R UTCOR
R ESEARCH
R EPORT

Preprocessing of Unconstrained Quadratic Binary Optimization

Endre Boros[a]   Peter L. Hammer[a]
Gabriel Tavares[a]

RRR 10–2006, April, 2006

[a]RUTCOR, Rutgers University, 640 Bartholomew Road, Piscataway, NJ 08854–8003, USA. E-Mail: {boros,hammer,gtavares}@rutcor.rutgers.edu

PREPROCESSING OF UNCONSTRAINED QUADRATIC BINARY OPTIMIZATION

Endre Boros   Peter L. Hammer   Gabriel Tavares

Abstract.

We propose several efficient preprocessing techniques for Unconstrained Quadratic Binary Optimization (QUBO), including the direct use of enhanced versions of known basic techniques (e.g., implications derived from first and second order derivatives and from roof–duality), and the integrated use (e.g., by probing and by consensus) of the basic techniques. The application of these techniques is implemented using a natural network flow model of QUBO. The use of the proposed preprocessing techniques provides: (i) a lower bound of the minimum of the objective function, (ii) the values of some of the variables in some or every optimum, (iii) binary relations (equations, inequalities, or non-equalities) between the values of certain pairs of variables in some or every optimum, and (iv) the decomposition (if possible) of the original problem into several smaller pairwise independent QUBO problems. Extensive computational experience showing the efficiency of the proposed techniques is presented. Substantial problem simplifications, improving considerably the existing techniques, are demonstrated on benchmark problems as well as on randomly generated ones. In particular, it is shown that 100% data reduction is achieved using the proposed preprocessing techniques for MAX-CUT graphs derived from VLSI design, MAX-Clique in c-fat graphs derived from fault diagnosis, and minimum vertex cover problems in random planar graphs of up to 500 000 vertices.

Contents

1 Introduction

Let n be a positive integer, $\mathbb{B} = \{0, 1\}$, and let $\mathbb{R}$ denote the set of reals. A function $f : \mathbb{B}^n \mapsto \mathbb{R}$ is called a *pseudo–Boolean function*. Every pseudo–Boolean function f is known (see [41]) to have a unique multilinear polynomial representation, the degree of which is called the *degree* of f. In this paper we shall consider the problem of *Quadratic Unconstrained Binary Optimization* (QUBO)

$$\min_{\mathbf{x} \in \mathbb{B}^n} f(\mathbf{x}), \tag{1}$$

concerning the minimization of a quadratic pseudo–Boolean function f given by

$$f(x_1, \cdots, x_n) = c_0 + \sum_{j=1}^{n} c_i x_i + \sum_{1 \leqslant i < j \leqslant n} c_{ij} x_i x_j, \tag{2}$$

where c_0, c_i for $i = 1, \cdots, n$ and c_{ij} for $1 \leqslant i < j \leqslant n$ are given reals.

Pseudo–Boolean functions and the optimization problem (1) were introduced in [41], and have since been shown to be a common model for a wide variety of discrete optimization problems arising in VLSI design ([9, 17, 19, 24, 48, 52, 78]), statistical mechanics ([9, 80, 71]), reliability theory and statistics ([67, 74, 75]), manufacturing ([2, 7, 8, 27, 53]), economics and finance ([42, 45, 54, 55, 57]), operations research and management science ([30, 37, 73, 84, 85]), and in numerous algorithmic problems of graph theory ([29, 38, 41, 69, 70, 72, 79]), etc.

Several exact algorithms (see e.g., [10, 14, 25, 80, 36, 44, 49, 62, 68, 86]) as well as numerous heuristic procedures (see e.g., [3, 11, 28, 33, 34, 35, 50, 51, 59, 60, 63, 64, 65, 66]) were proposed in the literature for the solution of (1), which is generally an NP-hard optimization problem.

In this paper we propose a family of *preprocessing* techniques, based on several computationally efficient transformations of problem (1), and aimed at simplifying it and possibly decomposing it into smaller problems of the same type. More precisely, the proposed preprocessing involves a series of transformations of the quadratic pseudo–Boolean objective function f, including:

(i) The fixation of some of the variables at values which must hold at every optimum, and the enforcement of certain binary relations (e.g., equations, inequalities, or non-equalities) between the values of certain pairs of variables, which must hold in every optimum;

(ii) The fixation of some of the variables and the enforcement of some binary relations between certain pairs of variables, which must hold in at least one optimal solution of QUBO;

(iii) The possible decomposition of the problem into several smaller QUBO problems involving pairwise disjoint subsets of the original variables.

As a result, we obtain a constant K and quadratic pseudo–Boolean functions $f_r : \mathbb{B}^{S_r} \mapsto \mathbb{R}$, $r = 1, \cdots, c$, where the sets of indices $S_r \subseteq \{1, \cdots, n\}$, $r = 1, \cdots, c$ are pairwise disjoint, such that

$$\min_{\mathbf{x} \in \mathbb{B}^n} f(\mathbf{x}) = K + \sum_{r=1}^{c} \min_{\mathbf{y} \in \mathbb{B}^{S_r}} f_r(\mathbf{y}).$$

The proposed method has several key ingredients. The first ingredient is the *representation* of a quadratic pseudo–Boolean function by means of a network (see [18, 20, 82]). This representation allows an efficient derivation via network flow computations of a lower bound on the minimum of the function and allows the identification of variables whose values in the optimum can be easily predicted, as well as of other information which makes the simplification of the problem possible. The second ingredient is the *methodology* primarily based on the theory of *roof–duality*, introduced in [40], combined with the use of first and second order derivatives and co-derivatives. This methodology provides an effective lower bound for the minimum of a quadratic pseudo–Boolean function, as well as information about the values of a subset of the variables in the optimum (so called *linear persistencies*), and about binary relations which must hold between certain pairs of variables in the optimum (so called *quadratic persistencies*). An additional component of the methodology is the decomposition of a large problem into several smaller ones, which can be derived at a low extra cost by combining the conclusions of roof–duality with those offered by the network representation of the problem (see Section 3.3.3). The third ingredient of the proposed method consists of the *integration* of conclusions obtained from subproblems associated to the original problem. This integration is realized by combining the conclusions of probing (i.e. multiple application of roof–duality to naturally associated subproblems), and of Boolean consensus (i.e. the exhaustive expansion of the detected linear and quadratic persistencies).

We present an extensive computational evaluation of the proposed preprocessing method in Section 7, using various benchmark sets and randomly generated test problems of various types, involving thousands of variables, as described in Section 6. Our experience shows that for dense problems the proposed preprocessing technique is less effective as the size of the problems grows. It is demonstrated on numerous publicly available test problems that for relatively sparse problems, including in particular certain classes of structured problems, the proposed preprocessing method achieves substantial reductions in size at a very low computational cost. For instance, applying the method to QUBO problems corresponding to vertex cover problems in planar graphs involving up to $500\,000$ nodes, lead to the optimal fixation of 100% of the variables, i.e. to the exact solution of the problem, in every single instance considered (see Section 8).

2   Definitions and Notations

Let $\mathbf{V} = \{1, \cdots, n\}$, and for a subset $S \subseteq \mathbf{V}$, denote by $\mathbf{1}^S \in \mathbb{B}^{\mathbf{V}}$ its characteristic vector, i.e. $\mathbf{1}_j^S = 1$ for $j \in S$ and $\mathbf{1}_j^S = 0$ for $j \notin S$.

We shall consider quadratic pseudo–Boolean functions in n binary variables, as in (2). We shall use $\mathbf{x} = (x_1, \cdots, x_n)$ to denote a binary vector, or the vector of variables on which the pseudo–Boolean function depend. The *complement* of a binary variable x_i is denoted by $\overline{x}_i \overset{\text{def}}{=} 1 - x_i$. Variables and their complements are called *literals*, and we denote by $\mathbf{L} \overset{\text{def}}{=} \{x_1, \cdots, x_n, \overline{x}_1, \cdots, \overline{x}_n\}$ the set of literals. In the sequel, the letters u, v and w refer to literals, while bold face letters $\mathbf{x}$, $\mathbf{y}$, etc., denote vectors.

Let us note that each literal $u \in \mathbf{L}$ itself can be considered as a simple pseudo–Boolean function, since for a particular binary assignment $\mathbf{y} \in \mathbb{B}^{\mathbf{V}}$ to the variables, we have $u(\mathbf{y}) = y_j \in \mathbb{B}$ if $u = x_j$ and $u(\mathbf{y}) = 1 - y_j \in \mathbb{B}$ if $u = \overline{x}_j$.

2.1   Posiforms

Pseudo–Boolean functions can also be represented as *posiforms*, i.e. multilinear polynomial expressions depending on all the literals

$$\phi(\mathbf{x}) = \sum_{T \subseteq \mathbf{L}} a_T \prod_{u \in T} u, \tag{3}$$

where $a_T \geqslant 0$ whenever $T \neq \emptyset$. If $\{u, \overline{u}\} \subseteq T$ for some $u \in \mathbf{L}$, then $\prod_{u \in T} u$ is identically zero over $\mathbb{B}^n$ and, therefore we shall assume that $a_T = 0$. Similarly to the case of polynomial expressions, the size of the largest subset $T \subseteq \mathbf{L}$ for which $a_T \neq 0$ is called the *degree* of the posiform ϕ, and is denoted by $\deg(\phi)$. The posiform ϕ is called *quadratic* (*linear, cubic*, etc.) if $\deg(\phi) \leqslant 2$ (1, 3, etc.) Note that a quadratic pseudo–Boolean function may also have representations by a posiform of degree higher than two. For instance, the quadratic function $f(x_1, x_2, x_3) = 1 - x_1 - x_2 - x_3 + x_1 x_2 + x_1 x_3 + x_2 x_3$ can also be represented by the cubic posiform $\phi = x_1 x_2 x_3 + \overline{x}_1 \overline{x}_2 \overline{x}_3$.

In this paper we shall be concerned only by quadratic posiforms of quadratic pseudo–Boolean functions. Let us note that any quadratic pseudo–Boolean function can be transformed to a quadratic posiform in time linear in its size, where the *size* of a pseudo–Boolean function e.g., as given by (2), or of a posiform e.g., as given in (3), is the number of its nonzero terms. To do this, in each quadratic term $c_{ij} x_i x_j$ in (2) having $c_{ij} < 0$, $i < j$ we replace x_j by $1 - \overline{x}_j$, and use the resulting transformation

$$c_{ij} x_i x_j = c_{ij} x_i (1 - \overline{x}_j) = c_{ij} x_i + (-c_{ij}) x_i \overline{x}_j.$$

If we do this for all quadratic terms of (2) with negative coefficients we find

$$f(\mathbf{x}) = c_0 + \sum_{j=1}^{n} c_j' x_j + \sum_{\substack{1 \leqslant i < j \leqslant n \\ c_{ij} > 0}} c_{ij} x_i x_j + \sum_{\substack{1 \leqslant i < j \leqslant n \\ c_{ij} < 0}} (-c_{ij}) x_i \overline{x}_j,$$

where

$$c_i' = c_i + \sum_{\substack{i < j \leqslant n \\ c_{ij} < 0}} c_{ij}$$

for $i = 1, \cdots, n$. Using now the same substitution $x_j = 1 - \overline{x}_j$, we rewrite those linear terms $c'_j x_j$ for which $c'_j < 0$ as

$$c'_j x_j = c'_j + (-c'_j)\overline{x}_j,$$

yielding at the end the quadratic posiform

$$f(\mathbf{x}) = c'_0 + \sum_{\substack{j=1 \\ c'_j > 0}}^{n} c'_j x_j + \sum_{\substack{j=1 \\ c'_j < 0}}^{n} (-c'_j)\overline{x}_j + \sum_{\substack{1 \leqslant i < j \leqslant n \\ c_{ij} > 0}} c_{ij} x_i x_j + \sum_{\substack{1 \leqslant i < j \leqslant n \\ c_{ij} < 0}} (-c_{ij}) x_i \overline{x}_j,$$

where

$$c'_0 = c_0 + \sum_{\substack{j=1 \\ c'_j < 0}}^{n} c'_j.$$

It is also easy to see that from a quadratic posiform we can also derive the unique polynomial form of the represented quadratic pseudo–Boolean function in linear time of its *size*.

Thus, we can assume without any loss of generality that the objective function of problem (1) is given as a quadratic posiform. To simplify notation, we shall assume in the sequel that the quadratic pseudo–Boolean function f is given (also) as a posiform

$$\phi(\mathbf{x}) = a_0 + \sum_{u \in \mathbf{L}} a_u u + \sum_{u,v \in \mathbf{L}} a_{uv} uv, \tag{4}$$

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

In the sequel, the letters f and g will denote pseudo–Boolean functions as well as their unique multi-linear polynomial expressions, while the Greek letters ϕ and ψ will denote posiforms.

2.2   Persistency

The concept of persistency was introduced in [40], referring to assignments which are valid at some or all optimal solutions of an optimization problem. More precisely, we say that for some index $j \in \mathbf{V}$ and binary value $\alpha \in \mathbb{B}$ *strong* (respectively, *weak*) *persistency* holds for problem (1) if $x_j = \alpha$ holds for all (respectively, for some) minimizing points of f. We shall equivalently say that the assignment $x_j = \alpha$ is strongly (respectively, weakly) persistent.

It will also be very useful to consider the somewhat more specific definition of *autarkies*. For a binary vector $\mathbf{x} \in \mathbb{B}^{\mathbf{V}}$ and subset $S \subseteq \mathbf{V}$ let us denote by $\mathbf{x}[S] \in \mathbb{B}^S$ the partial assignment (or subvector) defined as $\mathbf{x}[S] = (x_j | j \in S)$. For a subset $S \subseteq \mathbf{V}$, a partial assignment $\mathbf{y} \in \mathbb{B}^S$, and a binary vector $\mathbf{x} \in \mathbb{B}^{\mathbf{V}}$, let us denote by $\mathbf{z} = \mathbf{x}[S \leftarrow \mathbf{y}]$ the vector obtained from $\mathbf{x}$ by switching its S-components to $\mathbf{y}$, i.e. $z_j = y_j$ for $j \in S$, and $z_j = x_j$ for $j \notin S$.

Given a subset $S \subseteq \mathbf{V}$ we say that a partial assignment $\mathbf{y} \in \mathbb{B}^S$ is a *strong* (respectively, *weak*) *autarky* for f if the inequality $f(\mathbf{x}[S \leftarrow \mathbf{y}]) < f(\mathbf{x})$ (respectively, $f(\mathbf{x}[S \leftarrow \mathbf{y}]) \leqslant f(\mathbf{x})$) holds for all binary vectors $\mathbf{x} \in \mathbb{B}^{\mathbf{V}}$, $\mathbf{x}[S] \neq \mathbf{y}$.

It is easy to see that if a partial assignment $\mathbf{y} \in \mathbb{B}^S$ is a strong (respectively, weak) autarky for f, then the assignments $x_j = y_j$ for all $j \in S$ are strong (respectively, weak) persistencies for (1). Note also that while persistency is a property of individual assignments to variables, autarky is a property of partial assignments. For instance, if $\mathbf{y} \in \mathbb{B}^{\mathbf{V}}$ is a minimizing point of f, then it is also a weak autarky for f (strong if $\mathbf{y}$ is a unique minimum of f), though obviously not all partial assignments are autarkies of an arbitrary quadratic pseudo–Boolean function.

In this paper we also extend the notion of persistency for binary relations, which we call *quadratic persistencies*. Given two literals $u, v \in \mathbf{L}$, we say that the inequality $u \leqslant v$ is *strongly* (respectively, *weakly*) *persistent* for problem (1), if $u(\mathbf{y}) \leqslant v(\mathbf{y})$ holds for all (respectively, for some) optimal solutions $\mathbf{y} \in \mathbb{B}^{\mathbf{V}}$ of (1).

2.3   Implication network

Following [16, 18, 20, 82] we associate to a quadratic posiform ϕ given in (3) a capacitated network G_ϕ with node set N and with nonnegative capacities $\gamma_{uv} \geqslant 0$ for all $u, v \in N$, $u \neq v$.

We shall assume for the sake of simplicity that all the possible ordered pairs of nodes represent arcs of the network, some of which may have capacity 0. The complete set of arcs will be sometimes denoted by A. This assumption will simplify the statement of several of the properties which we shall need later, without limiting generality, since the complexity of computations to be performed in such networks will depend anyway only on the number of those arcs which have a positive capacity.

In order to define this network, let us first introduce an additional variable x_0 for which we shall assume the constant assignment $x_0 = 1$. Using this new variable (and the assumption that it is always equal to 1), we can rewrite (3) as $\phi(\mathbf{x}) - a_0 = \sum_{u \in \mathbf{L}} a_u x_0 u + \sum_{u,v \in \mathbf{L}} a_{uv} uv$, obtaining a *homogeneous* quadratic multilinear polynomial expression of $\phi - a_0$ with non-negative coefficients. To simplify notation, we introduce $N = \mathbf{L} \cup \{x_0, \overline{x}_0\}$ as the set of nodes of G_ϕ, and write the above homogeneous form as

$$\phi(\mathbf{x}) - a_0 = \sum_{u,v \in N, u \neq v} a_{uv} uv, \tag{5}$$

where we assume that the indices of the coefficients are sets of two literals (and not ordered pairs of literals). For the purpose of network definition, we also assume that (5) involves all pairs $u, v \in N$, $u \neq v$, in which we have $a_{uv} \geqslant 0$ for all terms with the possibility that $a_{uv} = 0$ for some of them.

The capacities of the arcs in G_ϕ are defined by

$$\gamma_{u\overline{v}} = \gamma_{v\overline{u}} = \frac{1}{2} a_{uv}$$

for all $u, v \in N$, $u \neq v$.

Example 1. *Consider the quadratic posiform ϕ given by*

$$
\begin{aligned}
\phi \;=\; & -8 + 2x_1 + 2\overline{x}_3 + 4\overline{x}_4 + 2\overline{x}_5 + 2x_1x_3 + 2x_1x_5 + \overline{x}_1x_2 + 2\overline{x}_1x_4 \\
& + x_2x_3 + \overline{x}_2x_4 + \overline{x}_2x_5 + 2x_3x_4 + 2\overline{x}_3x_5 + 2x_4x_5.
\end{aligned}
$$

Then, its homogeneous representation is

$$
\begin{aligned}
\phi + 8 \;=\; & 2x_0x_1 + 2x_0\overline{x}_3 + 4x_0\overline{x}_4 + 2x_0\overline{x}_5 + 2x_1x_3 + 2x_1x_5 + \overline{x}_1x_2 + 2\overline{x}_1x_4 \\
& + x_2x_3 + \overline{x}_2x_4 + \overline{x}_2x_5 + 2x_3x_4 + 2\overline{x}_3x_5 + 2x_4x_5.
\end{aligned}
$$

and the associated network G_ϕ is shown in Figure 1.
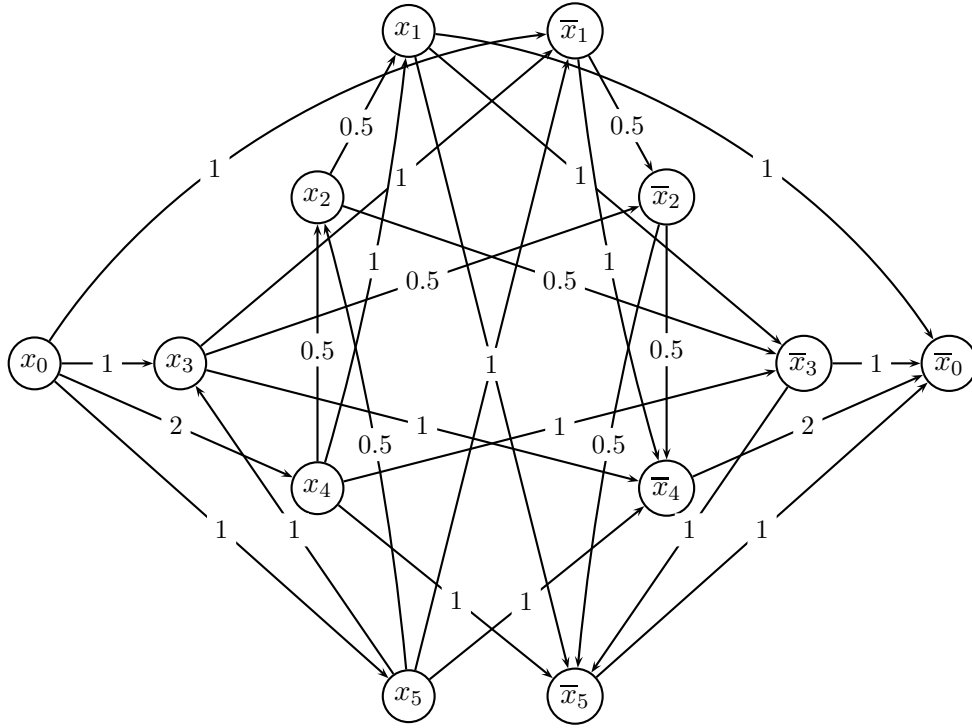
Figure 1: The network G_ϕ corresponding to the posiform ϕ of Example 1. We indicate only those arcs which have positive capacities.

