x_i x_j. \end{aligned} \tag{28}$$

$\square$

Linear l_2-approximations play an important role in game theory.

Let us consider $\mathbf{V}$ as the set of *players* of a multiperson game, where the subsets $S \subseteq \mathbf{V}$ are called *coalitions*. The game (more precisely, a game with side payments in characteristic function form) is given by a function $\nu : 2^{\mathbf{V}} \mapsto \mathbb{R}$, for which it is assumed that $\nu(\emptyset) = 0$. We can view the value $\nu(S)$ as the "worth" of coalition S. Identifying $2^{\mathbf{V}}$ with $\mathbb{B}^n$ in the usual natural way, we can view all such games as pseudo-Boolean functions ν for which $\nu(0, 0, ..., 0) = 0$.

A game is called *simple* if $\nu(\mathbf{x}) \in \mathbb{B}$ for all $\mathbf{x} \in \mathbb{B}^n$, and ν is monotone, i.e. if $\nu(S) \leq \nu(S')$ whenever $S \subseteq S'$. In this case, the coalitions $S \subseteq \mathbf{V}$ for which $\nu(\mathbf{1}^S) = 1$ are called *winning*, while the rest are *loosing*. Given a simple game ν, one would like to be able to compute real values p_i for $i \in \mathbf{V}$, representing the "influence" of the individual players on the game. Such values are called *power indices*. One of the power indices considered in the literature, and in many applications, computes p_i as the proportion of those loosing coalitions which turn winning if player i joins them. More precisely, the so called *Banzhaf index* (see [15]) is defined as

$$p_i = Ave \left[\frac{d\nu}{dx_i} \right]$$

for $i \in \mathbf{V}$. It can be seen easily from our analysis that these indices are in fact the linear coefficients of $A_1[\nu]$.

Another, similar index was introduced by Shapley (see [154]), and that also can be shown to correspond to the linear coefficients in a best "linear"-approximation of ν in a slightly differently weighted norm (see [80]).

5 Quadratic Pseudo-Boolean Functions

This part is devoted to the study of quadratic pseudo-Boolean functions, a class of functions arising naturally in the formulation of numerous combinatorial optimization problems. The importance of these functions is further explained by the fact – discussed in Section 4.4 – that all pseudo-Boolean optimization problems can be reduced to the quadratic case. In this section we shall survey results regarding bounds, structural properties, polyhedral representations, as well as heuristic and exact optimization algorithms for this class of pseudo-Boolean functions.

Let us remark that in the case of quadratic pseudo-Boolean functions, optimization of polynomial representations is computationally equivalent with the optimization of quadratic posiform representations, since transformations between these two forms are polynomial.

For the sake of uniformity, we shall present all results for the case of minimization problems, but obviously, all results mentioned in the sequel can naturally be re-formulated for the case of maximization problems.

Let us start by recalling some results regarding a useful upper bound of quadratic pseudo-Boolean functions, called the *roof dual*, introduced in [79], which was shown to be obtainable through several alternative approaches. Next, we show that these approaches, as well as the bound given by them, can be generalized to a hierarchy of increasingly tighter bounds. Following [135], we shall recall results about polyhedral formulations of these optimization problems, and finally survey some algorithms for quadratic pseudo-Boolean optimization.

Let us denote by $\mathcal{F}_2$ the family of quadratic pseudo-Boolean functions, and specializing the notations introduced earlier, let us assume that they are represented either by their (unique) quadratic polynomial expression

$$f(x_1, \cdots, x_n) = c_0 + \sum_{j=1}^{n} c_j x_j + \sum_{1 \le i < j \le n} c_{ij} x_i x_j, \tag{29}$$

or by a quadratic posiform

$$f(x_1, \cdots, x_n) = a_0 + \sum_{u \in \mathbf{L}} a_u u + \sum_{u,v \in \mathbf{L}, u \neq v} a_{uv} uv, \tag{30}$$

where, as before, $\mathbf{L}$ denotes the set of literals, $a_u \ge 0$ and $a_{uv} \ge 0$ for all $u, v \in \mathbf{L}$.

Let us remark that among the posiforms representing a quadratic pseudo-Boolean function there may also be some having degrees higher than 2.

Example 5.1 *Let us consider the following quadratic pseudo-Boolean function*

$$f(x_1, x_2, x_3) = 1 - x_1 - x_2 - x_3 + x_1 x_2 + x_1 x_3 + x_2 x_3$$

along with two different posiform representations of it:

$$\begin{aligned} f(x_1, x_2, x_3) &= -1 + \overline{x}_1 + x_1 x_2 + x_1 x_3 + \overline{x}_2 \overline{x}_3 \\ &= x_1 x_2 x_3 + \overline{x}_1 \overline{x}_2 \overline{x}_3. \end{aligned}$$

The equivalence of these expressions can be shown easily by expanding the complementations in the posiforms.

Let us denote by $\mathcal{P}_k$ the family of posiforms of degree at most k, for $k \ge 1$, let $\mathcal{P} = \bigcup_{k \ge 0} \mathcal{P}_k$, and for $k \ge \deg(f)$ let $\mathcal{P}_k(f)$ denote the family of those posiforms of degree at most k, which represent the pseudo-Boolean function f. Let us also denote by $C(\phi)$ the constant term of a posiform $\phi \in \mathcal{P}$, and let us note that $\phi - C(\phi) \ge 0$ holds for all posiforms $\phi \in \mathcal{P}$.

5.1 Roof Duality

Three different approaches were shown in [79] to yield the same upper bound for a quadratic pseudo-Boolean function $f \in \mathcal{F}_2$. This common upper bound was called the *roof dual* of f.

Here we recall these results reformulated for the case of lower bounds, for which the term floor dual would perhaps be more suitable. However, we keep the name *roof dual* in order to emphasize that all the results are perfectly analogous for the case of upper and lower bounds.

Roof duality has been studied in many papers since its introduction in [79], and its strong relation to several other basic techniques was demonstrated in several publications, together with numerous generalizations and algorithmic improvements (see e.g., [1, 2, 25, 28, 30, 31, 36, 38, 81, 94, 125, 156]).

We recall first the three basic techniques, leading to the same bound, as shown in [79].

5.1.1 Majorization

A term-by-term majorization procedure, introduced in [79], provides a useful upper bound to a pseudo-Boolean function f, by appropriately selecting the linear majorants of the terms of its polynomial representation. Analogously, we can find lower bounds based on linear minorants of the terms.

The method of developing a "good" linear majorant (or minorant) of a pseudo-Boolean function appeared in the literature much earlier (see e.g. [77, 89, 150]) and was also applied to graph stability [78]. Here we recall the technique from [79], translated to the case of minimization.

A linear function

$$l(\mathbf{x}) = l_0 + l_1 x_1 + \cdots + l_n x_n$$

is called a *linear minorant* of $f \in \mathcal{F}_2$, if $l(\mathbf{x}) \leq f(\mathbf{x})$ holds for all $\mathbf{x} \in \mathbb{B}^n$. Let us denote by $\mathcal{F}_1$ the family of linear functions.

A family $\mathcal{S}$ of linear minorants of f is called *complete* if f is the pointwise maximum of these linear functions, i.e. if

$$f(\mathbf{x}) = \max_{l \in \mathcal{S}} l(\mathbf{x})$$

for all $\mathbf{x} \in \mathbb{B}^n$. For a complete family of minorants, we have the equality

$$\min_{\mathbf{x} \in \mathbb{B}^n} f(\mathbf{x}) = \min_{\mathbf{x} \in \mathbb{B}^n} \max_{l \in \mathcal{S}} l(\mathbf{x}).$$

Interchanging the minimization and maximization on the right hand side above, we obtain a lower bound to the minimum of f:

$$M(f, \mathcal{S}) \stackrel{\text{def}}{=} \max_{l \in \mathcal{S}} \min_{\mathbf{x} \in \mathbb{B}^n} l(\mathbf{x}) \leq \min_{\mathbf{x} \in \mathbb{B}^n} f(\mathbf{x}). \tag{31}$$

For an efficiently computable lower bound of this type, we shall consider a complete family of linear minorants, formed by combining "best l_1-norm" linear minorants of the quadratic terms of $f \in \mathcal{F}_2$.

It can easily be seen that all linear minorants $\alpha + \beta x + \gamma y$ of the quadratic term xy, which minimize the sum of the gaps for all four possible binary substitutions, i.e. which are solutions of the problem (in variables α, β and γ)

$$\min \sum_{(x,y)\in\mathbb{B}^2} [xy - \alpha - \beta x - \gamma y]$$
$$s.t. \quad xy \geq \alpha + \beta x + \gamma y \text{ for all } (x,y) \in \mathbb{B}^2$$

are of the form $-\alpha = \beta = \gamma = \lambda$, for any $0 \leq \lambda \leq 1$. Similarly, it can be shown that the best l_1-norm linear minorants of $-xy$ are of the form $-\lambda x - (1-\lambda)y$, for any $0 \leq \lambda \leq 1$.

Given a quadratic pseudo-Boolean function $f \in \mathcal{F}_2$ as in (29), let us define a family of linear minorants of it, by taking the weighted sum of the best l_1-norm linear minorants of its terms, and using as weights the coefficients of the terms, i.e.

$$\mathcal{R}(f) \stackrel{\text{def}}{=} \left\{ \begin{array}{c} c_0 + \sum_{j=1}^n c_j x_j \\ +\sum_{\substack{1\leq i<j\leq n \\ c_{ij}>0}} c_{ij}\lambda_{ij}[x_i + x_j - 1] \\ +\sum_{\substack{1\leq i<j\leq n \\ c_{ij}<0}} c_{ij}[\lambda_{ij}x_i + (1-\lambda_{ij})x_j] \end{array} \middle| \begin{array}{c} 0 \leq \lambda_{ij} \leq 1 \\ 1 \leq i < j \leq n \\ c_{ij} \neq 0 \end{array} \right\}.$$

In [79] the analogously defined linear majorants are called *roof*s. Let us call here the corresponding lower bound

$$M_2(f) \stackrel{\text{def}}{=} M(f, \mathcal{R}(f)) = \max_{l\in\mathcal{R}(f)} \min_{\mathbf{x}\in\mathbb{B}^n} l(\mathbf{x}) \tag{32}$$

the *roof dual* of f.

It was demonstrated in [79] that the same bound can be computed by solving a linear programming problem. To see this, let us observe first that if $l(\mathbf{x}) = l_0 + l_1x_1 + \cdots + l_nx_n$, then $\min_{\mathbf{x}\in\mathbb{B}^n} l(\mathbf{x})$ is the maximum value of the linear program

$$l_0 + \textstyle\sum_{j=1}^n z_j \longrightarrow \max$$

$$l_j \geq z_j \text{ and } 0 \geq z_j \text{ for } j = 1, ..., n.$$

Replacing then the inner minimization in the right hand side of (32) with the above maximization, we find that $M_2(f)$ is the maximum value of the following linear program:

$$\begin{array}{rl} \text{Maximize} & c_0 - \displaystyle\sum_{\substack{1\leq i<j\leq n \\ c_{ij}>0}} \lambda_{ij}c_{ij} + \sum_{j=1}^n z_j \\ \text{subject to} & c_j + \displaystyle\sum_{\substack{i\neq j \\ c_{ij}>0}} \lambda_{ij}c_{ij} + \sum_{\substack{1\leq i<j \\ c_{ij}<0}} c_{ij}(1-\lambda_{ij}) + \sum_{\substack{j<i\leq n \\ c_{ij}<0}} c_{ij}\lambda_{ij} \geq z_j \\ & 0 \geq z_j \text{ for } j = 1, \ldots, n, \\ & 0 \leq \lambda_{ij} \leq 1 \text{ for } 1 \leq i < j \leq n,\ c_{ij} \neq 0. \end{array} \tag{33}$$

5.1.2 Linearization

Linearization is a standard, and quite natural technique to reduce nonlinear binary optimization to linear integer programming. The first publication of this nature is perhaps [60], and there were many others to follow, see e.g. [3, 12, 55, 67, 68, 88, 146, 163].

The basic idea in all these transformations is to replace a non-linear term by a new variable, and use linear inequalities to force the new variable to take in all feasible solutions the value of the corresponding term. For instance, to enforce the equality $u = xyz$ for binary variables $x, y, z \in \mathbb{B}$, we can write

$$u \leq x, \tag{34a}$$
$$u \leq y, \tag{34b}$$
$$u \leq z, \tag{34c}$$
$$u \geq x + y + z - 2, \tag{34d}$$
$$u \geq 0. \tag{34e}$$

It is easy to see that for all binary assignments to x, y and z, the only feasible assignment to the variable u is the value xyz. Let us also add that in a minimization problem the constraints (34a)-(34c) are redundant if u has a positive objective function coefficient, and the constraints (34d)-(34e) are redundant when the objective function coefficient of u is negative (and the other way around for maximization problems.)

Applying this re-formulation to the minimization of $f \in \mathcal{F}_2$ given by (29), and introducing new variables $y_{ij} = x_i x_j$ for all quadratic terms of f, we obtain the following equivalent linear $0 - 1$ programming problem:

$$\text{Minimize} \quad c_0 + \textstyle\sum_{j=1}^n c_j x_j + \sum_{1 \leq i < j \leq n} c_{ij} y_{ij}$$

subject to

$$\left.\begin{array}{r} y_{ij} \geq x_i + x_j - 1 \\ y_{ij} \geq 0 \end{array}\right\} \quad 1 \leq i < j \leq n, \quad c_{ij} > 0, \tag{35}$$

$$\left.\begin{array}{r} y_{ij} \leq x_i \\ y_{ij} \leq x_j \end{array}\right\} \quad 1 \leq i < j \leq n, \quad c_{ij} < 0,$$

$$x_j \in \mathbb{B}, \quad 1 \leq j \leq n.$$

Replacing in the above formulation the integrality conditions on $\mathbf{x}$ by the constraints $0 \leq x_j \leq 1$ for $j = 1, ..., n$, we obtain a linear programming relaxation, the optimum value of which will be denoted by $L_2(f)$. Clearly, $L_2(f)$ is a lower bound of the minimum of f.

There are several variations of such linearization methods. For instance, we could also consider a quadratic posiform $\phi \in \mathcal{P}_2(f)$, and introduce new variables for the products of literals appearing in ϕ. Assuming that ϕ is represented by the form (30), we obtain the formulation

$$
\begin{aligned}
&\text{Minimize} \;\; a_0 + \textstyle\sum_{u\in\mathbf{L}} a_u u + \sum_{u,v\in\mathbf{L},u\neq v} a_{uv} y_{uv} \\
&\text{subject to} \\
&\left.\begin{aligned} y_{uv} &\geq u+v-1 \\ y_{uv} &\geq 0 \end{aligned}\right\} \quad u,v\in\mathbf{L}, \quad a_{uv}>0 \\
&\qquad\qquad x_j \in \mathbb{B}, \quad 1\leq j\leq n.
\end{aligned}
\tag{36}
$$

(Here of course we write in the constraints $1 - x_j$ for $u = \overline{x}_j$ and x_j for $u = x_j$.) Let us replace again the integrality constraints $\mathbf{x} \in \mathbb{B}^n$ by $0 \leq x_j \leq 1$ for $j = 1, ..., n$, and let us denote by $L_2(\phi)$ the optimum value of this linear programming relaxation of problem (36). It is obvious again that $L_2(\phi)$ is a lower bound of the minimum of ϕ.

