Linear 0-1 Inequalities and Extended Clauses

Peter Barth

MPI–I–94–216 April 1994

Author's Address

Peter Barth
Max-Planck-Institut für Informatik
Im Stadtwald
66123 Saarbrücken, Germany
Email: Peter.Barth@mpi-sb.mpg.de

Publication Notes

An earlier version of this paper has appeared as [Bar93].

Acknowledgements

The author is grateful to Alexander Bockmayr for his valuable comments and fruitful discussions. The work was supported by the German Ministry for Research and Technology (BMFT) (contract ITS 9103), the ESPRIT Basic Research Project ACCLAIM (contract EP 7195) and the ESPRIT Working Group CCL (contract EP 6028).

Abstract

Extended clauses are the basic formulas of the 0-1 constraint solver for the constraint logic programming language CLP($\mathcal{PB}$). We present a method for transforming an arbitrary linear 0-1 inequality into a set of extended clauses, such that the solution space remains invariant. After applying well-known linearization techniques on non-linear 0-1 constraints followed by the presented transformation method, we are able to handle arbitrary 0-1 constraints in CLP($\mathcal{PB}$).

The transformation method presented relies on cutting planes techniques known from 0-1 integer programming. We develop specialized redundancy criteria and so produce the minimal number of extended clauses needed for preserving equivalence. The method is enhanced by using a compact representation of linear 0-1 inequalities and extended clauses. Unit resolution for classical clauses is generalized to pseudo-Boolean unit resolution for arbitrary linear 0-1 inequalities. We extend the transformation method to constrained transformation when the inequality to be transformed is part of a larger set of linear 0-1 inequalities. Furthermore the method can be used to obtain all strongest extended cover inequalities of a knapsack inequality.

Keywords

constraint logic programming, pseudo-Boolean constraints, 0-1 integer programming, extended clauses, extended cover inequalities.

Contents

1 Introduction

0-1 constraint satisfaction problems have been investigated for a long time. Instantiating the computational domain $\mathcal{X}$ in the CLP($\mathcal{X}$) scheme of Jaffar and Lassez [JL87] with 0-1 or pseudo-Boolean constraints was first introduced by Bockmayr [Boc93]. The resulting language CLP($\mathcal{PB}$) aims to be a logic programming environment suitable for various applications where 0-1 problems naturally arise, which is the case for many AI-applications [BB93] and operations research.

In the context of constraint logic programming there are several special requirements for the constraint solver. Ideally, a constraint solver should detect inconsistency of a set of constraints, or, if the constraint set is inconsistent, provide a *solved form*, that is a description of all solutions of the constraint set. Van Hentenryck [VH89] introduces constraints over finite domains. The finite domain constraint solver is based on local consistency techniques and is successfully applied to a large variety of combinatorial problems. The main disadvantage of this approach is that, in order to achieve good computational behavior, constraint solving is relaxed such that

- global consistency of the constraint set is not assured and
- there is no solved form.

Obviously, 0-1 constraint satisfaction problems can be easily expressed as finite domain problems by restricting the domain of each domain variable to $\{0, 1\}$. Solving 0-1 problems and classical Boolean problems with this approach can be quite efficient [CD93]. Our goal is to provide a *complete* constraint solver for 0-1 constraints that computes a solved form of the accumulated constraints and provides an easy check of logical entailment and therefore satisfiability. Especially for concurrent constraint programming languages [SR90], deciding logical entailment of constraints is essential.

Granot et al. [GH71] showed how to express 0-1 constraints as an equivalent set of classical clauses. 0-1 constraints can then be solved with resolution based methods [Rob65] in an incremental way [Jac92]. The solved form then is a set of prime implicants, which fits exactly into our requirements. Early attempts of transforming arbitrary pseudo-Boolean constraints into an equivalent set of clauses failed mainly because of the large number of generated clauses [GH71, GG80]. Recently [Hoo92] has presented a deductive system "Generalized Resolution" working with *extended clauses*, an extension of classical clauses. Semantically, a classical Boolean clause states that at least one of its literals has to be true. So a classical Boolean clause

$$L_1 \vee \ldots \vee L_n$$

can be expressed as linear 0-1 inequality

$$L_1 + \cdots + L_n \geq 1 \ ,$$

where a negated literal $\overline{X_i}$ corresponds to $(1 - X_i)$ and the truth values *false* and *true* correspond to 0 and 1 respectively. Extended clauses generalize classical clauses by allowing other values than 1 on the right-hand side. An extended clause is of the form

$$L_1 + \cdots + L_n \geq d \ ,$$

expressing that at least d of the n literals in the extended clause have to be 1. Hooker [Hoo92] shows that there exists an equivalent notion of prime implicants for a set of extended clauses, and the deductive system "Generalized Resolution" generates such a set of *prime extended clauses.* We propose to use extended clauses as basic constraints in our complete constraint solver, because

- there is a deductive system computing a solved form where satisfiability and logical entailment are easily decidable,
- the solved form describes all possible solutions and is *explanative* because of the natural d out of n interpretation,
- extended clauses are more expressive than classical clauses and provide a much more compact representation of information while preserving computational manageability.

For the latter point note that the equivalent representation of the extended clause $L_1 + \cdots + L_n \geq d$ using classical clauses without introducing new variables is

$$\bigwedge_{I \subseteq \{1,\ldots,n\} : |I| = n-d+1} (\vee_{i \in I} L_i) \enspace ,$$

a conjunction of $\binom{n}{n-d+1}$ clauses. Especially the more compact representation enables us to represent 0-1 constraints as an equivalent set of extended clauses, because the number of extended clauses needed is typically much smaller than the number of equivalent classical clauses. Applying standard linearization techniques to nonlinear constraints [BM84], we can transform arbitrary 0-1 constraints into a set of *linear 0-1 inequalities.* These linear 0-1 inequalities must then be transformed into an equivalent set of extended clauses in order to be processed by the constraint solver. In this paper we present a transformation method producing efficiently the minimal number of extended clauses that are equivalent to a given linear 0-1 inequality. We adopt cutting plane techniques, known from 0-1 integer programming, for generating valid extended clauses. Specialized redundancy criteria ensure that the minimal number of needed extended clauses is generated. Note that the number of needed extended clauses may be exponential w.r.t. the number of variables in the inequality. We introduce a compact representation of linear 0-1 inequalities and extended clauses and incorporate the technique into the transformation method. A linear 0-1 inequality is typically part of a set of linear 0-1 inequalities, and we show how to use this information in order to reduce the number of generated extended clauses. In this context we also present simplification methods and generalize the unit resolution procedure of classical clauses to linear 0-1 inequalities.

Beside the use of the presented method for the constraint solver of CLP($\mathcal{PB}$), we also relate extended cover inequalities [NW88] to the produced set of extended clauses. We show that the method generates the set of all *strongest, non redundant* extended cover inequalities, which can be used in a preprocessing phase of a 0-1 integer programming solver. Another application is the implementation of one of the basic deduction rules of "Generalized Resolution" that allows to apply several resolution steps at once.

The paper is organized as follows. In Sect. 2 we give basic definitions and introduce extended clauses and the pseudo-Boolean normal form of linear 0-1 inequalities. We discuss the generation of valid extended clauses from a pseudo-Boolean inequality in Sect. 3, and prove the equivalence of the set of generated extended clauses and the pseudo-Boolean inequality. Strong redundancy criteria, that allow an efficient redundancy test, are presented in Sect. 4. With help of the redundancy criteria only the minimal number of extended clauses is generated. We sketch an implementation of the transformation method in Sect. 5. An important improvement is the use of symmetries along with the compact form of pseudo-Boolean inequalities and extended clauses, introduced in Sect. 6. In Sect. 7 we generalize unit resolution for classical clauses to pseudo-Boolean unit resolution. Unit relaxation of a set of pseudo-Boolean inequalities is defined and we obtain a simple incomplete unsatisfiability check that is used to detect some of the fixed literals of a set of pseudo-Boolean inequalities. In Sect. 8 the transformation procedure is generalized to the case where the pseudo-Boolean inequality is part of a larger set of pseudo-Boolean inequalities. We present in this context also simplification routines for strengthening a pseudo-Boolean inequality and splitting a pseudo-Boolean inequality into a set of equivalent but simpler pseudo-Boolean inequalities. Two applications of the transformation routine, namely the implementation of one of the basic deduction rules of "Generalized Resolution" and traditional 0-1 integer programming, are described in Sect. 9. We show how to apply the presented techniques in a preprocessing phase of 0-1 integer programming problems and give some encouraging computational results followed by the conclusion in Sect. 10.

2 Preliminaries

We use the following conventions and abbreviations [Hoo92]. Let $\mathcal{B} = \{X_1, X_2, \ldots, X_n\}$ be a finite set of *Boolean variables* and the domain of the X_i be $\{0, 1\}$. A *literal* L_j is either a Boolean variable X_i (a positive literal), or the negation of a variable $\overline{X_i}$ (a negative literal). Let $\mathcal{L}$ be the set of all literals. The negation of a negative literal $\overline{\overline{X_i}}$ is always simplified to X_i. We denote by $\mathsf{Var}(X_i) = \mathsf{Var}(\overline{X_i}) = X_i$ the variable of a literal. An assignment is a mapping α from $\mathcal{B}$ to $\{0, 1\}$. An assignment can also be seen as a 0-1 vector of dimension n. We extend α to a mapping from $\mathcal{L}$ to $\{0, 1\}$ by defining $\alpha(\overline{X_i}) := 1 - \alpha(X_i)$. A set of literals $L = \{L_1, \ldots, L_n\}$ can also be seen as a sum of literals $L_1 + \cdots + L_n$. We extend α to a mapping from $2^{\mathcal{L}}$ to $\mathbb{N}$ by defining $\alpha(L) := \alpha(L_1) + \cdots + \alpha(L_n)$, that is the number of literals that are mapped to 1. A *product* c_iL_i is a pair of an integer coefficient c_i and a literal L_i. A set of products $cL = \{c_1L_1, \ldots, c_nL_n\}$ can also be seen as sum of products $c_1L_1 + \cdots + c_nL_n$. We extend α to a mapping from $2^{\mathbb{Z} \times \mathcal{L}}$ to $\mathbb{Z}$ by defining $\alpha(cL) := c_1\alpha(L_1) + \cdots + c_n\alpha(L_n)$, that is the sum over the coefficients whose literals are mapped to 1. For a set of products cL we denote by $s(c) := c_1 + \cdots + c_n$ the sum over the integer coefficients of cL.