Let us note that a network G_ϕ associated in this way to a quadratic posiform ϕ has a special symmetry, namely that we have

$$
\gamma_{uv} = \gamma_{\overline{v}\overline{u}} \quad \text{for all pairs of literals} \;\; u, v \in N, \; u \neq v. \tag{6}
$$

We shall call such a pair (G_ϕ, γ) an *implication network*. To justify this term, observe that if $a_{uv}uv$ is a term of ϕ then either it contributes a_{uv} to the value of ϕ, or we must have $u \leqslant \overline{v}$ (and equivalently $v \leqslant \overline{u}$). Similarly defined graphs, called *implication graphs* were associated to quadratic disjunctive normal forms in [4].

We remark here that given an arbitrary capacitated network G with capacity function γ, we can obtain an implication network $(G, \tilde{\gamma})$ from it by defining

$$\tilde{\gamma}_{uv} = \tilde{\gamma}_{\bar{v}\bar{u}} = \frac{\gamma_{uv} + \gamma_{\bar{v}\bar{u}}}{2},$$

for all pairs of literals $u, v \in N$, $u \neq v$.

Let us also note that given an arbitrary capacitated network (G, γ) on node set N, we can associate to it a quadratic posiform by defining

$$\psi_{(G,\gamma)} = \sum_{v \in \mathbf{L}} \gamma_{x_0,v} \bar{v} \; + \; \sum_{v \in \mathbf{L}} \gamma_{v,\bar{x}_0} v \; + \; \sum_{u,v \in \mathbf{L}, u \neq v} \gamma_{uv} u \bar{v}, \tag{7}$$

and observe that if $\phi = \psi_{(G,\gamma)}$, then G_ϕ is the implication network $(G, \tilde{\gamma})$, the symmetrized version of (G, γ) as defined above.

Thus, we have a one-to-one correspondence between implication networks and quadratic posiforms. As we shall see later, there is a much deeper connection between these objects.

Given a network G on node set N, we assume in the sequel for the sake of simplicity that $A = A(G)$ denotes the set of arcs of the network G, and that it consists of all pairs $u, v \in N$, $u \neq v$.

To be able to state further connections between quadratic posiforms and implication networks, we need to recall first that in network optimization theory, a mapping $\varphi : A \mapsto \mathbb{R}_+$ is called a *feasible flow* from source $x_0 \in N$ to sink $\bar{x}_0 \in N$, in a capacitated network G on node set N, with capacity function $\gamma \in \mathbb{R}_+^A$, if it satisfies the following constraints:

$$0 \leqslant \varphi(u, v) \leqslant \gamma_{uv} \qquad \text{for all arcs } (u, v) \in A$$

$$\sum_{(u,v) \in A} \varphi(u, v) \; = \; \sum_{(v,w) \in A} \varphi(v, w) \quad \text{for all nodes } v \in N \setminus \{x_0, \bar{x}_0\} . \tag{8}$$

The value of a feasible flow φ is defined as

$$v(\varphi) \; = \; \sum_{(x_0,v) \in A} \varphi(x_0, v) . \tag{9}$$

We shall denote by $v(G, \gamma)$ the maximum value of a feasible flow in the network (G, γ), i.e.

$$v(G, \gamma) \; = \; \max \{ v(\varphi) \mid \varphi \text{ satisfies (8)} \} .$$

Finally, the *residual network* is defined by the capacity function

$$\gamma_{uv}^\varphi = \gamma_{uv} - \varphi(u, v) + \varphi(v, u) \tag{10}$$

for all $(u, v) \in A$.

Let us note that if φ is a feasible flow, then $\gamma_{uv}^\varphi \geqslant 0$ for all $(u, v) \in A$, and hence $\psi_{(G,\gamma^\varphi)}$, as defined in (7), is a posiform.

3 Basic Preprocessing Tools

3.1 First order derivatives

The *i–th first order partial derivative* ($i \in \mathbf{V}$) of a quadratic pseudo–Boolean function f is given by

$$\Delta_i \left(x_1, \cdots, x_n\right) = c_i + \sum_{j=1}^{i-1} c_{ij} x_j + \sum_{j=i+1}^{n} c_{ij} x_j.$$

It is easy to show that in every local optimum,

Proposition 1.

(i) If $\Delta_i \left(\mathbf{x}\right) \leqslant 0$ for all $\mathbf{x} \in \mathbb{B}^{\mathbf{V}}$, then $x_i = 1$ is a weak persistency;

(ii) If $\Delta_i \left(\mathbf{x}\right) \geqslant 0$ for all $\mathbf{x} \in \mathbb{B}^{\mathbf{V}}$, then $x_i = 0$ is a weak persistency.

If $\Delta_i \left(\mathbf{x}\right)$ is strictly negative, respectively strictly positive, the above implications represent strong persistencies.

These simple relations have been already noticed in Hammer and Rudeanu [41], and represent an essential component of even the most recent work on preprocessing (see [6]).

Hammer, Hansen and Simeone [40] have shown that those strong persistencies which can be obtained from the analysis of partial derivatives, can also be obtained by roof–duality (a tool to be described in the next subsection). Moreover, roof–duality is a stronger preprocessing technique, as shown in [40], where an example is presented displaying persistencies found by roof–duality, but not following from the analysis of the signs of partial derivatives.

In view of these results, the preprocessing algorithm to be described in Section 5, which will exploit heavily roof–duality, will not explicitly include an analysis of the signs of partial derivatives, since the conclusions derivable from such an analysis will be automatically included among those provided by roof–duality.

3.2 Second order derivatives and co–derivatives

A natural generalization of the concept of the first order derivative allows us to establish some persistent binary relations to hold among the values taken in the optimum by certain pairs of variables.

Hammer and Hansen [39] called the linear function $\Delta_i \left(\mathbf{x}\right) - \Delta_j \left(\mathbf{x}\right) + \left(x_i - x_j\right) c_{ij}$ the $(i, j)th$ *second order derivative* of f and denoted it by Δ_{ij}. It can be seen that the linear function $\Delta_i \left(\mathbf{x}\right) + \Delta_j \left(\mathbf{x}\right) + c_{ij} \left(1 - \left(x_i + x_j\right)\right)$ has a similarly important role as Δ_{ij}; it will be called $(i, j)th$ *second order co-derivative* and will be denoted by ∇_{ij}. With these notations,

$$\begin{aligned}
\Delta_{ij} \left(\mathbf{x}\right) &= f \left(\mathbf{x} \left[\{i, j\} \leftarrow (1, 0)\right]\right) - f \left(\mathbf{x} \left[\{i, j\} \leftarrow (0, 1)\right]\right) \\
&\text{and} \\
\nabla_{ij} \left(\mathbf{x}\right) &= f \left(\mathbf{x} \left[\{i, j\} \leftarrow (1, 1)\right]\right) - f \left(\mathbf{x} \left[\{i, j\} \leftarrow (0, 0)\right]\right),
\end{aligned}$$

i.e. evaluate the effect of simultaneously changing the values of x_i and x_j, while keeping the values of the other variables unchanged.

The following statement identifies a series of weakly, respectively strongly, persistent binary (order) relations of the form $x_i x_j = 0$ (i.e. $x_i \leqslant \overline{x}_j$), $x_i \overline{x}_j = 0$ (i.e. $x_i \leqslant x_j$), $\overline{x}_i x_j = 0$ (i.e. $x_i \geqslant x_j$), and $\overline{x}_i \overline{x}_j = 0$ (i.e. $\overline{x}_i \leqslant x_j$), between certain pairs (x_i, x_j) of variables.

Proposition 2. *Let* $f(x_1, \cdots, x_n)$ *be a quadratic pseudo–Boolean function, and let* x_i *and* x_j *be two of its variables. Then,*

(i) If $\nabla_{ij}(\mathbf{x}) \geqslant 0$ *for every* $\mathbf{x} \in \mathbb{B}^{\mathbf{V}}$ *then* $x_i x_j = 0$ *is a weak persistency;*

(ii) If $\nabla_{ij}(\mathbf{x}) \leqslant 0$ *for every* $\mathbf{x} \in \mathbb{B}^{\mathbf{V}}$ *then* $\overline{x}_i \overline{x}_j = 0$ *is a weak persistency;*

(iii) If $\Delta_{ij}(\mathbf{x}) \geqslant 0$ *for every* $\mathbf{x} \in \mathbb{B}^{\mathbf{V}}$ *then* $x_i \overline{x}_j = 0$ *is a weak persistency;*

(iv) If $\Delta_{ij}(\mathbf{x}) \leqslant 0$ *for every* $\mathbf{x} \in \mathbb{B}^{\mathbf{V}}$ *then* $\overline{x}_i x_j = 0$ *is a weak persistency.*

If in any of the implications above the left hand side inequality is strict, then the corresponding persistency is strong.

3.3 Roof–duality

Let us recall first that the *roof–dual bound* proposed in [40], can be determined efficiently by maximum flow computations in the implication network G_ϕ ([16, 18, 20]).

Proposition 3. *For any quadratic posiform* ϕ *given by (4) and any feasible flow* φ *in the corresponding implication network* G_ϕ *the equality*

$$\phi = a_0 + v(\varphi) + \psi_{\left(G_\phi, \gamma^\varphi\right)} \tag{11}$$

holds, where the right hand side is a quadratic posiform. Therefore,

$$a_0 + v(G, \gamma) \leqslant \min_{\mathbf{x} \in \mathbb{B}^{\mathbf{V}}} \phi(\mathbf{x}). \tag{12}$$

Incidentally, for any network (G, γ) we have $v(G, \gamma) \leqslant v(G, \tilde{\gamma})$, thus we get the best lower bound in (12) exactly for the implication network form of a quadratic posiform. In fact the bound in (12) was shown in [16, 18] to be the same as the roof–dual value of ϕ, introduced in [40].

3.3.1 Strong persistency

Let us observe next that if φ is a feasible flow in an implication network (G, γ) then in view of conditions (6) and of the equality

$$v(\tilde{\varphi}) = v(\varphi),$$

the symmetrized flow $\tilde{\varphi}$ defined by

$$\tilde{\varphi}(u,v) = \tilde{\varphi}(\overline{v}, \overline{u}) = \frac{\varphi(u,v) + \varphi(\overline{v}, \overline{u})}{2} \text{ for all } u, v \in N, \ u \neq v$$

is also a feasible flow in (G, γ).

This implies that in any implication network, among the maximum flows there always exists a *symmetric* one, for which $\varphi = \tilde{\varphi}$ holds.

Let us consider therefore a symmetric maximum flow φ in the implication network (G_ϕ, γ), and let $\psi = \psi_{(G_\phi, \gamma^\varphi)}$. As we observed above, the corresponding implication network G_ψ is exactly the residual network of G_ϕ corresponding to the flow φ. Let us then define $S \subseteq N$ as the set of nodes $v \in N$ to which there exists a directed $x_0 \mapsto v$ path in G_ψ, each arc of which has a positive residual capacity (we assume that $x_0 \in S$). Furthermore, let $T = \{\overline{v} \mid v \in S\}$. Since φ is a maximum flow in G_ϕ, we cannot have $\overline{x}_0 \in S$, and consequently $T \cap S = \emptyset$ follows by the symmetry (6) of the implication network G_ψ.

The following fact is well–known in network optimization:

Proposition 4. *The set S is unique, and is independent of the choice of the maximum flow φ. It is in fact the intersection of the source sides of all minimum cuts of G_ϕ.*

Let us note also that $\{u, v\} \subseteq S$ cannot hold for any quadratic term $a_{uv}uv$ of ψ with positive coefficient a_{uv}, since otherwise we would have a positive capacity arc $(u, \overline{v})$ from $u \in S$ to $\overline{v} \in T$ by the definition of arc capacities in the implication network associated to ψ, leading to a positive capacity path from x_0 to $\overline{x}_0$, in contradiction with the maximality of φ. Thus, it follows that the assignment which sets all literals in S to 1 (there exists such an assignment, since $T \cap S = \emptyset$) makes all terms of ψ which involve literals from S or T vanish. Introduce

$$J = \{j \in \mathbf{V} \mid \{x_j, \overline{x}_j\} \cap S \neq \emptyset\}$$

and let $\mathbf{y} \in \mathbb{B}^J$ be the partial assignment for which $u(\mathbf{y}) = 1$ holds for all $u \in S$ (and consequently, $v(\mathbf{y}) = 0$ for all $v \in T$).

Proposition 5. *The partial assignment $\mathbf{y} \in \mathbb{B}^J$ is a strong autarky for ψ (and hence for ϕ). Consequently, the assignments $x_j = y_j$ for all $j \in J$ are strongly persistent for problem (1).*

Proof. See Theorem 11 in [18].   □

In fact the set of variables x_j, $j \in J$ consists exactly of those involved in the so called *master roof* as defined in [40]. This discussion shows that as a byproduct of a maximum flow computation, the above approach determines J in additional computing time, which is linear in the number of nonzero terms of ψ, i.e. linear in the size of ϕ.

3.3.2 Weak persistency

Let us consider now the directed graph $\widehat{G}$ obtained from G_ψ by keeping only those arcs which have a positive residual capacity. Since we can assume that the maximum flow φ is symmetric, we shall have

$$(\overline{v}, \overline{u}) \in A\left(\widehat{G}\right) \quad \text{whenever} \quad (u, v) \in A\left(\widehat{G}\right). \tag{13}$$

Let us compute the strong components of this directed graph (necessitating linear time in the number of arcs, i.e. linear time in the size of ψ [83]), and denote these strong components by K_i, $i = 1, \cdots, c$. It is easy to see that the symmetry (13) implies the following

Proposition 6. *For every strong component K_i of $\widehat{G}$ we have either*

$$\{\overline{v} \mid v \in K_i\} = K_i \tag{14}$$

or

$$\{\overline{v} \mid v \in K_i\} = K_{i'} \tag{15}$$

for some $i' \neq i$.

Proof. Follows readily by (13). $\qquad\square$

Let us label now as K_1, $K_{1'}$, K_2, $K_{2'}, \cdots, K_\ell$, $K_{\ell'}$, those strong components which satisfy (15), in such a way that

- there is no directed path in $\widehat{G}$ from K_i to $K_{i'}$ for $i = 1, \cdots, \ell$, and

- there is no directed path in $\widehat{G}$ from x_0 to $K_{i'}$ for $i = 1, \cdots, \ell$.

Since φ is a maximum flow, we cannot have a directed path from x_0 to $\overline{x}_0$ in $\widehat{G}$, and hence the symmetry conditions (13) imply the existence of such a labeling.

Let $J_i = \{j \in \mathbf{V} \mid \{x_j, \overline{x}_j\} \cap K_i \neq \emptyset\}$ and let $\mathbf{y}_i \in \mathbb{B}^{J_i}$ be the partial assignment for which $u(\mathbf{y}_i) = 1$ for all $u \in K_i$, for $i = 1, \cdots, \ell$.

Proposition 7. *The partial assignment $\mathbf{y}_i$ is an autarky for ψ, for $i = 1, \cdots, \ell$. Moreover, it is a strong autarky if there is a directed path in $\widehat{G}$ from x_0, or if there is a directed arc between K_i and $K_{i'}$. Consequently, the assignments $x_j = y_{ij}$ for all $j \in J_i$ and $i = 1, \cdots, \ell$ are all persistent assignments for problem (1).*

Proof. The claim is implied by the fact that the terms of ψ including variables in $J_i, i = 1, \cdots, \ell$ vanish. $\qquad\square$

Example 2. *Consider the quadratic pseudo–Boolean function of Example 1, but represented now by a different posiform ϕ' ($= \phi$),*

$$\begin{aligned} \phi' &= -3 + x_1\overline{x}_2 + 2x_1x_3 + 2x_1\overline{x}_4 + x_1x_5 + \overline{x}_1\overline{x}_5 + x_2x_3 \\ &\quad + x_2\overline{x}_5 + \overline{x}_2x_4 + 2\overline{x}_3\overline{x}_4 + 2\overline{x}_3x_5 + 2x_4x_5. \end{aligned}$$

In this case the maximum flow is $\varphi = 0$, and we have $\phi' = \psi$. The residual implication network $G_\psi = G_{\phi'}$ has two nontrivial strong components besides $\{x_0\}$ and $\{\overline{x}_0\}$, $K_1 = \{\overline{x}_1, \overline{x}_2, x_3, \overline{x}_4, x_5\}$ and $K_{1'} = \{x_1, x_2, \overline{x}_3, x_4, \overline{x}_5\}$, both of them of type (15) (see Figure 2). Then, by Proposition 7 the minimum value of ϕ' is $\phi'(0,0,1,0,1) = -3$, which is also its roof–dual value. In fact, there is an arc from $K_{1'}$ to K_1, and hence the assignment $\mathbf{y} = (0,0,1,0,1)$ is strongly persistent, implying that it is the unique minimum of ϕ', in this particular case.
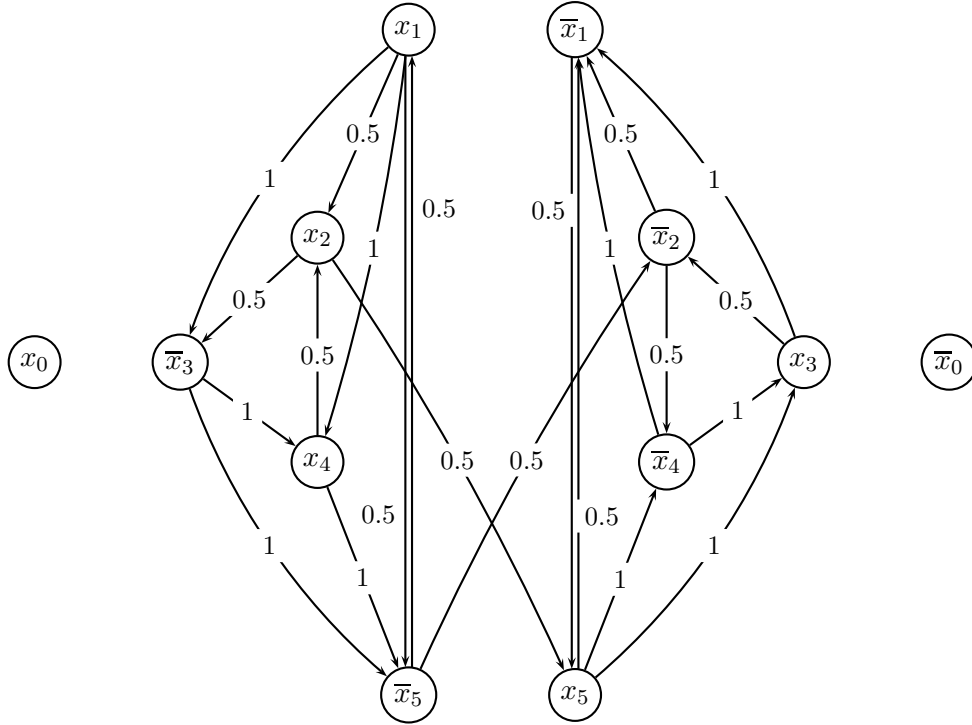
Figure 2: The implication network $G_{\phi'}$ of Example 2. We indicated only those arcs which have positive capacities.

Let us note that if there is a directed arc (u,v) between K_i and $K_{i'}$ for some $i = 1, \cdots, \ell$, then symmetry (13) implies that an arc $(\overline{v}, \overline{u})$ must also exist between K_i and $K_{i'}$. It is this property that makes the persistency result of Proposition 7 to be strong, in this particular situation.

In general, deciding if a partial assignment $\mathbf{y}_i$ is a strong autarky for ϕ, for $i = 1, \cdots, \ell$ is a NP–hard decision problem. This result can easily be established, since the outcome of this decision depends on the optimization of NP–hard sub-problems, each one associated to a component of type (14).

