**ISTITUTO DI ANALISI DEI SISTEMI ED INFORMATICA**
**"Antonio Ruberti"**
**CONSIGLIO NAZIONALE DELLE RICERCHE**

**C. Buchheim, G. Rinaldi**

# COMPACT INTEGER PROGRAMMING FORMULATIONS FOR BOOLEAN OPTIMIZATION PROBLEMS

**R. 672, Dicembre 2007**

**Christoph Buchheim** – Institut für Informatik, Universität zu Köln, Pohligstr. 1, 50969 Köln, Germany(buchheim@informatik.uni-koeln.de).

**Giovanni Rinaldi** – Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti" del CNR, viale Manzoni 30, 00185 Roma, Italy(rinaldi@iasi.cnr.it).

This work was partially supported by the Marie Curie RTN 504438 (ADONET) funded by the European Commission. The first author was supported by Deutsche Forschungsgemeinschaft (DFG) under grant BU 2313/1–1.

ISSN: 1128–3378

Collana dei Rapporti dell'Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti", CNR

viale Manzoni 30, 00185 ROMA, Italy

tel. ++39-06-77161
fax ++39-06-7716461
email: iasi@iasi.cnr.it
URL: http://www.iasi.cnr.it

## Abstract

We present a new polyhedral approach to nonlinear Boolean optimization. Compared to other methods, it produces much smaller integer programming models, making it more efficient from a practical point of view. We mainly obtain this by two different ideas: first, we do not require the objective function to be in any normal form. The transformation into a normal form usually leads to the introduction of many additional variables or constraints. Second, we reduce the problem to the degree-two case in a very efficient way, by slightly extending the dimension of the original variable space. The resulting model turns out to be closely related to the maximum cut problem; we show that the corresponding polytope is a face of a suitable cut polytope in most cases. In particular, our separation problem reduces to the one for the maximum cut problem.

In practice, the approach appears to be very competitive for unconstrained Boolean optimization problems. First experimental results, which have been obtained for some particularly hard instances of the Max-SAT Evaluation 2007, show that our very general implementation can outperform even special-purpose Max-SAT solvers. The software is accessible online under "`we.logoptimize.it`".

*Key words:* logic optimization, Boolean optimization, pseudo-boolean optimization, maximum satisfiability, polynomial optimization, cut polytope

*AMS subject classifications:* 90C57, 65K05, 03B70

## 1. Introduction

Nonlinear zero-one optimization problems are often solved by transforming the objective function into an appropriate normal form, e.g., into conjunctive normal form (CNF) or into a polynomial. The problem can then be addressed by a general solver for maximum satisfiability, polynomial zero-one optimization or some other standard problem. However, the transformation often increases the problem size significantly, since new variables or constraints have to be added. Depending on the normal form, an exponential blow-up might be unavoidable, e.g., if negations in a polynomial have to be resolved. But even if the increase is tractable from a theoretical point of view, in practice it might lead to a problem instance that is too large to be solved.

In this paper, we present a novel approach to nonlinear zero-one optimization that avoids the transformation into any normal form, by directly modeling arbitrarily constructed Boolean functions into an integer linear program (ILP). The strength of our approach lies in the fact that, nevertheless, a tight polyhedral description for the resulting model can be obtained. This description is based on a reduction of the general problem to the special case of unconstrained quadratic zero-one optimization, which is known to be equivalent to the maximum cut problem [8]. Our approach is mainly designed for unconstrained problems. Under general linear or nonlinear constraints, our polyhedral results do not hold anymore, however, our construction still gives rise to much tighter linear relaxations. Furthermore, certain classes of constraints do not harm our polyhedral results, as discussed below.

We first deal with the quadratic case, i.e., the case where all objective function terms contain at most one binary operator. We show that in this case the polytope corresponding to our formulation is isomorphic to a cut polytope, no matter which operators are considered. This is a generalization of a result of [6], where all operators are multiplications.

The situation is more complicated in the general case where we allow arbitrary Boolean functions defined recursively by binary operators. Again, our aim is to avoid introducing too many new variables or constraints, as done by other approaches such as lift-and-project. In [4], we developed a new approach for polynomial zero-one optimization problems that uses an efficient reduction to the quadratic case. The reduction can be applied after a slight extension of the variable space; the resulting polytope then turns out to be a face of a polytope corresponding to a quadratic instance of basically the same type. This allows to derive a polyhedral description for a general instance from the polyhedral description of an appropriate quadratic problem.

In the following, we generalize these results. In place of multiplications, we allow arbitrary binary operators. For problem instances not containing any exclusive disjunctions or equivalences, we show that the general polytope is still a face of an appropriate cut polytope, defined on roughly four times as many variables as in the original model. If exclusive disjunctions or equivalences are present, they can be replaced by at most three other operators each.

Our problem is closely related to the pseudo-boolean optimization problem, which has been investigated intensively in the literature. See [3] for a survey of applications and results. In pseudo-boolean optimization methods, polyhedral techniques are often used in order to derive stronger dual bounds on the optimal solutions, leading to smaller enumeration trees in general. In this spirit, Manquinho and Marques-Silva [11, 12] propose the use of cutting planes in SAT-based algorithms. Similarly, Joy *et al.* [10] devise classes of cutting planes for the basic integer programming model of Max-SAT. Cutting planes are also used in a hybrid approach presented by Sheini and Sakallah [14].

Our polyhedral approach presented in the following is different from these approaches, in that it does not aim at the identification of single classes of cutting planes that can be used to

tighten dual bounds, but at a full integer programming model of the problem, for which strong polyhedral results can be derived in a more general way.

We also give experimental evidence of the practical potential of our new approach. We present the results obtained by an implementation of this approach, on all instances of the Max-SAT Evaluation 2007 [1]. It turns out that our approach is very competitive and can even outperform special-purpose Max-SAT solvers on particularly hard instances.

## 2. The Problem

We consider an unconstrained Boolean optimization problem in the following form: a set of Boolean variables $x_i \in \{0,1\}$ for $i \in I$ is given, where $I$ is a finite index set. We set $n = |I|$. Moreover, we have a pseudo-boolean objective function

$$\min \sum_{k \in K} w_k f_k \ , \tag{1}$$

where each $f_k$ is a Boolean function $f : \{0,1\}^I \to \{0,1\}$ over the variables $x_i$ and the coefficients $w_k$ are arbitrary real numbers. For our purposes, the set of Boolean functions is defined recursively as follows: first, each variable $x_i$ for $i \in I$ corresponds to a Boolean function $f_i : \{0,1\}^I \to \{0,1\}$, defined by $f_i(x) = x_i$. Second, if $g$ and $h$ are Boolean functions and $\circ$ is any binary operator $\{0,1\}^2 \to \{0,1\}$, then $g \circ h$ is a Boolean function as well. Observe that in the evaluation of (1), all Boolean operators have precedence over the arithmetic ones.

In the following, our aim is to address problem (1) by an approach that is based on modeling each such Boolean function independently.

**Example 2.1.** All CNF and DNF clauses are Boolean functions. In particular, the maximum satisfiability problem is a special case of (1).

**Example 2.2.** Binary monomials are Boolean functions, since multiplication of binary variables can be considered a binary operator on Boolean variables. In particular, a special case of (1) is binary polynomial optimization.

**Example 2.3.** If $f$ is any Boolean function as defined above, then checking satisfiability of $f$ amounts to checking whether $\max(f) = 1$. Checking whether $f$ is a tautology amounts to checking whether $\min(f) = 1$. In particular, we can check equivalency of $f$ and any other logical formula $g$ by minimizing the Boolean function $f \Leftrightarrow g$.

A list of all possible 16 binary operators is given in Table 1. For ease of exposition, we assume throughout that all operators appearing in the functions are *proper binary operators*, i.e., we do not consider unary or constant operators. This situation can always be obtained as follows: constant operators can be resolved easily. The only non-trivial unary operator is negation. In the objective function, negations can be resolved by replacing $\neg f_k$ by $1 - f_k$. Elsewhere, negations can be merged into binary operators, e.g., we can consider $a \vee (\neg b)$ a binary operator in $a$ and $b$. In Table 2, we list the corresponding replacements explicitly. In summary, out of the 16 operators in Table 1, we only have to consider the first ten.