5.1.3 Complementation

The third approach considered in [79] is based on representing a quadratic pseudo-Boolean function $f \in \mathcal{F}_2$ by a quadratic posiform $\phi \in \mathcal{P}_2(f)$, and using the constant term $C(\phi)$ of ϕ as a lower bound of the minimum of f. Then the problem of finding the best lower bound of this type can be formulated as

$$C_2(f) \stackrel{\text{def}}{=} \max_{\phi\in\mathcal{P}_2(f)} C(\phi). \tag{37}$$

This problem can also be formulated as a linear program. For this, let us consider the coefficients of the posiform ϕ as unknowns, and let us formulate the conditions that this posiform (30) represents the function f given by (29). From these conditions we get the following equalities:

$$c_0 = a_0 + \sum_{j=1}^{n} a_{\overline{x}_j} + \sum_{1\leq i<j\leq n} a_{\overline{x}_i\overline{x}_j}, \tag{38a}$$

$$c_j = a_{x_j} - a_{\overline{x}_j} + \sum_{\substack{1\leq i\leq n\\ i\neq j}} (a_{\overline{x}_i x_j} - a_{\overline{x}_i\overline{x}_j}) \text{ for } j = 1,\ldots,n, \tag{38b}$$

$$c_{ij} = a_{x_j x_j} + a_{\overline{x}_i\overline{x}_j} - a_{x_i\overline{x}_j} - a_{\overline{x}_i x_j} \text{ for } 1\leq i<j\leq n. \tag{38c}$$

Expressing a_0 from the first equation (38a), we can write $C_2(f)$ equivalently as the maximum value of the linear programming problem

$$\text{Maximize} \quad c_0 - \sum_{j=1}^{n} a_{\overline{x}_j} - \sum_{1\le i<j\le n} a_{\overline{x}_i\overline{x}_j}$$

subject to

$$\begin{aligned} a_{x_j} - a_{\overline{x}_j} + \sum_{\substack{1\le i\le n\\ i\ne j}} (a_{\overline{x}_i x_j} - a_{\overline{x}_i\overline{x}_j}) &= c_j && \text{for } j = 1,\ldots,n,\\ a_{x_j x_j} + a_{\overline{x}_i\overline{x}_j} - a_{x_i\overline{x}_j} - a_{\overline{x}_i x_j} &= c_{ij} && \text{for } 1\le i<j\le n,\\ a_u, a_{uv} &\ge 0 && \text{for } u,v\in \mathbf{L},\ u\ne v. \end{aligned} \tag{39}$$

5.1.4 Equivalence of Formulations and Persistency

The first major result of [79] is that the three approaches described in the previous three subsections for obtaining a lower bound for the minimization of $f \in \mathcal{F}_2$, are in fact equivalent, in the sense that they all yield the same value:

Theorem 6 ([79]) *For any quadratic pseudo-Boolean function $f \in \mathcal{F}_2$, we have*

$$M_2(f) = L_2(f) = C_2(f) \le \min_{\mathbf{x}\in\mathbb{B}^n} f(\mathbf{x});$$

moreover, the equality of these lower bounds with the minimum of $f(\mathbf{x})$ can be tested in linear time by solving a 2-SAT problem. □

The somewhat technical proof, which the reader can find in [79], is based on showing that formulations (33) and (39) are both equivalent with the dual of (35). Using a similar proof, one can also show that

Theorem 7 *For a quadratic pseudo-Boolean function $f \in \mathcal{F}_2$ and for any posiform representation $\phi \in \mathcal{P}_2(f)$ of it, the equality $C_2(f) = L_2(\phi)$ holds.* □

Some other formulations are also known to be equivalent with roof duality; these include Lagrangean formulations, the so-called "paved duality", etc. (see e.g. [1, 2, 42, 77, 94, 125].)

The second major result of [79] provides a strong persistency property for quadratic pseudo-Boolean optimization:

Theorem 8 (Strong Persistency, [79]) *Given a quadratic pseudo-Boolean function $f \in \mathcal{F}_2$, let $\phi \in \mathcal{P}_2(f)$ be a posiform representing it (given as in (30)), such that $C(\phi) = C_2(\phi)$. Then ϕ has the property that if $a_u > 0$ for some literal $u \in \mathbf{L}$, then $u = 0$ in all binary vectors $\mathbf{x} \in \mathrm{Argmin}(f)$ minimizing f.*

Let us denote by $L(\phi)$ the *linear part* of the posiform (or pseudo-Boolean function) ϕ, e.g. if ϕ is given by (30), then $L(\phi) = a_0 + \sum_{u \in \mathbf{L}} a_u u$. The above theorem implies then that $L(\phi)$ must vanish at any minimum of f, whenever $C(\phi) = C_2(\phi)$. Let us denote by $\mathcal{P}_2^*(f)$ the family of such posiforms, i.e. $\mathcal{P}_2^*(f) = \{\phi \in \mathcal{P}_2(f) | C(\phi) = C_2(f)\}$. The linear parts $L(\phi)$ of such extremal quadratic posiforms $\phi \in \mathcal{P}_2^*(f)$ are the roofs of f. Clearly, $\mathcal{P}_2^*(f)$ is a convex subset of the family of quadratic pseudo-Boolean functions, implying

Proposition 11 ([79]) *For every $f \in \mathcal{F}_2$, there exists a posiform $\phi^* \in \mathcal{P}_2^*(f)$ such that $a_u > 0$ in ϕ^* if the literal u has a nonzero coefficient in some posiform $\phi \in \mathcal{P}_2^*(f)$.*

The linear part $L(\phi^*)$ of such a posiform is called a *master roof* of f.

For a given quadratic pseudo-Boolean function $f \in \mathcal{F}_2$ the convex cone $\mathcal{P}_2^*(f)$ maybe non-trivial, and therefore f may have many roofs. Since the sets of variables whose values in the optimum can be determined by using the strong persistency theorem vary with the particular roofs chosen, it is very important to note that the set of variables which can be fixed using a master roof is the union of all those sets of variables which can be fixed by any particular roof (see e.g. [77, 78] for applications of persistency in graph theory).

Even though the roof dual as well as a particular roof of a given quadratic pseudo-Boolean function can be computed in polynomial time by using one of the formulations in the previous subsections, the determination of a master roof may still not be obvious (see e.g. [25].) We shall show in the next subsection that in fact master roofs can be computed with the same effort as needed for finding the roof dual.

Example 5.2 *Let us consider the quadratic pseudo-Boolean function*

$$f(x_1, x_2, x_3) = 6 - x_1 - 4x_2 - x_3 + 3x_1x_2 + x_2x_3.$$

It can be shown that the roof dual value of this function is $C_2(f) = 2$, and two examples for posiforms belonging to $\mathcal{P}_2^*(f)$ are

$$g_1 = 2 + 2x_1 + 3\overline{x}_1\overline{x}_2 + \overline{x}_2\overline{x}_3 \text{ and} \tag{40}$$

$$g_2 = 2 + 2\overline{x}_2 + 2x_1x_2 + \overline{x}_1\overline{x}_2 + \overline{x}_2\overline{x}_3. \tag{41}$$

Clearly, the convex combination of g_1 and g_2

$$g = \frac{1}{2}g_1 + \frac{1}{2}g_2 = 2 + x_1 + \overline{x}_2 + x_1x_2 + 2\overline{x}_1\overline{x}_2 + \overline{x}_2\overline{x}_3$$

is also a member of $\mathcal{P}_2^*(f)$. As it will be seen in the next section, in fact $2 + x_1 + \overline{x}_2$ is a master roof of f.

The Strong Persistency Theorem implies then that $x_1 = \overline{x}_2 = 0$, i.e. $x_1 = 0$ and $x_2 = 1$ in all minima of f. Hence for this example we have

$$\min_{\mathbf{x} \in \mathbb{B}^3} f(\mathbf{x}) = \min_{x_3 \in \mathbb{B}} 2 = 2.$$

The above fortunate example illustrates that the variable fixation implied by persistency can sometimes simplify the minimization problem to a trivial one. Though this cannot be expected in general, the equality $C_2(f) = \min_{\mathbf{x}\in\mathbb{B}^n} f(\mathbf{x})$ can always be tested in polynomial time. To see this, let us compute first a posiform $\phi \in \mathcal{P}_2^*(f)$, then fix those variables which are implied by persistency, remove the constant term, and denote by ϕ' the quadratic posiform obtained in this way. According to the above discussion, the minimization of f is equivalent with the minimization of ϕ', thus $C_2(f) = \min_{\mathbf{x}\in\mathbb{B}^n} f(\mathbf{x})$ if and only if $\min_{\mathbf{x}\in\mathbb{B}^n} \phi'(\mathbf{x}) = 0$. Since ϕ' has only quadratic terms with positive coefficients, the latter is obviously equivalent with a 2-SAT problem, which is solvable in polynomial (in fact in linear) time.

A somewhat different but similar conclusion can also be drawn from the integrality of some of the components of an optimal solution to the linear programming formulation (35):

Theorem 9 (Weak Persistency, [79, 129, 139]) *Let $f \in \mathcal{F}_2$, and let $\tilde{\mathbf{x}}$ be an optimal solution of the linear program (35), for which $\tilde{x}_j = 1$ for $j \in O$, and $\tilde{x}_j = 0$ for $j \in Z$ (where O and Z are disjoint subsets of the indices). Then, for any minimizing vector $\mathbf{x}^* \in \operatorname{Argmin}(f)$ switching the components to $x_j^* = 1$ for $j \in O$ and $x_j^* = 0$ for $j \in Z$ will also yield a minimum of f.* □

While strong persistency's main advantage is that it allows a fixation of the values of a subset of variables, which holds in every optimum of the problem, the advantage of weak persistency is the fixation of a (usually) larger set of variables, valid in at least one of the optima of the problem.

5.1.5 Network flow model

According to the above discussion, the computation of the roof dual is a polynomial problem, since it can be carried out by solving a linear program. Because of its usefulness, several even more efficient combinatorial procedures were developed for its solution, see e.g. [30, 36, 38]. We shall recall here the network flow computation based method of [36], since this is perhaps the most efficient procedure to compute the roof dual, providing at the same time a master roof; moreover this method can also be used to obtain simple proofs for some of the theorems cited in the previous subsection.

Let us assume that the quadratic pseudo-Boolean function $f \in \mathcal{F}_2$ is given as a posiform $\phi \in \mathcal{P}_2(f)$ of the form (30). Let us associate to such a quadratic posiform a capacitated directed network $G_\phi = (N, A)$, where the node set is defined as $N = \mathbf{L} \cup \{x_0, \overline{x}_0\}$, where x_0 and $\overline{x}_0$ are two additional symbols representing the constants $x_0 = 1$ and $\overline{x}_0 = 0$, respectively. Let us associate two arcs to every quadratic term of ϕ, namely let us associate to term $c_{uv}uv$ the arcs $(\overrightarrow{u, \overline{v}})$ and $(\overrightarrow{v, \overline{u}})$, and let the capacity of both arcs be $\frac{1}{2}c_{uv}$. Similarly, let us homogenize the linear terms in ϕ by writing $c_u u = c_u u x_0$, and thus associating to this term the arcs $(\overrightarrow{u, \overline{x}_0})$ and $(\overrightarrow{x_0, \overline{u}})$, both with capacities $\frac{1}{2}c_u$. Let us note that the constant term of ϕ was disregarded in this construction.

Conversely, given a directed network $G = (N, A)$ with $N = \mathbf{L} \cup \{x_0, \overline{x}_0\}$, and with nonnegative capacities c_{uv} assigned to the arcs $(\overrightarrow{u,v}) \in A$, we can associate to it a quadratic posiform

$$\phi_G \stackrel{\text{def}}{=} \sum_{(\overrightarrow{u,v}) \in A} c_{uv} u\overline{v}.$$

Let us remark that in the above definition, all those terms which correspond to arcs entering x_0, or leaving $\overline{x}_0$ must vanish, since $u\overline{x}_0 = \overline{x}_0 v = 0$, due to the assumption $\overline{x}_0 = 0$.

It is easy to see that the above definitions imply

Proposition 12 *There is a one-to-one correspondence between quadratic posiforms $\phi \in \mathcal{P}_2$ for which $C(\phi) = 0$ and capacitated directed networks $G = (N, A)$ with node set $N = \mathbf{L} \cup \{x_0, \overline{x}_0\}$. Furthermore, the involution*

$$G_{\phi_G} = G \text{ and } \phi_{G_\phi} = \phi$$

holds for such corresponding pairs. □

Let us recall next that a feasible flow in the capacitated network $G = (N, A)$ with source x_0 and sink $\overline{x}_0$ is a mapping $\varphi : A \mapsto \mathbb{R}_+$, satisfying the constraints

$$\begin{array}{rcll} \varphi(u,v) & \leq & c_{uv} & \text{for all arcs } (\overrightarrow{u,v}) \in A, \text{ and} \\ \sum_{(\overrightarrow{u,v}) \in A} \varphi(u,v) & = & \sum_{(\overrightarrow{v,w}) \in A} \varphi(v,w) & \text{for all nodes } v \in \mathbf{L}. \end{array}$$

Given a capacitated network $G = (N, A)$ and a feasible flow φ in it, let us define the *residual network* $G[\varphi] = (N, A^\varphi)$ by setting the residual capacities

$$c^\varphi_{uv} = \begin{cases} c_{uv} - \varphi(u,v) & \text{for } (\overrightarrow{u,v}) \in A \\ \varphi(u,v) & \text{for } (\overrightarrow{v,u}) \in A. \end{cases}$$

Let us observe that, due to the special structure of this network (all arcs come in pairs $(\overrightarrow{u,v})$ and $(\overrightarrow{\overline{v},\overline{u}})$), a feasible flow can always be assumed to be *symmetric*, i.e. such that $\varphi(u,v) = \varphi(\overline{v},\overline{u})$ holds for every pair u, v.

An *augmenting path of capacity* ϵ in the residual network is a sequence of nodes v_0,...,v_k such that $v_0 = x_0$, $v_k = \overline{x}_0$, and $c^\varphi_{v_j,v_{j+1}} \geq \epsilon$ for $j = 0, 1, ..., k-1$. Let us note that for symmetric feasible flows φ in $G = (N, A)$ the path $\overline{v_k}$, $\overline{v}_{k-1}$, ..., $\overline{v}_0$ is also augmenting, and these two augmenting paths can actually share arcs! It is well known that all feasible flows can be obtained from the constant zero flow by iteratively increasing the flow along augmenting paths.

If u_1, u_2, ..., u_k are literals from $\mathbf{L}$, let us call an expression of the form

$$u_1 + \overline{u}_1 u_2 + \overline{u}_2 u_3 + \cdots + \overline{u}_{k-1} u_k + \overline{u}_k \tag{43}$$

an *alternating sum*. Let us say that a quadratic posiform ϕ *contains the alternating sum* (43) with weight ω, if we have $a_{u_1} \geq \omega$, $a_{\overline{u}_j u_{j+1}} \geq \omega$ for $j = 1, ..., k-1$, and $a_{\overline{u}_k} \geq \omega$ for all the corresponding coefficients of ϕ.

Proposition 13 *The following identity holds for alternating sums:*

$$u_1 + \overline{u}_1 u_2 + \overline{u}_2 u_3 + \cdots + \overline{u}_{k-1} u_k + \overline{u}_k = 1 + u_1 \overline{u}_2 + \cdots + u_{k-1} \overline{u}_k. \tag{44}$$

Proof. Immediate by elementary calculation. □

If a quadratic posiform ϕ contains the alternating sum (43) with weight ω, then it can be changed into an equivalent posiform with a larger constant term, as follows: first we re-group the terms of ϕ and write it as

$$\phi = \omega \left[u_1 + \overline{u}_1 u_2 + \overline{u}_2 u_3 + \cdots + \overline{u}_{k-1} u_k + \overline{u}_k\right] + \phi',$$

where ϕ' is also a quadratic posiform. Then we apply the identity (44) to get

$$\phi = \omega + \omega \left[u_1 + \overline{u}_1 u_2 + \overline{u}_2 u_3 + \cdots + \overline{u}_{k-1} u_k + \overline{u}_k\right] + \phi'.$$

Observing now that there is a one-to-one correspondence between alternating sequences contained in a posiform, and augmenting paths in the corresponding network, we have

Proposition 14 *Let us consider a posiform $\phi \in \mathcal{P}_2$, and a feasible flow φ in the corresponding capacitated network $G = G_\phi$. Then, x_0, u_1, ..., u_k, $\overline{x}_0$ is an augmenting path of capacity $\epsilon > 0$ in $G[\varphi]$ if and only if $u_1 + \overline{u}_1 u_2 + \cdots + \overline{u}_{k-1} u_k + \overline{u}_k$ is an alternating sum of weight ϵ in the corresponding posiform $\phi_{G[\varphi]}$.*

Proof. Follows easily form the above observations (see [36]). □

Two further consequences of the above are (see [36]):

Proposition 15 *Let $\phi \in \mathcal{P}_2(f)$ for a quadratic pseudo-Boolean function $f \in \mathcal{F}_2$, and let φ be a feasible flow in the corresponding network G_ϕ. Let us denote by $\nu(\varphi)$ the value of the flow (e.g. the total flow leaving the source), and let $\psi = \phi_{G_\phi[\varphi]}$ denote the posiform corresponding to the residual network. Then we have $C(\phi) + \nu(\varphi) + \psi \in \mathcal{P}_2(f)$.* □

Proposition 16 *If $\phi, \psi \in \mathcal{P}_2(f)$ for a quadratic pseudo-Boolean function f with $C(\phi) < C(\psi)$, then there is an augmenting path in the network G_ϕ.* □

Putting all these facts together, we can see that the quadratic posiforms representing a given quadratic pseudo-Boolean function f, and having the largest constant term, are those which correspond to the residual network of a maximum flow in the network associated to any quadratic posiform of f.