An *extended clause* is of the form

$$L_1 + \cdots + L_n \geq d$$

where $0 \leq d \leq n + 1$ and $\mathsf{Var}(L_i) \neq \mathsf{Var}(L_j)$ for all $1 \leq i, j \leq n$. An assignment α *satisfies* an extended clause $L \geq d$ if at least d of its n literals are mapped to 1, that is if $\alpha(L) \geq d$. If $d \geq n + 1$ then there is no assignment satisfying $L \geq d$ and we abbreviate $L \geq d$ by $\Box$. If $d = 0$ then every assignment is a satisfying assignment of $L \geq d$. We say then that $L \geq d$ is a *tautology* and abbreviate it by $\top$. We denote by $\deg(L \geq d) = d$ the *degree* of an extended clause $L \geq d$. Note that classical clauses are extended clauses with degree 1.

A *linear pseudo-Boolean (or 0-1) inequality* is of the form

$$c_1L_1 + \cdots + c_nL_n \geq d$$

where the c_i and d are integer numbers. An assignment α *satisfies* a linear pseudo-Boolean inequality $cL \geq d$ if $\alpha(cL) \geq d$. If there is no assignment satisfying $cL \geq d$ we abbreviate $cL \geq d$ by $\Box$. If every assignment satisfies $cL \geq d$, then $cL \geq d$ is a tautology and we abbreviate it by $\top$. As for extended clauses, we like to have only one occurrence of a variable in a linear pseudo-Boolean inequality, and we want to be able to immediately detect whether $cL \geq d$ is $\Box$, or whether it is a tautology.

Definition 2.1 A linear pseudo-Boolean inequality $cL \geq d$ is in (pseudo-Boolean) *normal form* if

$$d \geq c_1 \geq \cdots \geq c_n \geq 1 \text{ and } \mathsf{Var}(L_i) \neq \mathsf{Var}(L_j) \text{ for all } 1 \leq i < j \leq n \enspace .$$

We assume that $d \geq 1$ since otherwise the linear pseudo-Boolean inequality in normal form is a tautology, that is, it is valid for every assignment α and therefore need not be considered.

Proposition 2.2 *[HR68] For each non tautologous linear 0-1 inequality there is a linear pseudo-Boolean inequality in normal form admitting the same set of satisfying assignments.*

PROOF: We begin with an arbitrary linear pseudo-Boolean inequality

$$e_1L'_1 + \cdots + e_mL'_m \geq d' \enspace , \tag{1}$$

and construct in the following a linear pseudo-Boolean inequality in normal from. We first apply several arithmetic equivalence transformations. We rewrite (1) such that literals containing the same variable are grouped together and obtain

$$a_1X_1 + b_1\overline{X_1} + \cdots + a_nX_n + b_n\overline{X_n} \geq d' \enspace ,$$

where the X_i are pairwise different. For each i such that $a_i = b_i$ we can simplify $a_iX_i + b_i\overline{X_i}$ to the constant a_i and move the constant a_i to the right-hand side. So let us assume that $a_i \neq b_i$ for all $1 \leq i \leq n$. We next replace $a_iX_i + b_i\overline{X_i}$ by $c'_iL_i + c''_i$ for all $1 \leq i \leq n$ according to

$$c'_iL_i + c''_i := \begin{cases} (a_i - b_i)X_i + b_i & \text{if} \quad a_i > b_i \\ (b_i - a_i)\overline{X_i} + a_i & \text{if} \quad b_i > a_i \enspace . \end{cases}$$

Note that the c'_i are all positive. Bringing the constants c''_i to the right-hand side gives us the new right-hand side $d = d' - \sum_{i=1}^{n} c''_i$. After re-indexing according to the ordering restriction we have brought the linear 0-1 inequality into the form

$$c'_1L_1 + \cdots + c'_nL_n \geq d \enspace . \tag{2}$$

where $c'_1 \geq \cdots \geq c'_n \geq 1$ and $\mathsf{Var}(L_i) \neq \mathsf{Var}(L_j)$ for all $1 \leq i < j \leq n$. Note that $d \geq 1$, since otherwise (1) is a tautology. So far we have only applied arithmetic equivalence transformations, hence an assignment α satisfies (1) if and only if α satisfies (3). When constructing the pseudo-Boolean normal form of a linear pseudo-Boolean inequality we can detect at this point whether we have a tautology or not.

Suppose that $c'_i > d$ for some i, then every assignment α where $\alpha(L_i) = 1$ maps the left-hand side of (2) to an integer greater than d and satisfies (2). For all assignments α where $\alpha(L_i) = 0$ the value of the left-hand side is independent of c_i. Hence we can safely replace each c'_i by d if $c'_i > d$. This step is also called *coefficient reduction* [CJP83]. Formally we define

$$c_i := \begin{cases} c'_i & \text{if} \quad c'_i \le d \\ d & \text{if} \quad c'_i > d \end{cases}$$

for all $1 \le i \le n$, and thereby obtain the pseudo-Boolean normal form

$$c_1 L_1 + \cdots + c_n L_n \ge d \tag{3}$$

of (1), where $d \ge c_1 \ge \cdots \ge c_n \ge 1$. Obviously an assignment α satisfies (1) if and only if α satisfies (2) if and only if α satisfies (3). $\square$

Note that a linear pseudo-Boolean inequality in normal form $cL \ge d$ is satisfiable if and only if $s(c) \ge d$, because $c_i > 0$ for all $1 \le i \le n$. Hence $cL \ge d$ is unsatisfiable if and only if $s(c) < d$, and we can easily decide whether $cL \ge d$ is $\square$.

Example 2.3 *Let us transform the linear 0-1 inequality*

$$-6 \cdot X_1 + -5 \cdot X_6 + 4 \cdot X_3 + 3 \cdot X_6 + 3 \cdot X_4 + 3 \cdot X_3 + 2 \cdot X_1 + 2 \cdot \overline{X_3} + 2 \cdot X_5 + -2 \cdot \overline{X_6} + 1 \cdot X_2 \ge 7 \tag{4}$$

into pseudo-Boolean normal form using Prop. 2.2. We first regroup the products and obtain

$$4 \cdot X_3 + 3 \cdot X_3 + 2 \cdot \overline{X_3} + -6 \cdot X_1 + 2 \cdot X_1 + 3 \cdot X_4 + 2 \cdot X_5 + 1 \cdot X_2 + -5 \cdot X_6 + 3 \cdot X_6 + -2 \cdot \overline{X_6} \ge 7$$

which simplifies to

$$7 \cdot X_3 + 2 \cdot \overline{X_3} + -4 \cdot X_1 + 0 \cdot \overline{X_1} + 3 \cdot X_4 + 0 \cdot \overline{X_4} + 2 \cdot X_5 + 0 \cdot \overline{X_5} + 1 \cdot X_2 + 0 \cdot \overline{X_2} + -2 \cdot X_6 + -2 \cdot \overline{X_6} \ge 7 \ .$$

Because $a_6 = -b_6 = -2$ we simplify $-2 \cdot X_6 + -2 \cdot \overline{X_6}$ to -2 and derive

$$7 \cdot X_3 + 2 \cdot \overline{X_3} + -4 \cdot X_1 + 0 \cdot \overline{X_1} + 3 \cdot X_4 + 0 \cdot \overline{X_4} + 2 \cdot X_5 + 0 \cdot \overline{X_5} + 1 \cdot X_2 + 0 \cdot \overline{X_2} \ge 7 + 2 = 9 \ .$$

We next replace the $a_i X_i + b_i \overline{X_i}$ by $c'_i L_i + c''_i$ according to Prop. 2.2 and obtain

$$5 \cdot X_3 + 2 + 4 \cdot \overline{X_1} + -4 + 3 \cdot X_4 + 0 + 2 \cdot X_5 + 0 + 1 \cdot X_2 + 0 \ge 9 \ .$$

By bringing the constants to the right-hand side we have

$$5 \cdot X_3 + 4 \cdot \overline{X_1} + 3 \cdot X_4 + 2 \cdot X_5 + 1 \cdot X_2 \ge 9 - 2 + 4 = 11 \ .$$

After re-indexing we obtain the pseudo-Boolean normal form

$$5 \cdot L_1 + 4 \cdot L_2 + 3 \cdot L_3 + 2 \cdot L_4 + 1 \cdot L_5 \ge 11 \tag{5}$$

of (4), where $L_1 = X_3$, $L_2 = \overline{X_1}$, $L_3 = X_4$, $L_4 = X_5$, and $L_5 = X_2$.

Assumption 2.4 *From now on we assume that all linear 0-1 inequalities are in pseudo-Boolean normal form.*

Note that an extended clause is a linear pseudo-Boolean inequality $cL \geq d$ where all the c_i are 1.

Let us say that the *extension* $\mathsf{Ext}(I)$ of a linear pseudo-Boolean inequality I is the set of assignments α satisfying I. The extension $\mathsf{Ext}(S)$ of a set of linear pseudo-Boolean inequalities S is the intersection of the extensions of the linear pseudo-Boolean inequalities in S. A set of linear pseudo-Boolean inequalities S is *satisfiable* if $\mathsf{Ext}(S)$ is nonempty. A set of linear pseudo-Boolean inequalities S *(strictly) dominates* a set of pseudo-Boolean inequality S' if $\mathsf{Ext}(S)$ is a *(proper) subset* of $\mathsf{Ext}(S')$. We also write $S \models S'$ if S dominates S'. We abbreviate $\{I\} \models S'$ by $I \models S'$ and $S \models \{I\}$ by $S \models I$. Two sets of linear pseudo-Boolean inequalities S, S' are *equivalent* if $\mathsf{Ext}(S) = \mathsf{Ext}(S')$. If a set of linear pseudo-Boolean inequalities S dominates a linear pseudo-Boolean inequality I, we say I is *valid* w.r.t. S.

Let us say that a linear pseudo-Boolean inequality $cL \geq d$ *reduces* to $c'L \geq d'$ (the *reduction*) if $0 \leq c'_i \leq c_i$ for all $1 \leq i \leq n$ and $d' = d - (s(c) - s(c'))$. We require that $d' \geq 1$ and so forbid reduction to tautologies. Because $\alpha(cL) - d \leq \alpha(c'L) - d'$ for all assignments α, a linear pseudo-Boolean inequality dominates all its reductions. We denote by $\mathsf{Red}(cL \geq d)$ the set of all reductions of a linear pseudo-Boolean inequality $cL \geq d$. We say a linear pseudo-Boolean inequality $cL \geq d$ *strictly reduces* to $c'L' \geq d'$ if $c'L' \geq d'$ is a reduction and either $c'_i = c_i$ or $c'_i = 0$ for all $1 \leq i \leq n$. We say we have *eliminated* the literal L_i if $c'_i = 0$. We denote by $\mathsf{SRed}(cL \geq d)$ the set of all strict reductions of a linear pseudo-Boolean inequality $cL \geq d$. A reduction of an extended clause $L \geq d$ is $L \setminus L' \geq d - |L'|$ where $L' \subseteq L$.

Deciding domination between linear pseudo-Boolean inequalities is an NP-complete problem[1]. We show next that extended clauses are a generalization of classical clauses, where one of the main properties, namely domination, remains easily decidable.