In theory, problem (1) can model the minimization of any function $f : \{0,1\}^n \to \mathbb{R}$, as

$$f(x) = \sum_{t \in \{0,1\}^n} f(t) \cdot \Big( \bigwedge_{t_i = 1} x_i \Big) \wedge \Big( \bigwedge_{t_i = 0} \neg x_i \Big) \ . \tag{2}$$

| truth table | notation | equivalent to | description |
|---|---|---|---|
| $a = 0\,0\,1\,1$<br>$b = 0\,1\,0\,1$ | | | |
| 0 0 0 1 | $a \wedge b$ | $a \cdot b$, $\min(a, b)$ | conjunction |
| 0 0 1 0 | $a \not\Rightarrow b$ | $a > b$, $\min(a, 1-b)$ | non-implication |
| 0 1 0 0 | $a \not\Leftarrow b$ | $a < b$, $\min(1-a, b)$ | converse non-implication |
| 0 1 1 0 | $a \oplus b$ | $a \neq b$, $a + b \mod 2$ | exclusive disjunction |
| 0 1 1 1 | $a \vee b$ | $a + b - a \cdot b$, $\max(a, b)$ | disjunction |
| 1 0 0 0 | $a \bar{\vee} b$ | $(1-a) \cdot (1-b)$, $1 - \max(a, b)$ | nondisjunction |
| 1 0 0 1 | $a \Leftrightarrow b$ | $a \equiv b$, $a + b + 1 \mod 2$ | equivalence |
| 1 0 1 1 | $a \Leftarrow b$ | $a \geq b$, $\max(a, 1-b)$ | converse implication |
| 1 1 0 1 | $a \Rightarrow b$ | $a \leq b$, $\max(1-a, b)$ | implication |
| 1 1 1 0 | $a \bar{\wedge} b$ | $1 - a \cdot b$, $1 - \min(a, b)$ | non-conjunction |
| 0 0 0 0 | $a \perp b$ | 0 | constant 0 |
| 0 0 1 1 | $a \llcorner b$ | $a$ | left projection |
| 0 1 0 1 | $a \lrcorner b$ | $b$ | right projection |
| 1 0 1 0 | $a \urcorner b$ | $\neg b$, $1-b$ | right complementation |
| 1 1 0 0 | $a \ulcorner b$ | $\neg a$, $1-a$ | left complementation |
| 1 1 1 1 | $a \top b$ | 1 | constant 1 |

Table 1: All 16 binary operators.

| $\circ$ | $a \circ \neg b$ | $\neg a \circ b$ | $\neg a \circ \neg b$ |
|---|---|---|---|
| $\wedge$ | $a \not\Rightarrow b$ | $a \not\Leftarrow b$ | $a \bar{\vee} b$ |
| $\not\Rightarrow$ | $a \vee b$ | $a \bar{\wedge} b$ | $a \not\Leftarrow b$ |
| $\not\Leftarrow$ | $a \bar{\vee} b$ | $a \wedge b$ | $a \not\Rightarrow b$ |
| $\oplus$ | $a \Leftrightarrow b$ | $a \Leftrightarrow b$ | $a \oplus b$ |
| $\vee$ | $a \Leftarrow b$ | $a \Rightarrow b$ | $a \bar{\wedge} b$ |

| $\circ$ | $a \circ \neg b$ | $\neg a \circ b$ | $\neg a \circ \neg b$ |
|---|---|---|---|
| $\bar{\vee}$ | $a \not\Leftarrow b$ | $a \not\Rightarrow b$ | $a \wedge b$ |
| $\Leftrightarrow$ | $a \oplus b$ | $a \oplus b$ | $a \Leftrightarrow b$ |
| $\Leftarrow$ | $a \vee b$ | $a \bar{\wedge} b$ | $a \Rightarrow b$ |
| $\Rightarrow$ | $a \bar{\wedge} b$ | $a \vee b$ | $a \Leftarrow b$ |
| $\bar{\wedge}$ | $a \Rightarrow b$ | $a \Leftarrow b$ | $a \vee b$ |

Table 2: Equivalence relations among proper binary operators.

E.g., we could write $a \vee b$ as $(a \wedge b) + (a \wedge \neg b) + (\neg a \wedge b)$. Unfortunately, this representation of $f$ is useless, as it leads to a trivial problem: the optimal solution is readily obtained with a simple linear time algorithm that produces the solution right after reading the input to the problem. However, if a much more compact representation of $f$ than (2) is provided, then the problem may become much more difficult to solve and the approach presented in the following may become very efficient. Actually, the efficiency mainly depends on the total number of operators in all functions $f_k$, $k \in K$. Clearly, the number of functions $\{0,1\}^n \to \{0,1\}$ that can be modeled with a fixed number of arbitrary binary operators, as explained above, is much bigger than, e.g., the number of CNF formulae produced using the same number of operators.

In this context, we point out that some effort in compactifying the representation of $f$ can be worthwhile. Even if the original instance is given in some normal form such as CNF, it might well pay off to give up the normal form if this leads to a smaller number of operators. In general, the more the flexibility of our approach is exploited, the more it can be expected to outperform methods designed for instances with a specific structure.

## 3. Branch-and-Cut and the Maximum Cut Problem

In this section, we review the basic ideas of branch-and-cut algorithms, with a view towards the maximum cut problem. Branch-and-cut is an extension of branch-and-bound, in which the bounds in all enumeration nodes are computed by solving LP-relaxations of the problem [13].

In order to start a branch-and-cut approach, the problem must first be modeled as an integer linear program (ILP), i.e., a linear program with additional integrality constraints on all or some of the variables. In the following, all variable required to be integer will be binary, i.e., will take only the values 0 or 1. Therefore, we will only consider the branch-and-cut approach for the binary case. The branching step usually consists in creating two subproblems that replace the original one; each of the two subproblems is obtained by fixing an integer variable to one of its two possible values. The first LP-relaxation arises from ignoring all integrality requirements. The optimal solution of this LP-relaxation then yields a dual bound for the original problem (a lower or an upper bound depending on whether we minimize or maximize, respectively).

If the optimal LP-solution is fractional, i.e., if it does not meet all integrality constraints, then a branch-and-cut algorithm tries to improve the relaxation by adding further linear inequalities that are valid for all feasible solutions of the ILP but violated by the given fractional LP-solution. Such inequalities are called *cutting planes*. An algorithm trying to find such cutting planes is called a *separation algorithm*. Usually, a separation algorithm is designed to find cutting planes from within a specific class of candidate inequalities.

The separation problem leads to the mathematical task of understanding the polyhedral structure of the problem. More precisely, it is necessary to characterize classes of valid inequalities for the *polytope* defined as the convex hull of all feasible solutions of the ILP. The separation algorithm must then decide whether a certain fractional point can be separated from this polytope by some hyperplane from the known classes.

For an NP-hard problem, there is no hope of finding a complete characterization of the system of linear inequalities describing the polytope. However, in practice, the performance of a cutting plane algorithm generally increases even with a (very) limited knowledge of such a system. Moreover, for the polytope associated with these difficult problems the separation for several classes of valid inequalities is also NP-hard. Therefore, typically one has to resort to heuristic separation algorithms. When such an algorithm returns a cutting plane from a specific class, the answer is correct. However, if it terminates with the status "no separating hyperplanes found", this does not necessarily mean that indeed no such hyperplanes exist.

Due to the incomplete knowledge of the linear system, and due to the heuristic nature of some separation algorithms, the cutting-plane phase may terminate with an optimal LP-solution that still has some fractional components. The branch-and-cut algorithm then proceeds with a branching phase. Cutting-plane and branching phases are recursively applied to all generated subproblems.