Theorem 10 ([36]) *Given a quadratic pseudo-Boolean function f, and a posiform representation $\phi \in \mathcal{P}_2(f)$ of it, let us denote by ν^* the maximum flow value in the network G_ϕ. Then*

$$C_2(f) = C(\phi) + \nu^*.$$

□

Furthermore, from an easy analysis of the residual network corresponding to a maximum flow, we get the following

Theorem 11 *Let $\phi \in \mathcal{P}_2(f)$ for a quadratic pseudo-Boolean function f, let φ^* denote a maximum flow in $G = G_\phi$, and let $S \subseteq \mathbf{L}$ denote the set of nodes of G which are reachable from x_0 via a path with positive residual capacities. Then, $u(\mathbf{x}^*) = 1$ for all $u \in S$ and for all vectors $\mathbf{x}^* \in \mathrm{Argmin}(f)$ minimizing f.*

Proof. Let us denote by ψ the posiform corresponding to the residual network $G[\varphi^*]$. Then $C(\psi) = 0$ by definition, and every term of ψ with a positive coefficient involving a literal u or $\overline{u}$ for some $u \in S$ involves also a negated literal $\overline{v}$ for some $v \in S$, since there cannot be an arc in $G[\varphi^*]$ with positive residual capacity leaving the set of nodes S. Thus, the partial assignment $\mathbf{y}$ defined by $u(\mathbf{y}) = 1$ for all $u \in S$ is a contractor for ψ, and since $C_2(f) + \psi \in \mathcal{P}_2(f)$, weak persistency holds for $\mathbf{y}$ at the minima of f by Corollary 2. Adding that all nodes in S can be reached via arcs with positive residual capacity in $G[\varphi^*]$, it follows that $u(\mathbf{x}^*) = 1$ must hold for all $\mathbf{x}^* \in \mathrm{Argmin}(f)$, i.e. that strong persistency must also hold for $\mathbf{y}$. □

Example 5.3 *Let us illustrate the above discussion on the following example:*

$$f(x_1, x_2, x_3, x_4, x_5) = 10 - 4x_1 - 4x_3 - 2x_4 + 4x_1x_2 - 2x_2x_3 + 4x_3x_4 - 2x_4x_5.$$

Substituting the identity $x = 1 - \overline{x}$ for the first variables in each of the quadratic terms having a negative coefficient, as well as in the negative linear terms, we find that

$$\phi = -4 + 4\overline{x}_1 + 6\overline{x}_3 + 2\overline{x}_4 + 2\overline{x}_5 + 4x_1x_2 + 2\overline{x}_2x_3 + 4x_3x_4 + 2\overline{x}_4x_5$$

is a quadratic posiform representing f, i.e. $\phi \in \mathcal{P}_2(f)$. The corresponding network G_ϕ is shown in Figure 7.

Checking in Figure 7, we can see that in the network G_ϕ, unit flows can be pushed sequentially through each of the following augmenting paths:

$$\begin{array}{rl} x_0 \to x_1 \to \overline{x}_2 \to \overline{x}_3 \to \overline{x}_0 & \textit{and its twin } \; x_0 \to x_3 \to x_2 \to \overline{x}_1 \to \overline{x}_0, \\ x_0 \to x_3 \to \overline{x}_4 \to \overline{x}_0 & \textit{and its twin } \; x_0 \to x_4 \to \overline{x}_3 \to \overline{x}_0, \\ x_0 \to x_3 \to \overline{x}_4 \to \overline{x}_5 \to \overline{x}_0 & \textit{and its twin } \; x_0 \to x_5 \to x_4 \to \overline{x}_3 \to \overline{x}_0. \end{array}$$

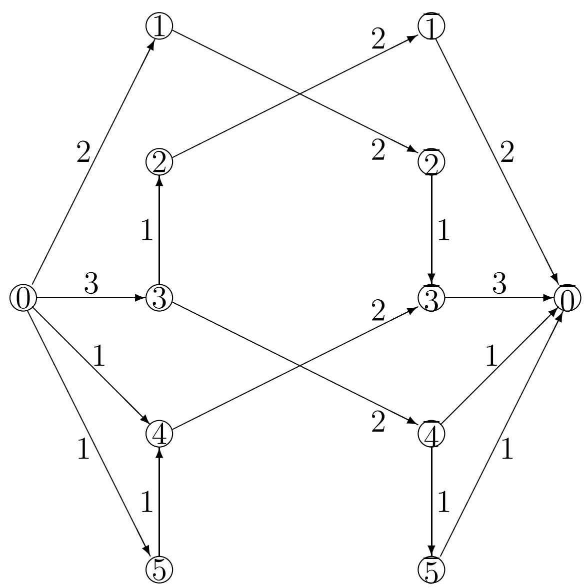


Figure 7: The capacitated network G_ϕ corresponding to the posiform ϕ of example 5.3.

These augmenting paths correspond, respectively, to the following alternating sequences:

$$\begin{array}{rcl} \overline{x}_1 + x_1x_2 + \overline{x}_2x_3 + \overline{x}_3 & \text{and} & \overline{x}_3 + x_3\overline{x}_2 + x_2x_1 + \overline{x}_1, \\ \overline{x}_3 + x_3x_4 + \overline{x}_4 & \text{and} & \overline{x}_4 + x_4x_3 + \overline{x}_3, \\ \overline{x}_3 + x_3x_4 + \overline{x}_4x_5 + \overline{x}_5 & \text{and} & \overline{x}_5 + x_5\overline{x}_4 + x_4x_3 + \overline{x}_3. \end{array}$$

Since there is no further augmenting path in the residual network shown in Figure 8, we have arrived to a maximum flow of value $\nu = 6$. The final network in Figure 8 corresponds indeed to the quadratic posiform

$$\psi = 2\overline{x}_1 + 2x_1x_2 + 2\overline{x}_1\overline{x}_2 + 2x_2\overline{x}_3 + 4\overline{x}_3\overline{x}_4 + 2x_4\overline{x}_5$$

for which we have $\phi = C(\phi) + \nu + \psi = 2 + \psi$. Hence $C_2(f) = 2$ and $2 + \psi \in \mathcal{P}_2^*(f)$ follow.

From the final network, we can see that nodes x_1 and $\overline{x}_2$ can be reached from the source via arcs with positive residual capacity, and hence $x_1 = 1$ and $x_2 = 0$ must hold in all minima of f. Substituting this into the last posiform ψ we can conclude that the minimization of f is equivalent with the minimization of

$$\psi^* = 4\overline{x}_3\overline{x}_4 + 2x_4\overline{x}_5.$$

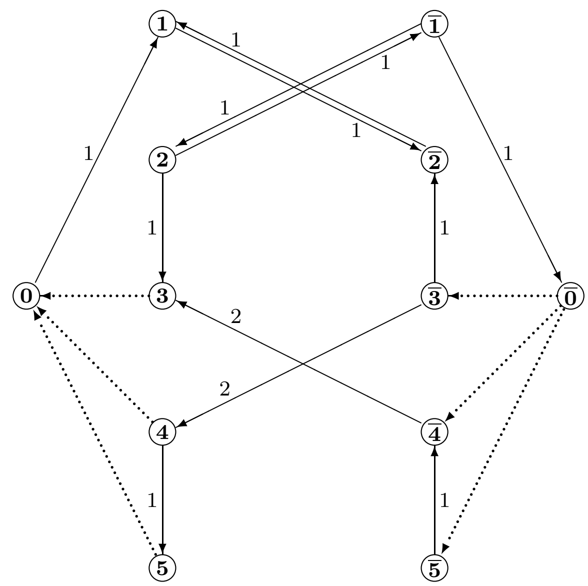


Figure 8: Residual network of Example 5.3. Only arcs with positive residual capacity are indicated. Dotted arcs, i.e. those which enter the source or leave the sink, have positive capacity but play no role in our analysis.

5.2 Hierarchy of Bounds

In this section we study the family of nonnegative pseudo-Boolean functions, along with their generators, and show that the roof dual bound can be extended to a complete hierarchy of bounds.

5.2.1 Cones of positive quadratic pseudo-Boolean functions

We shall view quadratic pseudo-Boolean functions in $|\mathbf{V}| = n$ variables as vectors of the $1 + n + \binom{n}{2}$ coefficients of their unique polynomial form, i.e., as vectors in $\mathbb{R}^{1+n+\binom{n}{2}}$.

For a subset $S \subseteq \mathbf{V}$ of the variables, let us denote by $\mathcal{F}_S$ the family of quadratic pseudo-Boolean functions depending only on variables from S, and let $\mathcal{F}_S^+ \subseteq \mathcal{F}_S$ denote the subfamily of nonnegative ones among them. Clearly, $\mathcal{F}_S$ is a subspace of dimension $1 + |S| + \binom{|S|}{2}$ of $\mathbb{R}^{1+n+\binom{n}{2}}$, and $\mathcal{F}_S^+$ is a convex cone in this subspace. It is also easy to see that $\mathcal{F}_S^+$ is finitely generated, since it is a polyhedral cone, described by the $2^{|S|}$ inequalities requiring the nonnegativity at each of the $2^{|S|}$ binary substitutions to these variables. Let us define next

$$\mathcal{Q}_k \stackrel{\text{def}}{=} \text{cone}\{\mathcal{F}_S^+ \mid S \subseteq \mathbf{V},\ |S| \leq k\}$$

for $2 \leq k \leq n$, as the convex cone in $\mathbb{R}^{1+n+\binom{n}{2}}$, generated by the unions of the above cones corresponding to at most k of the variables, chosen in all possible ways. It follows from these definitions that

$$\mathcal{Q}_2 \subseteq \mathcal{Q}_3 \subseteq \cdots \subseteq \mathcal{Q}_n, \tag{45}$$

and that all these cones are finitely generated. Let us denote by $\mathcal{B}(\mathcal{Q}_k)$ such a finite set of generators of the cone $\mathcal{Q}_k$. Finally let us note that $\mathcal{Q}_n = \mathcal{F}_{\mathbf{V}}^+$ is the family of all nonnegative quadratic pseudo-Boolean functions.

The characterization of the extremal elements of the above introduced cones is an interesting problem in itself, and as we shall see in the subsequent sections, it has its own importantce for several algorithmic problems. The full characterization however seems to be too difficult, and at this time we only have partial solution to it.

Let us consider the following special family of functions, which we recall from [28]:

$$b_{U,\alpha} \stackrel{\text{def}}{=} \binom{\sum_{u \in U} - \alpha}{2} \tag{46}$$

where $U \subseteq \mathbf{L}$ is a subset of the literals, not containing complemented pairs, and where $\alpha \in \mathbb{Z}$ is an integer. (Here we use $\binom{x}{2} = x(x-1)/2$ for all integers $x \in \mathbb{Z}$.) Clearly, the above defines a pseudo-Boolean function, and by using the identity $u^2 = u$ for the literals $u \in \mathbf{L}$, we can compute the quadratic polynomial representing these functions.

Example 5.4 *If* $U = \{x, \overline{y}\}$ *and* $\alpha = 1$*, we get*

$$\begin{aligned} b_{\{x,\overline{y}\},1} &= \tfrac{(x+\overline{y}-1)(x+\overline{y}-2)}{2} \\ &= \tfrac{x^2+\overline{y}^2+2x\overline{y}-3x-3\overline{y}+2}{2} \\ &= x\overline{y} - x - \overline{y} + 1 \\ &= \overline{x}y. \end{aligned}$$

Proposition 17 ([28]) *If* $U \subseteq \mathbf{L}$ *is a subset of the literals containing no complemented pairs, and* α *is an integer such that* $1 \leq \alpha \leq |U| - 2$ *for* $|U| \geq 3$*, and* $\alpha = 1$ *for* $|U| = 2$*, then* $b_{U,\alpha} \in \mathcal{B}(\mathcal{Q}_k)$ *for* $k \geq |U|$*.* □

The above statement implies that in the nested sequence of convex cones (45) there are no equalities:

$$\mathcal{Q}_2 \subset \mathcal{Q}_3 \subset \cdots \subset \mathcal{Q}_n. \tag{47}$$

Example 5.5 *For instance, if* $U = \{u, v, w\} \subseteq \mathbf{L}$ *is a subset of 3 distinct literals, and contains no complemented pair, then* $b_{U,1} = uvw + \overline{u}\overline{v}\overline{w} = uv + uw + vw - u - v - w + 1 \in \mathcal{Q}_3 \setminus \mathcal{Q}_2$*.*

For $k \leq 3$ the following characterization of the extremal elements of $\mathcal{Q}_k$ is known:

Proposition 18 ([29]) *Let*

$$\begin{aligned} \mathcal{B}_2 &= \{uv \mid u, v \in \mathbf{L}, u \neq v, u \neq \overline{v}\} \\ &= \{b_{U,1} \mid U \subseteq \mathbf{L} \text{ containing no complemented literals, } |U| = 2\}, \text{ and} \\ \mathcal{B}_3 &= \{uvw + \overline{u}\overline{v}\overline{w} \mid u, v, w \in \mathbf{L}, u \notin \{v, \overline{v}, w, \overline{w}\}, v \notin \{w, \overline{w}\}\} \\ &= \{b_{U,1} \mid U \subseteq \mathbf{L} \text{ containing no complemented literals, } |U| = 3\}. \end{aligned}$$

Then, we have $\mathcal{B}(\mathcal{Q}_2) = \mathcal{B}_2$ *and* $\mathcal{B}(\mathcal{Q}_3) = \mathcal{B}_2 \cup \mathcal{B}_3$*.* □

Not every generator of the cones $\mathcal{Q}_k$ is a function of the form (46). In [33] several families of extremal elements of $\mathcal{Q}_k$, $k \geq 10$ are exhibited, which are not in the form of this type.

5.2.2 Complementation, Majorization, Linearization

Following [28] we shall recall in this section a generalization of the three approaches yielding the roof dual value.

The first approach, the so called complementation, can be extended in a natural way as follows. Let us observe that since every quadratic pseudo-Boolean function f can be represented by a quadratic posiform ϕ (possibly with a negative constant term), there exists a constant C such that $f - C \in \mathcal{Q}_2 \subseteq \mathcal{Q}_k$ for $k \geq 2$. Thus, let us define

$$C_k(f) = \max\{C \in \mathbb{R} \mid f - C \in \mathcal{Q}_k\} \tag{48}$$

for $k = 2, 3, ..., n$. Since $\mathcal{Q}_k$ is a closed convex cone in $\mathbb{R}^{1+n+\binom{n}{2}}$ for $k = 2, 3, ..., n$, and since f takes only finitely many different values, in the above definition the maximum exists.

It is clear from the definition of the cones $\mathcal{Q}_k$ that

$$C_2(f) \leq C_3(f) \leq \cdots \leq C_n(f) = \min_{\mathbf{x}\in\mathbb{B}^n} f(\mathbf{x}),$$

implying that the values $C_k(f)$ form a sequence of increasingly better lower bounds of f.

Using a generating set $\mathcal{B}(\mathcal{Q}_k)$ of the cone $\mathcal{Q}_k$, we can also express the lower bound $C_k(f)$ as the optimum of a linear programming problem

$$\begin{array}{rl} C_k(f) & = \max C \\ \text{s.t.} \quad f - C & = \sum_{b\in\mathcal{B}(\mathcal{Q}_k)} \alpha_b b, \\ \alpha_b & \geq 0 \text{ for all } b \in \mathcal{B}(\mathcal{Q}_k), \end{array} \tag{49}$$

where the equations correspond to the $1 + n + \binom{n}{2}$ coefficients of f.

Using Proposition 18 and the above linear programming problem for $k = 2$, it can be verified that $C_2(f)$ is indeed the roof dual value of f, as defined in the previous sections. The linear programming problem corresponding to $k = 3$ was analyzed in [29], and $C_3(f)$ was introduced there as the *cubic dual* of f. It was also shown that $C_3(f) \geq C_2(f)$ with equality holding if and only if $C_2(f) = C_3(f) = \min_{\mathbf{x}\in\mathbb{B}^n} f(\mathbf{x})$, i.e. if and only if these bounds are tight.

The main idea of the second approach, *minorization*, is to consider a family of linear minorants of the quadratic pseudo-Boolean function f, and to find a “best” one among these linear functions. This idea was considered in [12, 79], and a generalization, which we recall here appears in [28].

For a *purely* quadratic function h of k variables (i.e., for which $h(0, 0, ..., 0) = h(1, 0, ..., 0) = \cdots = h(0, ..., 0, 1) = 0$) let

$$\mathcal{M}(h) \stackrel{\text{def}}{=} \{l \mid l \text{ linear}, \ l(\mathbf{x}) \leq h(\mathbf{x}) \text{ for all } \mathbf{x} \in \mathbb{B}^k\}$$

denote a family of linear minorants of h over the Boolean vectors.

For a quadratic pseudo-Boolean function f, and for a fixed integer k, let us consider a representation of f of the form

$$f = l + \sum_{j\in J} f_j, \tag{50}$$

where l is linear, and the f_j's are purely quadratic functions of at most k variables. Then, the linear functions of the form

$$p = l + \sum_{j\in J} l_j,$$

with $l_j \in \mathcal{M}(f_j)$ for $j \in J$, are linear minorants of f. Let $\mathcal{M}^k(f)$ denote the set of all linear minorants of f obtained in this way, varying (50) over all possible representations. Then, by definition,

$$\mathcal{M}(f) = \mathcal{M}^n(f) \supseteq \mathcal{M}^{n-1}(f) \supseteq \cdots \supseteq \mathcal{M}^3(f) \supseteq \mathcal{M}^2(f).$$

By defining

$$M_k(f) \stackrel{\text{def}}{=} \max_{p \in \mathcal{M}^k(f)} \min_{x \in \mathbb{B}^n} p(x), \tag{51}$$

we obtain another series of lower bounds of f

$$\min_{x \in \{0,1\}^n} f(x) = M_n(f) \geq M_{n-1}(f) \geq \cdots \geq M_3(f) \geq M_2(f).$$

The elements of $\mathcal{M}^2(f)$ were called *paved upper planes* in [79]. It was also shown there that the bound, obtained by taking the minimum in (51) for $k = 2$ over certain "minimal" elements $p(x)$ of $\mathcal{M}^2(f)$, rather than over all of $\mathcal{M}^2(f)$, is always equal to $C_2(f)$. As a consequence, it follows that $M_2(f) \geq C_2(f)$. Later, in [98] it has been proved that $C_2(f) = M_2(f)$ (see also [2, 42]).