Example 3. *Consider the quadratic posiform ϕ given by*

$$\phi = 2x_1\overline{x}_2 + 4\overline{x}_1x_2 + 2x_2\overline{x}_3 + 2\overline{x}_2x_3 + 2x_1x_3 + 4\overline{x}_1\overline{x}_3 + 2x_4\overline{x}_5 + 2\overline{x}_4x_5 + 2x_5\overline{x}_1,$$

and the associated network G_ϕ shown in Figure 3. The strong components of G_ϕ are:

$$
\begin{aligned}
K_1 &= \{x_0\}, \\
K_{1'} &= \{\overline{x}_0\}, \\
K_2 &= \{\overline{x}_4, \overline{x}_5\}, \\
K_{2'} &= \{x_4, x_5\} \ \ and \\
K_3 &= \{x_1, x_2, x_3, \overline{x}_1, \overline{x}_2, \overline{x}_3\}.
\end{aligned}
$$

Let us note that there is no directed path from K_2 to $K_{2'}$, there is no directed path from x_0 to $K_{2'}$, and that K_2 and $K_{2'}$ satisfy condition (15). Let $J_2 = \{4, 5\}$ and let $\mathbf{y_2} \in \mathbb{B}^{\mathbf{J_2}}$ be the partial assignment for which $\overline{x}_4 = \overline{x}_5 = 1$. By Proposition 7, $x_4 = x_5 = 0$ must hold in a minimizer of ϕ. Let us now show that these two assignments are not strongly presistent. In the first place one can verify that $x_1 = 1$ must hold in all minimizers of ϕ. Therefore, the only term connecting K_3 to the other components vanishes. So, any solution satisfying the equation $x_4\overline{x}_5 + \overline{x}_4 x_5 = 0$, including $x_4 = x_5 = 1$, must hold in a minimizer of ϕ.
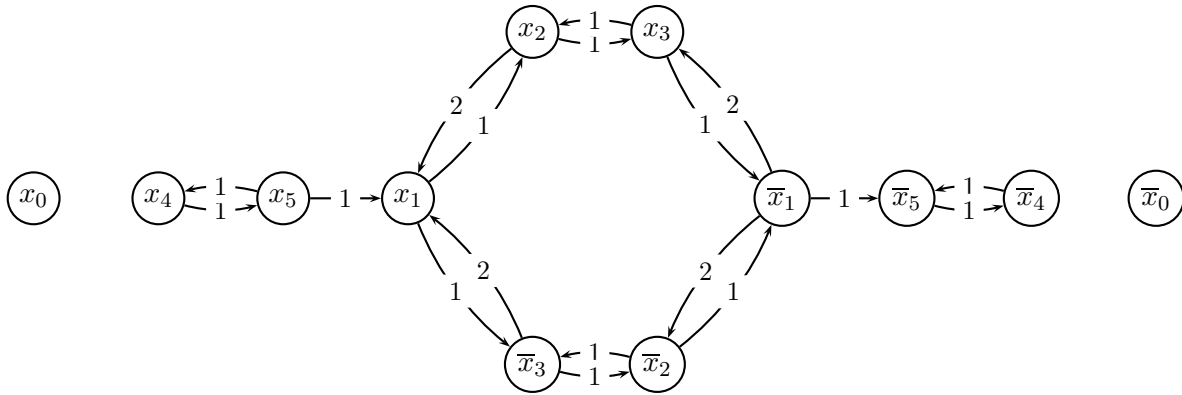
Figure 3: The network G_ϕ corresponding to the posiform ϕ of Example 3. We indicate only those arcs which have positive capacities.

3.3.3 Decomposition

Let us assume now that we have already fixed all weakly and strongly persistent variables (e.g., by Proposition 7), and that the strong components of the residual posiform ψ, K_i, $i = 1, \cdots, c$, are all of type (14).

Clearly, in this case the symmetry conditions (13) imply that there are no arcs between different strong components, i.e. ψ does not involve a quadratic term $a_{uv}uv$, $a_{uv} > 0$ for which $u \in K_i$, $v \in K_j$ and $i \neq j$. Thus, denoting by ψ_i the posiform corresponding to the induced subnetwork K_i, $i = 1, \cdots, c$, we have

$$
\psi = \sum_{i=1}^{c} \psi_i.
$$

Furthermore, due to property (14), these posiforms involve disjoint sets of variables. Hence, we have

Proposition 8.

$$\min_{\mathbf{x} \in \mathbb{B}^{\mathbf{V}}} \psi(\mathbf{x}) \; = \; \sum_{i=1}^{c} \left(\min_{\mathbf{x} \in \mathbb{B}^{J_i}} \psi_i(\mathbf{x}) \right). \tag{16}$$

A similar decomposition of quadratic posiforms which does not involve linear terms was already proposed in [13]. Let us note that after computing the maximum flow in the implication network G_ϕ, both the persistent assignments, as well as the above decomposition can be determined in linear time of the size of ϕ.

Example 4. *Consider ϕ to be an homogeneous quadratic posiform whose nonzero terms are:*

$$x_1\overline{x}_2, \overline{x}_1 x_2, x_2\overline{x}_3, \overline{x}_2 x_3, x_1 x_3, \overline{x}_1\overline{x}_3, x_4\overline{x}_5, \overline{x}_4 x_5, x_5\overline{x}_6, \overline{x}_5 x_6, x_4 x_6, \overline{x}_4\overline{x}_6,$$
$$x_7\overline{x}_8, \overline{x}_7 x_8, x_1\overline{x}_7, x_4\overline{x}_8, x_9\overline{x}_{10}, \overline{x}_9 x_{10}, x_7\overline{x}_9, \overline{x}_{11}, x_{11}\overline{x}_{12}, \overline{x}_{11} x_{12}, x_{10}\overline{x}_{11}.$$

The associated network G_ϕ, shown in Figure 4, has the following strong components:

$$
\begin{array}{llll}
K_1 &=& \{x_0\}, & \qquad K_{1'} \;=\; \{\overline{x}_0\}, \\
K_2 &=& \{x_{11}, x_{12}\}, & \qquad K_{2'} \;=\; \{\overline{x}_{11}, \overline{x}_{12}\}, \\
K_3 &=& \{x_9, x_{10}\}, & \qquad K_{3'} \;=\; \{\overline{x}_9, \overline{x}_{12}\}, \\
K_4 &=& \{x_7, x_8\}, & \qquad K_{4'} \;=\; \{\overline{x}_7, \overline{x}_8\}, \\
K_5 &=& \{x_1, x_2, x_3, \overline{x}_1, \overline{x}_2, \overline{x}_3\} & and \\
K_6 &=& \{x_4, x_5, x_6, \overline{x}_4, \overline{x}_5, \overline{x}_6\}.
\end{array}
$$

Let us first note that there is no directed path from x_0 to $\overline{x}_0$. Thus, by strong persistency (Proposition 5) $x_{11} = x_{12} = 1$ must hold for all minimizers of ϕ. Also, regardless of the values of the coefficients in the nontrivial terms of ϕ, by weak persistency (Proposition 7) the partial assignment $x_7 = x_8 = x_9 = x_{10} = x_{11} = x_{12} = 1$ must hold in a minimizer of ϕ. This partial assignment automatically cancels all those terms of ϕ which involve at least a variable from the set $\{x_7, x_8, x_9, x_{10}, x_{11}, x_{12}\}$. After eliminating these terms, the original problem is decomposed into two subproblems, involving disjoint sets of variables, coming respectively from K_5 and K_6. Obviously, these two subproblems can be optimized separately, and the sum of their optimal values will coincide with the minimum of ϕ.

4   Combining Basic Tools

4.1   Enhancing roof–duality by probing

It will be seen in this section that the results of roof–duality can be substantially strengthened by analyzing the roof–duals of several quadratic pseudo–Boolean functions associated to the original one. The analysis of the roof–duals of the $2n$ quadratic pseudo–Boolean functions
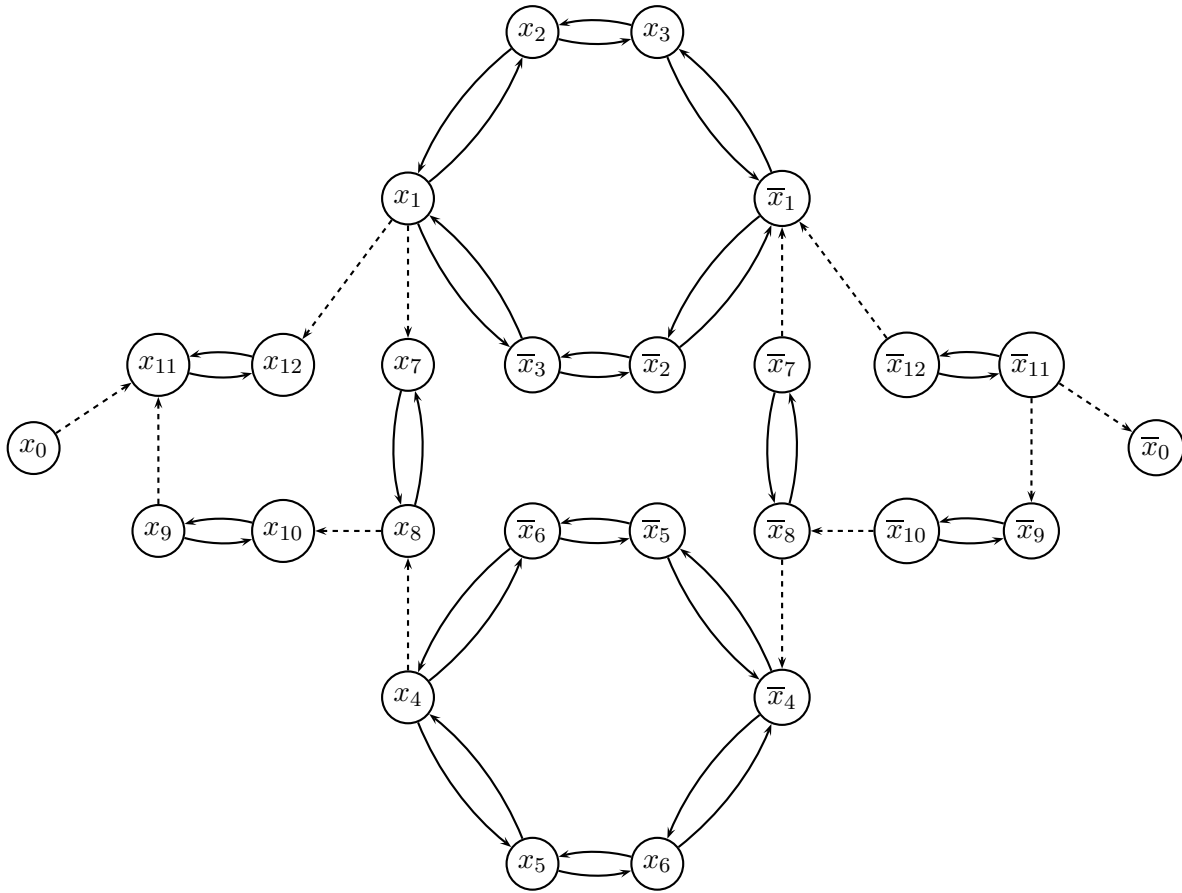
Figure 4: The network G_ϕ corresponding to the posiform ϕ of Example 4. We disregarded the values of the capacities, and indicated only those arcs which have positive capacities. The dashed arcs represent arcs connecting the strong components of G_ϕ.

obtained by fixing either to zero or to one the value of one of the n variables of a quadratic pseudo–Boolean function $f(x_1, \cdots, x_n)$, will be called *probing*.

Among the specific results of probing we mention the possibility of identifying an improved lower bound of the minimum of a quadratic pseudo–Boolean function, and of enlarging at the same time the set of variables and binary relations for which persistency conclusions apply.

In view of the fact that finding the roof–dual of a quadratic pseudo–Boolean function can be achieved by simply solving a maximum flow problem in a network, the calculation of the roof–duals of $2n$ quadratic pseudo–Boolean functions associated to the initially considered one is a feasible, easy–to–carry–out operation. Let us provide now some technical ideas on how to efficiently calculate the $2n$ roof–duals required by probing.

Let us assume first that using a simple heuristic (e.g. as proposed in [21, 22]) we have

found some upper bound U on the minimum of (1), say

$$\min_{\mathbf{x} \in \mathbb{B}^{\mathbf{V}}} \phi(\mathbf{x}) \leqslant U. \tag{17}$$

Let us now consider the most typical branching procedure, in which we split the problem into two somewhat smaller ones by fixing variable x_j at the two possible binary values. Since ϕ is a posiform, all of its terms – with the possible exception of the constant a_0 – contribute nonnegative quantities to the objective. Therefore, if $M > U - a_0$, then

$$\min_{\mathbf{x} \in \mathbb{B}^{\mathbf{V}}} \phi(\mathbf{x}) = \min \left\{ \min_{\mathbf{x} \in \mathbb{B}^{\mathbf{V}}} \phi(\mathbf{x}) + M x_j, \ \min_{\mathbf{x} \in \mathbb{B}^{\mathbf{V}}} \phi(\mathbf{x}) + M \overline{x}_j \right\}, \tag{18}$$

where a_0 is the constant in ϕ, as given in (4).

The two subproblems in (18) have simple network representations.

In order to calculate the roof–duals of $\min_{\mathbf{x} \in \mathbb{B}^{\mathbf{V}}} \phi(\mathbf{x}) + M x_j$ and of $\min_{\mathbf{x} \in \mathbb{B}^{\mathbf{V}}} \phi(\mathbf{x}) + M \overline{x}_j$, and to derive persistency relations from these, we should add to the original network an arc, $(x_0, \overline{x}_j)$ in the first case and (x_0, x_j) in the second case, and to assign to these the large capacity M.

Clearly, computationally it is simpler to increase the capacity of two arcs than to substitute $x_j = 0$, respectively $x_j = 1$, implying the deletion of nodes x_j and $\overline{x}_j$, and of all arcs incident to these nodes in the network. In addition, keeping the same network and updating the capacities of a few arcs at each branch evaluation, allows us to carry out computations without increasing the amount of computer memory needed to keep the network data necessary to find a maximum flow for each subproblem. From an implementational point of view, this approach has the added advantage of allowing the easy restoration of the network corresponding to the original problem, by simply finding an additional maximum flow – an option which turned out to be on the average to be much better than creating a copy to be reused after branching. It should be remarked that without these simplifying steps, the large scale QUBOs (including for instance those coming from finding optimal vertex covers of planar graphs with half a million vertices; see Section 8) could not have been handled.

We can similarly administer more complicated branching policies, as well. For instance, if $u, v \in \mathbf{L}$ are two literals, $u \neq v$, then branching on the binary relation $u \leqslant v$ can be written as

$$\min_{\mathbf{x} \in \mathbb{B}^{\mathbf{V}}} \phi(\mathbf{x}) = \min \left\{ \min_{\mathbf{x} \in \mathbb{B}^{\mathbf{V}}} \phi(\mathbf{x}) + M \overline{u} + M v, \ \min_{\mathbf{x} \in \mathbb{B}^{\mathbf{V}}} \phi(\mathbf{x}) + M u \overline{v} \right\} \tag{19}$$

for some $M > U - a_0$, resulting in the modification of 4 arc capacities in the first branch corresponding to $u = 1$ and $v = 0$, and of two arc capacities on the other branch corresponding to $u \leqslant v$.

We can also apply the above for handling persistencies. For instance, if we learn that $u \leqslant v$ is a persistent binary relation, then we can rewrite (19) as

$$\min_{\mathbf{x} \in \mathbb{B}^{\mathbf{V}}} \phi(\mathbf{x}) = \min_{\mathbf{x} \in \mathbb{B}^{\mathbf{V}}} \phi(\mathbf{x}) + M u \overline{v}. \tag{20}$$

Let us note that in all of the above cases, we had to increase the capacity of some of the arcs. Thus, as our procedure advances, and we learn more and more persistencies, at the same time the maximum flow value is also increasing. Hence, according to (12), as an added value we get better and better lower bounds on the minimum of (1).

To describe probing and its benefits, let us consider an arbitrary quadratic posiform ϕ, as given in (4). For a literal $u \in \mathbf{L}$ let us consider the posiform

$$\psi_u = \phi + M\overline{u},$$

where $M > U - a_0$ for an upper bound U satisfying (17). Let us further denote by $S_u \subseteq \mathbf{L}$ ($W_u \subseteq \mathbf{L}$) the set of strongly (weakly) persistent literals for ψ_u, as defined in Section 3.3.1 (3.3.2), and let L_u denote the roof–dual bound for ψ_u.

We can derive several consequences from the sets S_u, W_u and lower bounds L_u when generating these for all literals $u \in \mathbf{L}$.

Proposition 9. *Let U be an upper bound of $\min_{\mathbf{x} \in \mathbb{B}^\mathbf{V}} \phi(\mathbf{x})$, and let $u \in \mathbf{L}$ and $j \in \mathbf{V}$. Then*

- *The value $L^* = \max_{u \in L} \min \{L_u,\ L_{\overline{u}}\}$ is a lower bound on the minimum of ϕ.*

- *If $L_u > U$ then $u = 0$ is a strongly persistent assignment for ϕ.*

- *If $v \in S_{x_j} \cap S_{\overline{x}_j}$ ($v \in W_{x_j} \cap W_{\overline{x}_j}$) then $v = 1$ is a strongly (weakly) persistent assignment for ϕ.*

- *If $v \in S_{x_j}$ and $\overline{v} \in S_{\overline{x}_j}$ ($v \in W_{x_j}$ and $\overline{v} \in W_{\overline{x}_j}$) then $x_j = v$ is a strongly (weakly) persistent relation for ϕ.*

- *For all $v \in S_{x_j}$ and $w \in S_{\overline{x}_j}$ ($v \in W_{x_j}$ and $w \in W_{\overline{x}_j}$) the quadratic relations $x_j \leqslant v$, $\overline{x}_j \leqslant w$ and $\overline{w} \leqslant v$ are all strongly (weakly) persistent for ϕ.*

All these follow from the above definitions, by which the assignment $v = 1$ is strongly (weakly) persistent for ψ_u, for all $v \in S_u$ ($v \in W_u$). Let us note that by adding these new persistencies to ϕ, as in (18) and (20), we may increase both the roof–dual value as well as the set of strongly and weakly persistent literals of ϕ. Furthermore, the addition of these to ϕ may also change the sets S_v or W_v for some other literals $v \in \mathbf{L}$, $v \neq u$.

Let us remark that the lower bound derived from probing was also considered in [16, 20], and that analogous techniques were explored in the broader context of binary optimization in [5, 77].

4.2 Consensus

It has been remarked above that order relations between literals can be derived both from the signs of the second order derivatives and co–derivatives, as well as during the process of probing. The interactions of the various binary relations, and the conclusions obtained

by combining them can be very easily derived by the introduction of a Boolean quadratic equation $\Phi = 0$, where the terms of Φ are quadratic elementary conjunctions (representing the detected binary relations between literals) which must take the value zero in every minimum of (1). Moreover, the application of the consensus method to Φ allows the polynomial detection of all of its prime implicants (see [26]). Taking into account that the prime implicants of this function represent either variables with fixed values in the optimum, or order relations which must hold between pairs of literals in every optimum, it is clear that the detection of all prime implicants of Φ provides an enlarged set of strong persistencies.

Finally, we should remark that the conclusions that can be obtained from the knowledge of the prime implicants of Φ can also be obtained directly (using Proposition 9) during our implementation of probing, by appropriately transforming the original functions via term additions (as explained in the previous section) corresponding to the persistent binary relations found by the preprocessing tools considered.

5   Preprocessing Algorithm and Implementation

The proposed preprocessing algorithm is presented in this section, in which the tools described in the previous sections are used iteratively and recursively. The structure adopted is based on the network flow model of Section 2.3. Our goal is to find better and better lower bounds, weak (and strong) linear persistencies and weak (and strong) quadratic persistencies for ϕ, as well as to decompose the problem into several smaller problems, as explained in Section 3.3.3.

The PREPRO algorithm is described in Figure 5. The input of the algorithm is a quadratic posiform ϕ representing a quadratic pseudo–Boolean function f, as in (4). The output returned by PREPRO is a decomposed representation of the minimum of f, as in (16), where the subproblems on the right hand side of (16) involve pairwise disjoint sets of variables, together with a lower and an upper bound to the minimum of each subproblem, and with a partial (optimal) assignment to the derived persistent variables.

Four main components are part of PREPRO:

- NETWORK – This routine has a posiform ϕ as input. It first finds a maximum flow in the network G_ϕ as explained in Proposition 11. The maximum flow implementation that we have adopted is based on the shortest augmenting path algorithm, yielding a worst case time of $O(n^3)$, and is specially designed to obtain a residual network satisfying the symmetry conditions (6). When a minimum cut is found, a set of strong persistencies is obtained directly from a non–empty source side of a minimum cut (see Proposition 4) and saved accordingly in (V_0, V_1). The nodes of the residual network and corresponding arcs, which are associated to the set of strong persistencies is removed from the network. Using a linear time algorithm for identifying the strongly connected components of the residual network, a subset of weak persistencies is identified in every component of type (15) (see also Proposition 7), and saved accordingly in (V_0, V_1). The nodes of the residual network and corresponding arcs, which are associated to the

---

PrePro$(\phi, (V_0, V_1, E), \Theta)$

Input: A quadratic posiform ϕ representing a quadratic pseudo–Boolean function f, as in (4).

Step 1:   Call Network$(\phi, \Phi, (V_0, V_1, E))$.

Step 2:   If $\Phi = \emptyset$ then $\Theta \leftarrow \emptyset$ and STOP. Otherwise, for all $(\psi, L, U) \in \Phi$ do call Heuristic(ψ, U).

Step 3:   For all $(\psi, L, U) \in \Phi$ do call Coordination(ψ, U, E).

Step 4: $\mathcal{P} \leftarrow \{(\psi, L, U) \in \Phi \,|\, \psi$ is purely quadratic$\}$.