Branch-and-cut algorithms turned out to be very successful in practice for solving many hard optimization problems. One of these problems concerns cuts in graphs. In a graph $G = (V, E)$, with every subset $S \subseteq V$ we associate the set of all edges having exactly one end-node in $S$. Such an edge set is called the *cut* of $G$ defined by $S$ and is denoted by $\delta(S)$. Note that the sets $S$ and $V \setminus S$ define the same cut of $G$ and that $S$ can be empty (or can coincide with $V$), in which case the cut $\delta(S)$ is the empty set. Given a graph $G = (V, E)$ with edge weights $w_e \in \mathbb{R}$, the *maximum cut problem* (*max-cut*) is to find a cut in $G$ of maximum total weight. In the standard ILP formulation of this problem, we have a binary variable $x_e \in \{0, 1\}$ for each edge $e \in E$, such that $x_e = 1$ if and only if $e$ belongs to the cut being modeled. Such a binary vector associated

with a cut $\delta(S)$ is called its *incidence vector* and is denoted by $\chi^S \in \mathbb{R}^E$. The *cut polytope* $\mathcal{C}(G) \subset \mathbb{R}^E$ of $G$ is the convex hull of all incidence vectors of cuts in $G$.

Many classes of cutting planes as well as other structural properties of cut polytopes have been discovered; see [7] for a broad overview. In our implementation, we currently only use the most important class of cutting planes called *cycle inequalities*. These inequalities model the fact that every cut of $G$ must have an even intersection with every cycle in $G$, i.e., for every cycle $C$ in $G$ and every odd subset $F \subseteq C$, we have that the inequality

$$\sum_{e \in F} x_e - \sum_{e \in C \setminus F} x_e \leq |F| - 1$$

is satisfied by the incidence vectors of all cuts of $G$, thus it is as a valid inequality for the cut polytope $\mathcal{C}(G)$. In general, there are exponentially many cycle inequalities for a given graph; nevertheless, the corresponding separation problem can be solved in polynomial time [2].

Though polynomial, the exact separation algorithm for cycle inequalities is often too slow in practice. Therefore, the separation is often done heuristically. In our implementation, we use a heuristic based on maximum weight spanning trees, see Section 5 below.

## 4. New Formulation and Reduction to Maximum Cut

In the following, we develop our new model for Boolean optimization problems. This approach has two main advantages over other methods: the number of variables is kept small, even if the objective function does not conform to any normal form. The second advantage is that the corresponding polytope turns out to be a face of an appropriate cut polytope, if the model is slightly extended and no exclusive disjunctions or equivalences appear in the objective function. This allows us to address the general problem by a cutting plane method that is entirely based on separation algorithms for the maximum cut problem. An example illustrating the construction will be given in Section 4.3 below.

### 4.1. The Model

In order to develop a linear model for problem (1), we have to introduce further binary variables. First, we need to linearize the objective function by adding a variable $x_k \in \{0,1\}$ representing $f_k$, for every $k \in K$. The objective then translates to $\min c^\top x_K$, and it remains to model the connection between the *basic* variables $x_I$ and the *objective* variables $x_K$. More precisely, we have to make sure that

$$x_k = f_k(x_I) \text{ for every feasible solution } x \in \{0,1\}^{I \cup K} .$$

In order to bridge the gap between basic and objective variables, we additionally introduce *connection* variables $x_j \in \{0,1\}$, $j \in J$. Every such variable corresponds to an intermediate function in the recursive definition of the Boolean functions $f_k$.

More formally, we determine the set of connection variables recursively as follows: we start with $J = \emptyset$. Let $j \in J \cup K$ with $f_j = g \circ_j h$, for appropriate Boolean functions $g$ and $h$ and for a proper binary operator $\circ_j$. If $g = f_s$ for some $s \in I \cup J \cup K$, we define $l(j) = s$, otherwise we add a new index $l(j)$ to $J$, introduce a new variable $x_{l(j)}$ representing $g$, and define $f_{l(j)} = g$. Analogously, if $h = f_s$ for some $s \in I \cup J \cup K$, we define $r(j) = s$, otherwise we add a new index $r(j)$ to $J$, introduce a variable $x_{r(s)}$ representing $h$, and define $f_{r(j)} = h$. We continue in

this way until every $f_j$ with $j \in J \cup K$ is of the form $f_{l(j)} \circ f_{r(j)}$ for appropriate $l(j), r(j) \in I \cup J \cup K$. We finally extend each $f_s$ for $s \in I \cup J \cup K$ to a function $f_s : \{0,1\}^{I \cup J} \to \{0,1\}$, by ignoring all additional entries of $x \in \{0,1\}^{I \cup J}$.

We have thus constructed a set of variables $\{x_s \mid s \in I \cup J \cup K\}$ and a corresponding set of Boolean functions $F = \{f_s \mid s \in I \cup J \cup K\}$ defined on the domain $\{0,1\}^{I \cup J}$. In the following, the unit vector corresponding to $x_s$ will be denoted by $e_s$. Every non-basic function in $F$ is the result of applying some binary operator to an appropriate pair of other functions in $F$. Notice that the total number of connection and objective variables $|J \cup K|$ is at most the total number $m$ of operators in the objective function, so the total number of variables in our model is at most $n + m$. In practice, it is often possible to save a lot of these variables by intelligent decomposition of the objective function.

Every feasible solution in our model is uniquely determined by a truth assignment for all basic variables $x_i$, i.e., by a function $t : I \to \{0,1\}$. The corresponding characteristic vector $\chi_t \in \{0,1\}^{I \cup J \cup K}$ is defined in the obvious way—every component $(\chi_t)_s$ takes the value of $f_s$ under $t$, denoted by $t(f_s)$ in the following. Now we define

$$P = \operatorname{conv}\left\{\chi_t \mid t : I \to \{0,1\}\right\} \subset \mathbb{R}^{I \cup J \cup K} .$$

Equivalently, we have

$$P = \operatorname{conv}\left\{(x_I, x_J, x_K) \in \{0,1\}^{I \cup J \cup K} \mid x_s = f_s(x_I, x_J) \text{ for all } s \in J \cup K\right\} .$$

We can thus restate problem (1) as $\min c^\top x_K$ s.t. $x \in P$.

In order to solve this problem, it is necessary to find tight linear relaxations of the polytope $P$. The standard techniques for linearizing polynomial terms in binary programs could be adapted to our model, however, the resulting relaxations for $P$ are weak in general. Instead, we aim at generalizing the results we obtained for binary polynomial optimization, presented in [4], to the more general situation considered here. In the remainder of this section, we will show that $P$ is a face of an appropriate cut polytope of small dimension if the objective function does not contain exclusive disjunctions or equivalences.

The proof is done in two steps: first the result is shown in the quadratic case, i.e., when all objective terms $f_k$ contain at most one operator. In fact, $P$ is isomorphic to a cut polytope in this case. This is a generalization of a result given in [6], that states that binary quadric polytopes are isomorphic to cut polytopes. We obtain this result without restricting the set of allowed operators.

Second, we show that in the case of objective functions of arbitrary degree, the polytope $P$ is a face of a polytope $P^*$ defined by a quadratic instance of our problem, if the objective function does not contain exclusive disjunctions or equivalences. In other words, under this assumption, the case of arbitrary degree can be reduced to the quadratic case.

### 4.2. Quadratic Case

In the quadratic case, the polytope $P$ is always isomorphic to an appropriate cut polytope defined on the same number of variables.

**Lemma 4.1.** *Let $f_k$ contain at most one operator for all $k \in K$. Then the polytope $P$ is isomorphic to a cut polytope. The corresponding graph has $n + 1$ nodes and $n + m$ edges.*

*Proof:* In this case, we have $J = \emptyset$ and $l(k), r(k) \in I$ for all $k \in K$. We can thus define a graph $G = (V, E)$ by

$$\begin{aligned} V &= \{z\} \cup \{v_i \mid i \in I\} \\ E &= \{(z, v_i) \mid i \in I\} \cup \{(v_{l(k)}, v_{r(k)}) \mid k \in K\} \; . \end{aligned}$$

In the following, let $e_{(v,w)}$ denote the unit vector in $\mathbb{R}^E$ associated with an edge $(v, w) \in E$. Let $f_k = f_{l(k)} \circ_k f_{r(k)}$ for all $k \in K$. We define a linear map $\psi' : \mathbb{R}^E \to \mathbb{R}^{I \cup K}$ by fixing the images of all unit vectors of $\mathbb{R}^E$:

$$\begin{aligned} e_{(z,v_i)} \quad &\mapsto \quad e_i \\ &\quad + \sum_{\substack{k \in K \\ i = l(k)}} 1/2(-0 \circ_k 0 - 0 \circ_k 1 + 1 \circ_k 0 + 1 \circ_k 1) \cdot e_k \\ &\quad + \sum_{\substack{k \in K \\ i = r(k)}} 1/2(-0 \circ_k 0 + 0 \circ_k 1 - 1 \circ_k 0 + 1 \circ_k 1) \cdot e_k \\ e_{(v_{l(k)}, v_{r(k)})} \quad &\mapsto \quad 1/2(-0 \circ_k 0 + 0 \circ_k 1 + 1 \circ_k 0 - 1 \circ_k 1) \cdot e_k \; . \end{aligned}$$

As $\psi'$ is bijective, the map $\psi : \mathbb{R}^E \to \mathbb{R}^{I \cup K}$ given by $x \mapsto \psi'(x) + \sum_{k \in K} (0 \circ_k 0) e_k$ is an affine isomorphism. Hence it suffices to show that $\psi$ induces a bijection between the vertices of the cut polytope $\mathcal{C}(G)$ of $G$ and the vertices of $P$.