Using the notation $b = l_b + q_b$ for $b \in \mathcal{B}(\mathcal{Q}_k)$, where l_b denotes the linear part of b, and q_b is purely quadratic, we can obtain a computationally simpler derivation of $M_k(f)$, given by

$$M_k = \max_{\substack{l \text{ is linear}, \quad \alpha_b \geq 0 \\ f = l + \sum_{b \in \mathcal{B}(\mathcal{Q}_k)} \alpha_b q_b}} \min_{x \in \mathbb{B}^n} \left[l + \sum_{b \in \mathcal{B}(\mathcal{Q}_k)} \alpha_b l_b \right]. \tag{52}$$

The third approach, *linearization*, is a standard method to represent nonlinear expressions in terms of linear functions and inequalities.

To a quadratic pseudo-Boolean function over $\mathbb{B}^n$

$$f(\mathbf{x}) \stackrel{\text{def}}{=} q_0 + \sum_{j=1}^{n} q_j x_j + \sum_{1 \leq i < j \leq n} q_{ij} x_i x_j$$

we shall associate a linear function over $\mathbb{R}^{n+\binom{n}{2}}$, given by:

$$L_f(\mathbf{x}, \mathbf{y}) \stackrel{\text{def}}{=} q_0 + \sum_{j=1}^{n} q_j x_j + \sum_{1 \leq i < j \leq n} q_{ij} y_{ij},$$

where $(\mathbf{x}, \mathbf{y})$ denotes the vector $(x_1, ..., x_n, y_{12}, ..., y_{n-1,n}) \in \mathbb{R}^{n+\binom{n}{2}}$. This association establishes in fact a one-to-one correspondence between quadratic pseudo-Boolean functions in n variables and linear functions in $n + \binom{n}{2}$ variables.

Let us define then a polyhedron for $2 \leq k \leq n$ by

$$\mathbf{S}^{[k]} \stackrel{\text{def}}{=} \{(\mathbf{x}, \mathbf{y}) \mid L_b(\mathbf{x}, \mathbf{y}) \geq 0 \text{ for all } b \in \mathcal{B}(\mathcal{Q}_k)\},$$

and let

$$L_k(f) \stackrel{\text{def}}{=} \min L_f(\mathbf{x}, \mathbf{y}) \text{ s.t. } (\mathbf{x}, \mathbf{y}) \in \mathbf{S}^{[k]}. \tag{53}$$

It is easy to see by the definition of the cones $\mathcal{Q}_k$, $k = 2, ..., n$ that

$$\mathbf{S}^{[2]} \supseteq \mathbf{S}^{[3]} \supseteq \cdots \supseteq \mathbf{S}^{[n]}.$$

It follows by Proposition 18 that $\mathbf{S}^{[2]}$ is the same polyhedron appearing (with slight variations) in many publications (see e.g., [11, 31, 59, 60, 67, 68, 95, 122, 146]).

It has been shown in [135] that all fractional vertices of $\mathbf{S}^{[2]}$ are cut off by the so called *triangle inequalities*, i.e. by the inequalities $L_b(\mathbf{x}, \mathbf{y}) \geq 0$ for $b \in \mathcal{B}(\mathcal{Q}_3)$. A stronger statement was proved in [29]:

Proposition 19 ([29]) *The polyhedron* $\mathbf{S}^{[3]}$ *is the first Chvátal closure of* $\mathbf{S}^{[2]}$. □

Using Proposition 18, we can easily see that the roof dual of a quadratic pseudo-Boolean function f is the common value of the three bounds: $C_2(f) = M_2(f) = L_2(f)$. The main result of [28] generalizes this property:

Theorem 12 ([28]) *Given a quadratic pseudo-Boolean function f in n variables, the equalities*

$$C_k(f) = M_k(f) = L_k(f)$$

hold for all $k = 2, 3, ..., n$, providing increasingly tighter lower bounds on f, with

$$\min_{\mathbf{x} \in \mathbb{B}^n} f(\mathbf{x}) = C_n(f) = M_n(f) = L_n(f).$$

□

Definition (53) provides also a simple linear programming formulation to compute these bounds, at least for those cases when $\mathcal{B}(\mathcal{Q}_k)$ is known. For instance, we have

$$\mathbf{S}^{[2]} = \left\{ (\mathbf{x}, \mathbf{y}) \;\middle|\; \begin{array}{rrrl} & & -y_{ij} & \leq 0 \\ -x_i & & +y_{ij} & \leq 0 \\ & -x_j & +y_{ij} & \leq 0 \\ x_i & +x_j & -y_{ij} & \leq 1 \\ \multicolumn{4}{c}{\text{for } 1 \leq i < j \leq n} \end{array} \right\}, \tag{54}$$

and

$$\mathbf{S}^{[3]} = \mathbf{S}^{[2]} \cap \left\{ (\mathbf{x}, \mathbf{y}) \;\middle|\; \begin{array}{rrrrrrl} x_i & +x_j & +x_k & -y_{ij} & -y_{ik} & -y_{jk} & \leq 1 \\ -x_i & & & +y_{ij} & +y_{ik} & -y_{jk} & \leq 0 \\ & -x_j & & +y_{ij} & -y_{ik} & +y_{jk} & \leq 0 \\ & & -x_k & -y_{ij} & +y_{ik} & +y_{jk} & \leq 0 \\ \multicolumn{7}{c}{\text{for } 1 \leq i < j < k \leq n} \end{array} \right\}. \tag{55}$$

The sharpness of the roof dual bound, i.e. the validity of $C_2(f) = \min_{\mathbf{x}\in\mathbb{B}^n} f(\mathbf{x})$, can be recognized in $O(|f|)$ time (see [79]), while the sharpness of $C_3(f)$ was shown in [28] to be NP-complete. It was also shown there that $C_2(f) = C_3(f)$ only if these bounds are sharp, while $C_4(f) = C_3(f) < \min_{\mathbf{x}\in\mathbb{B}^n} f(\mathbf{x})$ is possible for some quadratic pseudo-Boolean functions.

5.3 Polyhedra

As for almost all combinatorial optimization problems, polyhedral formulations are quite natural, and are also frequent in the literature of quadratic pseudo-Boolean optimization. In fact, the technique of linearization has already associated some polyhedra with this problem. In this section we survey some of the relevant notions and results, selecting only the most closely related ones, from a very large and active area.

The most natural polytope associated to quadratic pseudo-Boolean optimization is the so called *Boolean quadric* polytope, introduced in [135], and defined by

$$\mathbf{Q} \stackrel{\text{def}}{=} \{(\mathbf{x},\mathbf{y}) \mid \mathbf{x} \in \mathbb{B}^n,\ y_{ij} = x_i x_j \text{ for all } 1 \le i < j \le n\}. \tag{56}$$

It was shown in [135] that $\mathbf{S}^{[2]} \supseteq \mathbf{Q}$, that the dimension of both polytopes is $n + \binom{n}{2}$; a number of facet defining inequalites were also introduced in the same paper. These polytopes were also shown to have several other interesting properties.

Let us call the 1-*skeleton* of a polytope P the family of its 1-faces, and let us say that a polytope P has the *Trubin property* with respect to another polytope $Q \subseteq P$ if the 1-skeleton of Q is a subfamily of the 1-skeleton of P (see [13, 14, 160]).

Proposition 20 ([135]) *The polytope $\mathbf{S}^{[2]}$ has the Trubin property with respect to $\mathbf{Q}$, furthermore, all vertices of $\mathbf{S}^{[2]}$ are half-integral.* $\square$

The relations

$$\mathbf{S}^{[2]} \supseteq \mathbf{S}^{[3]} \supseteq \cdots \supseteq \mathbf{S}^{[n]} = \mathbf{Q}$$

were observed in [28], together with the fact that the facets of $\mathbf{Q}$ correspond in a one-to-one way to the generators of the cone $\mathcal{Q}_n$ of nonnegative quadratic pseudo-Boolean functions.

Proposition 21 ([28]) *A quadratic pseudo-Boolean function f is nonnegative (i.e., $f \in \mathcal{Q}_n$), if and only if the inequality $L_f(\mathbf{x},\mathbf{y}) \ge 0$ is valid for $\mathbf{Q}$. Moreover, $f \in \mathcal{B}(\mathcal{Q}_n)$ if and only if $L_f(\mathbf{x},\mathbf{y}) \ge 0$ is facet defining for $\mathbf{Q}$.* $\square$

The fact that $\mathbf{S}^{[2]}$ has the Trubin property with respect to $\mathbf{Q}$ corresponds to the fact that $\mathcal{B}(\mathcal{Q}_2) \subset \mathcal{B}(\mathcal{Q}_n)$. We believe that a more general property must also hold, namely that

$$\mathcal{B}(\mathcal{Q}_2) \subset \mathcal{B}(\mathcal{Q}_3) \subset \cdots \subset \mathcal{B}(\mathcal{Q}_n),$$

implying that, $\mathbf{S}^{[i]}$ has the Trubin property with respect to $\mathbf{S}^{[j]}$ for all $2 \leq i < j \leq n$.

Another strongly related polytope is the so called *cut polytope*. Let $G = K_{n+1}$ denote the complete graph on vertices $v_0, v_1, ..., v_n$, and let $E = E(G)$ denote its edge set. Let us call an edges set $F \subseteq E$ a *cut set*, if the removal of F disconnects G, and let us define $\mathbf{C} \subseteq [0,1]^E$ as the convex hull of the characteristic vectors of cut sets.

It was noted in [49] that the mapping

$$\begin{aligned} z_{0i} &= x_i && \text{for } i = 1, ..., n, \\ z_{ij} &= x_i + x_j - 2y_{ij} && \text{for } 1 \leq i < j \leq n \end{aligned} \tag{57}$$

is an invertible linear mapping establishing a one-to-one correspondence between $\mathbf{Q}$ and $\mathbf{C}$. Combining this with the $f \longleftrightarrow L_f$ association, we get also a one-to-one correspondence between the points of the cut polytope, and the nonnegative quadratic-pseudo-Boolean functions. These mappings help to identify the corresponding results about the facial structure of the cut polytope (see e.g. [18, 19, 52, 53]), the Boolean quadric polytope ([135]) and the conical structure of nonnegative quadratic pseudo-Boolean functions ([28, 33]), and are also useful in extending those results.

For instance, the inequalities in (54) and (55) were shown to define facets for $\mathbf{Q}$ in [135], a fact which also follows easily from the characterization of $\mathcal{B}(\mathcal{Q}_k)$ for $k = 2, 3$ given in the previous section. It can be seen easily by (57) that the inequalities of (55) correspond to the so called *triangle inequalities* for the cut polytope.

Proposition 21 provides also a constructive approach to obtain valid inequalities for $\mathbf{Q}$ (and hence for $\mathbf{C}$), since nonnegative quadratic pseudo-Boolean functions can be constructed by various algebraic techniques.

It is easy to see that if $l : \mathbb{B}^n \longrightarrow \mathbb{Z}$ is an integer valued linear function, then $l(\mathbf{x})(l(\mathbf{x})-1)$ defines a nonnegative quadratic pseudo-Boolean function, and hence $L_{l(l-1)}(\mathbf{x}, \mathbf{y}) \geq 0$ is a valid inequality for $\mathbf{Q}$. It was shown in [33] that this family of valid inequalities includes all *hypermetric* inequalities of the cut polytope (see e.g. [52, 53]).

A further generalization was introduced in [33].

Proposition 22 ([33]) *Let $1 \leq p \leq n$, $k \geq 0$, let $l_j \geq 0$, $j = 1, .., p$ and $l_j < 0$ for $j = p+1, ..., n$ be integers, and let T be a spanning tree on vertices $\{1, ..., p\}$. Then the function*

$$f(\mathbf{x}) \stackrel{\text{def}}{=} \left(\sum_{i=1}^{n} l_i x_i\right) \left(\sum_{i=1}^{n} l_i x_i - (2k+1)\right) + k(k+1)\left(\sum_{j=1}^{p} x_j - \sum_{(i,j) \in E(T)} x_i x_j\right) \tag{58}$$

defines a nonnegative quadratic pseudo-Boolean function. $\square$

It can be seen that the facets of $\mathbf{C}$, defined through the above correspondence, include the *cycle inequalities* (for $k = 1$), and many other families of valid defining inequalities, giving at the same time rise to new families of facet defining inequalities:

Proposition 23 ([33]) *Let $r = k+1$, $n \geq 3k+7$, $p = [\frac{n+k+1}{k+2}]$ for $k \geq 1$, and let $l_1 = \cdots = l_p = r$, $l_{p+1} = \cdots = l_n = -1$. Let furthermore T be an arbitrary spanning tree on the vertices $\{1, 2, ..., p\}$. Then f, defined by (58), is an extremal element of $\mathcal{Q}_n$, i.e., $f \in \mathcal{B}(\mathcal{Q}_n)$.* $\square$

Let us also remark that the facet class defined in this way for $\mathbf{C}$ not only includes some facet defining cycle inequalities, but also has polynomial time separation (since a maximum weight spanning tree can be found in polynomial time.)

5.4 Heuristics

There is a large variety of techniques applied in the literature for solving problems that can be modelled by quadratic pseudo-Boolean functions. Many of the published algorithms are either branch-and-bound or branch-and-cut type, exploiting some of the results (bounds, facets, etc.) which were mentioned in the previous subsections, treating essentially those problems as integer programs.

In this brief survey of pseudo-Boolean optimization we shall restrict ourselves to recall a simple but very successful heuristic, which utilizes techniques highly specific to quadratic pseudo-Boolean functions. The DDT algorithm (Devour, Digest and Tidy-up) was introduced in [35] and was later applied successfully for delay-fault testing in VLSI design [39, 153], and to a number of other problems transformed to quadratic pseudo-Boolean optimization via a penalty function technique [66].

The basic idea in this heuristic is to represent the problem as the minimization of a quadratic posiform, and then sequentially restrict the Boolean cube to smaller and smaller subsets of it, on which the quadratic terms with the highest coefficients vanish. In this process the restriction is represented as the set of solutions to a quadratic Boolean equation, the consistency of which can be tested efficiently. In each iteration, logical consequences of the current Boolean equation are substituted back into the input formula, which is then simplified, before choosing the next term for elimination.

To make this procedure more precise, let us assume that we would like to minimize the quadratic posiform

$$\phi(\mathbf{x}) = \sum_{T \subset \mathbf{L}} a_T \prod_{u \in T} u, \tag{59}$$

where $a_T \geq 0$ for all $T \subseteq \mathbf{L}$, and $a_T > 0$ if and only if $|T| \leq 2$.

A straightforward one step implementation of the above idea would be to choose a smallest threshold $\theta \geq 0$ for which the quadratic Boolean equation $\bigvee_{T:a_T > \theta} T(\mathbf{x}) = 0$ is still consistent, and then output a solution of this equation. A refined sequential version, the so called *DDT algorithm*, is described below (see [35]).

DDT (Devour–Digest–Tidy-up)

Initialization: Input ϕ given by (59), and set $\mathcal{S} = \emptyset$, and $\tilde{\phi} = \phi$.

Devour: Find a term T of $\tilde{\phi}$ with the largest coefficient, and let $\mathcal{S} = \mathcal{S} \cup \{T\}$.

Digest: Draw all logical conclusions $\mathcal{C}$ of the Boolean equation

$$\bigvee_{T \in \mathcal{S}} T(\mathbf{x}) = 0 \qquad (\heartsuit)$$

Tidy-up: Substitute into $\tilde{\phi}$ the consequences $\mathcal{C}$, drawn in the previous step, and simplify the resulting posiform $\tilde{\phi}$. If $\tilde{\phi} \neq \mathit{const}$ then return to **Devour**.

Output: Output a solution $\mathbf{x}$ of the quadratic Boolean equation ($\heartsuit$), and STOP.

Let us add that the logical conclusions drawn in step **Digest** are of the form $u = 0$ or $uv = 0$ for some literals u and v, and that all such conclusions can be derived in polynomial time from the quadratic Boolean equation ($\heartsuit$). It is important to note that the completeness of this step implies that all terms which have positive coefficients after the substitution of the derived logical consequences in step **Tidy-up** are consistent with the current quadratic equation ($\heartsuit$), i.e. any one of those could be added to $\mathcal{S}$ without implying the inconsistency of the quadratic Boolean equation ($\heartsuit$).

Instead of giving a formal proof (simple, but technical, see [35]) for the correctness and finiteness of the above procedure, let us illustrate it on a small example.