Lemma 2.5 *$L \geq d$ dominates $L' \geq d'$ iff*

$$|L \setminus L'| \leq d - d' \quad . \tag{6}$$

PROOF:

"$\Leftarrow$": We eliminate all literals from L that are not in L' and obtain $L \cap L' \geq d - |L \setminus L'|$, that is a reduction of $L \geq d$. Since $|L \setminus L'| \leq d - d'$, we have $0 \leq d' \leq d - |L \setminus L'| \leq d$ and therefore $L \geq d$ also dominates $L \cap L' \geq d'$. Since $L \cap L' \subseteq L'$, we know that $L \geq d$ dominates $L' \geq d'$.

"$\Rightarrow$": We assume that $|L \setminus L'| > d - d'$ and show that in this case $L \geq d$ does not dominate $L' \geq d'$ by constructing an assignment α satisfying $L \geq d$, but not $L' \geq d'$. We choose $\alpha \in \mathsf{Ext}(L \geq d)$ such that $\alpha(L) = d$ and $\alpha(L_i) = 0$ for all $L_i \in L' \setminus L$. Then we know that $\alpha(L') = d - |L \setminus L'|$. Since $|L \setminus L'| > d - d'$, we have $d - |L \setminus L'| < d'$ and therefore $\alpha(L') < d'$. But then α is not in the extension of $L' \geq d'$, that is $L \geq d$ does not dominate $L' \geq d'$. □

When deciding domination between two classical clauses $L \geq 1$ and $L' \geq 1$, condition (6) reduces

[1] Note that a linear pseudo-Boolean inequality $cL \geq d$ dominates another linear pseudo-Boolean inequality $c'L' \geq d'$ iff the maximum of $c'L'$ subject to $cL \geq d$ is greater than d'. So deciding domination between two linear pseudo-Boolean inequalities involves solving a knapsack problem [NW88].

to $|L \setminus L'| \leq 0$, which is equivalent to $L \subseteq L'$; the usual condition for deciding domination (or implication) between classical clauses.

We say a set of extended clauses S is in *normal form* if no extended clause in S dominates another extended clause in S. Obviously every set of extended clauses S can be easily brought into normal form by deleting all extended clauses that are dominated by other extended clauses in S.

Assumption 2.6 *From now on we assume that all sets of extended clauses are in normal form.*

3 Generating Valid Extended Clauses

We show how to generate a set of extended clauses equivalent to a linear pseudo-Boolean inequality. We define the *strongest extended clause* of a linear pseudo-Boolean inequality, and relate it to *cutting plane* inequalities known from 0-1 integer programming [NW88]. The equivalence of the set of all strongest extended clauses of all strict reductions of a linear pseudo-Boolean inequality and the linear[2] pseudo-Boolean inequality itself, is the key theorem of this section.

3.1 The Strongest Extended Clause

Linear combination and integer rounding are well known techniques of 0-1 integer programming for deriving valid inequalities [NW88]. We adopt these techniques in order to generate valid extended clauses.

Let us divide a pseudo-Boolean inequality in normal form

$$c_1 L_1 + \cdots + c_n L_n \geq d \tag{7}$$

by a positive integer k and round the fractional coefficients of the left-hand side yielding

$$\lceil c_1/k \rceil L_1 + \cdots + \lceil c_n/k \rceil L_n \geq d/k \ ,$$

where $\lceil a \rceil = \min(\{b \in \mathbb{N} : b \geq a\})$. Because rounding only increases the left-hand side, we have generated a valid inequality w.r.t. (7). Since the coefficients $\lceil c_i/k \rceil$ are integer, the left-hand side is integer for all assignments. Therefore, the right-hand side d/k can be replaced by the largest integer less than or equal to d/k, that is $\lceil d/k \rceil$. We obtain the valid pseudo-Boolean inequalities

$$\lceil c_1/k \rceil L_1 + \cdots + \lceil c_n/k \rceil L_n \geq \lceil d/k \rceil \ ,$$

dominated by (7), for all positive integers k. These pseudo-Boolean inequalities are a subclass of the so called *cutting plane* inequalities [NW88], obtained from a linear combination of the pseudo-Boolean inequality and the valid bounds $L_i \geq 0$ and $-L_i \geq -1$ for the literals L_i followed by integer rounding of the coefficients and the right-hand side. We obtain the valid extended clause

$$L_1 + \cdots + L_n \geq \lceil d/c_1 \rceil \ , \tag{8}$$

dominated by (7), if we set $k = c_1$ since then the $\lceil c_i/k \rceil$ are all 1. The extended clause (8) derived from a pseudo-Boolean inequality $cL \geq d$ is denoted by $\mathsf{CP}(cL \geq d)$, and we know that $cL \geq d$ dominates $\mathsf{CP}(cL \geq d)$.

[2]Since we do not consider nonlinear pseudo-Boolean inequalities throughout the paper we may omit the word "linear" in the following.

Example 3.1 *Let $cL \geq d$ be*

$$5 \cdot L_1 + 4 \cdot L_2 + 3 \cdot L_3 + 2 \cdot L_4 + 1 \cdot L_5 \geq 11 \ ;$$

then $\mathsf{CP}(cL \geq d)$ *is*

$$L_1 + L_2 + L_3 + L_4 + L_5 \geq 3 \ .$$

For a set of pseudo-Boolean inequalities S we denote by $\mathsf{CP}(S)$ the set of extended clauses $\mathsf{CP}(cL \geq d)$ for all $cL \geq d$ in S. The cutting plane operation CP generates a valid extended clause $L \geq d'$ dominated by a pseudo-Boolean inequality $cL \geq d$. We are now interested in the greatest integer β such that $L \geq \beta$ is dominated by $cL \geq d$.

Proposition 3.2 *A pseudo-Boolean inequality $cL \geq d$ dominates the extended clause*

$$L \geq \beta \ , \tag{9}$$

where β is the smallest β such that $\sum_{i=1}^{\beta} c_i \geq d$, determined by the condition

$$\sum_{i=1}^{\beta-1} c_i < d \leq \sum_{i=1}^{\beta} c_i \ . \tag{10}$$

The extended clauses $L \geq \beta + l$ are not dominated by $cL \geq d$ for all $l \geq 1$. Hence (9) is the strongest extended clause w.r.t. $cL \geq d$ with left-hand side L.

PROOF:

[**a**] We show first that $cL \geq d$ dominates $L \geq \beta$. Because $cL \geq d$ is in pseudo-Boolean normal form, we know that $c_1 \geq \cdots \geq c_n$. Thus for all $1 \leq k \leq n$ we have $\sum_{i=1}^{k} c_i \geq \sum_{i \in K} c_i$ for all $K \subseteq \{1, \ldots, n\}$ and $|K| = k$. Let k be $\beta - 1$. Since $\sum_{i=1}^{\beta-1} c_i < d$ we know that $\sum_{i \in K} c_i < d$ for all $K \subseteq \{1, \ldots, n\}$ and $|K| = \beta - 1$. Consequently for all satisfying assignments $\alpha \in \mathsf{Ext}(cL \geq d)$ we have $\alpha(L) > \beta - 1$ and therefore $\alpha(L) \geq \beta$.

[**b**] We next show that $cL \geq d$ does not dominate $L \geq \beta + l$ for all $l \geq 1$. Let α be an assignment such that $\alpha(L_1) = \ldots = \alpha(L_\beta) = 1$ and $\alpha(L_{\beta+1}) = \ldots = \alpha(L_n) = 0$. Then $\alpha \in \mathsf{Ext}(cL \geq d)$ because $\sum_{i=1}^{\beta} c_i \geq d$, but $\alpha \notin \mathsf{Ext}(L \geq \beta + l)$ for all $l \geq 1$ because $\alpha(L) = \beta$. □

For a pseudo-Boolean inequality $cL \geq d$ we denote by $\mathsf{SCP}(cL \geq d)$ the strongest extended clause $L \geq \beta$ of Prop. 3.2. For a set of pseudo-Boolean inequalities S we denote by $\mathsf{SCP}(S)$ the set of all extended clauses $\mathsf{SCP}(cL \geq d)$ with $cL \geq d$ in S. From Proposition 3.2 we know that the degree of $\mathsf{SCP}(cL \geq d)$ is greater than or equal to the degree of $\mathsf{CP}(cL \geq d)$; therefore $\mathsf{SCP}(cL \geq d)$ dominates $\mathsf{CP}(cL \geq d)$. Note that $\mathsf{SCP}(cL \geq d)$ is also a cutting plane inequality, i.e. it can be obtained as a linear combination from $cL \geq d$ and the bounds $L_1 \geq 0$ and $-L_1 \geq -1$, followed by integer rounding. It is not necessary to construct the strongest extended clause using linear combination and integer rounding, because Prop. 3.2 guarantees that $\mathsf{SCP}(cL \geq d)$ is already the strongest extended clause.

Example 3.3 *Let $cL \geq d$ again be*

$$5 \cdot L_1 + 4 \cdot L_2 + 3 \cdot L_3 + 2 \cdot L_4 + 1 \cdot L_5 \geq 11$$

as in Example 3.1, then

$$\mathsf{CP}(cL \geq d) = \mathsf{SCP}(cL \geq d) = L_1 + L_2 + L_3 + L_4 + L_5 \geq 3 \ .$$

If $cL \geq d$ is

$$5 \cdot L_1 + 4 \cdot L_2 + 3 \cdot L_3 + 2 \cdot L_4 + 1 \cdot L_5 \geq 10$$

then

$$\mathsf{CP}(cL \geq d) \quad = \quad L_1 + L_2 + L_3 + L_4 + L_5 \geq 2$$

and

$$\mathsf{SCP}(cL \geq d) \quad = \quad L_1 + L_2 + L_3 + L_4 + L_5 \geq 3 \ .$$

Note that $\mathsf{SCP}(cL \geq d)$ strictly dominates $\mathsf{CP}(cL \geq d)$.

3.2 Strict Reductions

We know that all reductions of a pseudo-Boolean inequality $cL \geq d$ are valid w.r.t. $cL \geq d$. Hence for all strict reductions $c'L' \geq d'$ of $cL \geq d$ we know that $\mathsf{SCP}(c'L' \geq d')$ is a valid extended clause w.r.t. $cL \geq d$.

Example 3.4 *A strict reduction of*

$$5 \cdot L_1 + 4 \cdot L_2 + 3 \cdot L_3 + 2 \cdot L_4 + 1 \cdot L_5 \geq 11$$

is

$$5 \cdot L_1 + 4 \cdot L_2 + 3 \cdot L_3 + 2 \cdot L_4 \geq 10$$

obtained by eliminating L_5. Its strongest extended clause is

$$L_1 + L_2 + L_3 + L_4 \geq 3 \ .$$

We now show that $\mathsf{SCP}(\mathsf{SRed}(cL \geq d))$, the set of strongest extended clauses derived from all strict reductions of a pseudo-Boolean inequality $cL \geq d$, is equivalent to $cL \geq d$. For that we first show that a set of pseudo-Boolean inequalities dominates the set of all strongest extended clauses derived from the pseudo-Boolean inequalities in that set.