So consider the incidence vector $\chi^S \in \mathcal{C}(G)$ of any cut $\delta(S)$, $S \subseteq V$, where we may assume, without loss of generality, that $z \notin S$. Define a truth assignment $t : I \to \{0, 1\}$ by setting $t(i) = 1$ if and only if $v_i \in S$. We claim that $\psi(\chi_S) = \chi_t$. Indeed,

$$\chi_S = \sum_{\substack{i \in I \\ v_i \in S}} e_{(z,v_i)} + \sum_{\substack{k \in K \\ v_{l(k)} \in S \oplus v_{r(k)} \in S}} e_{(v_{l(k)}, v_{r(k)})} \; ,$$

thus

$$\psi(\chi_S) = \sum_{\substack{i \in I \\ t(i) = 1}} \psi(e_{(z,v_i)}) + \sum_{\substack{k \in K \\ t(l(k)) \oplus t(r(k)) = 1}} \psi(e_{(v_{l(k)}, v_{r(k)})}) \; ,$$

so that for $i \in I$ we have $\psi(\chi_S)_i = t(i) = (\chi_t)_i$ and, for $k \in K$,

$$\begin{aligned} \psi(\chi_S)_k \quad &= \quad 1/2(-0 \circ_k 0 + 0 \circ_k 1 + 1 \circ_k 0 - 1 \circ_k 1) \cdot t(l(k)) \oplus t(r(k)) \\ &\quad + 1/2(-0 \circ_k 0 - 0 \circ_k 1 + 1 \circ_k 0 + 1 \circ_k 1) \cdot t(l(k)) \\ &\quad + 1/2(-0 \circ_k 0 + 0 \circ_k 1 - 1 \circ_k 0 + 1 \circ_k 1) \cdot t(r(k)) + (0 \circ_k 0) \qquad (3) \\ &= \quad t(l(k)) \circ_k t(r(k)) = (\chi_t)_k \; . \end{aligned}$$

Conversely, for given $t : I \to \{0, 1\}$ we define a cut of $G$ by $S = \{v_i \in V \mid t(i) = 1\}$. This construction is obviously inverse to the one above, so the proof is complete. □

The main ingredient in the proof of Lemma 4.1 is the reformulation of an arbitrary binary operator as an affine combination of exclusive disjunction and basic variables, using (3). For the ten operators to be considered, the corresponding formulae are listed in Table 3.

| operator | reformulation |
|---|---|
| $a \wedge b$ | $1/2(\ \ a + b - a \oplus b)$ |
| $a \vee b$ | $1/2(\ \ a + b + a \oplus b)$ |
| $a \Rightarrow b$ | $1/2(-a + b - a \oplus b) + 1$ |
| $a \Leftarrow b$ | $1/2(\ \ a - b - a \oplus b) + 1$ |
| $a \bar{\wedge} b$ | $1/2(-a - b + a \oplus b) + 1$ |
| $a \bar{\vee} b$ | $1/2(-a - b - a \oplus b) + 1$ |
| $a \not\Rightarrow b$ | $1/2(\ \ a - b + a \oplus b)$ |
| $a \not\Leftarrow b$ | $1/2(-a + b + a \oplus b)$ |
| $a \oplus b$ | $a \oplus b$ |
| $a \Leftrightarrow b$ | $-a \oplus b\ \ +1$ |

Table 3: Reformulation of binary operators in terms of exclusive disjunctions.

### 4.3. General Case

In this section, we do not require a quadratic objective function any more. Moreover, we do not assume any normal form, all operators may be mixed arbitrarily. Nevertheless, we can show the following result.

**Theorem 4.2.** *Assume that no operator in the objective function is an exclusive disjunction or an equivalence. Then the polytope $P$ is isomorphic to a face of a cut polytope. The corresponding graph has at most $n + 4m$ edges.*

*Proof:* By the previous lemma, it suffices to show that $P$ is a face of some polytope $P^*$ that corresponds to a quadratic instance of our problem with at most $n + m$ basic variables and at most $3m$ operators in total. In order to construct this quadratic instance, define the set of basic variables to be $\{x_s^0 \mid s \in I \cup J\}$ and set $I^* = I \cup J$. Moreover, define a new set of quadratic objective terms over these variables as

$$\{x_s^1 = x_{l(s)}^0 \circ_s x_{r(s)}^0 \mid s \in J \cup K\} \cup \{x_s^2 = x_{l(s)}^0 \wedge x_s^0,\ x_s^3 = x_{r(s)}^0 \wedge x_s^0 \mid s \in J\}\ .$$

Let $K^*$ be an index set for these $3|J| + |K|$ objective terms. Denote the corresponding polytope in $\mathbb{R}^{I^* \cup K^*}$ by $P^*$. We will show that $P$ is a face of $P^*$.

For the following, we denote the unit vector in $\mathbb{R}^{I^* \cup K^*}$ associated with the variable $x_s^i$ by $e_s^i$, for $s \in I^* \cup K^*$ and $i = 0, 1, 2, 3$. Moreover, we define $c_s = 1 \circ_s 1 + 0 \circ_s 0 - 1 \circ_s 0 - 0 \circ_s 1$. Observe that $c_s \neq 0$ for all proper binary operators. We first claim that

$$P \cong \text{conv}\left(P^* \cap X \cap \{0,1\}^{I^* \cup K^*}\right), \tag{4}$$

where $X$ is the linear subspace of $\mathbb{R}^{I^* \cup K^*}$ given by the equations

$$\begin{aligned} x_s^1 &= x_s^0 && (5)\\ x_s^2 &= \big(c_s^{-1}(1 \circ_s 1 - 1 \circ_s 0)\big)x_s^0 && (6)\\ &\quad + \big(c_s^{-1}(1 \circ_s 1 - 1 \circ_s 0)(0 \circ_s 0 - 1 \circ_s 0) + (1 \circ_s 0)\big)x_{l(s)}^0\\ &\quad + \big(c_s^{-1}(1 \circ_s 1 - 1 \circ_s 0)(0 \circ_s 0 - 0 \circ_s 1)\big)x_{r(s)}^0\\ &\quad - c_s^{-1}(1 \circ_s 1 - 1 \circ_s 0)(0 \circ_s 0)\\ x_s^3 &= \big(c_s^{-1}(1 \circ_s 1 - 0 \circ_s 1)\big)x_s^0 && (7)\\ &\quad + \big(c_s^{-1}(1 \circ_s 1 - 0 \circ_s 1)(0 \circ_s 0 - 0 \circ_s 1) + (0 \circ_s 1)\big)x_{r(s)}^0\\ &\quad + \big(c_s^{-1}(1 \circ_s 1 - 0 \circ_s 1)(0 \circ_s 0 - 1 \circ_s 0)\big)x_{l(s)}^0\\ &\quad - c_s^{-1}(1 \circ_s 1 - 0 \circ_s 1)(0 \circ_s 0) \end{aligned}$$

for all $s \in J$. The isomorphism is induced by the linear map $\varphi : \mathbb{R}^{I^* \cup K^*} \cap X \to \mathbb{R}^{I \cup J \cup K}$ that is uniquely defined by $\varphi(e_s^0) = e_s$ for $s \in I^*$ and $\varphi(e_s^1) = e_s$ for $s \in K$. The images of all other unit vectors of $\mathbb{R}^{I^* \cup K^*}$ under $\varphi$ are determined by the equations (5) to (7).