Step 5:   For all $(\psi, L, U) \in \mathcal{P}$ do call Probing$(\psi, L, U, (V_0, V_1, E))$.

Step 6: $\mathcal{P} \leftarrow \{(\psi, L, U) \in \mathcal{P} \,|\, \psi$ is purely quadratic$\}$.

Step 7:   For all $(\psi, L, U) \in (\Phi \setminus \mathcal{P})$ do call PrePro $(\psi, (V_0, V_1, E), \mathcal{S})$. $\Theta \leftarrow \mathcal{P} \cup \mathcal{S}$.

Output: It returns in Θ a set of triplets, whose sum of all posiforms has the same minimum value as the minimum value of f. Additionally, a set of persistencies for the initial posiforms is returned as follows:

   V_0 – A set of variables with value 0,

   V_1 – A set of variables with value 1, and

   E – A set of equality relations between pairs of literals,

   which must hold in a minimizer of f.

Figure 5: PrePro algorithm.

set of weak persistencies are removed from the network. What is left after applying Network is a disjoint set of strong components, each corresponding to a subproblem (included in Φ) that can be optimized separately from the other subproblems.

- Heuristic – For each subproblem identified in Network, an upper bound U is found, and later used in the Coordination and Probing procedures. Any heuristic method valid for QUBO can be used. All of our experimental results include a fast one–pass heuristic based on a greedy approach, which ranks the set of non–fixed variables according to an approximated probability value of a variable to have a persistent value (some onepass variants are proposed in [33]).

- Coordination – This procedure has as input a posiform ψ, and the upper bound U found by Heuristic. Let us note that the minimum of any posiform ψ called within PrePro is *strictly* larger than its constant value a_0. This routine identifies binary persistent relations originated from the analysis of the second order derivative and co-derivative functions as explained in Proposition 2. The basic idea is to compute over all possible pair of variables $i < j$, the minimum and the maximum of the linear pseudo–Boolean functions Δ_{ij} and ∇_{ij}. A key element to save computing time in this operation is to stop it as soon as one learns that the minimum is strictly negative and the maximum is strictly positive. If a quadratic persistency $u \leqslant v$ is found,

then ψ is updated by adding a term $a_{u\overline{v}}u\overline{v}$ with a large enough coefficient (we use $a_{u\overline{v}} = 2(U - a_0) + 1$). Since the implication network structure was adopted in all tools, this last operation can be efficiently implemented by updating the coefficient of the arcs (u, v) and $(\overline{v}, \overline{u})$. Our data structure is also able to quickly identifying if the reverse relation $v \leqslant u$ is a quadratic persistency. In the affirmative case, E is updated to include the equality persistency $u = v$, and ψ is transformed by replacing v ($\overline{v}$) by u ($\overline{u}$). This routine stops as soon as a linear term or a equality persistency is found.

- PROBING – This procedure has as input a *purely* quadratic posiform ψ (i.e. a posiform that does not have linear terms) and the upper bound U found by HEURISTIC. The implication network structure plays a crucial role in this routine. Independently, each variable x_j is fixed to one of the possible binary values, and the resulting function is analyzed in terms of roof–duality and strong and weak persistencies. This operation can be accommodated easily in the network ψ as explained in Section 4.1. For a given assignment to x_j, a maximum flow algorithm is applied to the transformed network. All the strong and weak persistencies derived from the residual network are updated both in the network and in (V_0, V_1, E), as explained in Proposition 9. To analyze the complement assignment of x_j, a maximum flow algorithm is applied to the residual network. All the strong and weak persistencies derived from the residual network are again updated as before. A third maximum flow algorithm is applied to obtain a network which represents the same function as ψ. The use of maximum flow algorithms to recuperate the original function being optimized is possible due to Proposition 3. A clear advantage of this approach is that the amount of memory needed for the data structures remains about the same through every step of the procedure. Let us note that probing through the implication network is able to capture persistencies of transitive relations (see Section 4.2). For instance, suppose that $u \leqslant v$ and $v \leqslant w$ are quadratic persistencies, then if at some point $w \leqslant u$ is also found to be persistent, then the network and the set E are immediately updated with the equality relation $u = w$. The routine stops as soon as a linear term or a linear/equality persistency is found.

Step 4 of PREPRO selects the subproblems for which probing is applied. Step 6 of PREPRO selects the subproblems for which PREPRO is recursively applied. Obviously, the rules that govern these choices may vary. In our implementation, the rule adopted is to apply NETWORK to a subproblem whenever a new linear persistency or a new linear term was found by COORDINATION or PROBING.

All the tools considered in PREPRO are polynomial time algorithms:

NETWORK   – $O\left(n^3\right)$;
HEURISTIC   – $O\left(n^2\right)$;
COORDINATION   – $O\left(n^3 log\left(n\right)\right)$;
PROBING   – $O\left(n^4\right)$.

As a consequence of the previous complexity times, each run from Step 1 to Step 5 of PREPRO takes at most $O\left(n^4\right)$.

6 Test Problems

Most of the problem classes on which we have tested the proposed preprocessing procedures are benchmarks used in several other studies related to QUBO. These datasets include 85 problems with prescribed density, 13 graphs for MAX-Clique, 38 graphs for MAX–CUT, and 34 MAX–2–SAT formulas. Beside the above classes we have also carried out computational experiments on 436 randomly generated planar graphs for vertex cover optimization problems.

6.1 Benchmarks with prescribed density

The class of benchmarks with prescribed density consists of quadratic multilinear polynomials, which are randomly generated with a predetermined expected density[1] d, a discrete uniform distribution of values of the linear coefficients in an interval $[c^-, c^+]$, and a discrete uniform distribution of values of the the quadratic coefficients in an interval $[2q^-, 2q^+]$. The constant term of the function is zero. Since a quadratic term's probability to have a nonzero coefficient is d, the expected number of quadratic terms which include a specific variable and have a nonzero coefficient is $(n-1)\, d$.

This group of problems includes the test problems of Glover, Kochenberger and Alidaee [34], and the problems of Beasley [11] with at most 500 variables. The basic generation parameters of the sub–families containing these problems can be seen in Table 1, while the individual characteristics of the problems appear in Tables A.1 and A.2 of the Appendix. Obviously, Glover's and Beasley's maximization problems have been first converted to minimization problems in order to make the direct application of our proposed procedures possible.

Table 1: Benchmarks with prescribed densities for QUBO.

Family	Sub-Family	Number of Problems	Variables (n)	Density (d %)	Linear Coef. (c^-)	(c^+)	Quadr. Coef. (q^-)	(q^+)
GKA	A	8	30 to 100	6.5 to 50	-100	100	-100	100
	B	10	20 to 125	100	0	63	-100	0
	C	7	40 to 100	10 to 80	-50	50	-100	100
	D	10	100	6.5 to 50	-50	50	-75	75
	E	5	200	10 to 50	-50	50	-100	100
	F_1	5	500	10 to 100	-75	75	-50	50
Beasley	ORL-50	10	50	10	-100	100	-100	100
	ORL-100	10	100	10	-100	100	-100	100
	ORL-250	10	250	10	-100	100	-100	100
	ORL-500	10	500	10	-100	100	-100	100

---

[1]The density d of a quadratic pseudo–Boolean function represented by the polynomial expression (2) is defined as the number of nonzero coefficients c_{ij} $(1 \leqslant i < j \leqslant n)$ divided by $\binom{n}{2}$.

6.2   Maximum cliques of graphs

Let $G = (V, E)$ be an undirected graph, with vertex set V and edge set E. The *complement* $\overline{G}$ of G is the graph $\overline{G} = \left(V, \binom{V}{2} \setminus E\right)$, i.e. the graph having the same vertex set V and having as edge set the complement of the edge set E of G. A *clique* of the graph $G = (V, E)$ is a set of pairwise adjacent vertices. Naturally, a *maximum clique* (or MAX–Clique in short) is a clique of maximum cardinality. The size of a maximum clique is commonly called the *clique number* of G. We shall denote it as $\theta(G)$.

A subset $S \subseteq V$ is called *independent* (or *stable*), if no edge of G has both endpoints in S. A *maximum independent set* is a largest cardinality independent set; the cardinality of a maximum independent set will be denoted by $\alpha(G)$ and called the *stability number* of G.

An independent set in a graph G is a clique in the complement graph $\overline{G}$. Thus, $\alpha(G) = \theta(\overline{G})$. Moreover, since the complement $\overline{G}$ of G has a polynomial size representation of the input size of G, and since it can be obtained in a polynomial time function of the size of G, then there is a strong equivalence between the maximum clique and the maximum independent set problems. If a solution to one of the problems is available, a solution to the other problem can be obtained immediately.

It is not difficult to show (see e.g. [18]) that the cardinality of a maximum independent set of $G = (V, E)$ is given by

$$\alpha(G) = \max_{\mathbf{x} \in \mathbb{B}^V} \left(\sum_{i \in V} x_i - \sum_{(i,j) \in E} \left(1 + \epsilon_{(i,j)}\right) x_i x_j \right), \tag{21}$$

where $\epsilon_{(i,j)}$ $(\geqslant 0)$ are arbitrary nonnegative reals for all $(i, j) \in E$. Furthermore, if $\mathbf{x}^* = \mathbf{1}^S$ is a maximizing binary vector of (21), then a maximum cardinality set S^* $(\subseteq S)$ of G can be obtained in $O(n)$ time. In fact, if the coefficients ϵ_e, $e \in E$, are restricted to be positive, then $S^* = S$ is a maximum cardinality independent set of G. We note here that in our experiments it was assumed that $\epsilon_{(i,j)} = 0$ for all edges (i, j).

In order to facilitate comparisons among different exact and heuristic methods related to clique problems, a set of benchmark graphs has been constructed in conjunction with the 1993 DIMACS Challenge on maximum cliques, coloring and satisfiabilty ([47]). This study only reports preprocessing results for two families of this dataset, since for the other graphs PREPRO did not find any persistencies. Namely, we consider the benchmarks containing c–fat graphs or Hamming graphs:

- A major step in the algorithm of the fault diagnosis problem proposed by Berman and Pelc [12] is to find the maximum clique of a special class of graphs, called c–*fat* rings. In order to define a c–fat graph $G = (V, E)$, let us consider an arbitrary finite set of vertices V. Let c be a real parameter, $k = \left\lfloor \frac{|V|}{c \log |V|} \right\rfloor$, and let us consider a partition $W_0, \cdots, W_{k-1}$ of V, such that $c \log |V| \leqslant |W_i| \leqslant \lceil c \log |V| \rceil + 1$ for all $i = 0, \cdots, k-1$. The edge set E is defined as the set of those edges (u, v) which link distinct pairs of

vertices $u \in W_i$ and $v \in W_j$, such that $|i - j| \in \{0, 1, k - 1\}$. The DIMACS c–fat rings were created by Panos Pardalos using the c–fat rings generator of Hasselberg, Pardalos and Vairaktarakis [43].

- The *Hamming* graphs arise from coding theory problems ([81]). The Hamming distance between the binary vectors $\mathbf{u} = (u_1, \cdots, u_n)$ and $\mathbf{v} = (v_1, \cdots, v_n)$ is the number of indices $i = 1, \cdots, n$ where $u_i \neq v_i$. The Hamming graph $H(n, d)$ of size n and distance d is the graph whose vertex set is the set of all binary n–vectors, and whose edges link any two n–vectors at distance d or larger. Clearly, the graph $H(n, d)$ has 2^n vertices, $2^{n-1} \sum_{i=d}^{n} \binom{n}{i}$ edges, and the degree of each vertex is $\sum_{i=d}^{n} \binom{n}{i}$. A binary code consisting of a set of binary vectors, any two of which have Hamming distance greater or equal to d, can correct $\lfloor \frac{d-1}{2} \rfloor$ errors. Thus, a coding theorist (see [56]) would like to find the maximum number of binary vectors of size n with Hamming distance d, i.e. the maximum clique of $H(n, d)$. The DIMACS Hamming graphs were created by Panos Pardalos (for details see [43]).

6.3 Minimum vertex cover problems of planar graphs

The complement $V \backslash S$ of an independent set S of G is called a *vertex cover* of the graph. Let us denote the size of the smallest vertex cover of G as $\tau(G) = |V| - \alpha(G)$. Then using (21) it can be shown that

$$\tau(G) = \min_{\mathbf{x} \in \mathbb{B}^V} \left(\sum_{i \in V} x_i + \sum_{(i,j) \in E} \left(1 + \epsilon_{(i,j)} \right) \overline{x}_i \overline{x}_j \right). \tag{22}$$

The knowledge of any one of the three numbers $\theta(\overline{G})$, $\alpha(G)$ or $\tau(G)$ implies that the other two values can be immediately determined. In general finding any of these numbers is a NP–hard optimization problem ([31]). It is important to note that even for planar graphs it is known that solving the minimum vertex cover problem is NP–hard ([32]).

Motivated by a recent work by Alber, Dorn and Niedermeier [1] we have analyzed the performance of PREPRO in this class of graphs. The only difference between the two approaches is that the method of Alber, Dorn and Niedermeier considers "the influence of a clever, VERTEX COVER–specific data reduction" (see [1], page 220), whereas the results obtained by PREPRO are entirely due to its ways of simplifying QUBOs, since we have not introduced any specific adaptation of this method for the case of vertex cover problems.

In their experiments, [1] used a set of planar graphs randomly generated by the LEDA software package ([58]). Using the same LEDA generator we tried to replicate the experiment reported in [1], although it should be noted that not having access to the seeds used in [1], the graphs generated by us are not exactly identical to the ones used by Alber, Dorn and Niedermeier [1]. In order to distinguish between the two planar vertex cover benchmarks, we shall call those of [1] ADN benchmark graphs and the new ones *PVC LEDA benchmark*.

The total number of planar graphs that we have generated with LEDA is 400, partioned into 4 sets of 100 graphs, each subset having a specific number of vertices: 1000, 2000, 3000 and 4000. The *planar density* of each graph $G(V, E)$ was randomly determined, i.e. $|E| \backsim \textit{discrete uniform} (|V| - 1, 3|V| - 6)$. Some comparative statistical numbers about these two benchmarks are displayed in Table 2.

Table 2: Comparative statistical numbers about the LEDA benchmarks.

Benchmark	Vertices	Number of Graphs	Average Number of Edges	Average Maximum Degree	Average Degree	Average Minimum Vertex Cover
PVC LEDA	1000	100	2037.9	73.0	4.08	460.6
	2000	100	4068.6	106.7	4.07	921.1
	3000	100	6204.3	132.4	4.14	1391.2
	4000	100	8207.1	149.2	4.10	1848.2
ADN ([1])	1000	100	1978.9	73.3	3.96	453.9
	2000	100	3960.8	104.9	3.96	917.3
	3000	100	6070.6	129.6	4.05	1373.8
	4000	100	8264.5	146.6	4.13	1856.8

In addition to the PVC LEDA planar graphs we have also generated a dataset containing larger graphs with up to 500 000 vertices. These graphs were generated in order to analyze the scalability of the routine PREPRO. Because of size limitations associated to our trial license on LEDA, we used for this experiment Rinaldi's ([76]) generator called RUDY. With the RUDY program, we generated a total of 36 graphs whose sizes are of 50 000, 100 000, 250 000 and 500 000 vertices; for each of these graph sizes, we generated nine graphs: three instances with density of 10%, three with density of 50% and three with density of 90%. This set of benchmark graphs is called PVC RUDY.

6.4   Graphs for MAX–CUT

Let $G = (V, E)$ be an undirected graph, with vertex set V and edges set E. To each edge $e \in E$ of the graph, we may assign a real weight $w_e \in \mathbb{R}$.

A *cut* of the graph $G = (V, E)$ is defined by a partition of the vertex set V into two subsets S and $\overline{S}$, and consists of the set of edges with exactly one endpoint in S and another in $\overline{S}$. We shall denote the cut defined by S as $(S, \overline{S})$.

The *maximum cut* (or MAX–CUT in short) problem is to find a cut $(S, \overline{S})$ with the largest cardinality. If $\mathbf{x} = \mathbf{1}^S$ is the characteristic vector representing S, then it can be shown that

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

The *weighted MAX–CUT* problem in a graph $G = (V, E)$ with weights $\mathbf{w}^E$ is to find a cut $(S, \overline{S})$ for which the sum of the weights of the corresponding edges is maximum. If the

total weight of a cut is denoted by $W\left(S, \overline{S}\right)$, then the weighted maximum cut can be found by solving the problem

$$\max_{\mathbf{S} \subseteq V} W\left(S, \overline{S}\right) = \max_{\mathbf{x} \in \mathbb{B}^V}\left(\sum_{(i,j) \in E} w_{ij}\left(x_i \overline{x}_j + \overline{x}_i x_j\right)\right). \tag{23}$$

Let us remark that any optimal solution $\mathbf{x} = (x_1, \cdots, x_n)$ of the previous problem has a complementary solution $(\overline{x}_1, \cdots, \overline{x}_n)$. Thus, before calling PREPRO for a MAX–CUT problem, we select a variable and assign to it a 0–1 value.

Let us note that these problems are maximization problems. Therefore, we transformed the MAX–CUT problems (23) into quadratic posiform minimization problems as explained previously for the benchmarks in the fixed degree dataset.

We tested the PREPRO algorithm on the following graphs:

- *Torus graphs*: These graphs are 3D–toroidal graphs, originated from the Ising model of spin glasses in physics. They were taken from the DIMACS library of mixed semidefinite-quadratic-linear programs[2].

- *Via graphs* ([46]): These ten graphs are derived from layer assignment problems in the design process for VLSI chips. The characteristics of these graphs, and the largest cut information can be seen in Table A.5.

- *Sparse random graphs* ([46]): These eight graphs constitute the family R of Homer and Peinado [46], each having an edge probability of $10/n$, where the number of vertices n ranges from 1000 to 8000.

- *Gn.p graphs* ([51]): These graphs are randomly generated with n vertices, where an edge is placed between any two vertices with probability d, independent of all other edges. The probability d is chosen so that the expected vertex degree, i.e. $d\left(n-1\right)$, is p.

- *Un.p graphs* ([51]): These graphs are random geometric graphs on n vertices that lie in the unit square, and whose coordinates are chosen uniformly from the unit interval. There is an edge between any two vertices if their Euclidean distance is t or less, where $p = n\pi t^2$ is the expected vertex degree.

The general characteristics of the previous graphs, and the largest cut information are shown in Tables A.4 to A.6 of the Appendix.

---

[2]The DIMACS library of mixed semidefinite-quadratic-linear programs: *http://dimacs.rutgers.edu/Challenges/Seventh/Instances/*.

6.5   MAX–2–SAT formulas

The *maximum satisfiability problem* (or MAX–SAT in short) is a popular subject in applied mathematics and computer science, and it has a natural pseudo–Boolean formulation. The input of a MAX–SAT instance (usually called a *formula*) consists of a family $\mathcal{C}$ of subsets $C \subseteq \mathbf{L}$ of literals, called *clauses*. A binary assignment $\mathbf{x} \in \mathbb{B}^V$ *satisfies* a clause C, if at least one literal 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 in $\mathcal{C}$. It is easy to see that a clause C is satisfied by $\mathbf{x} \in \mathbb{B}^V$ if and only if $\prod_{u \in C} \overline{u} = 0$. Thus, MAX–SAT is equivalent to the problem

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

where $a_C = 1$ for all $C \in \mathcal{C}$. In the *weighted* maximum satisfiability problem there is also a nonnegative weight $a_C \in \mathbb{R}^+$ associated with each clause $C \in \mathcal{C}$, and the objective is to maximize the total weight of the satisfied clauses.

If the clauses C have at most k literals, then the (weighted) MAX–SAT problem is called the (weighted) MAX–k–SAT problem. In particular, the weighted MAX–2–SAT problem can be formulated as the optimization of a special quadratic *negaform*[3]:

$$\tau \left(\psi \right) = \max_{x \in B^V} \psi = \max_{x \in B^V} \left(\sum_{\{u\} \in \mathcal{C}} a_{\{u\}} \left(1 - \overline{u} \right) + \sum_{\{u,v\} \in \mathcal{C}} a_{\{u,v\}} \left(1 - \overline{u}\,\overline{v} \right) \right).$$

Let us denote the sum of the nontrivial coefficients of any given posiform θ as $\mathcal{A}\left(\theta\right) \overset{\text{def}}{=} \sum_{T \neq \emptyset} a_T$. If the previous negaform is denoted by ψ, then we consider the minimum of the quadratic posiform $\phi = \mathcal{A}\left(-\psi\right) - \psi$, i.e.

$$\tau\left(\phi\right) = \min_{x \in B^V} \phi = \min_{x \in B^V} \sum_{\{u\} \in \mathcal{C}} a_{\{u\}} \overline{u} + \sum_{\{u,v\} \in \mathcal{C}} a_{\{u,v\}} \overline{u}\,\overline{v}. \tag{24}$$

Since for any assignment of the formula $\mathcal{C}$, $\tau\left(\psi\right)$ is the maximum number of true clauses and $\nu\left(\phi\right)$ is the minimum number of false clauses, then it is simple to notice that $\tau\left(\psi\right) + \nu\left(\phi\right) = \mathcal{A}\left(-\psi\right) = \mathcal{A}\left(\phi\right)$.