Lemma 3.5 *Let S be a set of pseudo-Boolean inequalities, then*

$$\mathsf{Ext}(S) \subseteq \mathsf{Ext}(\mathsf{SCP}(S)) \ .$$

PROOF: The strongest extended clause of a pseudo-Boolean inequality $cL \geq d$ is valid w.r.t. $cL \geq d$. Therefore we have $\mathsf{Ext}(cL \geq d) \subseteq \mathsf{Ext}(\mathsf{SCP}(cL \geq d))$. Because the intersection of sets T_i is a subset of the intersection of sets T_i' if $T_i \subseteq T_i'$, the lemma is established. □

Theorem 3.6 *Let $cL \geq d$ be an arbitrary linear pseudo-Boolean inequality; then*

$$\mathsf{Ext}(cL \geq d) = \mathsf{Ext}(\mathsf{SCP}(\mathsf{SRed}(cL \geq d))) \ . \tag{11}$$

PROOF:

[a] $\mathsf{Ext}(cL \geq d) \subseteq \mathsf{Ext}(\mathsf{SCP}(\mathsf{SRed}(cL \geq d)))$:
For all strict reductions $c'L' \geq d'$ of $cL \geq d$ we know that $\mathsf{Ext}(cL \geq d) \subseteq \mathsf{Ext}(c'L' \geq d')$ because $cL \geq d$ dominates $c'L' \geq d'$. Hence

$$\mathsf{Ext}(cL \geq d) \subseteq \bigcap_{c'L' \geq d' \in \mathsf{SRed}(cL \geq d)} \mathsf{Ext}(c'L' \geq d') \quad = \quad \mathsf{Ext}(\mathsf{SRed}(cL \geq d)) \ .$$

By Lemma 3.5 we have $\mathsf{Ext}(\mathsf{SRed}(cL \geq d)) \subseteq \mathsf{Ext}(\mathsf{SCP}(\mathsf{SRed}(cL \geq d)))$.

[b] $\mathsf{Ext}(\mathsf{SCP}(\mathsf{SRed}(cL \geq d))) \subseteq \mathsf{Ext}(cL \geq d)$:
We show that if $\alpha \notin \mathsf{Ext}(cL \geq d)$, then $\alpha \notin \mathsf{Ext}(\mathsf{SCP}(\mathsf{SRed}(cL \geq d)))$ from which the theorem follows. Suppose that $\alpha \notin \mathsf{Ext}(cL \geq d)$. We show that in this case there is a strict reduction of $cL \geq d$ such that α is not a satisfying assignment of this reduction, and therefore $\alpha \notin \mathsf{Ext}(\mathsf{SCP}(\mathsf{SRed}(cL \geq d)))$. Since $\alpha \notin \mathsf{Ext}(cL \geq d)$, we know that $\alpha(cL) = d' < d$. Let Y be the set of literals L_i where $\alpha(L_i) = 1$ and $Z = L \setminus Y$, then $\sum_{i:L_i \in Y} c_i = d' < d$. Let $c'L' \geq d'$ be $\sum_{i:L_i \in Z} c_i L_i \geq d - d'$. Because $d' < d$, we know that $d - d' \geq 1$, and therefore $c'L' \geq d'$ is a strict reduction of $cL \geq d$, obtained by eliminating the literals that are mapped to zero by α. Let $\mathsf{SCP}(c'L' \geq d')$ be $L' \geq \beta$. We know that $\beta \geq 1$ because $d - d' \geq 1$. Because $\alpha(L') = 0$, we derive that $\alpha \notin \mathsf{Ext}(\mathsf{SCP}(c'L' \geq d'))$, and therefore $\alpha \notin \mathsf{Ext}(\mathsf{SCP}(\mathsf{SRed}(cL \geq d)))$, which proves the theorem. $\square$

Theorem 3.6 gives a simple procedure for transforming a linear pseudo-Boolean inequality into an equivalent set of extended clauses. Unfortunately a very large number of extended clauses may be produced. Obviously we preserve equivalence if we consider only extended clauses that are not dominated by others.

Example 3.7 *The set of all strongest extended clauses of all strict reductions of*

$$5 \cdot L_1 + 4 \cdot L_2 + 3 \cdot L_3 + 2 \cdot L_4 + 1 \cdot L_5 \geq 11 \tag{12}$$

is

$$\begin{array}{llll}
L_1 + L_2 + L_3 + L_4 + L_5 \geq 3 & L_1 + L_2 + L_3 + L_4 \geq 3 & L_1 + L_2 + L_3 + L_5 \geq 2 & \\
L_1 + L_2 + L_4 + L_5 \geq 2 & L_1 + L_3 + L_4 + L_5 \geq 2 & L_2 + L_3 + L_4 + L_5 \geq 2 & \\
L_1 + L_2 + L_3 \geq 2 & L_1 + L_2 + L_4 \geq 2 & L_1 + L_2 + L_5 \geq 2 & \\
L_1 + L_3 + L_4 \geq 2 & L_1 + L_3 + L_5 \geq 1 & L_1 + L_4 + L_5 \geq 1 & \\
L_2 + L_3 + L_4 \geq 2 & L_2 + L_3 + L_5 \geq 1 & L_2 + L_4 + L_5 \geq 1 & \\
L_3 + L_4 + L_5 \geq 1 & L_1 + L_2 \geq 1 & L_1 + L_3 \geq 1 & \\
L_1 + L_4 \geq 1 & L_1 + L_5 \geq 1 & L_2 + L_3 \geq 1 & \\
L_2 + L_4 \geq 1 & L_2 + L_5 \geq 1 & L_3 + L_4 \geq 1 & L_1 \geq 1 \ .
\end{array}$$

Note that there are 25 strict reductions and therefore 25 extended clauses. If we delete all extended clauses that are dominated by others, we obtain

$$L_1 + L_2 + L_3 + L_4 \geq 3 \qquad L_1 + L_2 + L_5 \geq 2 \qquad L_1 \geq 1 \ .$$

The set of these 3 extended clauses is equivalent to (12).

An interesting property is that the normal form of $\mathsf{SCP}(\mathsf{SRed}(cL \geq d))$ is *prime.*

Definition 3.8 A set of extended clauses S is called *prime* if for all extended clauses $L \geq d$ that are dominated by S there is an extended clause in S dominating $L \geq d$.

Proposition 3.9 *For all extended clauses $L' \geq d'$ dominated by $\mathsf{SCP}(\mathsf{SRed}(cL \geq d))$ there exists an extended clause in $\mathsf{SCP}(\mathsf{SRed}(cL \geq d))$ dominating $L' \geq d'$.*

> *PROOF:* Suppose that $L' \geq d'$ is dominated by $\mathsf{SCP}(\mathsf{SRed}(cL \geq d))$. Because $\mathsf{Ext}(cL \geq d) = \mathsf{Ext}(\mathsf{SCP}(\mathsf{SRed}(I)))$, we know that $cL \geq d$ dominates $L' \geq d'$. Obviously $cL \geq d$ dominates $L' \geq d'$ only if it dominates $L'' \geq d'$ where $L'' = L' \cap L$. We consider the strict reduction $c''L'' \geq d''$ of $cL \geq d$ and its corresponding strongest extended clause $L'' \geq \beta$. From Prop. 3.2 we know that $\beta \geq d'$, and therefore $L'' \geq \beta$ in $\mathsf{SCP}(SRed(cL \geq d))$ dominates $L' \geq d'$. $\square$

Hence, for a single pseudo-Boolean inequality $cL \geq d$ the normal form of $\mathsf{SCP}(\mathsf{SRed}(cL \geq d))$ is already the *solved form* that should be extracted by our constraint solver.

The challenge is to find the normal form of $\mathsf{SCP}(\mathsf{SRed}(cL \geq d))$ without completely constructing the dominated and therefore *redundant* extended clauses, and to minimize the number of domination checks. In the following we describe strong redundancy criteria identifying extended clauses that are dominated by other extended clauses.

4 Identifying Redundant Extended Clauses

The normal form of $\mathsf{SCP}(\mathsf{SRed}(cL \geq d))$, that is all non redundant extended clauses in $\mathsf{SCP}(\mathsf{SRed}(cL \geq d))$, is the set of extended clauses, equivalent to $cL \geq d$, that we want to build. A naive way is to completely generate $\mathsf{SCP}(\mathsf{SRed}(cL \geq d))$, and then delete the redundant extended clauses. Typically many of the generated extended clauses are redundant. We develop strong redundancy criteria that minimize the number of domination checks needed for deciding whether an extended clause is redundant, i.e. is dominated by another extended clause. We show how to decide redundancy of an extended clause in $\mathsf{SCP}(\mathsf{SRed}(cL \geq d))$ by checking domination against *two* other extended clauses. For this purpose we first arrange the elements of $\mathsf{SCP}(\mathsf{SRed}(cL \geq d))$ as a directed acyclic graph.