To show (4), consider a vertex $\chi_t$ of $P$ corresponding to an assignment $t : I \to \{0,1\}$. Extend it to $t^* : I^* \to \{0,1\}$ in the natural way, setting $t^*(s) = t(f_s)$ for all $s \in J$. Now by construction we have $\chi_t = \varphi(\chi^*_{t^*})$, where $\chi^*_{t^*}$ denotes the characteristic vector of $t^*$ in $P^*$. To see this, one can verify, by checking all possible operators $\circ_s$ and all possible truth assignments to the variables $x_{l(s)}$, $x_{r(s)}$, and $x_s$, that

$$x_{l(s)} \circ_s x_{r(s)} = c_s(x_{l(s)} \wedge x_{r(s)}) + (1 \circ_s 0 - 0 \circ_s 0)x_{l(s)} + (0 \circ_s 1 - 0 \circ_s 0)x_{r(s)} + 0 \circ_s 0$$

and that

$$\begin{aligned} (x_{l(s)} \wedge x_s) &= (1 \circ_s 1 - 1 \circ_s 0)(x_{l(s)} \wedge x_{r(s)}) + (1 \circ_s 0)x_{l(s)}\\ (x_{r(s)} \wedge x_s) &= (1 \circ_s 1 - 0 \circ_s 1)(x_{l(s)} \wedge x_{r(s)}) + (0 \circ_s 1)x_{r(s)} \ . \end{aligned}$$

Conversely, any point in $P^* \cap \{0,1\}^{I^* \cup K^*}$ is a vertex $\chi^*_{t^*}$ of $P^*$ corresponding to a truth assignment $t^* : I^* \to \{0,1\}$. Let $t = t^*|_I$. By (5), all vectors in $X$ satisfy

$$x_s^0 = x_s^1 = x_{l(s)}^0 \circ_s x_{r(s)}^0 \text{ for all } s \in J \ ,$$

so that we can inductively show that $x_s^0 = t(f_s)$ for $s \in I^*$, supposed that the same holds for $s \in I$. In other words, $t^*$ is the extension of $t$ described above, hence we have $\chi_t = \varphi(\chi^*_{t^*})$ again.

Having proved (4), it remains to show that $X$ induces a face of $P^*$, since this implies that $P^* \cap X$ is integer so that (4) yields an isomorphism $P \cong P^* \cap X$. This is true for all operators except for exclusive disjunctions and equivalences, i.e., for $\circ_s \in \{\wedge, \vee, \Rightarrow, \Leftarrow, \bar{\wedge}, \bar{\vee}, \not\Rightarrow, \not\Leftarrow\}$.

For each of these operators, we claim that the equations (6) and (7) hold as inequalities for $P^*$, the direction depending on the operator. Indeed, the right hand side of (6) reads

$$\begin{cases} x_s^0 & \text{for } \circ_s \in \{\wedge, \not\Rightarrow\}\\ x_{l(s)}^0 & \text{for } \circ_s \in \{\vee, \Leftarrow\}\\ x_s^0 + x_{l(s)}^0 - 1 & \text{for } \circ_s \in \{\Rightarrow, \bar{\wedge}\}\\ 0 & \text{for } \circ_s \in \{\bar{\vee}, \not\Leftarrow\} \ . \end{cases}$$

In the first two cases, this right hand side is greater or equal to $x^0_{l(s)} \wedge x^0_s = x^2_s$ for every integer point in $P^*$. In the other two cases, this right hand side is less or equal to $x^0_{l(s)} \wedge x^0_s = x^2_s$ for every integer point in $P^*$. Thus (6) induces a face of $P^*$. For (7), the same result follows by symmetry.

So let $F$ be the face of $P^*$ induced by the two equations (6) and (7). It remains to show that (5) induces a face of $F$. From (6) we derive

$$\begin{aligned}
x^0_s = 0 \;\text{ or }\; x^0_{l(s)} = 1 \quad &\text{if } \circ_s \in \{\wedge, \not\Rightarrow\} \\
x^0_s = 1 \;\text{ or }\; x^0_{l(s)} = 0 \quad &\text{if } \circ_s \in \{\vee, \Leftarrow\} \\
x^0_s = 1 \;\text{ or }\; x^0_{l(s)} = 1 \quad &\text{if } \circ_s \in \{\Rightarrow, \bar{\wedge}\} \\
x^0_s = 0 \;\text{ or }\; x^0_{l(s)} = 0 \quad &\text{if } \circ_s \in \{\bar{\vee}, \not\Leftarrow\}
\end{aligned}$$

and (7) yields

$$\begin{aligned}
x^0_s = 0 \;\text{ or }\; x^0_{r(s)} = 1 \quad &\text{if } \circ_s \in \{\wedge, \not\Leftarrow\} \\
x^0_s = 1 \;\text{ or }\; x^0_{r(s)} = 0 \quad &\text{if } \circ_s \in \{\vee, \Rightarrow\} \\
x^0_s = 1 \;\text{ or }\; x^0_{r(s)} = 1 \quad &\text{if } \circ_s \in \{\Leftarrow, \bar{\wedge}\} \\
x^0_s = 0 \;\text{ or }\; x^0_{r(s)} = 0 \quad &\text{if } \circ_s \in \{\bar{\vee}, \not\Rightarrow\}
\end{aligned}$$

and hence

$$\begin{aligned}
x^0_s \leq (x^0_{l(s)} \circ_s x^0_{r(s)}) = x^1_s \quad &\text{if } \circ_s \in \{\wedge, \bar{\vee}, \not\Rightarrow, \not\Leftarrow\} \\
x^0_s \geq (x^0_{l(s)} \circ_s x^0_{r(s)}) = x^1_s \quad &\text{if } \circ_s \in \{\vee, \Rightarrow, \Leftarrow, \bar{\wedge}\} \; .
\end{aligned}$$

This completes the proof. □

**Example 4.1.** To illustrate the construction of Theorem 4.2, consider the case of a single CNF-clause $x_1 \vee x_2 \vee x_3$ containing three non-negated variables, which is the smallest non-trivial example. Then the polytope $P$ corresponding to the problem

$$\begin{aligned}
\min \quad & x_1 \vee x_2 \vee x_3 \\
\text{s.t.} \quad & x_1, x_2, x_3 \in \{0, 1\}
\end{aligned}$$

is defined over five binary variables $x_1, \ldots, x_5$, where the first three are basic variables and can thus be chosen freely, while the other two variables are determined by the basic variables according to

$$\begin{aligned}
x_4 &= x_1 \vee x_2 \\
x_5 &= x_4 \vee x_3 = (x_1 \vee x_2) \vee x_3 \; .
\end{aligned}$$

We thus have $I = \{1, 2, 3\}$, $J = \{4\}$, and $K = \{5\}$, and $P$ is spanned by the $2^3$ vectors

$$\begin{array}{llll}
(0\;0\;0\;\;0\;0) & (0\;1\;0\;\;1\;1) & (1\;0\;0\;\;1\;1) & (1\;1\;0\;\;1\;1) \\
(0\;0\;1\;\;0\;1) & (0\;1\;1\;\;1\;1) & (1\;0\;1\;\;1\;1) & (1\;1\;1\;\;1\;1)
\end{array}$$

Now the constructed polytope $P^*$ is defined in $\mathbb{R}^{I^* \cup K^*}$, where $I^*$ contains indices for the new basic variables

$$x^0_1,\; x^0_2,\; x^0_3,\; x^0_4 = (x_1 \vee x_2)^0$$

and $K^*$ contains indices for the new quadratic terms

$$x_4^1 = x_1^0 \vee x_2^0,\ x_5^1 = x_4^0 \vee x_3^0,\ x_4^2 = x_1^0 \wedge x_4^0,\ x_4^3 = x_2^0 \wedge x_4^0\ .$$