Example 5.6 *Let us consider the minimization of the following quadratic posiform*

$$\begin{aligned}\phi(x_1, x_2, ..., x_5) \quad = \quad & 17\overline{x}_1 + 12x_1\overline{x}_2 + 10x_2\overline{x}_4 + 8x_1x_4 + 8x_4\overline{x}_5 \\ & + 7x_5 + 6x_3\overline{x}_4 + 5\overline{x}_3 + 4x_2x_3 + 4x_4 + \overline{x}_2,\end{aligned}$$

in which terms are listed in the decreasing order of their coefficients. The DDT algorithm will take 5 steps before termination, and will output $\mathbf{x} = (1, 1, 0, 0, 0)$*, which in this case is indeed a minimum of* ϕ*:*

1: $T = \overline{x}_1$, $\mathcal{S} = \{\overline{x}_1\}$, $\mathcal{C} = \{x_1 = 1\}$ *and*

$$\begin{aligned}\tilde{\phi} = \quad & +13\overline{x}_2 + 12x_4 + 10x_2\overline{x}_4 + 8x_4\overline{x}_5 + 7x_5 \\ & + 6x_3\overline{x}_4 + 5\overline{x}_3 + 4x_2x_3.\end{aligned}$$

2: $T = \overline{x}_2$, $\mathcal{S} = \{\overline{x}_1, \overline{x}_2\}$, $\mathcal{C} = \{x_1 = x_2 = 1\}$ *and*

$$\tilde{\phi} = 14 + 8x_4\overline{x}_5 + 7x_5 + 6x_3\overline{x}_4 + 2x_4 + \overline{x}_3.$$

3: $T = x_4\overline{x}_5$, $\mathcal{S} = \{\overline{x}_1, \overline{x}_2, x_4\overline{x}_5\}$, $\mathcal{C} = \{x_1 = x_2 = 1, x_4\overline{x}_5 = 0\}$ *and*

$$\tilde{\phi} = 14 + 7x_5 + 6x_3\overline{x}_4 + 2x_4 + \overline{x}_3.$$

4: $T = x_5$, $\mathcal{S} = \{\overline{x}_1, \overline{x}_2, x_4\overline{x}_5, x_5\}$, $\mathcal{C} = \{x_1 = x_2 = 1, x_4 = x_5 = 0\}$ *and*

$$\tilde{\phi} = 15 + 5x_3.$$

5: $T = x_3$, $\mathcal{S} = \{\overline{x}_1, \overline{x}_2, x_4\overline{x}_5, x_5, x_3\}$, $\mathcal{C} = \{x_1 = x_2 = 1, x_3 = x_4 = x_5 = 0\}$

$$\tilde{\phi} = 15.$$

A slightly simpler version of the DDT algorithm, in terms of signed graph balancing was also presented in [35], and it was shown that both version can be implemented to run in $O(n|\phi|)$ time.

Let us add that the selection of the next term to be eliminated in the DDT algorithm is based on a very simple "greedy" utility measure, namely the size of the coefficient of that term. One could further modify the above algorithm by making a more careful selection, based on some look-ahead procedure, analyzing several steps in advance before choosing the term to be eliminated, or based on some probabilistic analysis of the expected benefit from eliminating a particular term, etc. Several such variants were recently examined in [66].

6 Special Classes

In this section we survey some special classes of pseudo-Boolean functions, limiting our attention to those for which specialized optimization algorithms have been developed.

6.1 Sub- and supermodular functions

Among the most widely studied classes of set functions submodular and supermodular functions play a well-known and special role. A set function $f : 2^V \longrightarrow \mathbb{R}$ is called *submodular* if

$$f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y) \tag{60}$$

holds for all subsets X, Y of a base set V of n elements. Functions, for which the reverse inequality holds for all subsets $X, Y \subseteq V$ are called *supermodular*, while those for which $f(X) + f(Y) = f(X \cup Y) + f(X \cap Y)$ for all subsets $X, Y \subseteq V$ are called *modular*. Clearly,

f is submodular iff $-f$ is supermodular, and vice versa; furthermore, functions which are both sub- and supermodular, are the modular ones.

It is well-known that submodular functions can be minimized in polynomial time (see e.g. [71]), and even strongly polynomial algorithms are available for this task (see e.g. [108, 152]). Of course, the same applies to the maximization of supermodular functions.

It is an interesting problem to recognize if a given polynomial expression or a given posiform defines a sub- or supermodular set function.

It is easy to see that a set function is modular if and only if its unique polynomial expression is linear. This is also an easy consequence of the more general characterization of supermodular functions.

Proposition 24 ([58]) *A pseudo-Boolean function $f : \mathbb{B}^n \longrightarrow \mathbb{R}$ is supermodular if and only if its second order derivatives*

$$\Delta_{ij}(\mathbf{x}) \stackrel{\text{def}}{=} \frac{\partial f}{\partial x_i \partial x_j}(\mathbf{x}) \geq 0$$

are nonnegative for all $1 \leq i < j \leq n$ and for all $\mathbf{x} \in \mathbb{B}^n$. □

Clearly, f is modular iff both f and $-f$ are supermodular, i.e. if and only if $\Delta_{ij}(\mathbf{x}) \equiv 0$ for all $i \neq j$, and consequently, if and only if it is linear.

For a quadratic pseudo-Boolean function f it follows that f is supermodular iff all quadratic terms of it have non-negative coefficients, or equivalently, f is submodular iff its quadratic terms are non-positive. The minimization of such a quadratic submodular function is known to be equivalent with finding a minimum capacity cut in a corresponding network (see e.g. [72, 99]). Indeed, let us consider a quadratic posiform of a quadratic submodular function. According to the above, such a posiform ϕ can always be written, possibly after some simple transformations, as

$$\phi(x_1, ..., x_n) = \sum_{i \in P} a_i x_i + \sum_{j \in N} a_j \overline{x}_j + \sum_{1 \leq i < j \leq n} a_{ij} x_i \overline{x}_j$$

where $P, N \subseteq \mathbf{V}$, and where all the coefficients a_i ($i = P \cup N$) and a_{ij} ($1 \leq i < j \leq n$) are nonnegative. Let us associate then to ϕ a network N_ϕ on the node set $V(N_\phi) = \{s, t\} \cup \mathbf{V}$, with arcs corresponding to the terms of f:

$$A(N_\phi) = \{(s, j) \mid j \in N\} \cup \{(i, t) \mid i \in P\} \cup \{(i, j) \mid 1 \leq i < j \leq n\}$$

where the capacities of the arcs are defined as $c_{s,j} = a_j$ for $j \in N$, $c_{i,t} = a_i$ for $i \in P$, and $c_{i,j} = a_{ij}$ for $1 \leq i < j \leq n$. Then the s,t-cuts of this network are in a one-to-one correspondence with the binary vectors: $\mathbf{x} \in \mathbb{B}^n \longleftrightarrow S_{\mathbf{x}} \stackrel{\text{def}}{=} \{s\} \cup \{j \mid x_j = 1\}$. It is easy to check that with these definitions we have

$$\phi(\mathbf{x}) = \sum_{u \in S_{\mathbf{x}}, v \notin S_{\mathbf{x}}} c_{u,v}$$

for all $\mathbf{x} \in \mathbb{B}^n$, and hence the minimum of ϕ will correspond to a minimum capacity cut of N_ϕ.

The above analysis was extended to cubic posiforms, as well.

Proposition 25 ([24]) *A cubic posiform*

$$\psi(\mathbf{x}) = \sum_{T \subseteq \mathbf{L}} a_T \prod_{u \in T} u,$$

with $a_T \geq 0$ for all $T \subseteq \mathbf{L}$, and $a_T = 0$ for all $|T| > 3$, defines a supermodular set function if and only if all terms of it are pure, *i.e. if and only if $T \subseteq \{x_1, ..., x_n\}$ or $T \subseteq \{\overline{x}_1, ..., \overline{x}_n\}$ holds whenever $a_T > 0$.* □

Based on the above characterization, the maximization of supermodular cubic posiforms was also shown to be equivalent with a network flow computation in [24]. Several other related classes have also been considered in the literature, including almost positive functions, polar functions, and their switch equivalents (see e.g. [41, 45, 46, 100]).

The above results cannot be generalized easily. The recognition of sub(or super)modularity for higher degree posiforms turns out to be a hard problem, unless P=NP.

Proposition 26 ([62]) *The recognition of supermodularity of quartic (degree 4) posiforms is co-NP-complete.* □

Let us remark finally that even though the minimum of a submodular (or the maximum of a supermodular) set function can be found in polynomial time ([71]), or even in strongly polynomial time (see [108, 152]), the opposite optimization problems, i.e. the maximization of a submodular (or the minimization of a supermodular) set function is NP-hard (see e.g. [58, 130]).

Let us further add that a standard greedy procedure for the maximization of a submodular set function provides a $(1 - \frac{1}{e})$-approximation of the maximum, as shown in [58, 130].

6.2 Half-products

In this section we consider *half-products*, a special subclass of supermodular functions, defined by multilinear polynomial expressions of the following form

$$f(\mathbf{x}) \stackrel{\text{def}}{=} \sum_{1 \leq i < j \leq n} a_i b_j x_i x_j - \sum_{i=1}^{n} c_i x_i, \tag{61}$$

where $\mathbf{a} = (a_1, ..., a_n)$, $\mathbf{b} = (b_1, ..., b_n)$ and $\mathbf{c} = (c_1, ..., c_n)$ are non-negative integer vectors (in fact, the values of a_n and b_1 are not used).

These functions were considered independently in [9, 10, 118], and have attracted considerable attention, since a number of scheduling problems can be formulated as half-product minimization; examples where this model arises include the scheduling of two machines to minimize total weighted completion time or to minimize the makespan, and the scheduling of a single machine to minimize completion time variance, agreeably weighted completion time variance, or total weighted earliness and tardiness (see e.g. [9, 10, 40, 111, 118]). Half-products also occur in physics, e.g. in the infinite range Mattis model of a spin-glass, the energy function is in fact a half-product (see [6, 126]).

Half-products clearly form a subfamily of supermodular set functions by Proposition 24, since all quadratic terms of a half-product have a non-negative coefficient. It can be shown that the minimization of even these special supermodular functions remains NP-hard.

Proposition 27 ([10]) *The minimization of half-products is NP-hard.* □

Still, a fully polynomial time approximation scheme exists for the minimization of half-products, providing another attractive feature of this class.

Proposition 28 ([10]) *Given a half-product $f(\mathbf{x})$ by (61), let $A = \sum_{i=1}^{n-1} a_i$, and let $\mathbf{x}^*$ denote a minimum of f. Then, for every $\epsilon > 0$, one can find in $O((n^2 \ln A)/\epsilon)$ time a binary vector $\mathbf{x}^\epsilon$, for which*

$$f(\mathbf{x}^\epsilon) - f(\mathbf{x}^*) \leq \epsilon |f(\mathbf{x}^*)|.$$

□

The main component of the above approximation scheme is the existence of a pseudo-polynomial dynamic programming algorithm. Introducing

$$g_k(x_1, ..., x_k) \stackrel{\text{def}}{=} \sum_{1 \leq i < j \leq k} a_i b_j x_i x_j - \sum_{i=1}^{k} c_i x_i,$$

and

$$h_k(x_{k+1}, ..., x_n) \stackrel{\text{def}}{=} \sum_{k < i < j \leq n} a_i b_j x_i x_j - \sum_{i=k+1}^{n} c_i x_i$$

for $k = 1, 2, ..., n-1$, we can write the half-product f as

$$f(\mathbf{x}) = g_k(x_1, ..., x_k) + \left(\sum_{i=1}^{k} a_i x_i \right) \left(\sum_{j=k+1}^{n} b_j x_j \right) + h_k(x_{k+1}, ..., x_n).$$

From this it can be seen that if $\mathbf{x}$ and $\mathbf{y}$ are binary vectors for which $x_j = y_j$ for all $j > k$, $g_k(x_1, ..., x_k) \leq g_k(y_1, ..., y_k)$, and $\sum_{i=1}^{k} a_i x_i \leq \sum_{i=1}^{k} a_i y_i$, then $f(\mathbf{x}) \leq f(\mathbf{y})$. As a consequence, the first k components x_1^*, ..., x_k^* of a minimizing vector $\mathbf{x}^*$ of f will correspond to one of the minimal two-dimensional integer vectors of the form $(g_k(x_1, ..., x_k), \sum_{i=1}^k a_i x_i)$, $\mathbf{x} \in \mathbb{B}^n$, of which we have at most $\sum_{i=1}^k a_i \leq A$ different ones. Thus, updating recursively for $k = 1, 2, ..., n$ these (at most A) different two-dimensional vectors, we can determine the minimum of f in $O(nA)$ steps.

The above dynamic programming idea was further specialized for the case of the so called *ordered symmetric half-products* in [119]. A half-product f is called *symmetric* if it can be written as

$$f(\mathbf{x}) = \sum_{1 \leq i < j \leq n} a_i b_j (2x_i x_j - x_i - x_j)$$

for some non-negative integer vectors $\mathbf{a} = (a_1, ..., a_n)$ and $\mathbf{b} = (b_1, ..., b_n)$. It is called *ordered symmetric* if either $a_1 \leq a_2 \leq \cdots \leq a_n$, or $b_1 \geq b_2 \geq \cdots \geq b_n$.

Proposition 29 ([119]) *An ϵ-approximation of the minimum of an ordered symmetric half-product can be found in $O(n^2/\epsilon)$ time for any $\epsilon > 0$.* □

6.3 Hyperbolic pseudo-Boolean programming

An interesting special class of pseudo-Boolean optimization is the maximization (or minimization) of the ratio of two linear functions:

$$\max_{\mathbf{x} \in \mathbb{B}^n} f(\mathbf{x}) = \frac{a_0 + \sum_{j=1}^n a_j x_j}{b_0 + \sum_{j=1}^n b_j x_j} \tag{62}$$

This problem was introduced in [91] as *fractional programming*, and was studied later in [97, 128, 147]. Its applications include query optimization in data bases and information retrieval (see e.g. [97]).

It is known ([91]) that (62) has an easy polynomial time solution if

$$b_0 + \sum_{j=1}^n b_j x_j > 0 \quad \text{for all} \quad \mathbf{x} \in \mathbb{B}^n. \tag{63}$$

In order to see the solution of (62), let us observe first that condition (63) implies that strong persistency holds for f at the partial assignment $\mathbf{y}$ defined by

$$y_j = \begin{cases} 1 \text{ if } (a_j > 0 \text{ and } b_j \leq 0), & \text{or} \quad (a_j = 0 \text{ and } b_j < 0), \\ 0 \text{ if } (a_j < 0 \text{ and } b_j \geq 0), & \text{or} \quad (a_j = 0 \text{ and } b_j > 0). \end{cases} \tag{64}$$

Let us also note that if $a_j = b_j = 0$ then in fact $f(\mathbf{x})$ does not depend on x_j, and that by substituting $x_j = 1 - x'_j$ whenever $a_j < 0$ and $b_j < 0$ we can obtain an equivalent maximization problem, in which the coefficients of all variables are positive.

Thus, in case (63) holds, we can assume without any loss of generality that

$$b_j > 0 \text{ for } j = 0, 1, ..., n, \quad \text{and} \quad a_j > 0 \text{ for } j = 1, 2, ..., n. \tag{65}$$

Denoting by $\mathbf{x}^*$ an optimum of (62), let us observe next that $f(\mathbf{x}^*) \geq t$ if and only if

$$a_0 - tb_0 + \sum_{j=1}^{n} (a_j - tb_j) x_j^* \geq 0$$

or equivalently, if and only if

$$\max_{\mathbf{x} \in \mathbb{B}^n} \sum_{j=1}^{n} (a_j - tb_j) x_j \geq -a_0 + tb_0.$$

This latter optimization problem is trivial to solve, and for every value of the threshold t its optimal solution is one of the $n+1$ binary vectors of the form

$$x_{j_l} = 1 \text{ for } l \leq k \;\text{ and }\; x_{j_l} = 0 \text{ for } l > k, \tag{66}$$

for $k = 0, 1, ..., n$, where $(j_1, ..., j_n)$ is a permutation of the indices such that

$$\frac{a_{j_1}}{b_{j_1}} \geq \frac{a_{j_2}}{b_{j_2}} \geq \cdots \geq \frac{a_{j_n}}{b_{j_n}}.$$

Hence, the optimum of (62) is one of these $n+1$ binary vectors whenever (63) holds. Since these vectors can be generated in $O(n \log n)$ time, and all corresponding values of f can be determined in $O(n)$ time, problem (62) can be solved in this case in $O(n \log n)$ time.

Let us add that an analogous analysis with a similar conclusion can obviously be carried out in case (62) is a minimization problem.

Let us remark next that of course, if $b_0 + b_1 x_1 + \cdots + b_n x_n$ can also take the value zero, then (62) may not have a finite optimum. Furthermore, even if the condition

$$b_0 + \sum_{j=1}^{n} b_j x_j \neq 0 \;\text{ for all }\; \mathbf{x} \in \mathbb{B}^n \tag{67}$$

holds, but the denominator can take both negative and positive values (e.g. $b_0 < 0$), problem (62) is NP-hard. To see this latter claim, let us consider the problem of deciding if there exists a binary assignment $\mathbf{x} \in \mathbb{B}^n$ for which

$$\sum_{j=1}^{n} s_j x_j = S, \tag{68}$$

where $s_j > 0$, $j = 1, ..., n$ and S are given integers. This problem is known as the *subset sum* problem, a well-known NP-complete decision problem. Let us associate to it the hyperbolic pseudo-Boolean optimization problem

$$\max_{\mathbf{x}\in\mathbb{B}^n} \frac{-1}{(-1-2S)+\sum_{j=1}^{n} 2s_j x_j}. \tag{69}$$

It is easy to verify that the maximum of (69) is 1 if and only if (68) has a solution, implying that maximizing (69) cannot be easier than finding a solution to (68).