Let $T = T(cL \geq d)$ be a finite acyclic graph with root node where the nodes are labeled with the strict reductions of $cL \geq d$. The root of T is labeled by $cL \geq d$. The direct subgraphs of T are the graphs $T(c'L' \geq d')$ for all strict reductions $c'L' \geq d'$ of $cL \geq d$, where exactly one literal has been eliminated. We denote by $\mathcal{T}(T, j)$ the direct subgraph of T with the label $\sum_{1 \leq i \neq j \leq n} c_i L_i \geq d - c_j$, provided that $d - c_j \geq 1$. We abbreviate $\mathcal{T}(\mathcal{T}(\ldots(\mathcal{T}(T, j_1), j_2), \ldots), j_k)$ by $T(\{j_1, \ldots, j_k\})$. We say $j_1 \ldots j_k$ is a *path* to the graph $T(\{j_1, \ldots, j_k\})$. Note that every permutation of $j_1 \ldots j_k$ is a path to $T(\{j_1, \ldots, j_k\})$, and that every element of $\mathsf{SRed}(cL \geq d)$ is the label of exactly one node in $T(cL \geq d)$. We denote by T_j the strongest extended clause of the root label of $\mathcal{T}(T, j)$. The strongest extended clause of the root label of $T(\{j_1, \ldots, j_k\})$ is referenced by $T_{\{j_1, \ldots, j_k\}}$. Note that the extended clause $T_{\{j_1, \ldots, j_k\}}$ has $n - k$ literals, therefore the length of the path determines the number of literals of the label. A part of the graph view of $\mathsf{SCP}(\mathsf{SRed}(cL \geq d))$ is shown in

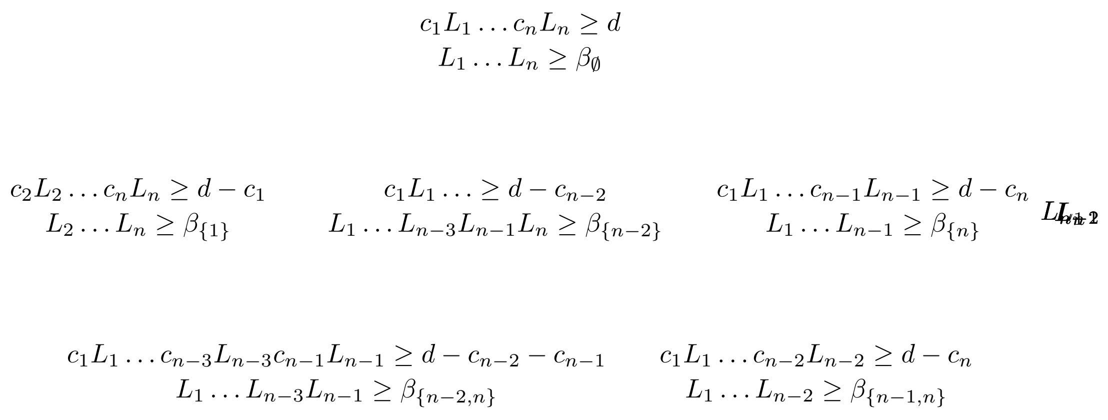
$c_1L_1 \dots c_nL_n \geq d$
$L_1 \dots L_n \geq \beta_\emptyset$
$c_2L_2 \dots c_nL_n \geq d - c_1$
$L_2 \dots L_n \geq \beta_{\{1\}}$
$c_1L_1 \dots \geq d - c_{n-2}$
$L_1 \dots L_{n-3}L_{n-1}L_n \geq \beta_{\{n-2\}}$
$c_1L_1 \dots c_{n-1}L_{n-1} \geq d - c_n$
$L_1 \dots L_{n-1} \geq \beta_{\{n\}}$
$c_1L_1 \dots c_{n-3}L_{n-3}c_{n-1}L_{n-1} \geq d - c_{n-2} - c_{n-1}$
$L_1 \dots L_{n-3}L_{n-1} \geq \beta_{\{n-2,n\}}$
$c_1L_1 \dots c_{n-2}L_{n-2} \geq d - c_n$
$L_1 \dots L_{n-2} \geq \beta_{\{n-1,n\}}$

Figure 1: Part of a graph view of $\mathsf{SCP}(\mathsf{SRed}(cL \geq d))$

Fig. 1. For each node we show the strict reduction and its corresponding strongest extended clause. Additionally the arcs are labeled by the literal L_i that we eliminated from father node to son node.

In the following we denote by $J_k := \{j_1, \dots, j_k\} \subset \{1, \dots, n\}$ a set of indices such that $T(J_k)$ is a valid strict reduction of $cL \geq d$. We denote by $\overline{J_k} := \{j'_1, \dots, j'_{n-k}\}$ the set of indices of the remaining literals in $T(J_k)$, that is $\overline{J_k} \cup J_k = \{1, \dots, n\}$ and $\overline{J_k} \cap J_k = \emptyset$. We then have

$$T(J_k) = \sum_{i=1}^{n-k} c_{j'_i} L_{j'_i} \geq d - \sum_{i=1}^{k} c_{j_i} \quad .$$

We abbreviate $\sum_{i=1}^{n-k} c_{j'_i} L_{j'_i}$ by $c(J_k)L(J_k)$ and $\sum_{i=1}^{n-k} L_{j'_i}$ by $L(J_k)$. We first investigate the degrees of the strongest extended clauses along a path.

Lemma 4.1 *For all J_k and $J_{k+1} = J_k \cup \{j_{k+1}\}$ we have*

$$\deg(T_{J_k}) \geq \deg(T_{J_{k+1}}) \geq \deg(T_{J_k}) - 1 \quad .$$

PROOF: Let T_{J_k} be $L_{j_{k+1}} + L(J_{k+1}) \geq \beta$ and $T_{J_{k+1}}$ be $L(J_{k+1}) \geq \beta'$. We need to show that $\beta \geq \beta' \geq \beta - 1$.

[a] $\beta \geq \beta'$:
We know that $L(J_{k+1}) \geq \beta'$ dominates $L_{j_{k+1}} + L(J_{k+1}) \geq \beta'$. Since $L_{j_{k+1}} + L(J_{k+1}) \geq \beta$ is a strongest extended clause, we have $\beta \geq \beta'$.

[b] $\beta' \geq \beta - 1$:
We know that $L_{j_{k+1}} + L(J_{k+1}) \geq \beta$ dominates its reduction $L(J_{k+1}) \geq \beta - 1$. Because $L(J_{k+1}) \geq \beta'$ is a strongest extended clause, we have $\beta' \geq \beta - 1$. □

Therefore the degree of the strongest extended clauses along a path from the root at each step either decreases by 1 or stays equal. We conclude that the following condition holds for all $J_l \subset J_k$.

$$|J_k| - |J_l| \geq \deg(T_{J_l}) - \deg(T_{J_k}) \qquad (13)$$

Lemma 4.2 *Let $T(J_k)$ be $c_{j_{k+1}}L_{j_{k+1}} + \sum_{i=1}^{n-(k+1)} c_{j'_i}L_{j'_i} \geq d' + c_{j_{k+1}}$. W.l.o.g. assume that $c_{j'_p} \geq c_{j'_q}$ for all $1 \leq p < q \leq n-(k+1)$. Let $\deg(T_{J_k})$ be β and $\deg(T_{J_{k+1}})$ be β'. If $c_{j_{k+1}} \geq c_{j'_{\beta'}}$ then $\beta' + 1 = \beta$.*

PROOF: The strict reduction $T(J_{k+1})$ of $T(J_k)$ is $\sum_{i=1}^{n-(k+1)} c_{j'_i}L_{j'_i} \geq d'$. From (10) in Prop. 3.2 we know that

$$\sum_{i=1}^{\beta'-1} c_{j'_i} < d' \leq \sum_{i=1}^{\beta'} c_{j'_i} \ .$$

By adding $c_{j_{k+1}}$ to the left inequality, we derive

$$c_{j_{k+1}} + \sum_{i=1}^{\beta'-1} c_{j'_i} < d' + c_{j_{k+1}} \ . \tag{14}$$

According to Prop. 3.2, β is the smallest number of the largest coefficients such that the sum of the largest coefficients is greater than the right-hand side $d' + c_{j_{k+1}}$. From $c_{j_{k+1}} \geq c_{j'_{\beta'}}$ and (14) we know that $c_{j_{k+1}}$ is one of the coefficients of this sum, and therefore $\beta > \beta'$. By Lemma 4.1 we have $\beta \leq \beta' + 1$, and therefore $\beta' + 1 = \beta$. □

Example 4.3 *Let the strict reduction $T(J_{k+1})$ be*

$$6 \cdot L_1 + 5 \cdot L_2 + 4 \cdot L_3 + 3 \cdot L_4 \geq 12 \ .$$

Its strongest extended clause is

$$L_1 + L_2 + L_3 + L_4 \geq 3$$

with degree 3, because $6 + 5 < 12 \leq 6 + 5 + 4$. Let c_iL_i be the product that has been eliminated, that is the label of the father node is

$$c_i \cdot L_i + 6 \cdot L_1 + 5 \cdot L_2 + 4 \cdot L_3 + 3 \cdot L_4 \geq 12 + c_i$$

and its strongest extended clause is

$$L_i + L_1 + L_2 + L_3 + L_4 \geq \beta.$$

By Lemma 4.2 we know that if $c_i \geq 4$ then $\beta = 4$. Suppose that $c_i = 4$; then condition (10) becomes $6+5+4 < 16 \leq 6+5+4+4$ and as expected the degree β is 4. Note that if $c_i = 3$ then β is still 4, that is the converse in Lemma 4.2 does not necessarily hold.

We now give a complete redundancy criterion for $\mathsf{SCP}(cL \geq d) = T_\emptyset$, which is also a sufficient redundancy criterion for all strongest extended clauses.

Lemma 4.4 *$T_\emptyset$ is redundant if and only if $\deg(T_\emptyset) = \deg(T_{\{n\}})$.*

PROOF: Let $T_\emptyset$ be $L \geq \beta$, let L' be $L \setminus \{L_n\}$ and let $T_{\{n\}}$ be $L' \geq \beta'$. Note that $c_n \leq c_i$ for all $1 \leq i \leq n$.

$\Leftarrow$: Since $\beta = \beta'$, we know that $L' \geq \beta$ dominates $L \geq \beta$ because $L' \subset L$. Hence $L \geq \beta$ is redundant.

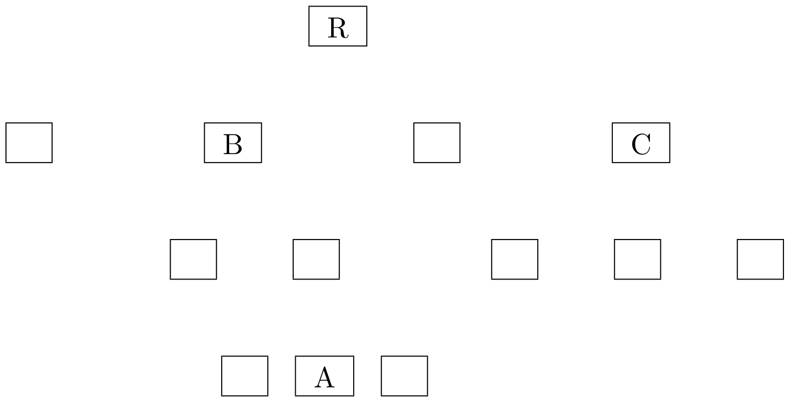

Figure 2: Redundancy caused by an extended clause in the subgraph

$\Rightarrow$: Suppose that $L \geq \beta$ is redundant. Then there exists $L'' \geq \beta''$ in $\mathsf{SCP}(\mathsf{SRed}(cL \geq d))$ that dominates $L \geq \beta$, and thus $\beta'' \geq \beta$. Since $L'' \geq \beta''$ is a strongest extended clause of a strict reduction of $cL \geq d$, we have by Lemma 4.1 that $\beta \geq \beta''$ and therefore $\beta = \beta''$. Because the degree of the strongest extended clauses along a path always decreases by 1 or stays equal, see (13), and since $L'' \subset L$, we know that all strongest extended clauses along the path from the root to the node with strongest extended clause $L'' \geq \beta$ have the degree β. Especially for a direct son of the root we have a strongest extended clause $L \setminus \{L_j\} \geq \beta$ dominating $L \geq \beta$. From Lemma 4.2 we know that the coefficient c_j of the eliminated literal L_j is smaller than $c_{\beta-1}$, because otherwise $\beta'' + 1 = \beta$. From $c_j < c_{\beta-1}$ we derive $j \geq \beta - 1$, since $cL \geq d$ is in pseudo-Boolean normal form and therefore $\sum_{i=1}^{\beta-1} c_i < d - c_j$. Because $c_j \geq c_n$, we have $d - c_j \leq d - c_n$ and therefore $\sum_{i=1}^{\beta-1} c_i < d - c_n$. But then $\beta' \geq \beta$, and because β' is the degree of a strongest extended clause on a path from the root, we have $\beta' \leq \beta$ and therefore $\beta' = \beta$. □