The $2^4$ vertices of $P^*$ are

$$\begin{array}{llll}
(0\ 0\ 0\ 0\ \ 0\ 0\ 0\ 0) & (0\ 1\ 0\ 0\ \ 1\ 0\ 0\ 0) & (1\ 0\ 0\ 0\ \ 1\ 0\ 0\ 0) & (1\ 1\ 0\ 0\ \ 1\ 0\ 0\ 0) \\
(0\ 0\ 0\ 1\ \ 0\ 1\ 0\ 0) & (0\ 1\ 0\ 1\ \ 1\ 1\ 0\ 1) & (1\ 0\ 0\ 1\ \ 1\ 1\ 1\ 0) & (1\ 1\ 0\ 1\ \ 1\ 1\ 1\ 1) \\
(0\ 0\ 1\ 0\ \ 0\ 1\ 0\ 0) & (0\ 1\ 1\ 0\ \ 1\ 1\ 0\ 0) & (1\ 0\ 1\ 0\ \ 1\ 1\ 0\ 0) & (1\ 1\ 1\ 0\ \ 1\ 1\ 0\ 0) \\
(0\ 0\ 1\ 1\ \ 0\ 1\ 0\ 0) & (0\ 1\ 1\ 1\ \ 1\ 1\ 0\ 1) & (1\ 0\ 1\ 1\ \ 1\ 1\ 1\ 0) & (1\ 1\ 1\ 1\ \ 1\ 1\ 1\ 1)
\end{array}$$

Now equation (5) reads $x_4^1 = x_4^0$. This equation excludes half of the vertices of $P^*$, namely those where the fourth and fifth entry do not agree, i.e., those not corresponding to solutions of the original problem. Equations (6) and (7) read $x_4^2 = x_1^0$ and $x_4^3 = x_2^0$. These equations automatically hold for the remaining $2^3$ vertices, but they are needed to ensure that $P$ is isomorphic to a face of $P^*$. We end up with a quadratic problem formulation

$$\begin{array}{lll}
\min & x_4^0 \vee x_3^0 & \\
\text{s.t.} & x_1^0 \vee x_2^0 & = \ x_4^0 \\
& x_1^0 \wedge x_4^0 & = \ x_1^0 \\
& x_2^0 \wedge x_4^0 & = \ x_2^0 \\
& x_1^0, x_2^0, x_3^0, x_4^0 \in \{0,1\}\ . &
\end{array}$$

**Example 4.2.** If one of the operators is an exclusive disjunction or an equivalence, it is not true in general that $\varphi(P)$ is a face of the polytope $P^*$ constructed in Theorem 4.2. To see this, consider the objective function $x_1 \circ (x_2 \oplus x_3)$, for any operator $\circ$. Then $P$ is defined over five variables $x_1, \ldots, x_5$ again, the last two of which correspond to quadratic terms

$$\begin{array}{lll}
x_4 & = & x_2 \oplus x_3 \\
x_5 & = & x_1 \circ x_4 = x_1 \circ (x_2 \oplus x_3)\ .
\end{array}$$

The polytope $P^*$ is now defined over the basic variables

$$x_1^0,\ x_2^0,\ x_3^0,\ x_4^0 = (x_2 \oplus x_3)^0$$

and the quadratic terms

$$x_4^1 = x_2^0 \oplus x_3^0,\ x_5^1 = x_1^0 \circ x_4^0,\ x_4^2 = x_2^0 \wedge x_4^0,\ x_4^3 = x_3^0 \wedge x_4^0\ .$$

By (4), the polytope $\varphi(P)$ is spanned by a subset of the vertices of $P^*$, determined by the equations (5) to (7). As $\circ_4 = \oplus$ and $c_4 = -2$, these equations read

$$\begin{array}{lll}
x_4^1 & = & x_4^0 \\
x_4^2 & = & 1/2 \cdot (x_4^0 + x_2^0 - x_3^0) \\
x_4^3 & = & 1/2 \cdot (x_4^0 - x_2^0 + x_3^0)\ .
\end{array}$$

Since $P^*$ is symmetric with respect to all hyperplanes defined by these equations, it follows that the barycenters of $\varphi(P)$ and $P^*$ agree, i.e., that $\varphi(P)$ cuts through the center of $P^*$. The same holds with $\Leftrightarrow$ in place of $\oplus$.

**Corollary 4.3.** *The polytope $P$ is isomorphic to a projection of a face of a cut polytope. The corresponding graph has at most $n + 12m$ edges.*

*Proof:* After replacing all exclusive disjunctions and equivalences using the identities

$$\begin{aligned} a \oplus b &= (a \not\Rightarrow b) \vee (a \not\Leftarrow b) \\ a \Leftrightarrow b &= (a \Rightarrow b) \wedge (a \Leftarrow b) \ , \end{aligned}$$

we get a new instance of our problem with at most $3m$ binary operators. Let $P'$ denote the polytope defined by this instance. Then, by Theorem 4.2, $P'$ is isomorphic to a face of a cut polytope on at most $n + 12m$ edges. On the other hand, it is clear by definition that $P$ is an orthogonal projection of $P'$. □

### 4.4. Constraints

So far we have discussed unconstrained Boolean optimization problems, where all Boolean functions in the problem formulation appear in the objective function. However, it is clear that the same approach works if we have constraints of the form $f_k = 0$ or $f_k = 1$, where $f_k$ is any Boolean function. In this case, we model $f_k$ exactly as we model the objective terms. If we consider the corresponding polytope $P$ and intersect it with the hyperplane $x_k = 0$ or $x_k = 1$, then we obviously get a face of $P$.

If we consider linear constraints instead of Boolean ones, the situation is more complicated. In general, we cannot rescue our polyhedral results in this case. However, in some special cases, the situation is again favorable. To give an example, consider the constraint

$$\sum_{s \in L} f_s \leq 1 \tag{8}$$

for an arbitrary subset $L \subseteq I \cup J \cup K$. This constraint states that at most one of the functions $f_s$ with $s \in L$ may evaluate to one. In order to model (8) without harming our polytope $P$, we introduce a zero-weight objective term $f_L = \bigvee_{s \in L} f_s$. Moreover, we have to add up to $|L| - 1$ connection variables. Then we can rephrase (8) as

$$\sum_{s \in L} f_s = f_L \ . \tag{9}$$

The latter formulation is preferable since $\sum_{s \in L} f_s \geq f_L$ is a valid constraint for $P$, so that (9) induces a face of $P$. In the same way, we can deal with an equation

$$\sum_{s \in L} f_s = 1 \ ,$$

stating that exactly one of the functions $f_s$ with $s \in L$ evaluates to one. Additionally to (9) we have to set $f_L = 1$ here, which again induces a face.

## 5. Experiments

Experimental results obtained with a straightforward implementation of our approach show that the increased modeling power leads to fast running times in practice. In this section, in order to demonstrate this by some examples, we shortly discuss two classes of instances taken from the

Max-SAT Evaluation 2007 [1, 9]. The evaluation [1] was based on several classes of instances. We chose those classes where the percentage of instances solved by the best participating algorithm was particularly small, namely the `logic-synthesis` and the `SPOT5` instances. Both classes belong to the partial Max-SAT category, i.e., some of the clauses have to be satisfied by every solution, while others have positive integer weights in the objective function.

In our implementation, we first apply some very simple preprocessing techniques. Their main objective is to reduce the number of variables in our model. Currently, we only do this in a very restricted way: we use the equivalent replacement

$$f_i \vee f_j \quad \text{for all } i,j \in I \text{ with } i \neq j \quad \Longleftrightarrow \quad \sum_{i \in I} (\neg f_i) \leq 1 \ ,$$

where each $f_i$ can be an arbitrary Boolean function. Notice that SAT solvers can only handle the former group of constraints, while in our approach we can also deal with the latter constraint, as explained in Section 4.4. This reduces the number of operators from $\binom{|I|}{2}$ to $|I|-1$. To detect such sets of constraints, we apply a simple algorithm for finding maximal cliques in the *conflict graph* defined as follows: the graph contains a vertex for every Boolean function $f_i$ appearing in the instance. Two vertices corresponding to Boolean functions $f_i$ and $f_j$ are adjacent if and only if the instance contains a subformula $f_i \vee f_j$.

In principle, we could apply much more sophisticated reformulation techniques in order to decrease the number of operators in our objective function, exploiting the fact that the result is not requested to be in CNF. However, developing such techniques was not the focus of our work. Moreover, in our opinion, the flexibility of our approach is best exploited if the user takes advantage of this flexibility already in the modeling phase.