In this work, we tested algorithm PREPRO in a set of random weighted and non–weighted MAX–2–SAT formulas proposed by Borchers and Furman [15]. These instances are publicly available on the Internet[4].

---

[3]In a similar way to a posiform, a negaform of a pseudo–Boolean function $f\left(x_1, \cdots, x_n\right)$ is defined as a polynomial $g\left(x_1, \overline{x}_1, \cdots, x_n, \overline{x}_n\right)$, taking the same values as f in every binary n–vector, and having the property that all its coefficients (with the possible exception of the free term) are non–positive

[4]MAXSAT, A Davis-Putnam like code for MAX-SAT Problems (12/18/04): http://www.nmt.edu/~borchers/maxsat.html

The list of problems contains 17 standard formulas and 17 formulas with weights associated to the clauses, ranging from one to ten. The parameters of the non-weighted formulas can be seen in Table A.7, and those of the weighted formulas can be seen in Table A.8. We solve the (weighted) MAX–2–SAT problem by associating to it a quadratic posiform ϕ (see (24)), for which the minimum value $\nu(\phi)$ is the minimum weighted set of unsatisfied clauses.

7 Computational Experiments

7.1 Test environment

The algorithm PREPRO was implemented in C++, compiled using the Microsoft Windows 32-bit C/C++ Optimizing Compiler (version 12) for 80x86, and linked with the Microsoft Incremental Linker (version 6).

Except for the experiments on large scale planar graphs (previously defined as the RUDY benchmark), all the remaining computations were carried out on a computer with a 2.80 GHz Pentium 4, 512 MB of RAM, and hyper-threading technology. The experiments on the large planar graphs were carried out on a computer system with 3.5 GB of RAM, and a 3.06 GHz Xeon. Both computers have installed the Windows XP Professional (version 2002) operating system.

7.2 Results

Given a posiform ϕ, its roof–dual bound, a set of strong/weak persistencies and a decomposition of type (16) of it, can easily be derived from the residual network resulting after aplying a maximum flow algorithm to the implication network associated to ϕ. We consider this preprocessing step (entirely based on the roof–duality theory) as a *standard* preprocessing tool in all experiments that we have carried out. We tested four possible preprocessing strategies:

A Only the standard tool is considered;

B Standard tool and coordination are considered;

C Standard tool and probing are considered; and

D All preprocessing tools are considered.

Since strategy **D** usually provides the best data reduction of the problems, we have included in Appendix B several statistical results about the preprocessing performance of this strategy in all benchmarks.

At first glance, we have tried to understand how any of the preprocessing techniques would impact in the reduction of the problem's size. Table 3 provides preprocessing results for the test beds, whose values are averages for groups of problems belonging to the same

family. Strategy **D** provides 100% data reduction for the following benchmarks: MAX–Clique problems in all HAM–2 graphs and all c–FAT graphs; minimum vertex cover problems in all PVC LEDA planar graphs; MAX–CUT problems in all VIA graphs. These results clearly indicate that one can expect getting an oustanding data reduction level in these special well structured problems.

It should be remarked that the border separating successful from unsuccessful preprocessing cases is very thin. For instance, all the Hamming graphs in HAM–2 were optimally solved with the standard preprocessing tool. However, in the closely related family HAM–4 there was no reduction found for any of the graphs, even when all the preprocessing tools were considered. We also remark the fact that strategy **C** provided optimallity for all MAX-Clique problems and all MAX–CUT problem in the VIA graphs, using a substantial smaller computing time than the one corresponding to strategy **D**. Strategy **C** with an average value of 99.9%, provided also a very good data reduction on the minimum vertex cover problems in the PVC LEDA planar graphs.

Table 4 suggests the particular preprocessing techniques which can be recommended for each of the problem families considered, in order to achieve (on the average) as high a data reduction as possible within a limited amount of time. In view of the relatively uniform behavior of problems within the same family, the recommendation of a common strategy for problems within a family seems reasonable. Here are some remarks and some recommendations for the examined groups:

- *Problems with prescribed density* – Coordination does not have a practical influence in the preprocessing results. Probing should be used in the cases where density is low. In general, the probing tool should be used in this class, if the condition $nd \leqslant 20$ is satisfied. Let us also note that family B consists of very dense submodular maximization problems, and for which the preprocessing outcome changed considerably, in comparison with the other problems. Six of the 10 problems in the B group were solved optimally, and for the unsolved cases, a large number of quadratic persistencies was found.

- *Minimum vertex cover of planar graphs* – Probing when used with the *coordination* method provides slighty better preprocessing data reduction, without degrading the computing times returned by probing only.

- *MAX–CUT in torus graphs* – The standard preprocessing tool should be used for the graphs with ± 1 weights in the edges (see also Table B.5). Probing should be used in the other weighted graphs.

- *MAX–CUT in VIA graphs* – All the problems in the VIA.CY family are solved optimally by the basic preprocessing tool (see also Table B.5). Every instance in this group of problems is solved in less than 0.2 seconds. The VIA.CN problems are all optimally solved with probing, taking an average computing time of 13.2 seconds. In two of the VIA.CN problems, the analysis of the starting implication network found 2 components which were preprocessed separately under the result of Proposition 16.

Table 3: Average QUBO simplifications and decomposition after preprocessing.

Type of Problem	Family Name	Roof-Duality		Roof-Duality and Coordination				Roof-Duality and Probing				ALL Tools			
		Total Time	Variab. Reduc.	Total Time	Relat. Gap	Quad. Rel.	Variab. Reduc.	Total Time	Relat. Gap	Quad. Rel.	Variab. Reduc.	Total Time	Relat. Gap	Quad. Rel.	Variab. Reduc.
Fixed Degree	A	0.0	52.3%	0.0	8.2%	1	52.9%	0.0	5.1%	1	64.0%	0.0	5.1%	1	64.0%
	B	0.0	0.9%	0.1	85.3%	1762	0.9%	1.2	34.9%	785	72.2%	2.9	41.9%	801	68.2%
	C	0.0	18.1%	0.0	22.4%	0	18.5%	0.1	19.8%	1	30.2%	0.1	19.8%	1	30.2%
	D	0.0	0.6%	0.0	56.8%	0	0.6%	1.2	55.4%	3	1.6%	1.2	55.4%	3	1.6%
	E	0.0	0.0%	0.1	57.9%	0	0.0%	5.2	57.2%	6	0.0%	5.3	57.2%	6	0.0%
	F1	1.1	0.0%	1.6	78.7%	0	0.0%	149.8	78.6%	0	0.0%	150.1	78.6%	0	0.0%
	ORL-50	0.0	94.2%	0.0	0.2%	2	94.4%	0.0	0.0%	0	100.0%	0.0	0.0%	0	100.0%
	ORL-100	0.0	4.9%	0.0	13.3%	0	5.2%	0.3	8.8%	26	36.3%	0.4	8.8%	26	36.3%
	ORL-250	0.0	0.0%	0.1	44.1%	0	0.0%	3.3	43.5%	1	0.0%	3.4	43.5%	1	0.0%
	ORL-500	0.1	0.0%	0.4	60.6%	0	0.0%	28.4	60.3%	0	0.0%	28.5	60.3%	0	0.0%
MAX Clique	C-FAT-200	0.0	0.0%	1.3	68.7%	29	0.0%	4.8	0.0%	0	100.0%	14.1	0.0%	0	100.0%
	C-FAT-500	0.1	0.0%	21.9	77.0%	56	0.0%	80.8	0.0%	0	100.0%	327.4	0.0%	0	100.0%
	HAM-2	0.0	100.0%	0.0	0.0%	0	100.0%	0.0	0.0%	0	100.0%	0.0	0.0%	0	100.0%
	HAM-4	0.1	0.0%	12.8	89.6%	0	0.0%	79.0	88.3%	0	0.0%	91.5	88.3%	0	0.0%
MIN Vert. Cov.	LEDA-1000	0.0	75.3%	0.1	0.0%	0	99.9%	0.1	0.0%	0	99.9%	0.1	0.0%	0	100.0%
	LEDA-2000	0.1	74.5%	0.2	0.0%	0	99.9%	0.2	0.0%	0	99.8%	0.2	0.0%	0	100.0%
	LEDA-3000	0.2	75.0%	0.3	0.0%	0	99.9%	0.3	0.0%	0	99.8%	0.3	0.0%	0	100.0%
	LEDA-4000	0.4	75.1%	0.5	0.0%	0	99.8%	0.6	0.0%	0	99.9%	0.5	0.0%	0	100.0%
MAX Cut	Torus	0.1	0.2%	6.4	39.2%	0	0.2%	220.0	38.3%	2	3.1%	853.3	38.3%	2	3.1%
	R	0.4	0.0%	64.7	28.9%	0	0.0%	1818.1	28.7%	1	0.2%	3187.8	28.7%	1	0.2%
	VIA.CN	0.1	4.0%	1.1	4.6%	3	4.0%	13.2	0.0%	0	100.0%	100.5	0.0%	0	100.0%
	VIA.CY	0.1	100.0%	0.1	0.0%	0	100.0%	0.1	0.0%	0	100.0%	0.1	0.0%	0	100.0%
	G500	0.0	4.2%	0.5	23.1%	3	4.2%	6.2	22.3%	5	19.0%	11.8	22.3%	5	19.0%
	G1000	0.1	2.7%	2.2	23.5%	5	2.7%	36.1	23.0%	7	18.8%	100.8	23.0%	7	18.8%
	U500	0.0	1.0%	2.6	35.2%	31	2.6%	9.1	36.0%	0	5.7%	14.3	33.8%	32	11.5%
	U1000	0.1	0.6%	14.7	35.2%	57	2.5%	39.0	35.9%	0	7.6%	76.2	34.2%	83	9.9%
MAX 2–Sat	BF-50	0.0	9.3%	0.0	240.0%	1	11.3%	0.1	121.2%	8	25.8%	0.1	121.2%	8	25.8%
	BF-100	0.0	12.6%	0.0	691.1%	1	13.8%	0.1	271.9%	24	27.6%	0.2	271.9%	25	27.6%
	BF-150	0.0	17.1%	0.0	908.1%	0	17.1%	0.3	258.3%	55	42.2%	0.5	258.3%	55	42.2%
	BFW-50	0.0	6.2%	0.0	391.4%	1	6.2%	0.1	133.2%	8	24.2%	0.1	133.2%	8	24.2%
	BFW-100	0.0	14.2%	0.0	1731.4%	5	14.2%	0.2	335.5%	39	26.2%	0.3	335.5%	39	26.2%
	BFW-150	0.0	17.1%	0.0	3214.3%	4	17.1%	0.7	263.1%	68	45.6%	0.9	265.2%	67	45.8%

Table 4: Preprocessing strategies recommended for the benchmarks.

Type of Problem	Family Name	Number of Instances	Density	Best Strategy
Fixed Degree	A	8	18.78%	**C**
	B	10	98.83%	**C**
	C	7	36.01%	**C**
	D	10	54.31%	**A**
	E	5	29.62%	**A**
	F1	5	51.56%	**A**
	ORL-50	10	9.75%	**C**
	ORL-100	10	9.75%	**C**
	ORL-250	10	9.94%	**A**
	ORL-500	10	9.90%	**A**
MAX Clique	C-FAT-200	3	77.82%	**C**
	C-FAT-500	4	83.28%	**C**
	HAM-2	3	4.55%	**A**
	HAM-4	3	39.42%	**A**
MIN Vertex Cover	LEDA-1000	100	0.41%	**D**
	LEDA-2000	100	0.20%	**D**
	LEDA-3000	100	0.14%	**D**
	LEDA-4000	100	0.10%	**D**
MAX Cut	Torus	4	0.68%	**C**
	R	8	0.34%	**C**
	VIA.CN	5	0.32%	**C**
	VIA.CY	5	0.37%	**A**
	G500	4	1.87%	**C**
	G1000	4	0.95%	**C**
	U500	4	3.40%	**D**
	U1000	4	1.72%	**D**
MAX 2–SAT	BF-50	9	19.98%	**C**
	BF-100	5	7.53%	**C**
	BF-150	3	3.91%	**C**
	BFW-50	9	21.18%	**C**
	BFW-100	5	7.77%	**C**
	BFW-150	3	3.93%	**C**

- *MAX–CUT in Gn.p graphs* – The preprocessing efficiency decreases with density for the graphs with the same number of vertices (see also Table B.5). Probing helped increasing data reduction for graphs with densities up to 5%, and attained "as expected" better performance for graphs with 500 vertices, than for those with 1000 vertices.

- *MAX–CUT in Un.p graphs* – The preprocessing efficiency decreases with density for the graphs with the same number of vertices (see also Table B.5). In this category, the standard preprocessing tool found some non trivial decomposition, and both coordination and probing helped improving the average data reduction rates from 2–3% to about 11%.

- *MAX–2–SAT* – The preprocessing efficiency decreases with the number of clauses when the number of variables is fixed (see also Table B.6). Both in the non–weighted and weighted formulas, the probing technique provided better reduction indicators.

In conclusion it can be seen that the choice "best strategy" is highly problem family dependent. It should also be remarked that only three out of the four examined strategies turn out to provide the "best" performance for some of the considered group of problems; strategy **B** (consisting of the application of the standard tool and coordination, but not of probing) did not give best results in any of the examined cases. Table 4 indicates the best recommended strategies for each of the examined families of problems.

8   Application: Optimal vertex covers of planar graphs

In view of the outstanding results obtained by applying PrePro to the minimum vertex cover problem in random planar graphs, we have tried to refine this method to the point where it would not only preprocess the problem but actually produce an optimal solution of it. As it will be seen in this section, the resulting method allowed the efficient detection of minimum vertex covers in planar graphs of impressive dimensions, including some having 500 000 vertices.

Although the vertex cover problem is known to be NP–hard in the class of planar graphs ([32]), our computational experiments with a large collection of benchmark planar graphs indicate that, in all likelihood, finding vertex covers in planar graphs may be frequently tractable. This conclusion provides the motivation for the work reported in this section.

Before presenting the results of our computational experiments we would like to emphasize that PrePro is not capable of solving the QUBO problems associated to *every* planar graph, and that it may encounter problems even in the case of very small graphs. For example there are no persistencies in the QUBO associated to the "toy box" graph of Figure 6.

(a) 3–dimensional drawing.   (b) Planar drawing.
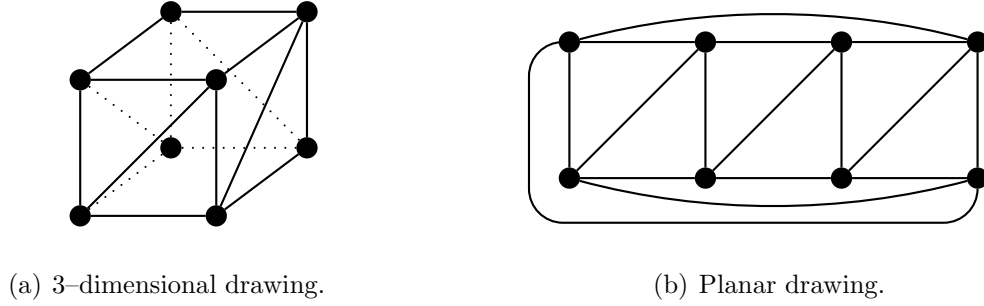
Figure 6: Planar graph for which PREPRO does not find any persistent result.

8.1   Deriving minimum vertex cover from QUBO's optimum

We have seen in Section 6.3 that finding a minimum vertex cover of a graph $G = (V, E)$ is a QUBO, and we have also noticed in Section 7.2 that out of the 400 QUBOs coming from vertex cover problems in randomly generated *planar* graphs, every single QUBO was solved to optimality by PREPRO. The only matter which needs special attention is that – due to the fact that in (22) we have fixed to zero the values of $\epsilon_{(i,j)}$ for every edge (i, j) – it may happen that the optimal 0–1 solution of a QUBO defines a vertex set which does not cover every edge. Let us prove now that there is a simple polynomial time transformation, which associates to the optimal solution of QUBO an optimal vertex cover of G.

Proposition 10. *Let $G = (V, E)$ be a graph, and let us associate to it the quadratic pseudo–Boolean function $f(x_1, \cdots, x_n) = \sum_{i \in V} x_i + \sum_{(i,j) \in E} \overline{x}_i \overline{x}_j$ and the QUBO (22). Let further $f(x_1^*, \cdots, x_n^*)$ be a minimum of f, and let S^* be the set of vertices $j \in V$ for which $x_j^* = 1$. Then, the size of a minimum vertex cover is $f(x_1^*, \cdots, x_n^*)$, and the set S^* can be enlarged to a minimum vertex cover $\widehat{S} \supseteq S^*$ in $O(|E|)$ time.*

Proof. If S^* is a vertex cover, it is clearly minimal, and since $\overline{x}_i^* \overline{x}_j^* = 0$ for every $(i, j) \in E$, the problem is solved. Let us assume that S^* is not a vertex cover, and let h and k be two adjacent vertices with $x_h^* = x_k^* = 0$. Let us consider now the vector $(x_1^{**}, \cdots, x_n^{**})$ defined by

$$x_l^{**} = \begin{cases} x_l^* & \text{if } l \neq k \\ 1 & \text{if } l = k \end{cases}.$$

If N_k^* is the neighborhood of k in the set $V \setminus S^*$, then clearly

$$f(x_1^{**}, \cdots, x_n^{**}) = f(x_1^*, \cdots, x_n^*) + 1 - \sum_{t \in N_k^*} \overline{x}_t^*,$$

and since $h \in N_k^*$, the set N_k^* is not empty, and therefore

$$f(x_1^{**}, \cdots, x_n^{**}) \leqslant f(x_1^*, \cdots, x_n^*).$$

On the other hand, from the minimality of f in $(x_1^*, \cdots, x_n^*)$ it follows that

$$f\left(x_1^{**}, \cdots, x_n^{**}\right) = f\left(x_1^*, \cdots, x_n^*\right).$$

By repeating this transformation several times (at most $|V \setminus S^*|$ times), the set S^* will be eventually transformed to a vertex cover $\widehat{S}$ of the graph G, which has to have a minimum size. □

While the above proposition holds in any graph, it is particularly useful in classes of graphs for which the associated QUBO (22) can be solved to optimality.

As a consequence of the discussion above, we have supplemented PREPRO with the simple procedure outlined in the proof of Proposition 10 to derive an optimal vertex cover from the optimal solution of the corresponding QUBO problem (22). This amended version of the proposed algorithm will be called PREPRO$^+$.

In this section the preprocessing strategies **A**, **B** and **C** of the previous section will not be considered, i.e. all the experiments below were carried out by using strategy **D**, which involves all the preprocessing tools of Section 7.2.

The results obtained by PREPRO$^+$ for moderately sized graphs (i.e. having up to a few thousand vertices), have been compared with those of the recent paper of Alber, Dorn and Niedermeier (or ADN in short) reported in [1], which is essentially based on the data reduction results of Nemhauser and Trotter [61].

8.2 Preprocessing

Table 5 provides averages of results obtained by preprocessing minimum vertex cover problems on 400 random planar graphs generated by the LEDA package (see Section 6.3). Four groups of 100 graphs each have been considered, each graph in these sets containing respectively 1000, 2000, 3000 and 4000 vertices.

Table 5: Comparative preprocessing results for minimum vertex cover problems in planar graphs.

Number of		Time (sec)		Variables Fixed (%)		Size of Residual Problem	
Graphs per Family	Vertices per Graph	ADN ([1])	PREPRO	ADN ([1])	PREPRO	ADN ([1])	PREPRO
100	1 000	4.06	0.05	68.4	100	315.8	0
100	2 000	12.24	0.16	67.4	100	652.9	0
100	3 000	30.90	0.27	65.5	100	1036.1	0
100	4 000	60.45	0.53	62.7	100	1492.9	0

Remarkably, PREPRO achieved 100% data reduction in all PVC LEDA graphs, whereas the ADN method obtained between 63% and 68% average data reduction on their LEDA benchmarks, which have similar characteristics to the PVC LEDA graphs (see Table 2). It can also be seen that the best performance of the ADN method (68.4% reduction of vertex set) occurs on the group of relatively smaller graphs, while the performance of PREPRO

(100% reduction of vertex set) does not seem to be influenced by the size of the graphs considered.

8.3   Optimization

While the results reported in Table 5 refer to the preprocessing by PREPRO of the minimum vertex cover problem, we shall discuss below the results of applying PREPRO$^+$ for actually finding optimal solutions to this problem.

It is important to remark that PREPRO$^+$ assumes the knowledge of the exact optimum of the associated QUBO. If this optimum is not available PREPRO$^+$ is further enhanced to an algorithm PREPRO*, by adding a branch–and–bound component to it, in order to handle minimum vertex cover problems even in this case. However, the use of PREPRO* turned out not to be necessary in any of the 400 test problems randomly generated with the LEDA software package, which were all solved to optimality without the branch–and–bound component having been called.