Proposition 4.5 *If T_{J_k} is dominated by T_{J_l} and $J_l \supset J_k$, then $T_{J_{k+1}} := T_{J_k \cup \{j_{k+1}\}}$ dominates T_{J_k}, where $c_{j_{k+1}} \leq c_i$ for all $i \in \overline{J_k}$.*

PROOF: The proposition follows immediately from Lemma 4.4 if we view $T(J_k)$ as an independent graph. □

With Prop. 4.5 we can decide whether T_{J_k} is dominated by an extended clause in a subgraph of the graph with root label $T(J_k)$ by checking domination against one specific extended clause. In Fig. 2 Lemma 4.4 is illustrated. The graph corresponds to a part of the graph view of $\mathsf{SCP}(\mathsf{SRed}(cL \geq d))$. Assume that the sons of each node are ordered such that the coefficient of the eliminated literal of a son node is greater than or equal to all the coefficients of the eliminated literals of its right neighbors. Now Lemma 4.4 says that there is an extended clause A under the extended clause R dominating R if and only if C dominates R. In the proof of Lemma 4.4 we show that if there is an A dominating R, then B is dominating R, from which we show that C is dominating R. The key idea is to show that the strongest extended clauses A,B, and C all have the same degree.

We now give a similar criterion for extended clauses lying on a path to $T(J_k)$, and, as in Prop. 4.5, identify a single specific extended clause that needs to be considered.

Lemma 4.6 *For $J_l \subset J_k$ the extended clause T_{J_k} is dominated by T_{J_l} if and only if*

$$|J_k| - |J_l| \leq \deg(T_{J_l}) - \deg(T_{J_k}) \ .$$

PROOF: Because $|\{1,\dots,n\} \setminus J_l| - |\{1,\dots,n\} \setminus J_k| = |J_k| - |J_l|$ the lemma follows immediately from Lemma 2.5. □

Lemma 4.7 *If $T_{J_{k+1}}$ is dominated by T_{J_l} where $J_l \subset J_{k+1}$, then $T_{J_{k+1} \setminus \{j_i\}}$ dominates $T_{J_{k+1}}$ for all $j_i \in J_{k+1} \setminus J_l$.*

PROOF: From (13) we have $|J_{k+1}| - |J_l| \geq \deg(T_{J_l}) - \deg(T_{J_{k+1}})$ and from Lemma 4.6 we have $|J_{k+1}| - |J_l| \leq \deg(T_{J_l}) - \deg(T_{J_{k+1}})$, therefore we know that $|J_{k+1}| - |J_l| = \deg(T_{J_l}) - \deg(T_{J_{k+1}})$. Since in this case the degree of all extended clauses on a path from T_{J_l} to $T_{J_{k+1}}$ decreases always by one, we have that the degree of all direct fathers, $T_{J_{k+1} \setminus \{j_i\}}$ for all $j_i \in J_{k+1} \setminus J_l$, is the degree of $T_{J_{k+1}}$ plus one. □

We identify similar to Proposition 4.5 a single specific extended clause that needs to be tested for application of Lemma 4.6.

Proposition 4.8 *If $T_{J_{k+1}}$ is dominated by T_{J_l} where $J_l \subset J_{k+1}$, then there exists $J_{k+1} \setminus \{j_{\max}\}$ such that $T_{J_{k+1}} \setminus \{j_{\max}\}$ dominates $T_{J_{k+1}}$ and $c_{j_{\max}} \geq c_{j_i}$ for all $1 \leq i \leq k+1$.*

PROOF: Let us consider a specific path $H_{k+1} := h_1 \dots h_{k+1}$ from the root to $T(J_{k+1})$ where $c_{h_1} \leq \dots \leq c_{h_{k+1}} = c_{j_{\max}}$ and $h_{k+1} = j_{\max}$, i.e. always the smallest coefficient becomes eliminated next. For all direct fathers of $T(J_{k+1})$ the coefficient of the literal that becomes eliminated is therefore less than or equal to $c_{h_{k+1}}$. Because of Lemma 4.7 we have a strict reduction

$$R := \sum_{i \in \overline{J_{k+1}}} c_i L_i + c_x L_x \geq d + c_x - \sum_{i \in J_{k+1}} c_i \ ,$$

where $x \in J_l$ and $\deg(\mathsf{SCP}(R)) = \beta + 1$. Using our fixed path H_{k+1}, we know that there is another strict reduction

$$H := \sum_{i \in \overline{J_{k+1}}} c_i L_i + c_{j_{\max}} L_{j_{\max}} \geq d + c_{j_{\max}} - \sum_{i \in J_{k+1}} c_i \ .$$

We now show that $\deg(\mathsf{SCP}(R)) = \deg(\mathsf{SCP}(H)) = \beta + 1$.

Suppose that $c_{j_{\max}} \geq c_{j'_\beta}$. As in Lemma 4.2 we then derive that $\deg(\mathsf{SCP}(H)) > \beta$, hence $\deg(\mathsf{SCP}(H)) = \beta + 1$.

Now suppose that $c_{j_{\max}} < c_{j'_\beta}$. Since $c_{j_{\max}} \geq c_x$, we have $c_x < c_{j'_\beta}$. Because $\deg(\mathsf{SCP}(R)) = \beta + 1$, we have

$$\sum_{1 \leq i \leq \beta} c_{j'_i} < d + c_x - \sum_{i \in J_{k+1}} c_i \ .$$

With $c_{j_{\max}} \geq c_x$ we get

$$\sum_{1 \leq i \leq \beta} c_{j'_i} < d + c_{j_{\max}} - \sum_{i \in J_{k+1}} c_i \ ,$$

which also implies $\deg(\mathsf{SCP}(H)) > \beta$ and therefore $\deg(\mathsf{SCP}(H)) = \beta + 1$. Let $J_{k+1} \setminus \{j_{\max}\}$ be $\{h_1, \dots, h_k\}$, then the proposition follows. □

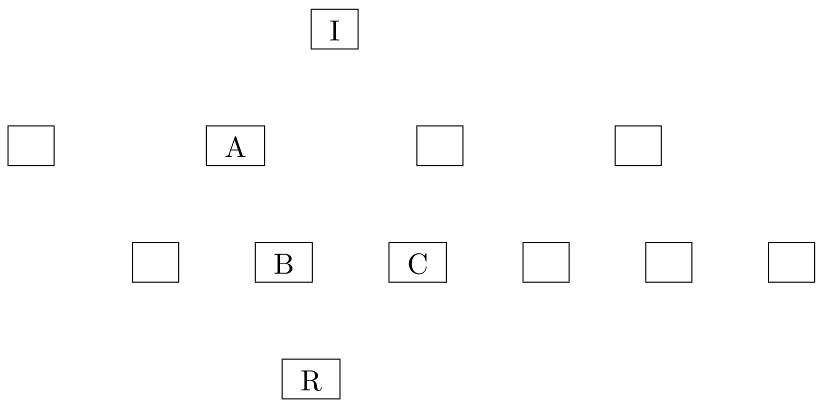

Figure 3: Redundancy caused by an extended clause above in the graph

Consider the illustration of Prop. 4.8 in Fig. 3. Prop. 4.8 says that there is an A dominating R if and only if C dominates R. In the graph we assume that the fathers of each node are ordered such that the coefficient of the eliminated literal to come from a father node to a son node is greater than or equal to all the coefficients of the eliminated literals of its *left* neighbors. In the proof of Prop. 4.8 we show first that if A dominates R, then there is a B dominating R, and that all degrees along a path from R to A decrease by one. The double line corresponds to the specific path H_{k+1}, and we show that C, lying on this path, has the same degree as B and therefore dominates R.

We combine now Prop. 4.5 and Prop. 4.8 to a complete redundancy criterion.

Theorem 4.9 *The strongest extended clause $T_{J_{k+1}}$ is redundant if and only if*

[a] *$T_{J_{k+2}}$ dominates $T_{J_{k+1}}$ where $c_{j_{k+2}} \leq c_i$ for all $i \in \overline{J_{k+1}}$,* *or*

[b] *$T_{J_{k+1} \setminus \{j_{\max}\}}$ dominates $T_{J_{k+1}}$ where $c_{j_{\max}} \geq c_i$ for all $i \in J_{k+1}$.*

PROOF:

$\Leftarrow$: **[a]** from Prop. 4.5 and **[b]** from Prop. 4.8.

$\Rightarrow$: Suppose that neither **[a]** nor **[b]** is satisfied for $T_{J_{k+1}}$, but $T_{J_{k+1}}$ is redundant. Then there is a strongest extended clause $L'' \geq \beta$ that dominates $T_{J_{k+1}} := L' \geq \beta'$ and $L'' \neq L'$. From Prop. 4.8 we have $L'' \not\supset L'$ and from Prop. 4.5 we have $L'' \not\subset L'$. Therefore there is at least one literal in L'' that is not in L'. Following the proof of Lemma 2.5 a valid reduction of $L'' \geq \beta$ is $L'' \cap L' \geq \beta - |L'' \setminus L'|$, which also dominates $L' \geq \beta'$. The strongest extended clause containing exactly the literals in $L'' \cap L'$ therefore dominates $L \geq \beta'$ and hence $T_{J_{k+1}}$. Because $L'' \cap L' \subset L''$ $(L'' \setminus L' \neq \emptyset)$ Prop. 4.5 applies and therefore condition **[a]**. $\square$

For illustrating statement **[b]** of Theorem 4.9 we look again at Fig. 2. We have shown that if there is an X dominating R and X is not on a path also including R then we will find and A in the subgraph under X such that Prop. 4.5 applies.

By Theorem 4.9 we can decide redundancy of a strongest extended clause by investigating exactly 2 other strongest extended clauses. We end this section with an example.