6.4 Products of linear functions

Another interesting special case of pseudo-Boolean optimization is the maximization (or minimization) of the product of two linear functions over binary variables:

$$\max_{\mathbf{x}\in\mathbb{B}^n} f(\mathbf{x}) = l_1(\mathbf{x})l_2(\mathbf{x}), \tag{70}$$

where

$$l_1(\mathbf{x}) = a_0 + a_1x_1 + \cdots + a_nx_n, \quad \text{and} \quad l_2(\mathbf{x}) = b_0 + b_1x_1 + \cdots + b_nx_n.$$

This problem was considered in [76], and was shown to be an NP-hard optimization problem via a reduction from the subset sum problem cited above. The continuous relaxation

$$\max_{\mathbf{q}\in\mathbb{U}^n} f(\mathbf{q}) = l_1(\mathbf{q})l_2(\mathbf{q}), \tag{71}$$

was also considered in [76], and a polynomial $O(n\log n)$ time algorithm was presented for it. Based on this, a branch and bound algorithm was developed for (70). Computational experiments were carried out with problems of this type involving up to 10000 variables.

7 Approximation Algorithms

7.1 MAX-SAT and variants

The *maximum satisfiability* problem is one of the central problems of computer science and combinatorial optimization. Given a family of weighted elementary disjunctions (so called *clauses*), the maximum satisfiability problem consists in finding a binary assignment to the Boolean variables which maximizes the total weight of satisfied clauses. Using the equality

$$\bigvee_{u\in C} u = 1 - \prod_{u\in C} \overline{u}$$

for subsets $C \subseteq \mathbf{L}$ of literals, we can reformulate this problem as a pseudo-Boolean optimization problem, as shown in Section 3: Given a family $\mathcal{C}$ of literals, and nonnegative weights $a_C \in \mathbb{R}_+$ associated to the clauses $C \in \mathcal{C}$, the maximum satisfiability problem can be stated as

$$\max_{\mathbf{x}\in\mathbb{B}^n} \sum_{C\in\mathcal{C}} a_C \left(1 - \prod_{u\in C} \overline{u}\right). \tag{72}$$

The maximum satisfiability problem is a well-known NP-hard optimization problem, a common generalization of many other combinatorial optimization problems (e.g., maximum cut in graphs, maximum directed cut in directed graphs, etc.), which is also known to have good approximations, as well as inapproximability results.

For a maximization problem $\max f(\mathbf{x})$ a vector $\tilde{\mathbf{x}}$ is called an *α-approximation*, if

$$\frac{f(\tilde{\mathbf{x}})}{\max f(\mathbf{x})} \geq \alpha.$$

It is important to note that this measure of approximability is not invariant under several simple operations, which otherwise do not change the optimization problem. For instance, the same $\tilde{\mathbf{x}}$ vector may not be an α-approximation of the objective function $f(\mathbf{x}) - K$, where K is a nonnegative constant, though the maximization of $f(\mathbf{x}) - K$ is clearly equivalent with the maximization of $f(\mathbf{x})$. It is also important to point out here that while the maximum satisfiability problem is equivalent with the minimization of the corresponding posiform

$$\sum_{C \in \mathcal{C}} a_C \prod_{u \in C} \overline{u}$$

the latter problem cannot be approximated well, unless $P = NP$. For a more precise treatment of approximation algorithms and related notions of complexity see [136].

Returning to approximations of the maximum satisfiability problem, there are many relevant results to mention here, with a particular increase in research volume in the last decade, due to two important new techniques: on the one hand, semidefinite formulations yielded new efficient approximations to several variants of the maximum satisfiability problem, while, on the other hand, the development of probabilistic proof verification techniques made it possible to prove inapproximability beyond certain rates, assuming $P \neq NP$.

Before giving a brief overview of these results, let us recall an extended terminology for the many variants of this problem. MAX SAT refers to the maximum satisfiability problem as stated in (72). If $|C| \leq k$ for all clauses $C \in \mathcal{C}$, the problem is called MAX-k-SAT. Particular attention is given in the literature to MAX-2-SAT, which is, as an optimization problem, equivalent with quadratic pseudo-Boolean optimization.

A natural generalization of maximum satisfiability is called *maximum constraint satisfaction problem*, or MAX CSP. Let g be an arbitrary Boolean expression, and let us denote by $g(S)$ the value of this function when applied to the set of literals $S \subseteq \mathbf{L}$. For any fixed Boolean expression g, MAX CSP(g) denotes the following problem: Given a collection $\mathcal{S}$ of subsets of literals, and nonnegative weights $a_S \in \mathbb{R}_+$ associated to these subsets $S \in \mathcal{S}$, find a binary assignment to the variables, which maximizes the function

$$\sum_{S \in \mathcal{S}} a_S g(S). \tag{73}$$

It is easy to see that for most interesting cases in the literature, g has a very short posiform representation, and hence MAX CSP(g) is a special case of pseudo-Boolean optimization in all these cases. For instance, denoting by

$$OR(S) = \bigvee_{u \in S} u = 1 - \prod_{u \in S} \overline{u}$$

we find that MAX CSP(OR) is the maximum satisfiability problem as written in (72), while if

$$AND(S) = \bigwedge_{u \in S} u = \prod_{u \in S} u$$

then MAX CSP(AND) is the posiform maximization as considered in Section 4.7.

It is customary to indicate in the lower index if the size of the sets S to which these functions are applied are limited in size, e.g. MAX CSP(OR_2) denotes MAX-2-SAT, etc. For instance, if

$$XOR_2(u, v) = u\overline{v} + \overline{u}v$$

then MAX CSP(XOR_2) is in fact a generalization of the maximum cut problem in graphs and signed graphs. Another interesting case is the majority function

$$MAJ_3(u, v, w) = \begin{cases} 1 & \text{if } u + v + w \geq 2, \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to see that MAX CSP(MAJ_3) is the problem of finding the maximum number of clauses in a given 3-CNF, which can be switched to a Horn formula, a problem arising in artificial intelligence (see [26]).

A $\frac{1}{2}$-approximation algorithm for MAX-SAT is simply the application of RoundUp (see subsection 4.1) starting with $(\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2})$, as proposed in [109]. In fact that algorithm provides a $(1 - \frac{1}{2^k})$-approximation whenever all terms are of degree k or larger, implying e.g. a $\frac{3}{4}$-approximation for MAX-2-SAT in which there are no linear terms. For the case, when the linear terms are present, but cannot be trivially simplified a $\frac{\sqrt{5}-1}{2}$-approximation (0.618...-approximation) was obtained by [123]. In [116] a randomized algorithm is presented yielding a $\frac{2}{3}$-approximation, in expected value. A transformation producing an equivalent problem without linear terms, and yielding a $\frac{3}{4}$-approximation with the help of RoundUp starting from $(\frac{1}{2}, \frac{1}{2}, \cdots, \frac{1}{2})$ was presented in [166]. In fact this method for MAX-2-SAT is simply a combination of roof-duality and the RoundUp procedure. We shall recall this procedure in more detail in the next subsection.

In [69] a new $\frac{3}{4}$-approximation algorithm was provided by a combination of the rounding method and another $(1 - \frac{1}{e})$-approximation obtained via a linear programming formulation. Though this latter method solves the problem by solving a higher dimensional model first, it can be simplified somewhat by using a simple convex majorant of the input posiform, and a robust $\frac{3}{4}$-approximation algorithm can be obtained via convex programming in the original space of the variables [9].

A substantial improvement was presented in [70] based on an $O(n^2)$-dimensional semidefinite formulation of the problem, yielding a 0.87856-approximation for both MAX-2-SAT and

MAX CUT (in fact for MAX CSP(XOR_2)). This method also yielded a 0.758-approximation for MAX SAT. This has further been improved to a 0.93109-approximation for MAX-2-SAT and an 0.859-approximation for quadratic posiform maximization (MAX CSP(AND_2)) in [56].

This novel approach of solving a semidefinite relaxation and then rounding the resulted $O(n^2)$ dimensional fractional solution to a binary n-vector has been applied in the last few years to a large variety of combinatorial optimization problems. Among these we mention the $\frac{7}{8}$-approximation for MAX-3-SAT by [113], a $\frac{1}{2}$-approximation for MAX CSP(AND_3), and a $\frac{2}{3}$-approximation for MAX CSP(MAJ_3) by [168]. Let us add that for the latter problem a robust but weaker $\frac{40}{67}$-approximation can be obtained using pseudo-Boolean techniques without solving a large semidefinite relaxation [26].

On the negative side, the development of the theory of probabilistically checkable proofs [7, 8] lead to a series of inapproximability results. For instance the results mentioned above for the MAX-3-SAT, MAX CSP(AND_3) and MAX CSP(MAJ_3) problems are all known to be best possible, unless P=NP (see [20, 101, 159]). It is also known that for MAX-2-SAT it is impossible to obtain in polynomial time an approximation better than $\frac{21}{22}$, unless P=NP (see [159]).

To give a more detailed overview of all these results and techniques is beyond the scope of our survey of pseudo-Boolean optimization. We shall recall only two results, where the applied techniques are specific to the theory of pseudo-Boolean functions. These results also show that a robust and reasonably good approximation can be achieved without solving a high dimensional relaxation.

Let us point out that most approaches which do not employ semidefinite programming used in fact a variant of the RoundUp procedure with $(\frac{1}{2}, \cdots, \frac{1}{2})$ as a starting point. We would like to show that by applying algebraic transformation to the input posiform first, and using persistency before applying RoundUp (see e.g. [166]), as well as precomputing a better starting point than $(\frac{1}{2}, \cdots, \frac{1}{2})$ (see e.g. [9, 27]) can lead to substantial performance improvements.

7.2 3/4-approximation of MAX-2-SAT via roof-duality

As a first example, let us describe here a simple proof for the $\frac{3}{4}$-approximation algorithm of [166] for the MAX-2-SAT problem. Let us consider a MAX-2-SAT instance as given by

$$f(\mathbf{x}) = \sum_{u \in \mathbf{L}} a_u(1 - \overline{u}) + \sum_{\substack{u,v \in \mathbf{L} \\ u \neq v}} a_{uv}(1 - \overline{u}\overline{v}). \tag{74}$$

By introducing

$$\phi(\mathbf{x}) = \sum_{u \in \mathbf{L}} a_u \overline{u} + \sum_{\substack{u,v \in \mathbf{L} \\ u \neq v}} a_{uv} \overline{u}\overline{v}$$

and recalling that $A(\phi)$ denotes the sum of the coefficients of the posiform ϕ, we can write the above MAX-2-SAT problem as

$$\max_{\mathbf{x}\in\mathbb{B}^n} f(\mathbf{x}) = \max_{\mathbf{x}\in\mathbb{B}^n}(A(\phi) - \phi(\mathbf{x})) = A(\phi) - \min_{\mathbf{x}\in\mathbb{B}^n} \phi(\mathbf{x}). \tag{75}$$

By applying roof-duality (see subsection 5.1.3) we can bring the quadratic posiform ϕ to the form

$$\phi(\mathbf{x}) = C_2(\phi) + l(\mathbf{x}) + \psi(\mathbf{x}), \tag{76}$$

where $C_2(\phi)$ is the roof-dual value of ϕ, $l(\mathbf{x})$ is a linear posiform, and $\psi(\mathbf{x})$ is a pure quadratic posiform.

Let us also recall (see section 5.1.5) that the right hand side of (76) can be obtained by iteratively applying identities of the form (44)

$$u_1 + \overline{u}_1 u_2 + \overline{u}_2 u_3 + \cdots + \overline{u}_{k-1} u_k + \overline{u}_k = 1 + u_1\overline{u}_2 + \cdots + u_{k-1}\overline{u}_k.$$

Since the sum of coefficients is exactly one more on the left hand side than on the right hand side above, the following equation is implied: $A(\phi) = 2C_2(\phi) + A(l) + A(\psi)$. Since both $A(l)$ and $C_2(\phi)$ are nonnegative, we can conclude that

$$A(\psi) \leq A(\phi) - C_2(\phi). \tag{77}$$

Let us now denote by $\mathbf{x}^*$ an optimum of the above MAX-2-SAT instance, i.e. a minimum of both ϕ and ψ according to (76). Furthermore, let $\mathbf{x}^r$ denote the binary vector obtained by eliminating first the variables which are fixed by strong persistency (see Theorem 8), and then by applying $\textsc{RoundDown}(\psi,(\frac{1}{2},\cdots,\frac{1}{2}))$ to derive the values of the rest of the variables. Thus, for both vectors

$$l(\mathbf{x}^*) = l(\mathbf{x}^r) = 0 \tag{78}$$

holds by the above definitions, implying by (76) that

$$\min_{\mathbf{x}\in\mathbb{B}^n} \phi(\mathbf{x}) = \phi(\mathbf{x}^*) = C_2(\phi) + \psi(\mathbf{x}^*), \tag{79}$$

and

$$\phi(\mathbf{x}^r) = C_2(\phi) + \psi(\mathbf{x}^r). \tag{80}$$

Furthermore, by Proposition 5 we have

$$\psi(\mathbf{x}^*) \leq \psi(\mathbf{x}^r) \leq \psi(\frac{1}{2},\frac{1}{2},\cdots,\frac{1}{2}) = Exp[\psi]. \tag{81}$$

Here the expectation is taken in a probability space, as in Proposition 5, in which the variables are pairwise independent random variables with probabilities $Prob(x_j = 1) = Prob(x_j = 0) = \frac{1}{2}$, $j = 1, ..., n$. Since here $Prob(uv = 1) = \frac{1}{4}$ for any pair of literals with $u \notin \{v, \overline{v}\}$, and since ψ is a pure quadratic posiform, we obtain from (81) that

$$\psi(\mathbf{x}^r) \leq \frac{1}{4}A(\psi). \tag{82}$$

Putting all the above together we get

$$\frac{f(\mathbf{x}^r)}{f(\mathbf{x}^*)} = \frac{A(\phi) - C_2(\phi) - \psi(\mathbf{x}^r)}{A(\phi) - C_2(\phi) - \psi(\mathbf{x}^*)} \geq \frac{A(\phi) - C_2(\phi) - \frac{1}{4}A(\psi)}{A(\phi) - C_2(\phi)} \geq \frac{3}{4}.$$

Here the first equality follows by (79)-(80). The first inequality follows by (82) and by $\psi(\mathbf{x}^*) \geq 0$, while the last inequality is implied by (77).

Thus, the above inequality shows that $\mathbf{x}^r$ provides a $\frac{3}{4}$-approximation of the problem (75). Let us add that $\mathbf{x}^r$ can be derived in $O(n^3)$ time from f, first determining the roof dual (76) by computing the maximum flow in a network of $2n+2$ nodes (see subsection 5.1.5) in $O(n^3)$ time, next fixing some of the variables by strong persistency in $O(|\phi|) = O(n^2)$ time, and finally applying RoundDown(ψ,$(\frac{1}{2}, \cdots, \frac{1}{2})$) in $O(|\phi|) = O(n^2)$ time again.

7.3 3/4-approximation of MAXSAT via convex majorization

In this section we recall a $\frac{3}{4}$-approximation algorithm for the MAX SAT problem from [9]. This algorithm computes a vector $\mathbf{p}^* \in \mathbb{U}^n$ via convex programming without increasing the dimensions of the problem, and then constructs an approximative solution by applying the RoundDown algorithm starting from $\mathbf{p}^*$. The proof of correctness is very analogous to the one given in [69] with the notable difference that here only one starting point is needed for the rounding procedure, and even that point has to be determined only up to a certain fixed precision, allowing thus a faster and more robust computation.

Let us consider a posiform given as

$$\phi(\mathbf{x}) = \sum_{C \in \mathcal{C}} a_C \prod_{u \in C} \overline{u} \tag{83}$$

where $\mathcal{C}$ is a given family of subsets of literals, and a_C are positive integers for $C \in \mathcal{C}$. Furthermore, let us consider the corresponding MAX SAT instance

$$\max_{\mathbf{x} \in \mathbb{B}^n} \sum_{C \in \mathcal{C}} a_C (1 - \prod_{u \in C} \overline{u}) = \max_{\mathbf{x} \in \mathbb{B}^n} \left(A(\phi) - \phi(\mathbf{x})\right), \tag{84}$$

where $A(\phi) = \sum_{C \in \mathcal{C}} a_C$ denotes again the sum of the coefficients of the posiform ϕ.