Table 6: Average computing times of optimal vertex covers for graphs belonging to the LEDA benchmarks.

Algorithm	ADN ([1])	PREPRO* in the PVC LEDA Benchmark	
Computer System (speed)	750 MHz Linux 720 MB RAM	500 MHz Pentium III Windows 98 96 MB RAM (slower)	2.8 GHz Pentium 4 Windows XP 512 MB RAM (faster)
1 000 vertices	5.75 sec	0.24 sec	0.06 sec
2 000 vertices	19.93 sec	0.64 sec	0.18 sec
3 000 vertices	51.54 sec	1.07 sec	0.31 sec
4 000 vertices	109.84 sec	1.71 sec	0.56 sec
Average Speedup		51 times	169 times

Table 6 provides comparative results for finding optimal vertex covers for graphs belonging to the LEDA benchmarks. It includes computing times for the exact algorithm of Alber, Dorn and Niedermeier [1] and solution times for PREPRO* (which coincide with PREPRO$^+$ for all the 400 cases). We would like to recall the fact that – not having had access to the test problems of [1] – we have randomly generated our problems, but made sure (as explained in Section 6.3) that the parameters used in the random graph generation process were chosen so as to match exactly does of [1].

In order to be able to differentiate between the acceleration due to computer systems and those due to algorithms, all the experiments reported in Table 6 have been carried out twice, first on a somewhat slower computer system (500 MHz Pentium III, 98 MB RAM, Windows 98) than the one used by [1] (715 MHz, 720 MB RAM, Linux), and second on a faster system (2.8 GHz Pentium 4, 512 MB RAM, Windows XP).

The basic conclusion of this set of experiments is that using the slower computer system, PREPRO* is about 50 times faster than that of [1], on average.

8.4 Minimum vertex covers of very large planar graphs

Based on the high efficiency of PREPRO* when applied to the optimization of vertex cover problems in planar graphs, we have investigated the possibility of using it on substantially larger planar graphs. The relevant experiments were carried out on the set of 36 benchmark problems contained in the RUDY list (described in Section 6.3), which contains graphs whose vertex sets contain 50 000, 100 000, 250 000 and 500 000 vertices, and have planar densities of 10%, 50% and 90%. For each particular number of vertices and each density the list contains three graphs.

Table 7: Average computing times over 3 experiments of optimal vertex covers for graphs belonging to the PVC RUDY benchmark.

	Planar Density		
Vertices	10%	50%	90%
50 000	1.2 min	3.7 min	1.8 min
100 000	4.8 min	16.2 min	7.4 min
250 000	30.4 min	107.7 min	48.2 min
500 000	124.7 min	422.4 min	195.3 min

Table 7 presents the average computing times needed by PREPRO* for finding minimum vertex cover sets in *all* the graphs contained in the RUDY list. Each of the computing times reported in the table represents the average needed for solving the three problems with a fixed number of vertices and a fixed planar density contained in the RUDY list. The average computing times range from 2.2 minutes for the graphs with 50 000 vertices up to 4.1 hours for the graphs with 500 000 vertices. Clearly, the computing times vary with the size of the vertex set. A similarly foreseable phenomenon happens with the dependency of computing times and densities. Indeed, the average computing time for the low density graphs is 40 minutes, for medium density graphs this increases to 2.3 hours, and for high density graphs it drops to 1 hour.

More detailed information about the performance of PREPRO can be read from the statistics presented in Table B.4 in the Appendix, where specific data are given for each of the 36 problems of the RUDY list. First, it can be seen that almost all of the computing time (78.7%) is spent on calculating the roof duals; moreover, most of this time (99.9%) is spent on calculating the very first roof dual.

The large investment of computing time in the calculation of roof duals brings returns in the form of graph size reductions (which are due to strong and weak persistency) and in the form of decompositions.

- The detailed analysis of the problem size reductions occurring in PREPRO shows that

roof–duality accounts for 99.8% of these reductions for planar graphs of density 10%, 93.2% for the 50% dense graphs, and 51.8% for the 90% dense graphs.

- It is interesting to note the extremely small size of the average components of the graphs left after applying decomposition and strong and weak persistency. Indeed, the average size of these components for graphs of 10%, 50% and 90% density is of 3.1, 4.4 and 14.4 vertices, respectively.

Beside roof–duality, important simplifications of the remaining QUBOs were obtained by the coordination method and by probing. It can be seen in column (n_e) of Table B.4 of the Appendix that the number of equality relations between pairs of variables or their complements, discovered by the coordination method is an increasing monotone function of planar density. Also, column (n_f) of Table B.4 shows that the number of variables whose values are fixed by probing reaches maximum values for the medium density graphs. In conclusion it can be seen that there is substantial complementarity in the effect of applying the basic preprocessing techniques considered in this paper. Indeed,

- 10% dense planar graphs derive almost the entire solution from the application of roof–duality;

- 50% dense planar graphs derive a considerable reduction through probing; and

- 90% dense planar graphs derive a considerable reduction through the coordination method.

However, the most important conclusion is that PREPRO$^+$ found optimal vertex covers for all the 36 benchmarks in the RUDY list.

9   Conclusions

This study is devoted to the systematic simplification of QUBOs. The proposed method uses enhanced versions of several basic techniques (e.g., extraction of conclusions from the analysis of first and second order derivatives [39], and from roof–duality [40]) and several integrative techniques (e.g., probing, consensus) for combining the conclusions provided by the basic techniques. The application of these techniques is implemented using the network flow model of [18, 20].

The use of the proposed preprocessing techniques provides:

(*i*)   A lower bound of the minimum of the objective function;

(*ii*)   The values of some of the variables in some or every optimum;

(*iii*)   Binary relations (equations, inequalities, or non-equalities) between the values of certain pairs of variables in some or every optimum;

(*iv*) The decomposition (if possible) of the original problem into several smaller pairwise independent minimization problems.

The efficiency of the proposed techniques is demonstrated through numerous computational experiments carried both on benchmark problems and on randomly generated ones.

The simplifications obtained with the proposed methods exceed substantially those reported in the literature. An interesting example is the minimum vertex cover problem for which [1] reports a preprocessing stage reduction of dimensionality by 62.7%–68.4%, while the method proposed here achieves 100% reduction (i.e. exact solution) in each of the test problems. Moreover, while the computing times reported in [1] for finding optimal vertex covers for graphs from 1 000 to 4 000 vertices range from 5.75 to 109.84 seconds, those required by the proposed method range from 0.24 to 1.71 seconds using a somewhat slower computer, or from 0.06 to 0.56 seconds using a somewhat faster one.

The experiments show that the methods can be applied successfully to problems of unusually large size, for instance:

- MAX–Clique on graphs derived from fault diagnosis having up to 500 vertices;

- MAX–CUT problems on graphs derived from VLSI design having thousands of vertices;

- Minimum vertex cover problems on randomly generated planar graphs (an NP-hard problem [32]) having up to 500 000 vertices.

It should be added that the proposed preprocessing technique have not only simplified the above problems but have in fact produced their exact optimum solutions. As far as we know there are no reports in the literature about methods capable of providing optimal solutions to vertex cover problems in planar graphs with the investigated sizes.

References

[1] Alber, J., F. Dorn and R. Niedermeier. Experimental evaluation of a tree decomposition based algorithm for vertex cover on planar graphs. *Discrete Applied Mathematics* **145**, (2005), pp. 219–231.

[2] Alidaee, B., G.A. Kochenberger and A. Ahmadian. 0-1 quadratic programming approach for the optimal solution of two scheduling problems. *International Journal of Systems Science* **25**, (1994), pp. 401–408.

[3] Amini, M.M., B. Alidaee and G.A. Kochemberger. A scatter search approach to unconstrained quadratic binary programs. In *New ideas in optimisation* (D. Corne, M. Dorigo and F. Glover, eds.), pp. 317–329 (McGraw-Hill, London, 1999).

[4] Aspvall, B., M.F. Plass and R.E. Tarjan. A linear time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters* **8**, (1979), pp. 121–123.

[5] Atamtürk, A., G.L. Nemhauser and M.W.P. Savelsbergh. Conflict graphs in solving integer programming problems. *European Journal of Operational Research* **121**, (2000), pp. 40–55.

[6] Axehill, D. and A. Hansson. *A preprocessing algorithm for MIQP solvers with applications to MPC.* Tech. rep., Department of Electrical Engineering, Linköping University (2004).

[7] Badics, T. *Approximation of some nonlinear binary optimization problems.* Ph.D. thesis, RUTCOR, Rutgers University (1996).

[8] Badics, T. and E. Boros. Minimization of half-products. *Mathematics of Operations Research* **23**, (1998), pp. 649–660.

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

[10] Barahona, F., M. Jünger and G. Reinelt. Experiments in quadratic 0-1 programming. *Mathematical Programming* **44**, (1989), pp. 127–137.

[11] Beasley, J.E. *Heuristic algorithms for the unconstrained binary quadratic programming problem.* Tech. rep., Management School, Imperial College, London, UK (1998).

[12] Berman, P. and A. Pelc. Distributed fault diagnosis for multiprocessor systems. In *Proceedings of the 20th annual international symposium on fault-tolerant computing* (Newcastle, UK, 1990), pp. 340–346.

[13] Billionnet, A. and B. Jaumard. A decomposition method for minimizing quadratic pseudo-boolean functions. *Operations Research Letters* **8**(3), (1989), pp. 161–163.

[14] Billionnet, A. and A. Sutter. Minimization of a quadratic pseudo-boolean function. *European Journal of Operational Research* **78**, (1994), pp. 106–115.

[15] Borchers, B. and J. Furman. A two-phase exact algorithm for max-sat and weighted max-sat problems. *Journal of Combinatorial Optimization* **2**(4), (1998), pp. 299–306.

[16] Boros, E. and P.L. Hammer. *A max-flow approach to improved roof-duality in quadratic 0-1 minimization.* Research Report RRR 15-1989, RUTCOR, Rutgers University (1989).

[17] 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), pp. 151–180.

[18] Boros, E. and P.L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics* **123**, (2002), pp. 155–225.

[19] 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.

[20] Boros, E., P.L. Hammer, R. Sun and G. Tavares. *A max-flow approach to improved lower bounds for quadratic 0-1 minimization.* Research Report RRR 7-2006, RUTCOR, Rutgers University (2006).

[21] Boros, E., P.L. Hammer and X. Sun. *The DDT method for quadratic 0-1 optimization.* Research Report RRR 39-1989, RUTCOR, Rutgers University (1989).

[22] Boros, E., P.L. Hammer and G. Tavares. *Probabilistic One-Pass Heuristics for Quadratic Unconstrained Binary Optimization (QUBO).* Technical Report 1-2006, RUTCOR, Rutgers University (2006).

[23] Burer, S., R.D.C. Monteiro and Y. Zhang. Rank-two relaxation heuristics for max-cut and other binary quadratic programs. *SIAM Journal on Optimization* **12**(2), (2001), pp. 503–521.

[24] Bushnell, M.L. and I.P. Shaik. Robust delay fault built-in self-testing method and apparatus. *United States Patent # 5,422,891* .

[25] Carter, M.W. The indefinite zero-one quadratic problem. *Discrete Applied Mathematics* **7**, (1984), pp. 23–44.

[26] Crama, Y. and P.L. Hammer. *Boolean functions: Theory, algorithms and applications* (Cambridge University Press, 2006, forthcoming).

[27] Crama, Y. 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), pp. 99–128.

[28] Festa, P., P.M. Pardalos, M.G.C. Resende and C.C. Ribeiro. Randomized heuristics for the max-cut problem. *Optimization Methods and Software* **7**, (2002), pp. 1033–1058.

[29] Fraenkel, A.S. and P.L. Hammer. Pseudo-boolean functions and their graphs. *Annals of Discrete Mathematics* **20**, (1984), pp. 137–146.

[30] Gallo, G., P.L. Hammer and B. Simeone. Quadratic knapsack problems. *Mathematical Programming* **12**, (1980), pp. 132–149.

[31] Garey, M.R. and D.S. Johnson. *Computers and intractability: An introduction to the theory of NP-completeness* (W. H. Freeman, San Francisco, 1979).

[32] Garey, M.R., D.S. Johnson and L. Stockmeyer. Some simplified np-complete graph problems. *Theoretical Computer Science* **1**, (1976), pp. 237–267.

[33] Glover, F., B. Alidaee, C. Rego and G. Kochenberger. One-pass heuristics for large-scale unconstrained binary quadratic problems. *European Journal of Operational Research* **137**, (2002), pp. 272–287.

[34] Glover, F., G. Kochenberger and B. Alidaee. Adaptative memory tabu search for binary quadratic programs. *Management Science* **44**(3), (1998), pp. 336–345.

[35] Glover, F., G.A. Kochenberger, B. Alidaee and M. Amini. Tabu search with critical event memory: An enhanced application for binary quadratic programs. In *Meta-heuristics - Advances and trends in local search paradigms for optimization* (S. Voss, S. Martello, I. Osman and C. Roucairol, eds.), pp. 83–109 (Kluwer Academic Publishers, 1998).

[36] Gulati, S.K., S.K. Gupta and A.K. Mittal. Unconstrained quadratic bivalent programming problem. *European Journal of Operational Research* **15**, (1980), pp. 121–125.

[37] Hammer, P.L. Plant location - a pseudo-boolean approach. *Israel Journal of Technology* **6**, (1968), pp. 330–332.

[38] Hammer, P.L. Pseudo-boolean remarks on balanced graphs. *International Series of Numerical Mathematics* **36**, (1977), pp. 69–78.

[39] Hammer, P.L. and P. Hansen. Logical relations in quadratic 0-1 programming. *Romanian Journal of Pure and Applied Mathematics* **26**(3), (1981), pp. 421–429.

[40] 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.

[41] Hammer, P.L. and S. Rudeanu. *Boolean methods in operations research and related areas* (Springer-Verlag, Berlin, Heidelberg, New York, 1968).

[42] Hammer, P.L. and E. Shliffer. Applications of pseudo-boolean methods to economic problems. *Theory and Decision* **1**, (1971), pp. 296–308.

[43] Hasselberg, J., P.M. Pardalos and G. Vairaktarakis. Test case generators and computational results for the maximum clique problem. *Journal of Global Optimization* **3**(4), (1993), pp. 463–482.

[44] Helmberg, C. and F. Rendl. Solving quadratic (0,1)-problems by semidefinite programs and cutting planes. *Mathematical Programming* **82**, (1998), pp. 291–315.

[45] Hillier, F.S. *The evaluation of risky interrelated investments* (North-Holland, Amsterdam, 1969).

[46] Homer, S. and M. Peinado. Design and performance of parallel and distributed approximation algorithms for maxcut. *Journal of Parallel and Distributed Computing* **46**, (1997), pp. 48–61.

[47] Johnson, D.S. and M.A. Trick. Cliques, coloring, and satisfiability. In *2nd DIMACS Implementation Challenge in New Brunswick, NJ, October 11-13, 1993* (D.S. in Discrete Mathematics and Theoretical Computer Science (26), eds.) (American Mathematical Society, Providence, RI, 1996).

[48] Jünger, M., A. Martin, G. Reinelt and R. Weismantel. Quadratic 0-1 optimization and a decomposition approach for the placement of electronic circuits. *Mathematical Programming* **63**, (1994), pp. 257–279.

[49] Kalantari, B. and A. Bagchi. An algorithm for quadratic zero-one programs. *Naval Research Logistics* **37**, (1990), pp. 527–538.

[50] Katayama, K. and H. Narihisa. Performance of simulated annealing-based heuristic for the unconstrained binary quadratic programming problem. *European Journal of Operational Research* **134**, (2001), pp. 103–119.

[51] Kim, S.H., Y.H. Kim and B.R. Moon. A hybrid genetic algorithm for the max cut problem. In *Genetic and Evolutionary Computation Conference* (2001), pp. 416–423.

[52] Krarup, J. and P.M. Pruzan. Computer-aided layout design. *Mathematical Programming Study* **9**, (1978), pp. 75–94.

[53] Kubiak, W. New results on the completion time variance minimization. *Discrete Applied Mathematics* **58**, (1995), pp. 157–168.

[54] Laughhunn, D.J. Quadratic binary programming with applications to capital budgeting problems. *Operations Research* **18**, (1970), pp. 454–461.

[55] Laughhunn, D.J. and D.E. Peterson. Computational experience with capital expenditure programming models under risk. *J. Business Finance* **3**, (1971), pp. 43–48.

[56] MacWilliams, F.J. and N.J.A. Sloane. *The theory of error-correcting codes* (North-Holland, Amsterdam, 1979).

[57] McBride, R.D. and J.S. Yormark. An implicit enumeration algorithm for quadratic integer programming. *Management Science* **26**, (1980), pp. 282–296.

[58] Mehlhorn, K. and S. Näher. *The LEDA platform of combinatorial and geometric computing* (Cambridge University Press, Cambridge, England, 1999).

[59] Merz, P. and B. Freisleben. Genetic algorithms for binary quadratic programming. In *Proceedings of the 1999 international Genetic and Evolutionary Computation Conference (GECCO'99)* (1999), pp. 417–424.

[60] Merz, P. and B. Freisleben. Greedy and local search heuristics for the unconstrained binary quadratic programming problem. *Journal of Heuristics* **8**(2), (2002), pp. 197–213.

[61] Nemhauser, G.L. and L.E. Trotter. Vertex packings: Structural properties and algorithms. *Mathematical Programming* **8**, (1975), pp. 232–248.

[62] Palubeckis, G. A heuristic-based branch and bound algorithm for unconstrained quadratic zero-one programming. *Computing* **54**, (1995), pp. 283–301.

[63] Palubeckis, G. Steepest ascent: A generalization of the greedy algorithm for the maximum clique problem. *Information Technology and Control* **28**(3), (2003), pp. 7–13.

[64] Palubeckis, G. Multistart tabu search strategies for the unconstrained binary quadratic optimization problem. *Annals of Operations Research* **131**, (2004), pp. 259–282.

[65] Palubeckis, G. and V. Krivickiene. Application of multistart tabu search to the max-cut problem. *Information Technology and Control* **31**(2), (2004), pp. 29–35.

[66] Palubeckis, G. and A. Tomkevièius. Grasp implementations for the uncostrained binary quadratic optimization problem. *Information Technology and Control* **24**(3), (2002), pp. 14–20.

[67] Papaioannou, S.G. Optimal test generation in combinational networks by pseudo-boolean programming. *IEEE Transactions on Computers* **26**, (1977), pp. 553–560.

[68] Pardalos, P.M. and G.P. Rodgers. Computational aspects of a branch and bound algorithm for quadratic 0-1 programming. *Computing* **45**, (1990), pp. 131–144.

[69] Pardalos, P.M. and G.P. Rodgers. A branch and bound algorithm for the maximum clique problem. *Computers and Operations Research* **19**, (1992), pp. 363–375.

[70] Pardalos, P.M. and J. Xue. The maximum clique problem. *Journal of Global Optimization* **4**, (1994), pp. 301–328.

[71] Phillips, A.T. and J.B. Rosen. A quadratic assignment formulation for the molecular conformation problem. *Journal of Global Optimization* **4**, (1994), pp. 229–241.

[72] Picard, J.C. and H.D. Ratliff. Minimum cuts and related problems. *Networks* **5**, (1975), pp. 357–370.

[73] Picard, J.C. and H.D. Ratliff. A cut approach to the rectilinear facility location problem. *Operations Research* **26**, (1978), pp. 422–433.

[74] Ranyard, R.H. An algorithm for maximum likelihood ranking and slater's i from paired comparisons. *British Journal of Mathematical and Statistical Psychology* **29**, (1976), pp. 242–248.

[75] Rao, M.R. Cluster analysis and mathematical programming. *Journal of the American Statistical Association* **66**, (1971), pp. 622–626.

[76] Rinaldi, G. Rudy: A generator for random graphs (1996).

[77] Savelsbergh, M.W.P. Preprocessing and probing for mixed integer programming problems. *ORSA Journal on Computing* **6**, (1994), pp. 445–454.

[78] Shaik, I.P. *An optimization approach to robust delay-fault built-in testing.* Ph.D. thesis, Electrical Engineering Department, Rutgers University (1996).

[79] Simeone, B. *Quadratic 0-1 programming, Boolean functions and graphs.* Ph.D. thesis, University of Waterloo (1979).

[80] Simone, C.D., M. Diehl, M. Jünger, P. Mutzel, G. Reinelt and R. G. 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.

[81] Sloane, N.J.A. Unsolved problems in graph theory arising from the study of codes. *Graph Theory Notes of New York XVIII* pp. 11–20.

[82] Sun, X. *Combinatorial algorithms for boolean and pseudo-Boolean functions.* Ph.D. thesis, RUTCOR, Rutgers University (1991).

[83] Tarjan, R.E. Depth-first search and linear graph algorithms. *SIAM Journal of Computing* **1**, (1972), pp. 146–160.