Example 4.10 *Let us transform the pseudo-Boolean inequality*

$$5 \cdot L_1 + 4 \cdot L_2 + 3 \cdot L_3 + 2 \cdot L_4 + 1 \cdot L_5 \geq 11 \ .$$

Its strongest extended clause is

$$L_1 + L_2 + L_3 + L_4 + L_5 \geq 3 \ .$$

Let us compute the strict reductions $T(\{i\})$ and their corresponding strongest extended clauses $T_{\{i\}}$.

i	$T(\{i\})$	$T_{\{i\}}$
1	$4 \cdot L_2 + 3 \cdot L_3 + 2 \cdot L_4 + 1 \cdot L_5 \geq 6$	$L_2 + L_3 + L_4 + L_5 \geq 2$
2	$5 \cdot L_1 + 3 \cdot L_3 + 2 \cdot L_4 + 1 \cdot L_5 \geq 7$	$L_1 + L_3 + L_4 + L_5 \geq 2$
3	$5 \cdot L_1 + 4 \cdot L_2 + 2 \cdot L_4 + 1 \cdot L_5 \geq 8$	$L_1 + L_2 + L_4 + L_5 \geq 2$
4	$5 \cdot L_1 + 4 \cdot L_2 + 3 \cdot L_3 + 1 \cdot L_5 \geq 9$	$L_1 + L_2 + L_3 + L_5 \geq 2$
5	$5 \cdot L_1 + 4 \cdot L_2 + 3 \cdot L_3 + 2 \cdot L_4 \geq 10$	$L_1 + L_2 + L_3 + L_4 \geq 3$

We see that $T_{\{5\}}$ dominates $T_\emptyset$ (Theorem 4.9[a] $\Leftarrow$) and that $T_{\{1\}}, T_{\{2\}}, T_{\{3\}}, T_{\{4\}}$ are dominated by $T_\emptyset$ (Theorem 4.9[b] $\Leftarrow$). We calculate $T_{\{4,5\}} = L_1 + L_2 + L_3 \geq 2$ and conclude that $T_{\{5\}}$ is not redundant(Theorem 4.9; completeness). Proceeding further we arrive at the set of non redundant extended clauses as in Example 3.7.

5 Implementation

We sketch an implementation of the transformation algorithm using the strong redundancy criteria presented in Sect. 4.

Theorem 4.9 is a complete redundancy criterion for a strongest extended clause that reduces the number of needed domination checks to 2. We denote by $\beta' = \deg(T_{J_{k+2}})$ the degree of the strongest extended clause of the specific father of $T(J_{k+1})$ and with $\beta'' = \deg(T_{J_k \setminus \{j_{\max}\}})$ the degree of the strongest extended clause of the specific son of $T(J_{k+1})$. The degree of the extended clause for which we want to check redundancy is denoted by $\beta = \deg(T_{J_{k+1}})$. From Theorem 4.9 we see immediately that $T_{J_{k+1}}$ is redundant iff

$$\beta' - 1 = \beta \qquad \text{or} \qquad \beta = \beta'' \ .$$

Hence, for deciding redundancy of a strongest extended clause only the degrees of two strongest extended clauses (the specific father and the specific son) are needed.

We describe an algorithm that computes from a given linear pseudo-Boolean inequality $cL \geq d$ in normal form a set of equivalent non redundant extended clauses. We assume that $\sum_{i=1}^{n} c_i \geq d$, since otherwise $cL \geq d$ is unsatisfiable. We also assume that $\sum_{i=2}^{n} c_i \geq d$, since otherwise $\alpha(L_1) = 1$ for all $\alpha \in \mathsf{Ext}(cL \geq d)$, and we can consider the simpler problem of transforming

$$\sum_{i=2}^{n} c_i L_i \geq d - c_1 \ .$$

In the following we assume that $cL \geq d$ is satisfiable, in pseudo-Boolean normal form and that for every literal $L_i \in L$ there are assignments $\alpha_1, \alpha_2 \in \mathsf{Ext}(cL \geq d)$ such that $\alpha_1(L_i) = 1 - \alpha_2(L_i)$. Note that for all generated strongest extended clauses $L' \geq \beta$ we have $|L'| > \beta$.

In the program fragments we use the following nonstandard notation. Given a pseudo-Boolean term cL we select the last (resp. first) coefficient/literal pair $c_l L_l$ and the remaining term $c'L'$ by $cL = c'L' + c_l L_l$ (resp. $cL = c_l L_l + c'L'$). We give first the procedure `get_beta` calculating the degree β of the strongest extended clause of its argument, a strict reduction, and β'', the degree of the strongest extended clause of the strict reduction, where we have eliminated the literal with the smallest coefficient. The right-hand side β'' then is the degree of the strongest extended clause of the specific son node as in Prop. 4.8.

```
get_beta($cL \geq d$)
        $i, \beta, sum, prevsum := 1, 0, 0, 0$
        **while** $sum < d$
                $cL = c_l L_l + c'L'$
                $prevsum := sum$
                $sum, \beta, cL := sum + c_l, \beta + 1, c'L'$
        **endwhile**
        **while** $cL = c_l L_l + c'L'$
                $cL := c'L'$
        **endwhile**
        (* $c_l$ is the smallest coefficient *)
        $\beta'' :=$ **if** $prevsum \geq d - c_l$ **then** $\beta - 1$ **else** $\beta$ **endif**
        **return** $(\beta, \beta'')$
**end** get_beta
```

We obtain β by summing up the coefficients according to (9), and count the number of added coefficients until we reach the right-hand side. Since the specific son is the one where the literal with the smallest coefficient has been eliminated, this coefficient can not be part of the sum. Note that $\sum_{i=2}^{n} c_i \geq d$. Therefore the degree of the specific son is one smaller if and only if the actual sum minus the last added coefficient, that is the previous sum, is already greater than or equal to the right-hand side $d - c_l$ of the specific son. We see that the overhead for calculating β'' is not high. In the actual implementation we store the value of the smallest coefficient for each strict reduction and so avoid searching for it.

For an efficient implementation we must not generate the whole graph. Instead we implicitly visit the nodes, that is the strict reductions, in a specific order such that for each strict reduction the redundancy test becomes trivial. We assure that each strict reduction $T(J_{k+1})$ is visited only once through the path H_{k+1} (see Proof of Prop. 4.8). This is achieved by splitting the pseudo-Boolean term (the left-hand side of the pseudo-Boolean inequality) into two parts, one containing the literals that must be eliminated and one part containing the literals that need not be eliminated. We eliminate literals having a smaller coefficient first and then forbid further reduction on these literals by moving them from the first part to the second part. We thus make sure that the strict reductions are visited only through the specific path. We give a recursive procedure `transform` generating each possible strict reduction only once and adding its strongest extended clause to the output set if and only it is not redundant. The procedure `transform` has as parameters a pseudo-Boolean inequality, which is split into the two parts and the right-hand side, and the right-hand side β' of the strongest extended clause of the father in the path H_{k+1}.

```
transform($cL, \hat{c}\hat{L}, d, \beta'$)
    $S := \emptyset$
    $(\beta, \beta'') =$ get_beta($cL + \hat{c}\hat{L} \geq d$)
    **if** $\beta' - 1 \neq \beta \wedge \beta'' \neq \beta$ **then** (* not redundant *)
        $S := S \cup \{L + \hat{L} \geq \beta\}$ (* add the non redundant clause to the output set *)
    **endif**
    (* As long as $cL$ contains elements and there are valid strict reductions *)
    **while** $c'L' + c_l L_l = cL \wedge d - c_l \geq 1$ (* eliminate $c_l L_l$ *)
        $S := S \cup$ transform($c'L', \hat{c}\hat{L}, d - c_l, \beta$)
        $\hat{c}\hat{L} := \hat{c}\hat{L} + c_l L_l$ (* forbid further reduction on $L_l$ *)
        $cL := c'L'$
    **endwhile**
    **return** $S$
**end** transform
```

Note that in the **while**-loop we select first the product with the smallest coefficient and then forbid further reduction on it by moving the product to the second part. The set of all non redundant strongest extended clauses equivalent to the pseudo-Boolean inequality $cL \geq d$ then is `transform`$(cL, \emptyset, d, 0)$. We finish this section with some examples.

Example 5.1 *The examples are taken from [NW88].*

[a] *[NW88, page 266] Let $cL \geq d$ be*

$$79 \cdot X_1 + 53 \cdot X_2 + 53 \cdot X_3 + 45 \cdot X_4 + 45 \cdot X_5 \leq 178 \ .$$

The pseudo-Boolean normal form of $cL \geq d$ is

$$79 \cdot \overline{X_1} + 53 \cdot \overline{X_2} + 53 \cdot \overline{X_3} + 45 \cdot \overline{X_4} + 45 \cdot \overline{X_5} \geq 97 \ .$$

The equivalent set of extended clauses is

$$\begin{array}{rcl} \{\overline{X_1} + \overline{X_2} + \overline{X_3} + \overline{X_4} + \overline{X_5} & \geq & 2, \\ \overline{X_1} + \overline{X_2} + \overline{X_3} & \geq & 1\} \ . \end{array}$$

[b] *[NW88, page 460] Let $cL \geq d$ be*

$$774 \cdot X_1 + 76 \cdot X_2 + 22 \cdot X_3 + 42 \cdot X_4 + 21 \cdot X_5 + 760 \cdot X_6 + 818 \cdot X_7 + 62 \cdot X_8 + 785 \cdot X_9 \leq 1500 \ .$$

The transformation procedure generates the single extended clause

$$\overline{X_1} + \overline{X_6} + \overline{X_7} + \overline{X_9} \geq 3$$

which is equivalent to $cL \geq d$.

[c] *[NW88, page465] Let $cL \geq d$ be*

$$\begin{array}{lr} 300 \cdot X_3 + 300 \cdot X_4 + 285 \cdot X_5 + 285 \cdot X_6 + 265 \cdot X_8 + 265 \cdot X_9 + 230 \cdot X_{12} + & \\ 230 \cdot X_{13} + 190 \cdot X_{14} + 200 \cdot X_{22} + 400 \cdot X_{23} + 200 \cdot X_{24} + 400 \cdot X_{25} + & \\ 200 \cdot X_{26} + 400 \cdot X_{27} + 200 \cdot X_{28} + 400 \cdot X_{29} + 200 \cdot X_{30} + 400 \cdot X_{31} & \leq \ 2700 \end{array}$$

which is a linear 0-1 inequality from the constraint set of a 0-1 integer programming problem. Transforming $cL \geq d$ generates 4282 non redundant strongest extended clauses. They have been calculated with a PROLOG-implementation of the algorithm in 4.8 seconds cpu-time on a SPARC-10/31. All 15 inequalities of the problem produce together 8710 extended clauses where 749 extended clauses are dominated by extended clauses generated from another inequality such that 7961 extended clauses remain. Note that the number of non redundant classical clauses that are equivalent to $cL \geq d$, as needed in [GH71], is 117520.