Let us next consider the following convex majorant of ϕ

$$g(\mathbf{x}) = \sum_{C \in \mathcal{C}} a_C \left(\frac{\sum_{u \in C} \overline{u}}{|C|}\right)^{|C|} \tag{85}$$

formed as the sum of convex termwise majorants. It is easy to verify that

$$g(\mathbf{p}) \geq \phi(\mathbf{p}) \text{ holds for every } \mathbf{p} \in \mathbb{U}^n. \tag{86}$$

Let us further fix a small positive constant

$$\epsilon = \frac{3}{16} - \frac{1}{2e} \approx 0.00356. \tag{87}$$

Let $\mathbf{p}^*$ be the minimum of $g(\mathbf{x})$ over $\mathbb{U}^n$ (since g is a smooth convex function, $\mathbf{p}^*$ is unique), and let $\hat{\mathbf{p}} \in \mathbb{U}^n$ be such that

$$g(\hat{\mathbf{p}}) \leq g(\mathbf{p}^*) + \epsilon. \tag{88}$$

It is well-known that such a $\hat{\mathbf{p}}$ can be determined in polynomial time in the size of g, and hence in the size of ϕ, see e.g. [131]. In fact, due to the tolerance ϵ, most convex optimization algorithms provide such a vector in a very fast and numerically robust way.

Let us denote again by $\mathbf{x}^* \in \mathbb{B}^n$ a minimum of ϕ, and finally let $\mathbf{x}^r \in \mathbb{B}^n$ denote the binary vector obtained by the application of RoundDown(ψ,$\hat{\mathbf{p}}$), unless ϕ is a quadratic posiform and $\phi(\mathbf{x}^*) = 0$, in which case let $\mathbf{x}^r = \mathbf{x}^*$.

Theorem 13 ([9]) *The vector $\mathbf{x}^r$ is a $\frac{3}{4}$-approximative solution of the MAX SAT problem (84).*

Proof. Clearly, if ϕ is a quadratic posiform, the equality $\phi(\mathbf{x}^*) = 0$ can be detected in $O(|\phi|)$ time, by solving the corresponding quadratic Boolean equation. Obviously, $\mathbf{x}^r = \mathbf{x}^*$ is the exact optimum of (84) in this case, and hence the statement holds.

Let us assume in the sequel that either ϕ is not quadratic, or $\phi(\mathbf{x}^*) > 0$, and let us compute $\mathbf{x}^r$ in polynomial time, as described above, by solving a convex minimization problem first, up to a precision of ϵ, and then running RoundDown(ψ,$\hat{\mathbf{p}}$). We have then the inequalities

$$\phi(\mathbf{x}^r) \leq \phi(\hat{\mathbf{p}}) \leq g(\hat{\mathbf{p}}) \leq g(\mathbf{p}^*) + \epsilon \leq g(\mathbf{x}^*) + \epsilon \tag{89}$$

and

$$g(\hat{\mathbf{p}}) \leq g(\mathbf{p}^*) + \epsilon \leq g(\frac{1}{2}, \cdots, \frac{1}{2}) + \epsilon, \tag{90}$$

implying thus

$$\phi(\mathbf{x}^r) \leq \frac{g(\frac{1}{2}, \cdots, \frac{1}{2}) + g(\mathbf{x}^*)}{2} + \epsilon. \tag{91}$$

Let us also introduce the notation

$$l_C = \sum_{u \in C} \overline{u}(\mathbf{x}^*) \tag{92}$$

for $C \in \mathcal{C}$. With this notation we have $\phi(\mathbf{x}^*) = \sum_{C \in \mathcal{C}:\ l_C = |C|} a_C$, and thus

$$f(\mathbf{x}^*) = A(\phi) - \phi(\mathbf{x}^*) = \sum_{C \in \mathcal{C}:\ l_C < |C|} a_C. \tag{93}$$

Furthermore, we can write (91) as

$$\begin{aligned} f(\mathbf{x}^r) &= A(\phi) - \phi(\mathbf{x}^r) \\ &\geq A(\phi) - \epsilon - \tfrac{g(\frac{1}{2}, \cdots, \frac{1}{2}) + g(\mathbf{x}^*)}{2} \\ &\geq -\epsilon + \textstyle\sum_{C \in \mathcal{C}} a_c \left(1 - \frac{1}{2}\left(\frac{l_C}{|C|}\right)^{|C|} - \frac{1}{2^{|C|+1}}\right) \end{aligned} \tag{94}$$

It is easy to verify by elementary calculations that

$$1 - \frac{1}{2}\left(\frac{l_C}{|C|}\right)^{|C|} - \frac{1}{2^{|C|+1}} \geq \begin{cases} \frac{1}{4} & \text{if } l_C = |C|,\ |C| > 0, \\ \frac{3}{4} & \text{if } l_C < |C|,\ |C| \leq 2, \\ \frac{3}{4} + \epsilon & \text{if } l_C < |C|,\ |C| \geq 3. \end{cases} \tag{95}$$

Substituting these estimates back into (94) we obtain

$$f(\mathbf{x}^r) \geq \frac{3}{4} \sum_{\substack{C \in \mathcal{C} \\ l_C < |C|}} a_C + \frac{1}{4} \sum_{\substack{C \in \mathcal{C} \\ l_C = |C|}} a_C + \epsilon \left(-1 + \sum_{\substack{C \in \mathcal{C} \\ l_C < |C| \\ |C| \geq 3}} a_C \right) \tag{96}$$

Since the coefficients a_C for $C \in \mathcal{C}$ are assumed to be positive integers, and since $\frac{1}{4} > \epsilon$, we have

$$\frac{1}{4} \sum_{\substack{C \in \mathcal{C} \\ l_C = |C|}} a_C + \epsilon \left(-1 + \sum_{\substack{C \in \mathcal{C} \\ l_C < |C| \\ |C| \geq 3}} a_C \right) \leq 0$$

if and only if

$$\sum_{\substack{C \in \mathcal{C} \\ l_C = |C|}} a_C = 0 \quad \text{and} \quad \sum_{\substack{C \in \mathcal{C} \\ l_C < |C| \\ |C| \geq 3}} a_C = 0,$$

i.e, if and only if ϕ is a quadratic posiform and $\phi(\mathbf{x}^*) = 0$. Since this case was treated separately at the beginning of this proof, we can conclude from (96) that in all other cases

$$f(\mathbf{x}^r) \geq \frac{3}{4} \sum_{\substack{C \in \mathcal{C} \\ l_C < |C|}} a_C \tag{97}$$

must hold, implying thus by (93) that

$$\frac{f(\mathbf{x}^r)}{f(\mathbf{x}^*)} \geq \frac{3}{4}. \tag{98}$$

□

References

[1] Adams, W.E., A. Billionnet, and A. Sutter. Unconstrained $0-1$ optimization and Lagrangean relaxation. First International Colloquium on Pseudo-Boolean Optimization and Related Topics (Chexbres, 1987). *Discrete Appl. Math.* **29** (1990), no. 2-3, 131–142.

[2] Adams, W.P. and P.M. Dearing. On the equivalence between roof duality and Lagrangian duality for unconstrained $0-1$ quadratic programming problems. *Discrete Appl. Math.* **48** (1994), pp. 1–20.

[3] Adams, W.P. and H.D. Sherali. A tight linearization and an algorithm for zero-one quadratic programming problems. *Management Sci.* **32** (1986), no.10, 1274-1290.

[4] Alexe, G., and P.L. Hammer. Boolean simplifications of the stability problem in graphs. RUTCOR Research Report, RUTCOR, August, 2001.

[5] Alon N. and J.H. Spencer. *The probabilistic method*. John Wiley and Sons, New York, 1992.

[6] Amit, D.J. *Modeling brain function: the world of attractor neural networks*. (Cambridge University Press, 1989).

[7] Arora, S., C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science* Pittsburgh, Pennsylvania, 1992, pp. 14-23.

[8] Arora, S., and M. Safra. Probabilistic checking of proofs: a new characterization of NP. In *Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science* Pittsburgh, Pennsylvania, 1992, pp. 2-13.

[9] Badics, T. *Approximation of some Nonlinear Binary Optimization Problems*. Ph.D. Thesis, RUTCOR, Rutgers University, 1996.

[10] Badics T. and E. Boros. Minimization of Half-Products. *Mathematics of Operations Research*, **23** (1998), pp. 649-660.

[11] Balas, E. Extension de l'algorithme additif a la programmation en nombres at a la programmation non lineare. *C.R. Acad. Sci. Paris* **258** (1967) 5136-5139.

[12] Balas, E. and J.B. Mazzola. Nonlinear $0-1$ programming: I. Linearization techniques and II. Dominance relations and algorithms. *Mathematical Programming* **30** (1984), 1-45.

[13] Balas, E. and M. Padberg, On the set covering problem, *Operations Research* **20** (1972) 1152-1161.

[14] Balas, E. and M. Padberg, Set partitioning: a survey, *SIAM Review* **18** (1976) 710-760.

[15] Banzhaf, JF III. Weighted voting does not work: A mathematical analysis. *Rutgers Law Review* **19** (1965) pp. 317-343.

[16] Barahona, F. A solvable case of quadratic 0-1 programming. *Discrete Appl. Math.* **13** (1986), no. 1, 23-26.

[17] Barahona, F., M. Grötschel, M. Jünger and G. Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Operation Research* **36** (1988), 493-513.

[18] Barahona, F., M. Grötschel and A.R. Mahjoub. Facets of the bipartite subgraph polytope. *Math. of Opns. Res* **10** (1985), 340-358.

[19] Barahona, F. and A.R. Mahjoub. On the cut polytope. *Math. Prog.* **36** (1986), 157-173.

[20] Bellare, M., O. Goldreaich, and M. Sudan. Free bits, PCPsw and non-approximability – towards tight results. In *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science* Milwaukee, Wisconsin, 1995, pp. 422-431.

[21] Benzaken, C., Y. Crama, P.L. Hammer, F. and Maffray. More characterizations of triangulated graphs. RUTCOR Research Report 46-87, Rutgers University, 1987.

[22] Benzaken, C., S. Boyd, P.L. Hammer and B. Simeone. Adjoints of pure bidirected graphs. *Congressus Numerantium* **39** (1983), 123-144.

[23] Benzaken, C., P.L. Hammer and B. Simeone. Some remarks on conflict graphs of quadratic pseudo-Boolean functions. *International Series of Numerical Mathematics* **55** (1980), 9-30.

[24] Billionnet, A. and M. Minoux. Maximizing a supermodular pseudo-boolean function: a polynomial algorithm for supermodular cubic functions. *Discrete Appl. Math.* **12** (1985), no. 1, 1-11.

[25] Billionnet, A. and A. Sutter. Persistency in quadratic $0-1$ optimization. *Math. Programming* **54** (1992), no. 1, Ser. A, pp. 115–119.

[26] Boros E. Maximum renamable Horn sub-CNFs. *Discrete Applied Mathematics*, **96-97** (1999) pp. 29-40.

[27] Boros, E., Y. Caro, Z. Füredi, and R. Yuster. Covering non-uniform hypergraphs. *Journal of Combinatorial Theory (B)* **82** (2001) pp. 270-284.

[28] Boros, E., Y. Crama, and P.L. Hammer. Upper bounds for quadratic $0-1$ maximization. *Operations Research Letters*, **9** (1990), 73-79,

[29] Boros, E., Y. Crama and P.L. Hammer. Chvátal cuts and odd cycle inequalities in quadratic $0-1$ optimization. *SIAM Journal on Discrete Mathematics*, **5** (1992), 163-177.

[30] Boros E. and P.L. Hammer. A max-flow approach to improved roof-duality in quadratic $0-1$ minimization. RUTCOR Research Report RRR 15-1989, RUTCOR, March 1989.

[31] Boros E. and P.L. Hammer. The Max-Cut problem and Quadratic $0-1$ Optimization, Polyhedral Aspects, Relaxations and Bounds. *Annals of Operations Research*, **33** (1991), 151-180.

[32] Boros E. and P.L. Hammer. A generalization of the pure literal rule for satisfiability problems. RUTCOR Research Report RRR 20-92, RUTCOR, April 1992; *DIMACS Technical Report 92-19*

[33] Boros E. and P.L. Hammer. Cut-Polytopes, Boolean Quadric Polytopes and Nonnegative Quadratic Pseudo-Boolean Functions. *Mathematics of Operations Research*, **18** (1993) 245-253.

[34] Boros, E., P.L. Hammer, M. Minoux and D. Rader. Optimal Cell Flipping to Minimize Channel Density in VLSI Design and Pseudo-Boolean Optimization. *Discrete Applied Mathematics*, **90** (1999) pp. 69-88.

[35] Boros, E., P.L. Hammer and X. Sun. The DDT method for quadratic $0-1$ optimization. RUTCOR Research Report RRR 39-1989, RUTCOR, October 1989.

[36] Boros, E., P.L. Hammer and X. Sun. Network flows and minimization of quadratic pseudo-Boolean functions. RUTCOR Research Report RRR 17-1991, RUTCOR, May 1991.

[37] Boros E. and A. Prékopa. Probabilistic bounds and algorithms for the maximum satisfiability problem. *Annals of Operations Research* **21** (1989), pp. 109-126.

[38] Bourjolly, J-M., P.L. Hammer, W.R. Pulleyblank and B. Simeone. Boolean-combinatorial bounding of maximum 2-satisfiability. *Computer science and operations research (Williamsburg, VA 1992)*, pp. 23-42. (Pergamon, Oxford, 1992)

[39] Bushnell, M.L., and I.P. Shaik. U.S. Patent # 5,422,891, Robust delay-fault built-in testing method and its apparatus. June 6, 1995. (Patents pending in EEC, Canada, Japan and South Korea.)

[40] Cheng, J., and W. Kubiak. A faster fully polynomial approximation scheme for agreeable weighted completion time variance. Working Paper, Faculty of Business Administration, Memorial University of New-Foundland, 2000.

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

[42] Crama, Y. Concave extensions for nonlinear 0-1 maximization problems. *Mathematical Programming* **61** (1993) 53-60.

[43] Crama, Y. and P.L. Hammer. Recognition of quadratic graphs and adjoints of bidirected graphs. In : *Combinatorial Mathematics : Proceedings of the Third International Conference*, G.S. Bloom, R.L. Graham and J. Malkevitch (eds.), *Annals of the New York Academy of Sciences*, Vol. 555, 1989, pp. 140-149.

[44] Y. Crama and P.L. Hammer. Bimatroidal independence systems. *Zeitschrift fr Operations Research* **33** (1989) 149-165.

[45] Y. Crama, P.L. Hammer and R. Holzman. A characterization of a cone of pseudo-Boolean functions via supermodularity-type inequalities. In: *Quantitative Methoden in den Wirtschaftswissenschaften*, P. Kall, J. Kohlas, W. Popp and C.A. Zehnder (eds.), Springer-Verlag, Berlin-Heidelberg, 1989, pp. 53-55.

[46] Crama, Y., P.L. Hammer and T. Ibaraki. Strong unimodularity for matrices and hypergraphs. *Discrete Applied Mathematics* **15** (1986), 221-239.

[47] Crama, Y., P. Hansen and B. Jaumard. The basic algorithm for pseudo-Boolean programming revisited. *Discrete Appl. Math.* **29** (1990), no. 2-3, 171–185.

[48] Y. Crama, and J.B. Mazzola. Valid inequalities and facets for a hypergraph model of the nonlinear knapsack and FMS part-selection problems. *Annals of Operations Research* **58** (1995) 99-128.

[49] De Simone, C. The cut polytope and the Boolean quadric polytope. *Discrete Mathematics* **79** (1989/90), 71-75.

[50] De Simone, C., M. Diehl, M. Jünger P. Mutzel, G. Reinelt, G., Rinaldi. Exact ground states of Ising spin glasses: New experimental results with a branch and cut algorithm. *Journal of Statistical Physics* **80** (1995) pp. 487-496.

[51] de Werra, D. On some properties of the struction of a graph. *SIAM J. Algebraic Discrete Methods* **5** (1984), no. 2, pp. 239–243.

[52] Deza, M. and M. Laurent. Facets for the cut cone I. *Mathematical Programming, Ser. A*, **56** (1992) 121-160.

[53] Deza, M. and M. Laurent. Facets for the cut cone II: clique-web inequalities. *Mathematical Programming, Ser. A*, **56** (1992) 161-188.

[54] Ebenegger, C., P.L. Hammer and D. de Werra. Pseudo-Boolean functions and stability of graphs. *Annals of Discrete Mathematics* **19** (1984), 83-98.

[55] Fabian, CS., S. Rudeanu, GH. Weisz. Rezolvarea problemelor de programare pseudobooleana liniara cu ajutorul unui calculator electronic. *Studii Si Cercetari* **10**, tomul 20, 1968

[56] U. Feige and M.X. Goemans. Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT. Proceedings of the Third Israel Symposium on Theory of Computing and Systems, Tel Aviv, Israel, 182–189, 1995.

[57] Feige, U., S. Goldwasser, L. Lovasz, and S. Safra. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM* **43** (1996) pp. 268-292.

[58] Fisher, M.L., G.L. Nemhauser and L.A. Wolsey. An analysis of approximations for maximizing submodular setfunctions - I. *Mathematical Programming* **14** (1978) 265-294.

[59] Fortet, R. L'algebre de Boole en recherche operationelle. *Cahiers du Centre de Recherche Operationelle* **1** (1959), pp. 5-36.

[60] Fortet, R. Applications de l'algebre de Boole en recherche operationelle. *Revue Francaise de Recherche Operationelle* **4** (1960), pp. 17-26.

[61] Fraenkel, A.S. and P.L. Hammer. Pseudo-Boolean functions and their graphs. *Annals of Discrete Mathematics* **20** (1984), 137-146.