It is also possible to save variables without decreasing the number of operators. For this, note that the variables corresponding to expressions $f_i \circ_1 f_j$ and $f_i \circ_2 f_j$, containing the same operands $f_i$ and $f_j$, determine each other: by (3), the variables for $f_i$, $f_j$, $f_i \circ_1 f_j$, and $f_i \circ_2 f_j$ satisfy an affine equation. In other words, we only need one variable for modeling both expressions $f_i \circ_1 f_j$ and $f_i \circ_2 f_j$.

Generally speaking, it is thus desirable to have a large number of expressions with the same pair of operands. In our implementation, we try to increase this number by exploiting associativity and commutativity of $\wedge$ and $\vee$: for all subformulae of the objective function containing only conjunctions or only disjunctions, we can group the operands in an arbitrary way, e.g., we can write $f_i \vee f_j \vee f_k$ as $(f_i \vee f_j) \vee f_k$, $(f_i \vee f_k) \vee f_j$, or $(f_j \vee f_k) \vee f_i$, which leads to different sets of variables in our model. Considering such expressions at different positions in the objective function simultaneously, one can possibly save many variables by choosing a nesting that leads to a large number of equivalent variables.

In our preprocessing phase, we determine the nesting in a greedy way: we always choose a pair of operands $(f_i, f_j)$ that jointly appears in a maximal number of such expressions, and introduce a variable for $f_i \circ f_j$. In other words, we consider $(f_i \circ f_j)$ as a single operand in the following, and proceed like this until all conjunctions and disjunctions are defined on exactly two operands. For example, having two formulae $f_i \vee f_j \vee f_k$ and $f_i \wedge f_j \wedge f_l$, we would determine the nesting as $(f_i \vee f_j) \vee f_k$ and $(f_i \wedge f_j) \wedge f_l$, as we only need one variable for both formulae $f_i \vee f_j$ and $f_i \wedge f_j$.

After preprocessing, an initial ILP formulation of the reduced instance is handed over to the CPLEX 11.1 MIP solver [5], which tries to optimize it using a classical branch-and-cut algorithm, see Section 3. Our only (but crucial) extension of the standard solver concerns the separation phase, where we make use of our results presented in Section 4: we first create the graph

corresponding to the cut polytope constructed in the proof of Lemma 4.1, after transforming the original problem to a quadratic one as described in the proof of Theorem 4.2. Whenever CPLEX tries to separate a fractional LP-solution, we first transform this solution to the variable space of the corresponding cut polytope. Then we can apply any separation algorithm for max-cut on this graph, with the transformed fractional point as input.

In our current implementation, we only separate cycle inequalities. We do this heuristically in the following way: we first compute a maximum weight spanning tree with respect to the current fractional LP-values. Then we consider all fundamental cycles defined by this tree, the number of which is linear in the number of edges of the graph. We only add inequalities defined on cycles not exceeding a given length (in our current implementation, the limit is $|V|/50$). Any resulting cutting plane can easily be transformed back to the original variable space. Note that this heuristic approach is much faster than the exact separation algorithm, but it may fail to find a violated cycle inequality even if there is one. Moreover, it does not necessarily find the most violated cycle inequalities, like the exact separation algorithm does.

An important advantage of our approach is the fact that all other components of the branch-and-cut algorithm can be applied to the original set of variables, since all cutting planes generated by our max-cut based separation algorithm are expressed in these variables. In particular, the solution of LP-relaxations as well as the branching can be performed on the original set of variables. The transformation to a max-cut problem defined in a higher-dimensional space is only necessary within the separation phase.

All experimental results reported in the following were obtained on an Intel Xeon processor with 2.33 GHz running Linux, i.e., on a machine that is slightly faster than the one used for the Max-SAT Evaluation 2007. As it was done with the codes evaluated in this contest, we set the cpu time limit of 30 minutes for all instances. We first report our results for two classes of particularly hard instances.

Instances in the class `logic-synthesis` are unweighted. The running times obtained with our approach described above turn out to be more than competitive: In 30 cpu minutes, we could solve to optimality 15 out of the 17 instances in this class. On the contrary, half of the 10 participants of the Max-SAT Evaluation could not solve a single of these instances, while the others could solve between 1 and 4 of them. More detailed results are displayed in Table 4. We state the number of solved instance and the average values of the running time in cpu seconds, the running time without preprocessing (i.e., the running time for the branch-and-cut algorithm), the number of subproblems in the enumeration tree, the number of nodes and edges in the auxiliary max-cut graph, and the number of cycle inequalities generated. We noticed in our experiments that generating a relatively small number of cycle inequalities can already lead to a significant reduction of the number of subproblems in the enumeration tree. Adding more cycle inequalities can reduce the number of subproblems even further, but at the expense of a longer separation time, which only pays off for larger instances. Notice that preprocessing uses a large portion of running time for the `logic-synthesis` problems, which however pays off as it shrinks these instances by up to 90 %. Here we would like to point out again that all preprocessing techniques we apply are very straightforward and are not adjusted to the given type of instances.

The second example class consists of the weighted `SPOT5` instances. Our results are again very positive: as shown in Table 4, we could solve 17 instances in the subclass `DIR` and 15 in `LOG`. In the Max-SAT Evaluation, the best participant could solve only 6 instances in each class. So it seems that our method is consistently superior to other approaches when applied to very hard Max-SAT instances, in spite of the fact that it is designed for much more general problems.

| instances | solved | total time | b & c time | b & c nodes | auxiliary graph nodes | auxiliary graph edges | cycle ineqs |
|---|---|---|---|---|---|---|---|
| `logic-synthesis` | 15/17 | 126.56 | 8.83 | 7.5 | 4357.0 | 9580.7 | 131.9 |
| `SPOT5/DIR` | 17/21 | 62.96 | 59.31 | 2430.7 | 1001.1 | 6877.0 | 241.1 |
| `SPOT5/LOG` | 15/21 | 202.88 | 130.06 | 1751.3 | 1636.6 | 8119.6 | 1039.9 |

Table 4: Experimental results for `logic-synthesis` and `SPOT5` instances.

The presented running times, in particular those for `logic-synthesis`, suggest that a lot of work is done in the preprocessing. As discussed above, the preprocessing aims at a more compact reformulation of the problem. Indeed, when running the same instances without preprocessing, running times are significantly longer and less instances can be solved, as shown in Table 5 (where the running time for the branch-and-cut algorithm now coincides with the total time). This is mainly due to the larger size of the auxiliary graphs used in the max-cut separation. Nevertheless, our algorithm still outperforms all participants of the Max-SAT Evaluation on these instances.

| instances | solved | total time | b & c nodes | auxiliary graph nodes | auxiliary graph edges | cycle ineqs |
|---|---|---|---|---|---|---|
| `logic-synthesis` | 12/17 | 139.62 | 18.7 | 13410.4 | 37155.5 | 415.9 |
| `SPOT5/DIR` | 16/21 | 29.62 | 2114.9 | 1001.8 | 6434.6 | 145.8 |
| `SPOT5/LOG` | 6/21 | 9.00 | 913.0 | 555.3 | 1459.2 | 621.8 |

Table 5: Experimental results without preprocessing.

The instances discussed so far turned out to be the hardest instances for all solvers in the Max-SAT Evaluation 2007. On these instances, our approach performs particularly well. However, it is also competitive on most of the other instances; see Table 6 for a complete overview of our experimental results. The results show that our approach "prefers" sparse instances, i.e., instances with a small ratio between the number of operators and the number of basic variables. This could be expected, as it is well-known that integer programming approaches to max-cut perform much better on sparse instances. For dense instances, we plan to develop other methods in the future, e.g., using approaches based on semidefinite programming.

Moreover, as our method mainly addresses the dual side of the optimization, while not containing any nontrivial primal heuristics, it is not surprising that traditional Max-SAT solvers perform better on instances where all clauses can be satisfied simultaneously. On such instances, the optimization aspect takes a back seat, and dual bounds are necessarily trivial. However, it is clearly plausible that our algorithm can be improved by combining it with any primal heuristic for SAT or Max-SAT.