[84] Warszawski, A. Pseudo-boolean solutions to multidimensional location problems. *Operations Research* **22**, (1974), pp. 1081–1085.

[85] Weingartner, H.M. Capital budgeting of interrelated projects: Survey and synthesis. *Management Science* **12**, (1966), pp. 485–516.

[86] Williams, A.C. *Quadratic 0-1 programming using the roof dual with computational results.* Research Report RRR 8-1985, RUTCOR, Rutgers University (December 1985).

Appendix

A   Characteristics of Benchmarks

Table 1: QUBO problems of Glover, Kochenberger and Alidaee [34].

Sub-Family	Problem Number	Variables (n)	Density (d) (%)	Best known maximum (x*)		
				$f(\mathbf{x}^*)$	1st reporter	Optimal?
A	1	50	10	3 414	[34]	yes
	2	60	10	6 063	[34]	yes
	3	70	10	6 037	[34]	yes
	4	80	10	8 598	[34]	yes
	5	50	20	5 737	[34]	yes
	6	30	40	3 980	[34]	yes
	7	30	50	4 541	[34]	yes
	8	100	6.25	11 109	[34]	yes
B	1	20	100	133	[34]	yes
	2	30	100	121	[34]	yes
	3	40	100	118	[34]	yes
	4	50	100	129	[34]	yes
	5	60	100	150	[34]	yes
	6	70	100	146	[34]	yes
	7	80	100	160	[34]	yes
	8	90	100	145	[34]	yes
	9	100	100	137	[34]	yes
	10	125	100	154	[34]	yes
C	1	40	80	5 058	[34]	yes
	2	50	60	6 213	[34]	yes
	3	60	40	6 665	[34]	yes
	4	70	30	7 398	[34]	yes
	5	80	20	7 362	[34]	yes
	6	90	10	5 824	[34]	yes
	7	100	10	7 225	[34]	yes
D	1	100	10	6 333	[34]	yes
	2	100	20	6 579	[34]	yes
	3	100	30	9 261	[34]	yes
	4	100	40	10 727	[34]	yes
	5	100	50	11 626	[34]	yes
	6	100	60	14 207	[34]	yes
	7	100	70	14 476	[34]	yes
	8	100	80	16 352	[34]	yes
	9	100	90	15 656	[34]	yes
	10	100	100	19 102	[34]	yes
E	1	200	10	16 464	[34]	yes
	2	200	20	23 395	[34]	n/a
	3	200	30	25 243	[34]	n/a
	4	200	40	35 594	[34]	n/a
	5	200	50	35 154	[34]	n/a
F_1	1	500	10	61 194	[34]	n/a
	2	500	25	100 161	[34]	n/a
	3	500	50	138 035	[34]	n/a
	4	500	75	172 771	[34]	n/a
	5	500	100	190 507	[34]	n/a

Table 2: QUBO problems of Beasley [11].

Sub-Family	Problem Number	Variables (n)	Best known maximum (x*)		
			f (x*)	1ˢᵗ reporter	Optimal?
ORL-50	1	50	2 098	[11]	yes
	2	50	3 702	[11]	yes
	3	50	4 626	[11]	yes
	4	50	3 544	[11]	yes
	5	50	4 012	[11]	yes
	6	50	3 693	[11]	yes
	7	50	4 520	[11]	yes
	8	50	4 216	[11]	yes
	9	50	3 780	[11]	yes
	10	50	3 507	[11]	yes
ORL-100	1	100	7 970	[11]	yes
	2	100	11 036	[11]	yes
	3	100	12 723	[11]	yes
	4	100	10 368	[11]	yes
	5	100	9 083	[11]	yes
	6	100	10 210	[11]	yes
	7	100	10 125	[11]	yes
	8	100	11 435	[11]	yes
	9	100	11 455	[11]	yes
	10	100	12 565	[11]	yes
ORL-250	1	250	45 607	[11]	yes
	2	250	44 810	[11]	yes
	3	250	49 037	[11]	yes
	4	250	41 274	[11]	yes
	5	250	47 961	[11]	yes
	6	250	41 014	[11]	n/a
	7	250	46 757	[11]	yes
	8	250	35 726	[11]	n/a
	9	250	48 916	[11]	yes
	10	250	40 442	[11]	yes
ORL-500	1	500	116 586	[11]	n/a
	2	500	128 339	[59]	n/a
	3	500	130 812	[11]	n/a
	4	500	130 097	[11]	n/a
	5	500	125 487	[11]	n/a
	6	500	121 772	[59]	n/a
	7	500	122 201	[11]	n/a
	8	500	123 559	[11]	n/a
	9	500	120 798	[11]	n/a
	10	500	130 619	[11]	n/a

Table 3: c–fat and Hamming graphs from the DIMACS benchmark for MAX–Clique.

Sub-Family	Problem Name	Vertices (|V|)	Edges (|E|)	Density (d %)	Maximum Clique
C–FAT	c–fat200–1	200	1 534	92.29	12
	c–fat200–2	200	3 235	83.74	24
	c–fat200–5	200	8 473	57.42	58
	c–fat500–1	500	4 459	96.43	14
	c–fat500–2	500	9 139	92.67	26
	c–fat500–5	500	23 191	81.41	64
	c–fat500–10	500	46 627	62.62	126
Hamming	hamming6-2	64	1 824	9.52	32
	hamming8-2	256	31 616	3.14	128
	hamming10-2	1024	518 656	0.98	512
	hamming6-4	64	704	65.08	4
	hamming8-4	256	20 864	36.08	16
	hamming10-4	1024	434 176	17.11	$\geqslant$40

Table 4: Torus graphs.

Problem Name	Vertices ($|V|$)	Edges ($|E|$)	Density (d %)	Maximum Cut $W\left(S,\overline{S}\right)$	
				$W\left(S,\overline{S}\right)$	1st reporter
pm3–8–50	512	1 536	1.17	⩾458	[65]
pm3–15–50	3375	10 125	0.18	⩾3 016	[65]
g3–8	512	1 536	1.17	41 684 814	[23]
g3–15	3375	10 125	0.18	⩾285 790 637	[23]

Table 5: Graphs of Homer and Peinado [46].

Family	Problem Name	Vertices ($|V|$)	Edges ($|E|$)	Density (d %)	Maximum Cut $W\left(S,\overline{S}\right)$	
					$W\left(S,\overline{S}\right)$	1st reporter
random	R1000	1000	5 033	1.01	⩾3 687	[51]
	R2000	2000	9 943	0.50	⩾7 308	[51]
	R3000	3000	14 965	0.33	⩾10 997	[46]
	R4000	4000	19 939	0.25	⩾14 684	[46]
	R5000	5000	24 794	0.20	⩾18 225	[46]
	R6000	6000	29 862	0.17	⩾21 937	[46]
	R7000	7000	35 110	0.14	⩾25 763	[46]
	R8000	8000	39 642	0.12	⩾29 140	[51]
via	via.c1n	828	1 389	0.41	6 150	[46]
	via.c2n	980	1 712	0.36	7 098	[46]
	via.c3n	1327	2 393	0.27	6 898	[46]
	via.c4n	1366	2 539	0.27	10 098	[46]
	via.c5n	1202	2 129	0.29	7 956	[46]
	via.c1y	829	1 693	0.49	7 746	[46]
	via.c2y	981	2 039	0.42	8 226	[46]
	via.c3y	1328	2 757	0.31	9 502	[46]
	via.c4y	1367	2 848	0.31	12 516	[46]
	via.c5y	1203	2 452	0.34	10 248	[46]

Table 6: Graphs of S. Kim, Y. Kim and Moon [51].

Family	Problem Name	Vertices ($|V|$)	Edges ($|E|$)	Density (d %)	Maximum Cut $\left(S,\overline{S}\right)$	
					$\left(S,\overline{S}\right)$	1st reporter
random	g500.2.5	500	625	0.50	574	[51]
	g500.05	500	1 223	0.98	⩾1 008	[51]
	g500.10	500	2 355	1.89	⩾1 735	[51]
	g500.20	500	5 120	4.10	⩾3 390	[51]
	g1000.2.5	1000	1 272	0.25	⩾1 173	[51]
	g1000.05	1000	2 496	0.50	⩾2 053	[51]
	g1000.10	1000	5 064	1.01	⩾3 705	[20]
	g1000.20	1000	10 107	2.02	⩾6 729	[51]
geometric	U500.05	500	1 282	1.03	900	[51]
	U500.10	500	2 355	1.89	⩾1 546	[51]
	U500.20	500	4 549	3.65	⩾2 783	[51]
	U500.40	500	8 793	7.05	⩾5 181	[51]
	U1000.05	1000	2 394	0.48	⩾1 711	[51]
	U1000.10	1000	4 696	0.94	⩾3 073	[51]
	U1000.20	1000	9 339	1.87	⩾5 737	[51]
	U1000.40	1000	18 015	3.61	⩾10 560	[51]

Table 7: MAX–2–SAT formulas of Borchers and Furman [15].

Sub-Family	Problem (φ) Name	Variables (n)	Clauses (A(φ))	Density (d %)	False Clauses (ν(φ))
BF–50	BF–50–100	50	100	7.59	4
	BF–50–150	50	150	10.86	8
	BF–50–200	50	200	14.37	16
	BF–50–250	50	250	17.22	22
	BF–50–300	50	300	20.90	32
	BF–50–350	50	350	23.10	41
	BF–50–400	50	400	25.22	45
	BF–50–450	50	450	29.88	63
	BF–50–500	50	500	30.69	66
BF–100	BF–100–200	100	200	3.88	5
	BF–100–300	100	300	5.88	15
	BF–100–400	100	400	7.47	29
	BF–100–500	100	500	9.56	44
	BF–100–600	100	600	10.85	65
BF–150	BF–150–300	150	300	2.67	4
	BF–150–450	150	450	3.94	22
	BF–150–600	150	600	5.12	38

Table 8: Weighted MAX–2–SAT formulas of Borchers and Furman [15].

Sub-Family	Problem (φ) Name	Variables (n)	Clauses Number	Weight (A(φ))	Density (d %)	ρ %	p %	False Clauses Weight (ν(φ))
BFW–50	BFW–50–100	50	100	554	7.76	51.31	57.68	16
	BFW–50–150	50	150	800	11.18	49.91	50.66	34
	BFW–50–200	50	200	1,103	15.18	51.11	44.03	69
	BFW–50–250	50	250	1,361	18.94	50.68	45.52	96
	BFW–50–300	50	300	1,634	21.06	49.98	36.85	132
	BFW–50–350	50	350	1,936	24.57	48.80	38.95	211
	BFW–50–400	50	400	2,204	27.51	53.01	33.96	211
	BFW–50–450	50	450	2,519	30.53	52.09	36.18	257
	BFW–50–500	50	500	2,820	33.88	48.74	29.58	318
BFW–100	BFW–100–200	100	200	1,103	3.94	48.71	65.70	7
	BFW–100–300	100	300	1,634	5.92	51.43	49.81	67
	BFW–100–400	100	400	2,204	7.90	52.17	43.89	119
	BFW–100–500	100	500	2,820	9.62	51.63	37.75	241
	BFW–100–600	100	600	3,369	11.47	49.17	39.59	266
BFW–150	BFW–150–300	150	300	1,634	2.65	50.80	57.71	24
	BFW–150–450	150	450	2,519	3.92	50.88	51.77	79
	BFW–150–600	150	600	3,369	5.23	50.43	44.44	189

B   Preprocessing Statistics

Several statistical counters were included in the algorithm PREPRO. Let us remark that all the computing times presented in this Appendix include the time needed for this statistical gathering. Next, we describe the statistical counters that we have placed in PREPRO:

- t_n –   Time in seconds used by NETWORK;

- n_s –   Number of strong persistencies found by NETWORK;

- n_w –   Number of weak persistencies found by NETWORK;

- t_p –   Time used by PROBING in seconds;

- n_b –   Number of persistencies found by bounding in PROBING;

- n_f –   Number of linear persistencies found by quadratic consensus in PROBING;

- n_e –   Number of equality relations found by quadratic consensus in PROBING;

- t_c –   Time used by COORDINATION in seconds;

- n_c –   Number of equality relations found in COORDINATION;

We have also obtained several statistics immediately after the execution of PREPRO. Namely:

- t –   Total computing time of PREPRO in seconds;

- q –   Number of quadratic persistencies of the resulting posiform;

- g –   Relative gap $\frac{|U-L|}{|U|}$ between the upper bound U and the lower bound L returned by PREPRO;

- c –   Number of strong components of the resulting posiform;

- s –   Number of variables of the largest strong component of the resulting posiform;

- r –   Relative percentage of the number of variables fixed by PREPRO.

Table 1: PREPRO statistical report on the QUBO problems of Glover, Kochenberger and Alidaee [34].

		Preprocessing Tools Statistics									After Preprocessing				
		Roof-Duality			Probing				Coordination		Total	Quad.	Relat.	Comp.	Left	Variab.
Problem		Time	Strong	Weak	Time	Bound.	Fix.	Eq.	Time	Eq.	Time	Rel.	Gap	Num.	Larg.	Reduc.
Family	Numb.	(t_n)	(n_s)	(n_w)	(t_p)	(n_b)	(n_f)	(n_e)	(t_c)	(n_c)	(t)	(q)	(g)	(c)	(s)	(r)
A	1	0.0	48	2	0.0	0	0	0	0.0	0	0.0	0	0.0%	0	0	100.0%
	2	0.0	60	0	0.0	0	0	0	0.0	0	0.0	0	0.0%	0	0	100.0%
	3	0.0	17	0	0.0	3	31	19	0.1	0	0.1	0	0.0%	0	0	100.0%
	4	0.0	78	0	0.0	0	0	0	0.0	2	0.0	0	0.0%	0	0	100.0%
	5	0.0	5	1	0.0	0	0	0	0.0	0	0.0	4	8.7%	1	44	12.0%
	6	0.0	0	0	0.0	0	0	0	0.0	0	0.0	2	15.9%	1	30	0.0%
	7	0.0	0	0	0.0	0	0	0	0.0	0	0.0	0	16.4%	1	30	0.0%
	8	0.0	100	0	0.0	0	0	0	0.0	0	0.0	0	0.0%	0	0	100.0%
B	1	0.0	8	0	0.0	11	0	0	0.0	1	0.0	0	0.0%	0	0	100.0%
	2	0.0	9	0	0.0	18	0	1	0.0	2	0.1	0	0.0%	0	0	100.0%
	3	0.0	6	1	0.0	28	0	3	0.1	2	0.1	0	0.0%	0	0	100.0%
	4	0.0	1	1	0.2	13	1	0	0.2	0	0.4	466	77.9%	1	34	32.0%
	5	0.0	1	0	0.2	39	0	0	0.4	0	0.6	141	66.3%	1	20	66.7%
	6	0.1	6	0	0.3	62	0	1	0.8	1	1.2	0	0.0%	0	0	100.0%
	7	0.1	4	0	0.7	67	5	4	1.5	0	2.4	0	0.0%	0	0	100.0%
	8	0.1	0	1	1.8	31	0	0	2.2	0	4.2	1325	90.0%	1	58	35.6%
	9	0.1	0	1	2.8	20	0	0	2.8	0	5.8	2585	92.0%	1	79	21.0%
	10	0.3	0	3	6.1	30	0	0	7.5	0	14.0	3495	93.1%	1	92	26.4%
C	1	0.0	0	0	0.1	0	0	0	0.0	0	0.1	0	36.0%	1	40	0.0%
	2	0.0	0	0	0.1	0	0	0	0.0	0	0.1	0	36.5%	1	50	0.0%
	3	0.0	0	0	0.1	0	0	0	0.0	0	0.1	0	27.7%	1	60	0.0%
	4	0.0	0	0	0.1	0	0	0	0.0	0	0.1	2	24.1%	1	70	0.0%
	5	0.0	6	0	0.4	2	0	1	0.0	0	0.4	5	14.7%	1	71	11.3%
	6	0.0	87	0	0.0	3	0	0	0.0	0	0.0	0	0.0%	0	0	100.0%
	7	0.0	100	0	0.0	0	0	0	0.0	0	0.0	0	0.0%	0	0	100.0%
D	1	0.0	8	0	0.6	6	0	2	0.1	0	0.8	19	6.6%	1	84	16.0%
	2	0.0	0	0	0.3	0	0	0	0.0	0	0.3	14	45.7%	1	100	0.0%
	3	0.0	0	0	0.4	0	0	0	0.0	0	0.4	0	47.5%	1	100	0.0%
	4	0.0	0	0	0.7	0	0	0	0.0	0	0.7	0	56.1%	1	100	0.0%
	5	0.0	0	0	0.9	0	0	0	0.0	0	0.9	0	62.1%	1	100	0.0%
	6	0.0	0	0	1.2	0	0	0	0.0	0	1.3	0	61.3%	1	100	0.0%
	7	0.0	0	0	1.5	0	0	0	0.0	0	1.5	0	66.8%	1	100	0.0%
	8	0.0	0	0	1.8	0	0	0	0.0	0	1.8	0	67.0%	1	100	0.0%
	9	0.0	0	0	2.0	0	0	0	0.0	0	2.1	0	71.2%	1	100	0.0%
	10	0.0	0	0	2.4	0	0	0	0.0	0	2.4	0	69.4%	1	100	0.0%
E	1	0.0	0	0	1.5	0	0	0	0.0	0	1.6	32	29.6%	1	200	0.0%
	2	0.0	0	0	3.3	0	0	0	0.0	0	3.4	0	53.8%	1	200	0.0%
	3	0.0	0	0	5.1	0	0	0	0.0	0	5.2	0	65.2%	1	200	0.0%
	4	0.1	0	0	7.0	0	0	0	0.1	0	7.1	0	64.8%	1	200	0.0%
	5	0.1	0	0	8.9	0	0	0	0.1	0	9.0	0	72.6%	1	200	0.0%
F_1	a	0.1	0	0	28.2	0	0	0	0.3	0	28.6	0	61.4%	1	500	0.0%
	b	0.3	0	0	66.3	0	0	0	0.4	0	67.0	0	75.0%	1	500	0.0%
	c	0.9	0	0	132.8	0	0	0	0.5	0	134.1	0	82.7%	1	500	0.0%
	d	1.6	0	0	211.1	0	0	0	0.5	0	213.4	0	85.6%	1	500	0.0%
	e	2.6	0	0	304.1	0	0	0	0.7	0	307.5	0	88.1%	1	500	0.0%

Table 2: PREPRO statistical report on the QUBO problems of Beasley [11].