[62] Gallo, G. and B. Simeone. On the supermodular knapsack problem. *Mathematical Programming Study*, **45** (1989) 295-309.

[63] Garey, M.R. and D.S. Johnson. *Computers and Intractability: An Introduction to the Theory of NP-Completeness.* (W.H. Freeman, San Francisco, 1979)

[64] Glover F. Future paths for Integer Programming and Links to Artificial Intelligence. *Computers and Operations Research*, **5** (1986) pp. 533-549.

[65] Glover F. Tabu Search - Part I. *ORSA Journal on Computing*, **1** (1989) pp. 190-206.

[66] Glover, F., B. Alidaee, C. Rego, and G. Kochenberger. One-Pass Heuristics for Large Scale Unconstrained Binary Quadratic Problems. Technical Report, HCES-09-00, September 2000, Hearin Center for Enterprise Science.

[67] Glover, F., and R.E. Woolsey. Further reduction of zero-one polynomial programs to zero-one linear programming problems. *Operations Research* **21** (1973), 156-161.

[68] Glover, F., and R.E. Woolsey. Note on converting the $0-1$ polynomial programming problems to zero-one linear programming problems. *Operations Research* **22** (1974), 180-181.

[69] M.X. Goemans and D.P. Williamson. New $\frac{3}{4}$-approximation algorithms for the maximum satisfiability problem. *SIAM J. Discr. Math.* **7** (1994), pp. 656-666.

[70] M.X. Goemans and D.P. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *Journal of the ACM* **42** (1995) pp. 1115-1145.

[71] Grötschel, M., L. Lovász, and L. Scrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica* **1** (1981) 169-197.

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

[73] Hammer, P.L. Plant location - a pseudo-Boolean approach. *Israel Journal of Technology* **6** (1968), 330-332.

[74] Hammer, P.L. Pseudo-Boolean remarks on balanced graphs. *International Series of Numerical Mathematics* **36** (1977), 69-78.

[75] Hammer, P.L. The conflict graph of a pseudo-Boolean function. Bell Laboratories, Technical Report, August 1978.

[76] Hammer, P.L., P. Hansen, P.M. Pardalos and D.J. Rader, Jr. Maximizing the product of two linear functions in $0-1$ variables. RUTCOR Research Report RRR 2-97, RUTCOR, February 1997.

[77] P.L. Hammer, P. Hansen, B. Simeone. Upper planes of quadratic 0-1 functions and stability in graphs. In: *O. Mangasarian, R.R. Meyer, S.M. Robinson (eds.): Nonlinear Programming 4* (Academic Press, New York) 1981, pp. 395-414.

[78] Hammer, P.L., P. Hansen and B; Simeone. On vertices belonging to all or to no maximum stable sets of a graph. *SIAM Journal of Algebraic and Discrete Methods* **3** (1982), 511-522.

[79] Hammer, P.L., P. Hansen and B. Simeone. Roof duality, complementation and persistency in quadratic $0-1$ optimization. *Mathematical Programming* **28** (1984), pp. 121-155.

[80] Hammer, P.L. and R. Holzman. Approximations of pseudo-Boolean functions; Applications to Game Theory. *ZOR – Methods and Models of Operations Research* **36** (1992) pp. 3-21.

[81] Hammer, P.L. and B. Kalantari. A bound on the roof duality gap. *Combinatorial Optimization*, Lecture Notes in Mathematics **1403** (1989) pp. 254-257.

[82] Hammer, P.L., T. Liebling, B. Simeone, and D. de Werra. From linear separability to unimodality: a hierarchy of pseudo-Boolean functions. ORWP 84-3, Ecole Polytechnique Federale de Lausanne. To appear in: SIAM Journal on Algebraic and Discrete Methods.

[83] Hammer, P.L., N.V.R. Mahadev, and D. de Werra. Stability in CAN-free graphs. *Journal of Combinatorial Theory (B)* **38** (1985), 23-30.

[84] Hammer, P.L., N.V.R. Mahadev, and D. de Werra. The struction of a graph: Application to CN-free graphs. *Combinatorica* **5** (1985), 141-147.

[85] Hammer, P.L. and U.N. Peled. On the maximization of a pseudo-Boolean function. *Journal of the Association for Computing Machinery* **19** (1972), 265-282.

[86] Hammer, P.L., U.N. Peled and S. Sorensen. Pseudo-Boolean functions and game theory. I. Core elements and Shapley value. *Cahiers du Centre d'Etudes de Recherche Operationnelle* **19** (1977), 159-176.

[87] Hammer, P.L., I. Rosenberg, and S. Rudeanu. On the determination of the minima of pseudo-Boolean functions. *Stud. Cerc. Mat.* **14** (1963) pp. 359-364 (in Romanian).

[88] Hammer, P.L., I. Rosenberg, and S. Rudeanu. Application of discrete linear programming to the minimization of Boolean functions. *Rev. Mat. Pures Appl.* **8** (1963) pp. 459-475.

[89] Hammer, P.L. and I. Rosenberg. Linear decomposition of a positive group-Boolean function. In: Numerische Methoden bei Optimierung, Vol. II, pp 51-62, L. Collatz and W. Wetterling (eds.). Birkhauser-Verlag, Basel-Stuttgart, 1974.

[90] Hammer, P.L. and A.A. Rubin. Some remarks on quadratic programming with $0-1$ variables. *Revue Frances d'Informatique et de Recerche Operationelle* **4** (1970) pp. 67-79.

[91] Hammer, P.L. and S. Rudeanu. *Boolean Methods in Operations Research and Related Areas.* (Springer-Verlag, Berlin, Heidelberg, New York, 1968.)

[92] Hammer, P.L. and E. Shliffer. Applications of pseudo-Boolean methods to economic problems. *Theory and Decision* **1** (1971), 296-308.

[93] Hammer, P.L. and B. Simeone. Quasimonotone boolean functions and bistellar graphs. *Annals of Discrete Mathematics* **9** (1980), 107-119.

[94] Hammer, P.L. and B. Simeone. Quadratic functions of binary variables. *Combinatorial Optimization*, Lecture Notes in Mathematics **1403** (1989) pp. 1-56.

[95] Hansen, P. Methods of nonlinear $0-1$ programming. *Annals of Discrete Mathematics* **5** (1979), pp. 53-70.

[96] Hansen, P. The Steepest Ascent Mildest Descent heuristic for combinatorial programrning. Presented at the congress on Numerical Methods in Combinatorial Optimization, Capri, March 1986.

[97] Hansen, P., M. Poggi de Aragão, C.C. Ribeiro. Boolean Query Optimization and the 0-1 Hyperbolic Sum Problem. *Annals of Mathematics and Artificial Intelligence* **1** (1990) 97-109.

[98] Hansen, P., S.H. Lu and B. Simeone. On the equivalence of paved-duality and standard linearization in nonlinear optimization. *Discr. Appl. Math.* **29** (1990) pp. 187–193.

[99] Hansen, P. and B. Simeone. A class of quadratic pseudoboolean fuctions whose maximization is reducible to a network flow problem. CORR 79-39,Dept. Comb. Optim., University of Waterloo

[100] Hansen, P. and B. Simeone. Unimodular functions. *Discrete Applied Mathematics* **14** (1986), 269-281.

[101] Håstad, J. Some optimal inapproximability results. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, El Paso, Texas, 1997, pp. 1-10.

[102] Hertz, A. Quelques utilisations de la struction. *Discrete Math.* **59** (1986), no. 1-2, 79–89.

[103] Hertz, A. Polynomially solvable cases for the maximum stable set problem. *ARIDAM VI and VII (New Brunswick, NJ, 1991/1992). Discrete Appl. Math.* **60** (1995) pp. 195–210.

[104] Hertz, A. On the use of Boolean methods for the computation of the stability number. *Discrete Applied Mathematics* **76** (1997), no. 1-3, 183–203.

[105] Hillier, F.S. *The evaluation of risky interrelated investments*. North-Holland, Amsterdam, 1969.

[106] Hoke, K.W., and M.F. Troyon. The struction algorithm for the maximum stable set problem revisited. *Discrete Mathematics* **131** (1994) pp. 105-113.

[107] Hopfield, J.J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the USA* (Biophysics) **79**, 2554-2558.

[108] Iwata, S., L. Fleischer, and S. Fujishige. A Strongly Polynomial-Time Algorithm for Minimizing Submodular Functions. *Proceedings of the 32nd ACM Symposium on Theory of Computing*, May, 2000.

[109] Johnson, D.S. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences* **9** (1974), 256-278.

[110] Johnson, D.S., C.H. Papadimitriou and M. Yannakakis. How easy is local search. *Proceedings of the 26th Annual Symposium on the Foundations of Computer Science*, 39-42, 1985.

[111] Jurisch, B., W. Kubiak, and J. Józefowska. Algorithms for minclique scheduling problems. *Discrete Applied Mathematics* **72** (1997), 115-139.

[112] Kalantari, B. Quadratic functions with exponential number of local maxima. *Operations Research Letters* **5** (1986), 47-49.

[113] Karloff, H., and U. Zwick. A 7/8-approximation algorithm for MAX 3SAT? In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science*, Miami Beach, Florida, 1997, pp. 406-415.

[114] Karp, R.M. Reducibility among combinatorial problems, In: *Complexity of Computer Computation.* pp. 85-104. R.G. Miller and J.W. Thatcher (eds.), Plenum Press, New York, 1972.

[115] Kirkpatrick, S., C. D. Gelatt,Jr. and M. P. Vecchi. Optimization by simulated annealing. *Science*, **220** (1983) pp. 671-680.

[116] Kohli, R., and R. Krishnamurti. Average performance of heuristics for satisfiability. *SIAM Journal on Discrete Mathematics* **2** (1989) pp. 508-523.

[117] Korner, F. An effective branch-and-bound algorithm for Boolean quadratic optimization problems. *Z. Angew. Maht. Mech.* **65** (1985), no. 8, 392-394.

[118] Kubiak, W. New results on the completion time variance minimization. *Discrete Applied Mathematics* **58** (1995) 157-168.

[119] Kubiak, W. Minimization of ordered, symmetric half-products. Research Report, Faculty of Business Administration, Memorial University of Newfoundland, 2000.

[120] Laughhunn, D.J. Quadratic binary programming with applications to capital budgeting problems. *Operations Research* **18** (1970), 454-461.

[121] Laughhunn, D.J. and D.E. Peterson. Computational experience with capital expenditure programming models under risk. *J. Business Finance* **3** (1971), 43-48.

[122] Lawler, E. The quadratic assignment problem. *Management Science* **9** (1963) 586-599.

[123] Lieberherr, K.J. and E. Specker. Complexity of partial satisfaction. *Journal of the ACM* **28** (1981), 411-421.

[124] Lovász, L. Submodular functions and convexity. *Mathematical Programming – The State of the Art*, (A. Bachem, M. Grotschel, and B. Korte, eds.), Springer-Verlag 1983, 235-257.

[125] Lu, S.H. and A.C. Williams. Roof duality for $0-1$ nonlinear optimization. *Math. Prog.* **37** (1987), 357-360.

[126] Mattis, D.C. Solvable spin systems with random interaction. *Phys. Letters* **56** (1976) 412.

[127] Monien , B. and E. Speckenmeyer. Solving satisfiability in less than 2^n steps. *Discrete Applied Mathematics* **10** (1985) 287-295.

[128] Nagih, A. *Sur la résolution des programmes fractionnaires en variables* $0-1$. Ph.D. thesis, Université Paris 13, 1996.

[129] Nemhauser, G.L. and L.E. Trotter. Vertex packing: Structural properties and algorithms. *Mathematical Programming* **8** (1978), 232-248.

[130] Nemhauser, G. L. and L. Wolsey. Maximizing submodular set functions: formulations and analysis of algorithms. In *Studies of Graphs and Discrete Programming*, North-Holland, Amsterdam, 1981, pp. 279-301.

[131] Nesterov, Y. and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. SIAM Studies in Applied Mathematics, 1994.

[132] Neumann, J. von and O. Morgenstern. *Theory of games and economic behavior*. Princeton, Princeton University Press, 1944.

[133] Nieminen, J. A linear pseudo-Boolean viewpoint on matching and other central concepts in graph theory. *Zastos. Mat.* **14** (1974), 365-369

[134] Owen, G. *Game Theory*. Academic Press, New York, 1982.

[135] Padberg, M. The Boolean quadric polytope: Some characteristics, facets and relatives. *Mathematical Programming* **45** (1989) pp. 139-172.

[136] Papadimitriou, C.H., and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences* **43** (1991) pp. 425-440.

[137] Papaioannou, S.G. Optimal test generation in combinational networks by pseudo-Boolean programming. *IEEE Transactions on Computers* **26** (1977), 553-560.

[138] Personnaz, L., I. Guyon, and G. Dreyfus. Collective computational properties of neural networks: new learning mechanisms. *Physical Review A*, **34**, (1986), 4217-4228.

[139] Picard, J.C. and M. Queyranne. On the integer-valued variables in the linear vertex packing problem. *Mathematical Programming* **12** (1977), 97-101.

[140] Picard, J.C. and H.D. Ratliff. Minimum cuts and related problems. *Networks* **5** (1975), 357-370.

[141] Picard, J.C. and H.D. Ratliff. A cut approach to the rectilinear facility location problem. *Operations Research* **26** (1978), 422-433.

[142] Raghavan, P. Probabilistic construction of deterministic algorithms: Approximating packing integer programs. *J. Comput. System. Sci.* **37** (1988), pp. 130-143.

[143] Raghavan, P. and C. Thompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica* **7** (1987), pp. 365-374.

[144] Ranyard, R.H. An algorithm for maximum likelihood ranking and Slater's i from paired comparisions. *British Journal of Mathematical and Statistical Psychology* **29** (1976), 242-248.

[145] Rao, M.R. Cluster analysis and mathematical programming. *Journal of the American Statistical Association* **66** (1971), 622-626.

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

[147] Robillard, P. $(0,1)$ hyperbolic programming problems. *Naval Research Logistic Quarterly* **18** (1971) pp. 47-57.

[148] Rosenberg, I.G. $0-1$ optimization and non-linear programming. *Revue Française d'Automatique, d'Informatique et de Recherche Opérationnelle (Série Bleue)* **2** (1972), 95-97.

[149] Rosenberg, I.G. Reduction of bivalent maximization to the quadratic case. *Cahiers du Centre d'Etudes de Recherche Operationnelle* **17** (1975), 71-74.

[150] Rosenberg, I.G. Linear decompositions of positive real function of binary arguments. *Utilitas Math.* **17** (1980), 17-34.

[151] Rödl, V. and C.A. Tovey. Multiple optima in local search. *Journal of Algorithms* **8**(2) (1987) pp. 250–259.

[152] Schrijver, A. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *J. Combinatorial Theory (B)*, submitted, 2000.

[153] Shaik, I.P. An Optimization Approach to Robust Delay-Fault Built-In Testing. Ph.D. in Electrical Engineering, Rutgers University, 1996.

[154] Shapley, L.S. A value for n-person games. In: Kuhn H.W., Tucker A.W. (eds) *Contributions to the theory of games II.* Princeton University Press, Princeton, 1953.

[155] Shepanik, I. Use of pseudo-boolean preference functions in metagame analysis. Ph. D. Thesis, Dept. of system Design, 1973

[156] Simeone, B. Quadratic 0-1 programming, Boolean functions and graphs. Ph.D. thesis in COMBINATORICS AND OPTIMIZATION, Waterloo, 1979

[157] Tovey, C.A. Hill climbing with multiple local optima. *SIAM Journal on Algebraic and Discrete Methods* **6** (1985), 384-393.

[158] Tovey, C.A. Low order polynomial bounds on the expected performance of local improvement algorithms. *Mathematical Programming* **35** (1986), 193-224.

[159] Trevisan, L., G.B. Sorkin, M. Sudan, and D.P. Williamson. Gadgets, approximation, and linear programming (extended abstract). In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, Burlington, Vermont, 1996, pp. 617-626.

[160] Trubin, V. On a method of solution of integer linear programming problems of special kind. *Soviet Mathematics Doklady* **10** (1969) 1544-1546.

[161] Tuza, Z. Maximum cuts: improvements and local algorithmic analogues of the Edwards-Erdös inequality. *Discrete Mathematics* **194** (1999) pp. 39-58.

[162] Warszawski, A. Pseudo-Boolean solutions to multidimensional location problems. *Operations Research* **22**, 1081-1085, 1974

[163] Watters, L.G. Reduction of integer polynomial problems to zero-one linear programming problems. *Operations Research* **15** (1967), 1171-1174.

[164] Welsh, D.J.A. *Matroid theory.* Academic Press, London, 1976.

[165] Weingartner, H.M. Capital budgeting of interrelated projects: survey and synthesis. *Management Science* **12** (1966), 485-516.

[166] M. Yannakakis. On the approximation of maximum satisfiability. *Journal of Algorithms* **17** (1994) pp. 475-502.

[167] Young, A.P. Spin glasses. *Journal of Statistical Physics* **34** (1984), 871-881.

[168] U. Zwick. Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint. *Proc. of 9th SODA* (1998), pp. 201-210.