## 6. Conclusion

We presented a novel integer programming approach to nonlinear Boolean optimization problems. Unlike other approaches, it avoids adding many artificial variables to the model, at the

| | instances | solved | total time | b & c time | b & c nodes |
|---|---|---|---|---|---|
| | MAX3SAT/40VARS | 16/40 | 417.38 | 417.35 | 1282.4 |
| | MAX3SAT/50VARS | 10/40 | 311.04 | 311.02 | 3137.8 |
| | MAX3SAT/60VARS | 10/40 | 176.65 | 176.64 | 2154.6 |
| | MAX3SAT/70VARS | 10/40 | 89.68 | 89.67 | 868.0 |
| | SPINGLASS | 20/20 | 45.43 | 45.42 | 1679.0 |
| | RAMSEY | 24/48 | 168.92 | 168.42 | 302.4 |
| | MAX2SAT/100VARS | 106/110 | 260.72 | 260.70 | 12504.4 |
| | MAX2SAT/140VARS | 78/110 | 189.44 | 189.43 | 7999.4 |
| | MAX2SAT/60VARS | 110/110 | 4.98 | 4.96 | 112.2 |
| | MAX3SAT/40VARS | 23/50 | 409.10 | 409.07 | 943.4 |
| | MAX3SAT/60VARS | 10/50 | 0.52 | 0.51 | 10.2 |
| | MAX3SAT/80VARS | 11/50 | 45.20 | 45.19 | 129.0 |
| | MAXCUT/DIMACS_MOD | 32/62 | 292.89 | 292.87 | 5483.7 |
| | MAXCUT/RANDOM | 19/40 | 633.38 | 633.36 | 35030.7 |
| ⋆ | MAXCUT/SPINGLASS | 5/5 | 231.79 | 231.74 | 1214.2 |
| | RAMSEY | 24/48 | 119.43 | 118.94 | 108.5 |
| | WMAX2SAT | 82/90 | 184.32 | 184.31 | 10402.9 |
| | WMAX3SAT | 11/80 | 219.02 | 219.00 | 2371.1 |
| | WMAXCUT/DIMACS_MOD | 38/62 | 284.39 | 284.38 | 4767.1 |
| | WMAXCUT/RANDOM | 33/40 | 542.00 | 541.97 | 31876.1 |
| ⋆ | WMAXCUT/SPINGLASS | 5/5 | 46.71 | 46.66 | 107.0 |
| | RANDOM/PMAX2SAT | 53/90 | 463.60 | 463.44 | 2037.4 |
| | RANDOM/PMAX3SAT | 18/60 | 88.05 | 88.02 | 536.1 |
| | MAXCLIQUE/RANDOM | 96/96 | 65.74 | 64.75 | 7314.9 |
| | MAXCLIQUE/STRUCTURED | 30/62 | 266.95 | 213.55 | 17505.0 |
| | MAXONE/3SAT | 58/80 | 92.38 | 92.35 | 437.5 |
| | MAXONE/STRUCTURED | 41/60 | 65.88 | 61.69 | 373.8 |
| ⋆ | PSEUDO/garden | 6/7 | 215.79 | 215.78 | 6511.0 |
| ⋆ | PSEUDO/logic-synthesis | 15/17 | 126.56 | 8.83 | 7.5 |
| | PSEUDO/primes-dimacs-cnf | 78/148 | 38.11 | 37.41 | 649.0 |
| | PSEUDO/routing | 12/15 | 47.26 | 28.81 | 11.1 |
| | WCSP/MAXCSP/DENSE_LOOSE | 20/20 | 39.94 | 39.90 | 5698.6 |
| | WCSP/MAXCSP/DENSE_TIGHT | 20/20 | 34.27 | 34.24 | 35760.1 |
| | WCSP/MAXCSP/SPARSE_LOOSE | 20/20 | 14.91 | 14.87 | 4897.7 |
| | WCSP/MAXCSP/SPARSE_TIGHT | 20/20 | 47.94 | 47.92 | 18979.6 |
| ⋆ | WCSP/WQUEENS | 7/7 | 0.98 | 0.69 | 24.3 |
| | RANDOM/WPMAX2SAT | 56/90 | 532.59 | 532.41 | 2065.8 |
| | RANDOM/WPMAX3SAT | 18/60 | 209.05 | 209.01 | 1615.3 |
| ⋆ | AUCTIONS/AUC_PATHS | 88/88 | 0.07 | 0.01 | 1.0 |
| | AUCTIONS/AUC_REGIONS | 84/84 | 4.09 | 0.26 | 1.0 |
| ⋆ | AUCTIONS/AUC_SCHEDULING | 84/84 | 0.93 | 0.04 | 1.0 |
| | PSEUDO/factor | 186/186 | 35.84 | 34.87 | 126.7 |
| | PSEUDO/miplib | 5/16 | 14.08 | 14.01 | 773.8 |
| | QCP | 18/25 | 75.71 | 70.11 | 135.6 |
| | WCSP/PLANNING | 71/71 | 94.65 | 4.29 | 9.8 |
| ⋆ | SPOT5/DIR | 17/21 | 62.96 | 59.31 | 2430.7 |
| ⋆ | SPOT5/LOG | 15/21 | 202.88 | 130.06 | 1751.3 |

Table 6: Experimental results for all instances of the Max-SAT Evaluation 2007, the runtime limit is 30 min. On instances marked with a star, we significantly outperform all participating solvers.

same time allowing to derive tight linear relaxations of the corresponding polytope. In the special case of binary polynomial optimization, which has been investigated in [4], our approach proved to be very successful in practical experiments. Computational results for hard Max-SAT instances reported in this paper seem to confirm the good performance of our approach also for other nonlinear optimization problems.

The software described in the paper is accessible online at the web site

`we.logoptimize.it`

## Acknowledgments

We thank the anonymous referees for their comments that helped us to improve the readability of the paper.

## References


[1] J. Argelich, C. M. Li, F. Manyà, and J. Planes, "The first and second max-sat evaluations," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 4, pp. 251–278, 2008.

[2] F. Barahona and A. R. Mahjoub, "On the cut polytope," *Mathematical Programming*, vol. 36, pp. 157–173, 1986.

[3] E. Boros and P. L. Hammer, "Pseudo-boolean optimization," *Discrete Applied Mathematics*, vol. 123, no. 1–3, pp. 155–225, 2002.

[4] C. Buchheim and G. Rinaldi, "Efficient reduction of polynomial zero-one optimization to the quadratic case," *SIAM Journal on Optimization*, vol. 18, no. 4, pp. 1398–1413, 2007.

[5] CPLEX 11.1, `www.ilog.com/products/cplex`.

[6] C. De Simone, "The cut polytope and the Boolean quadric polytope," *Discrete Mathematics*, vol. 79, no. 1, pp. 71–75, 1990.

[7] M. Deza and M. Laurent, *Geometry of Cuts and Metrics*, vol. 15 of *Algorithms and Combinatorics*. Springer-Verlag, 1997.

[8] P. Hammer, "Some network flow problems solved with pseudo-boolean programming," *Operations Research*, vol. 13, pp. 388–399, 1965.

[9] F. Heras, J. Larrosa, S. de Givry, and T. Schiex, "2006 and 2007 max-sat evaluations: Contributed instances," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 4, pp. 239–250, 2008.

[10] S. Joy, J. Mitchell, and B. Borchers, "A branch-and-cut algorithm for MAX-SAT and weighted MAX-SAT," in *Satisfiability Problem: Theory and Applications*, vol. 35 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pp. 519–536, American Mathematical Society, 1997.

[11] V. M. Manquinho and J. Marques-Silva, "On applying cutting planes in DLL-based algorithms for pseudo-boolean optimization," in *Theory and Applications of Satisfiability Testing*, vol. 3569 of *LNCS*, pp. 451–458, Springer, 2005.

[12] V. M. Manquinho and J. Marques-Silva, "On using cutting planes in pseudo-boolean optimization," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 2, pp. 209–219, 2006.

[13] M. Padberg and G. Rinaldi, "A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems," *SIAM Review*, vol. 33, pp. 60–100, 1991.

[14] H. M. Sheini and K. A. Sakallah, "Pueblo: A hybrid pseudo-boolean SAT-solver," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 2, pp. 165–189, 2006.