Problem		Preprocessing Tools Statistics									Total Time	After Preprocessing				
		Roof-Duality			Probing				Coordination			Quad. Rel.	Relat. Gap	Comp. Left		Variab. Reduc.
Family	Numb.	Time	Strong	Weak	Time	Bound.	Fix.	Eq.	Time	Eq.	Time			Num.	Larg.	
		(t_n)	(n_s)	(n_w)	(t_p)	(n_b)	(n_f)	(n_e)	(t_c)	(n_c)	(t)	(q)	(g)	(c)	(s)	(r)
ORL-50	1	0.0	46	4	0.0	0	0	0	0.0	0	0.0	0	0.0%	0	0	100.0%
	2	0.0	45	5	0.0	0	0	0	0.0	0	0.0	0	0.0%	0	0	100.0%
	3	0.0	50	0	0.0	0	0	0	0.0	0	0.0	0	0.0%	0	0	100.0%
	4	0.0	48	2	0.0	0	0	0	0.0	0	0.0	0	0.0%	0	0	100.0%
	5	0.0	49	1	0.0	0	0	0	0.0	0	0.0	0	0.0%	0	0	100.0%
	6	0.0	49	0	0.0	0	0	0	0.0	0	0.0	0	0.0%	0	0	100.0%
	7	0.0	49	1	0.0	0	0	0	0.0	0	0.0	0	0.0%	0	0	100.0%
	8	0.0	50	0	0.0	0	0	0	0.0	0	0.0	0	0.0%	0	0	100.0%
	9	0.0	48	0	0.0	2	0	0	0.0	0	0.0	0	0.0%	0	0	100.0%
	10	0.0	45	3	0.0	2	0	0	0.0	0	0.0	0	0.0%	0	0	100.0%
ORL-100	1	0.0	4	0	0.3	0	0	4	0.1	0	0.4	45	19.3%	1	92	8.0%
	2	0.0	0	1	0.4	2	1	2	0.0	0	0.5	37	6.4%	1	94	6.0%
	3	0.0	85	0	0.3	8	6	1	0.1	0	0.4	0	0.0%	0	0	100.0%
	4	0.0	1	0	0.2	0	0	0	0.0	0	0.2	43	11.3%	1	99	1.0%
	5	0.0	3	2	0.4	2	1	0	0.0	0	0.4	39	11.1%	1	92	8.0%
	6	0.0	5	0	0.3	0	0	1	0.0	0	0.3	24	19.6%	1	94	6.0%
	7	0.0	0	2	0.2	0	0	0	0.0	0	0.2	44	14.1%	1	98	2.0%
	8	0.0	23	2	0.2	5	1	1	0.1	0	0.3	25	6.3%	1	68	32.0%
	9	0.0	81	1	0.2	14	0	0	0.1	4	0.3	0	0.0%	0	0	100.0%
	10	0.0	7	1	0.5	17	74	1	0.2	0	0.7	0	0.0%	0	0	100.0%
ORL-250	1	0.0	0	0	3.4	0	0	0	0.0	0	3.4	8	41.5%	1	250	0.0%
	2	0.0	0	0	3.2	0	0	0	0.1	0	3.3	1	42.9%	1	250	0.0%
	3	0.0	0	0	3.2	0	0	0	0.1	0	3.3	0	38.8%	1	250	0.0%
	4	0.0	0	0	3.4	0	0	0	0.0	0	3.4	0	45.1%	1	250	0.0%
	5	0.0	0	0	3.3	0	0	0	0.0	0	3.4	0	39.7%	1	250	0.0%
	6	0.0	0	0	3.4	0	0	0	0.0	0	3.5	0	47.8%	1	250	0.0%
	7	0.0	0	0	3.3	0	0	0	0.0	0	3.4	0	41.1%	1	250	0.0%
	8	0.0	0	0	3.2	0	0	0	0.0	0	3.3	0	51.5%	1	250	0.0%
	9	0.0	0	0	3.4	0	0	0	0.0	0	3.4	0	40.0%	1	250	0.0%
	10	0.0	0	0	3.2	0	0	0	0.0	0	3.3	0	46.2%	1	250	0.0%
ORL-500	1	0.1	0	0	28.2	0	0	0	0.2	0	28.5	0	62.9%	1	500	0.0%
	2	0.1	0	0	27.9	0	0	0	0.2	0	28.2	0	58.5%	1	500	0.0%
	3	0.1	0	0	28.8	0	0	0	0.3	0	29.2	0	58.8%	1	500	0.0%
	4	0.1	0	0	27.9	0	0	0	0.2	0	28.3	0	58.8%	1	500	0.0%
	5	0.1	0	0	28.3	0	0	0	0.2	0	28.6	0	60.0%	1	500	0.0%
	6	0.1	0	0	28.0	0	0	0	0.2	0	28.3	0	61.1%	1	500	0.0%
	7	0.1	0	0	28.3	0	0	0	0.2	0	28.6	0	61.2%	1	500	0.0%
	8	0.1	0	0	28.0	0	0	0	0.2	0	28.4	0	60.8%	1	500	0.0%
	9	0.1	0	0	28.1	0	0	0	0.2	0	28.5	0	62.0%	1	500	0.0%
	10	0.1	0	0	28.1	0	0	0	0.2	0	28.5	0	59.1%	1	500	0.0%

Table 3: PREPRO statistical report on the c–fat and Hamming graphs.

		Preprocessing Tools Statistics									Total	After Preprocessing				
		Roof-Duality			Probing				Coordination			Quad.	Relat.	Comp. Left		Variab.
Problem		Time	Strong	Weak	Time	Bound.	Fix.	Eq.	Time	Eq.	Time	Rel.	Gap	Num.	Larg.	Reduc.
Family	Name	(t_n)	(n_s)	(n_w)	(t_p)	(n_b)	(n_f)	(n_e)	(t_c)	(n_c)	(t)	(q)	(g)	(c)	(s)	(r)
c–fat	c-fat200-1	0.5	17	3	0.9	5	40	134	19.6	1	21.1	0	0.0%	0	0	100.0%
	c-fat200-2	0.3	53	1	0.8	13	20	113	16.1	0	17.4	0	0.0%	0	0	100.0%
	c-fat200-5	0.0	1	0	0.2	0	30	169	3.3	0	3.7	0	0.0%	0	0	100.0%
	c-fat500-1	6.4	9	3	12.3	12	106	369	655.3	1	677.0	0	0.0%	0	0	100.0%
	c-fat500-2	3.2	29	5	7.9	5	51	409	364.1	1	377.1	0	0.0%	0	0	100.0%
	c-fat500-5	1.3	35	0	4.8	26	11	428	159.1	0	166.2	0	0.0%	0	0	100.0%
	c-fat500-10	0.6	1	0	3.0	0	126	373	85.5	0	89.5	0	0.0%	0	0	100.0%
Hamming	hamming6-2	0.0	0	64	0.0	0	0	0	0.0	0	0.0	0	0.0%	0	0	100.0%
	hamming8-2	0.0	0	256	0.0	0	0	0	0.0	0	0.0	0	0.0%	0	0	100.0%
	hamming10-2	0.1	0	1024	0.0	0	0	0	0.0	0	0.1	0	0.0%	0	0	100.0%
	hamming6-4	0.0	0	0	0.1	0	0	0	0.0	0	0.1	0	84.6%	1	64	0.0%
	hamming8-4	0.0	0	0	7.8	0	0	0	1.2	0	8.9	0	86.8%	1	256	0.0%
	hamming10-4	0.1	0	0	229.0	0	0	0	36.2	0	265.4	0	93.6%	1	1024	0.0%

Table 4: PREPRO statistical report on the minimum vertex cover of the RUDY planar graphs.

		Preprocessing Tools Statistics										Total	After Preprocessing				
		Roof-Duality			Probing				Coordination				Quad. Rel.	Relat. Gap	Comp. Num.	Left Larg.	Variab. Reduc.
Problem		Time	Strong	Weak	Time	Bound.	Fix.	Eq.	Time	Eq.	Time						
Family	Name	(t_n)	(n_s)	(n_w)	(t_p)	(n_b)	(n_f)	(n_e)	(t_c)	(n_c)	(t)	(q)	(g)	(c)	(s)	(r)
50 000	50000-10-1	26.2	9581	40419	0.0	0	0	0	0.0	0	65.7	0	0.0%	0	0	100.0%
(10%)	50000-10-2	23.7	9377	40619	0.0	0	0	4	0.0	0	63.3	0	0.0%	0	0	100.0%
	50000-10-3	23.3	9311	40689	0.0	0	0	0	0.0	0	62.5	0	0.0%	0	0	100.0%
50 000	50000-50-1	194.9	23591	26295	0.0	0	24	86	0.0	4	206.3	0	0.0%	0	0	100.0%
(50%)	50000-50-2	197.3	23560	26342	0.0	0	23	73	0.0	2	209.0	0	0.0%	0	0	100.0%
	50000-50-3	204.4	23673	26208	0.0	0	30	88	0.0	1	216.2	0	0.0%	0	0	100.0%
50 000	50000-90-1	85.3	8120	41871	0.0	0	0	4	0.0	5	94.0	0	0.0%	0	0	100.0%
(90%)	50000-90-2	85.2	8338	41639	0.0	0	6	12	0.1	5	94.6	0	0.0%	0	0	100.0%
	50000-90-3	86.2	8575	41408	0.0	0	4	10	0.1	3	94.8	0	0.0%	0	0	100.0%
100 000	100000-10-1	109.0	18986	81002	0.0	0	2	10	0.0	0	266.5	0	0.0%	0	0	100.0%
(10%)	100000-10-2	99.5	19138	80858	0.0	0	0	4	0.0	0	256.4	0	0.0%	0	0	100.0%
	100000-10-3	105.4	18929	81067	0.0	0	0	4	0.0	0	263.1	0	0.0%	0	0	100.0%
100 000	100000-50-1	836.7	47472	52358	0.0	0	39	126	0.0	5	883.3	0	0.0%	0	0	100.0%
(50%)	100000-50-2	871.4	47671	52145	0.0	0	40	141	0.0	3	917.2	0	0.0%	0	0	100.0%
	100000-50-3	887.8	47900	51974	0.0	1	32	93	0.0	0	933.5	0	0.0%	0	0	100.0%
100 000	100000-90-1	354.5	16065	83897	0.0	1	4	20	0.2	13	386.8	0	0.0%	0	0	100.0%
(90%)	100000-90-2	360.0	16517	83426	0.0	1	9	32	0.2	15	392.7	0	0.0%	0	0	100.0%
	100000-90-3	358.0	16675	83286	0.0	0	4	20	0.2	15	391.7	0	0.0%	0	0	100.0%
250 000	250000-10-1	634.6	47140	202852	0.0	0	1	7	0.0	0	1619.8	0	0.0%	0	0	100.0%
(10%)	250000-10-2	671.5	47175	202817	0.0	0	1	7	0.0	0	1655.7	0	0.0%	0	0	100.0%
	250000-10-3	689.2	47367	202625	0.0	0	1	7	0.0	0	1669.5	0	0.0%	0	0	100.0%
250 000	250000-50-1	5754.9	119519	129972	0.0	0	109	392	0.0	8	6047.3	0	0.0%	0	0	100.0%
(50%)	250000-50-2	5612.8	119704	129840	0.0	1	111	335	0.2	9	5906.4	0	0.0%	0	0	100.0%
	250000-50-3	6054.2	120021	129512	0.0	1	102	356	0.1	8	6346.9	0	0.0%	0	0	100.0%
250 000	250000-90-1	2304.1	42022	207847	0.0	1	31	75	0.6	24	2507.9	0	0.0%	0	0	100.0%
(90%)	250000-90-2	2370.9	43822	206093	0.0	0	15	57	0.5	13	2574.0	0	0.0%	0	0	100.0%
	250000-90-3	2318.8	41872	207970	0.0	0	59	77	0.6	22	2528.4	0	0.0%	0	0	100.0%
500 000	500000-10-1	3026.0	94462	405510	0.0	0	5	23	0.0	0	6943.9	0	0.0%	0	0	100.0%
(10%)	500000-10-2	2850.7	94371	405609	0.0	0	4	16	0.0	0	6767.6	0	0.0%	0	0	100.0%
	500000-10-3	2680.8	95190	404798	0.0	0	2	10	0.0	0	6601.3	0	0.0%	0	0	100.0%
500 000	500000-50-1	22013.2	240672	258449	0.0	1	205	656	0.2	17	23175.5	0	0.0%	0	0	100.0%
(50%)	500000-50-2	24020.2	238932	260334	0.0	1	163	543	0.2	27	25188.5	0	0.0%	0	0	100.0%
	500000-50-3	22178.5	239687	259501	0.1	0	165	632	0.3	15	23340.3	0	0.0%	0	0	100.0%
500 000	500000-90-1	9564.2	86359	413422	0.0	1	46	108	1.3	64	10363.3	0	0.0%	0	0	100.0%
(90%)	500000-90-2	9427.4	83741	416025	0.0	1	40	136	1.3	57	10219.4	0	0.0%	0	0	100.0%
	500000-90-3	9547.5	84980	414793	0.0	0	47	134	1.4	46	10364.0	0	0.0%	0	0	100.0%

Table 5: PREPRO statistical report on the MAX–CUT graphs.

		Preprocessing Tools Statistics										After Preprocessing				
		Roof-Duality			Probing				Coordination		Total	Quad. Rel.	Relat. Gap	Comp. Left		Variab. Reduc.
Problem		Time	Strong	Weak	Time	Bound.	Fix.	Eq.	Time	Eq.	Time			Num.	Larg.	
Family	Name	(t_n)	(n_s)	(n_w)	(t_p)	(n_b)	(n_f)	(n_e)	(t_c)	(n_c)	(t)	(q)	(g)	(c)	(s)	(r)
Torus	pm3-8-50	0.0	0	0	3.7	0	0	0	0.2	0	4.0	0	43.6%	1	511	0.2%
	pm3-15-50	0.1	0	0	267.0	0	0	0	11.8	0	279.3	0	45.5%	1	3374	0.0%
	g3-8	0.1	1	0	7.1	0	0	31	7.2	0	14.7	3	31.2%	1	479	6.4%
	g3-15	6.7	0	0	509.7	0	0	196	2513.8	0	3115.0	4	33.0%	1	3178	5.8%
R	R1000	0.1	0	0	46.8	0	0	1	5.7	0	52.7	0	29.0%	1	998	0.2%
	R2000	0.3	0	0	224.3	0	0	4	64.7	0	290.1	0	28.6%	1	1995	0.2%
	R3000	0.6	0	0	525.9	0	0	7	247.8	0	777.4	6	28.8%	1	2992	0.3%
	R4000	0.7	0	0	1205.3	0	0	7	436.1	0	1647.5	0	28.7%	1	3992	0.2%
	R5000	1.3	0	0	1697.1	0	0	12	1150.6	0	2862.9	0	28.6%	1	4987	0.3%
	R6000	1.7	0	0	2152.8	0	0	12	1625.3	0	3799.1	0	28.8%	1	5987	0.2%
	R7000	2.6	0	2	3784.1	0	0	16	2952.4	0	6773.9	0	28.8%	1	6981	0.3%
	R8000	3.5	0	0	4752.3	0	0	19	4489.5	0	9298.4	3	28.7%	1	7980	0.2%
Via	via.c1n	0.1	455	115	0.3	4	84	169	1.0	0	1.5	0	0.0%	0	0	100.0%
	via.c2n	0.4	102	67	3.3	3	18	789	45.5	0	52.6	0	0.0%	0	0	100.0%
	via.c3n	0.9	147	18	10.2	3	9	1149	76.3	0	94.5	0	0.0%	0	0	100.0%
	via.c4n	1.1	151	18	10.2	0	5	1191	185.8	0	208.9	0	0.0%	0	0	100.0%
	via.c5n	0.9	21	23	6.8	1	66	1090	129.4	0	145.1	0	0.0%	0	0	100.0%
	via.c1y	0.0	775	53	0.0	0	0	0	0.0	0	0.0	0	0.0%	0	0	100.0%
	via.c2y	0.1	928	52	0.0	0	0	0	0.0	0	0.1	0	0.0%	0	0	100.0%
	via.c3y	0.1	1256	71	0.0	0	0	0	0.0	0	0.1	0	0.0%	0	0	100.0%
	via.c4y	0.1	1317	49	0.0	0	0	0	0.0	0	0.1	0	0.0%	0	0	100.0%
	via.c5y	0.1	1119	83	0.0	0	0	0	0.0	0	0.1	0	0.0%	0	0	100.0%
G–500	G500.2.5	0.1	0	72	1.0	0	0	235	7.6	0	9.8	12	7.7%	1	192	61.6%
	G500.05	0.2	0	7	4.9	0	0	60	13.1	0	18.8	9	19.1%	1	432	13.6%
	G500.10	0.0	0	0	5.9	0	0	1	1.0	0	6.9	0	27.2%	1	498	0.4%
	G500.20	0.0	0	0	10.8	0	0	0	1.0	0	11.9	0	35.2%	1	499	0.2%
G–1000	G1000.2.5	0.6	1	95	6.3	0	0	521	91.2	0	108.5	18	8.3%	1	382	61.8%
	G1000.05	1.0	0	6	30.7	0	0	121	154.6	0	191.0	9	19.8%	1	872	12.8%
	G1000.10	0.1	0	0	40.0	0	0	2	8.8	0	49.0	0	29.1%	1	997	0.3%
	G1000.20	0.1	0	0	50.4	0	0	0	4.3	0	54.8	0	35.0%	1	999	0.1%
U–500	U500.05	0.1	14	39	1.1	0	8	150	4.9	0	7.3	59	20.2%	4	151	42.4%
	U500.10	0.1	0	4	12.0	0	0	11	7.5	0	20.1	37	34.0%	1	484	3.2%
	U500.20	0.2	0	0	8.0	0	0	0	2.4	0	10.9	25	39.1%	1	499	0.2%
	U500.40	0.1	0	0	14.5	0	0	0	4.2	0	18.9	7	41.7%	1	499	0.2%
U–1000	U1000.05	0.9	16	89	8.3	0	6	264	57.5	0	74.5	160	21.6%	3	316	37.6%
	U1000.10	0.7	0	6	43.7	0	0	12	43.0	0	91.3	96	34.4%	1	981	1.9%
	U1000.20	0.7	0	0	36.5	0	0	0	12.0	0	51.7	44	38.8%	1	999	0.1%
	U1000.40	0.8	0	0	63.0	0	0	0	21.8	0	87.2	30	41.9%	1	999	0.1%

Table 6: PREPRO statistical report on the MAX–2-SAT formulas of Borchers and Furman [15].

Problem		Roof-Duality			Probing				Coordination		Total Time	Quad. Rel.	Relat. Gap	Comp. Left		Variab. Reduc.
		Time	Strong	Weak	Time	Bound.	Fix.	Eq.	Time	Eq.	Time			Num.	Larg.	
Family	Name	(t_n)	(n_s)	(n_w)	(t_p)	(n_b)	(n_f)	(n_e)	(t_c)	(n_c)	(t)	(q)	(g)	(c)	(s)	(r)
BF–50	BF-50-100	0.0	2	38	0.0	0	0	6	0.0	4	0.0	0	0.0%	0	0	100.0%
	BF-50-150	0.0	1	18	0.0	4	15	12	0.0	0	0.0	0	0.0%	0	0	100.0%
	BF-50-200	0.0	1	5	0.0	0	0	2	0.0	0	0.0	15	100.0%	1	42	16.0%
	BF-50-250	0.0	0	1	0.1	2	0	2	0.0	0	0.2	14	109.5%	1	45	10.0%
	BF-50-300	0.0	0	0	0.0	0	0	0	0.0	0	0.0	14	204.8%	1	50	0.0%
	BF-50-350	0.0	0	0	0.1	1	0	0	0.0	0	0.1	13	186.7%	1	49	2.0%
	BF-50-400	0.0	1	0	0.1	1	0	0	0.0	0	0.1	14	141.0%	1	48	4.0%
	BF-50-450	0.0	0	0	0.0	0	0	0	0.0	0	0.1	3	190.9%	1	50	0.0%
	BF-50-500	0.0	0	0	0.0	0	0	0	0.0	0	0.0	0	157.7%	1	50	0.0%
BF–100	BF-100-200	0.0	2	50	0.0	0	17	31	0.0	0	0.0	0	0.0%	0	0	100.0%
	BF-100-300	0.0	0	14	0.2	0	0	9	0.1	0	0.2	49	328.6%	1	77	23.0%
	BF-100-400	0.0	0	10	0.2	0	0	1	0.0	0	0.2	43	215.8%	1	89	11.0%
	BF-100-500	0.0	0	3	0.1	0	0	0	0.0	0	0.1	25	475.0%	1	97	3.0%
	BF-100-600	0.0	0	1	0.1	0	0	0	0.0	0	0.1	6	340.0%	1	99	1.0%
BF–150	BF-150-300	0.0	1	79	0.1	0	31	34	0.1	5	0.2	0	0.0%	0	0	100.0%
	BF-150-450	0.0	0	22	0.4	0	0	5	0.1	0	0.5	85	475.0%	1	123	18.0%
	BF-150-600	0.0	0	4	0.6	0	3	6	0.2	0	0.8	80	300.0%	1	137	8.7%
BFW–50	BFW-50-100	0.0	4	13	0.0	2	14	17	0.0	0	0.0	0	0.0%	0	0	100.0%
	BFW-50-150	0.0	11	28	0.1	5	1	4	0.0	1	0.1	0	0.0%	0	0	100.0%
	BFW-50-200	0.0	1	3	0.1	1	0	1	0.0	0	0.1	14	102.8%	1	44	12.0%
	BFW-50-250	0.0	0	0	0.1	0	0	0	0.0	0	0.1	32	209.1%	1	50	0.0%
	BFW-50-300	0.0	0	1	0.1	0	0	0	0.0	0	0.1	12	149.1%	1	49	2.0%
	BFW-50-350	0.0	0	0	0.1	0	0	0	0.0	0	0.1	6	236.2%	1	50	0.0%
	BFW-50-400	0.0	0	0	0.1	2	0	0	0.0	0	0.1	4	126.5%	1	48	4.0%
	BFW-50-450	0.0	0	0	0.1	0	0	0	0.0	0	0.1	1	169.1%	1	50	0.0%
	BFW-50-500	0.0	0	0	0.1	0	0	0	0.0	0	0.1	0	205.7%	1	50	0.0%
BFW–100	BFW-100-200	0.0	54	45	0.0	1	0	0	0.0	0	0.0	0	0.0%	0	0	100.0%
	BFW-100-300	0.0	1	10	0.4	1	0	10	0.1	0	0.4	69	193.6%	1	78	22.0%
	BFW-100-400	0.0	0	2	0.4	0	1	5	0.1	0	0.5	81	476.7%	1	92	8.0%
	BFW-100-500	0.0	0	1	0.1	0	0	0	0.0	0	0.2	25	557.1%	1	99	1.0%
	BFW-100-600	0.0	0	0	0.2	0	0	0	0.0	0	0.2	18	450.0%	1	100	0.0%
BFW–150	BFW-150-300	0.0	39	55	0.5	4	7	44	0.3	1	0.8	0	0.0%	0	0	100.0%
	BFW-150-450	0.0	1	24	0.9	2	2	15	0.3	0	1.2	142	175.0%	1	106	29.3%
	BFW-150-600	0.0	0	6	0.6	0	3	3	0.1	0	0.7	59	620.7%	1	138	8.0%