Example 5.1 **[c]** demonstrates the advantage of extended clauses versus classical clauses. Because of the large number of classical clauses needed to represent a pseudo-Boolean inequality, clausal satisfiability methods can not be applied to typical 0-1 integer programming problems [GG80]. The reformulation of a pseudo-Boolean inequality as a set of extended clauses is more likely to be applicable on general 0-1 problems because of its compact representation. In the next section we give an even more compact representation of pseudo-Boolean inequalities and extended clauses that allows to speed up the computation, and brings even larger problems into the scope of this symbolic method.

6 Symmetries

We introduce the concept of symmetries for a set of literals w.r.t. to a pseudo-Boolean inequality and then define the *compact set* representation of pseudo-Boolean inequalities and extended clauses. The more compact formulation helps to avoid *redundant computation* in the transformation method.

Let us transform the pseudo-Boolean inequality

$$5 \cdot A + 5 \cdot B + 4 \cdot C + 3 \cdot D + 3 \cdot E + 3 \cdot F \geq 12 \ . \tag{15}$$

Three strict reductions of (15) are for example

$$5 \cdot A + 5 \cdot B + 4 \cdot C + 3 \cdot E + 3 \cdot F \geq 9 \tag{16}$$
$$5 \cdot A + 5 \cdot B + 4 \cdot C + 3 \cdot D + 3 \cdot F \geq 9 \tag{17}$$
$$5 \cdot A + 5 \cdot B + 4 \cdot C + 3 \cdot D + 3 \cdot E \geq 9 \tag{18}$$

obtained by eliminating D for (16), E for (17) and F for (18). Note that these three strict reductions are identical except for the names of the last two literals. Therefore these three extended clauses yield exactly the same set of non redundant extended clauses except for the names of the literals. We say that D, E and F are *symmetric* in (15). In the following we describe how to avoid redundant computations occurring while eliminating symmetric literals. For that we first define a compact representation of a set of pseudo-Boolean inequalities.

Definition 6.1 For all $1 \leq i < j \leq m$ let S_i and S_j be sets of literals such that $\mathsf{Var}(L_l) \neq \mathsf{Var}(L_k)$ for all $L_l \in S_i$ and $L_k \in S_j$. We say that

$$\sum_{i=1}^{m} c_i S_i^{k_i} \geq d \ , \tag{19}$$

where $d \geq c_1 > \cdots > c_m \geq 1$ and $1 \leq k_i \leq |S_i|$ for all $1 \leq i \leq m$, is a *compact pseudo-Boolean inequality set* of a set of pseudo-Boolean inequalities. The compact pseudo-Boolean inequality set (19) represents the set of all pseudo-Boolean inequalities

$$\left\{ cL \geq d \;\middle|\; \begin{array}{l} |L| = \sum_{i=1}^{m} k_i \text{ and} \\ \text{for all } c_l L_l \in cL \text{ there is an } S_i \text{ such that } L_l \in S_i \text{ and } c_l = c_i \text{ and} \\ |S_i \cap L| = k_i \text{ for all } 1 \leq i \leq m \ . \end{array} \right\} \tag{20}$$

In other words, all pseudo-Boolean inequalities where from each set of literals S_i exactly k_i occur in the pseudo-Boolean inequality with coefficient c_i. We say we *expand* a compact pseudo-Boolean inequality set if we replace $\sum_{i=1}^{m} c_i S_i^{k_i} \geq d$ by (20) and define $\mathsf{expand}(\sum_{i=1}^{m} c_i S_i^{k_i} \geq d) := (20)$. For a set S of compact pseudo-Boolean inequality sets we define $\mathsf{expand}(S) := \bigcup_{I \in S} \mathsf{expand}(I)$.

Example 6.2 *We can represent the single pseudo-Boolean inequality (15) by the compact pseudo-Boolean inequality set*

$$5 \cdot \{A, B\}^2 + 4 \cdot \{C\}^1 + 3 \cdot \{D, E, F\}^3 \geq 12 \ .$$

The three strict reductions (16), (17) and (18) can be represented by

$$5 \cdot \{A, B\}^2 + 4 \cdot \{C\}^1 + 3 \cdot \{D, E, F\}^2 \geq 9 \ .$$

Note that every pseudo-Boolean inequality can be represented as compact pseudo-Boolean inequality set where $k_i = |S_i|$.

The compact pseudo-Boolean inequality set representation is the key to exploit the symmetry property of literals in a pseudo-Boolean inequality. The possible strict reductions that lead to an identical search can now be represented by a single compact pseudo-Boolean inequality set. We define similar to Def. 6.1 a compact set representation for extended clauses.

Definition 6.3 For all $1 \leq i < j \leq m$ let S_i and S_j be sets of literals such that $\mathsf{Var}(L_l) \neq \mathsf{Var}(L_k)$ for all $L_l \in S_i$ and $L_k \in S_j$. We say that

$$\sum_{i=1}^{m} S_i^{k_i} \geq \beta \ , \tag{21}$$

where $1 \leq k_i \leq |S_i|$ for all $1 \leq i \leq m$, is a *compact extended clause set* of a set of extended clauses. The compact extended clause set (21) represents the set of all extended clauses

$$\left\{ L \geq \beta \;\middle|\; \begin{array}{l} |L| = \sum_{i=1}^{m} k_i \text{ and} \\ |S_i \cap L| = k_i \text{ for all } 1 \leq i \leq m \ . \end{array} \right\} \tag{22}$$

We say we *expand* a compact extended clause set if we replace $\sum_{i=1}^{m} S_i^{k_i} \geq \beta$ by the set of extended clauses (22) and define $\mathsf{expand}(\sum_{i=1}^{m} S_i^{k_i} \geq \beta) := (22)$. For a set S of compact extended clause sets we define $\mathsf{expand}(S) := \bigcup_{I \in S} \mathsf{expand}(I)$.

We generalize now the generation of strongest extended clauses of Prop. 3.2. For a compact pseudo-Boolean inequality set $\sum_{i=1}^{m} c_i S_i^{k_i} \geq d$ the corresponding strongest compact extended clause set is

$$\sum_{i=1}^{m} S_i^{k_i} \geq \beta \tag{23}$$

where

$$\sum_{i=1}^{\beta-1} c_i \cdot k_i < d \leq \sum_{i=1}^{\beta} c_i \cdot k_i \ . \tag{24}$$

Note that (24) corresponds to (9) but takes into account the compact set representation. We denote by $\mathsf{SCP}'(\sum_{i=1}^{m} c_i S_i^{k_i} \geq d) := \sum_{i=1}^{m} S_i^{k_i} \geq \beta$ the strongest compact extended clause set of a compact pseudo-Boolean inequality set. Obviously for each pseudo-Boolean inequality in $\mathsf{expand}(\sum_{i=1}^{m} c_i S_i^{k_i} \geq d)$ there is a corresponding strongest extended clause in $\mathsf{expand}(\sum_{i=1}^{m} S_i^{k_i} \geq \beta)$ and we have

$$\mathsf{expand}(\mathsf{SCP}'(\sum_{i=1}^{m} c_i S_i^{k_i} \geq d)) = \mathsf{SCP}(\mathsf{expand}(\sum_{i=1}^{m} c_i S_i^{k_i} \geq d)) \ .$$

A compact strict reduction of a compact pseudo-Boolean inequality set $\sum_{i=1}^{m} c_i S_i^{k_i} \geq d$ is

$$\sum_{i=1}^{m} c_i S_i^{k_i'} \geq d - \sum_{i=1}^{m} c_i \cdot (k_i - k_i')$$

where either $k_i' = k_i$ or $k_i' = k_i - 1$ for all $1 \leq i \leq m$. Note that $c_i S_i^{k_i'}$ disappears if $k_i' = 0$. We denote by $\mathsf{SRed}'(\sum_{i=1}^{m} c_i S_i^{k_i} \geq d)$ the set of all compact strict reductions of a compact pseudo-Boolean inequality set. Obviously we have

$$\mathsf{expand}(\mathsf{SRed}'(\sum_{i=1}^{m} c_i S_i^{k_i} \geq d)) = \mathsf{SRed}(\mathsf{expand}(\sum_{i=1}^{m} c_i S_i^{k_i} \geq d)) \ .$$

The transformation method using compact sets is now straightforward. For a compact pseudo-Boolean inequality set $\sum_{i=1}^{m} c_i S_i^{k_i} \geq d$ we have

$$\mathsf{expand}(\mathsf{SCP}'(\mathsf{SRed}'(\sum_{i=1}^{m} c_i S_i^{k_i} \geq d))) = \mathsf{SCP}(\mathsf{SRed}(\mathsf{expand}(\sum_{i=1}^{m} c_i S_i^{k_i} \geq d))) \ .$$

The practical advantage is that we avoid redundant computation of identical strict reductions modulo literal renaming because they are grouped together in a single compact pseudo-Boolean inequality set. Generalizing Theorem 4.9 for compact sets is immediate and an implementation of the transformation method using compact sets is similar to the implementation sketched in Sect. 5.

Example 6.4 [a] *Let us transform the pseudo-Boolean inequality*

$$5 \cdot A + 5 \cdot B + 4 \cdot C + 3 \cdot D + 3 \cdot E + 3 \cdot F \geq 12$$

then 32 strict reductions need to be considered. If we transform the equivalent compact pseudo-Boolean inequality set

$$5 \cdot \{A, B\}^2 + 4 \cdot \{C\}^1 + 3 \cdot \{D, E, F\}^3 \geq 12$$

then only 11 compact strict reductions are possible. The set of equivalent strongest extended clause is

$$\begin{aligned}\{A+B+C+D+E+F &\geq 3,\\ A+B+C+D &\geq 2,\\ A+B+C+E &\geq 2,\\ A+B+C+F &\geq 2\}\ .\end{aligned}$$

The compact variant generates

$$\begin{aligned}\{\{A\}^1+\{B\}^1+\{C\}^1+\{D\}^1+\{E\}^1+\{F\}^1 &\geq 3,\\ \{A\}^1+\{B\}^1+\{C\}^1+\{D,E,F\}^1 &\geq 2\}\ .\end{aligned}$$

So by exploiting the symmetry property of literals we avoid redundant computation and obtain a more compact result.

[b] *The pseudo-Boolean inequality of Example 5.1[c] generates 4282 extended clauses in 4.8 seconds cpu time on a SPARC-10/31. The implementation using symmetries takes 0.77 seconds cpu time for generating the 4282 extended clause where 0.49 seconds are used to expand the 253 compact non redundant extended clause sets. The transformation itself takes only 0.18 seconds.*

The very compact representation of Example 6.4**[b]** suggests to further investigate the compact extended clause set representation of extended clauses. If we can find efficient symbolic solution methods working directly on the compact representation even large 0-1 problems are in the scope of these methods. This issue will be investigated in the future.

7 Detection of Fixed Literals

We present a method for checking whether a set of pseudo-Boolean inequalities S dominates an extended clause $L_i \geq 1$ for some literal L_i. For such an L_i we then have $\alpha(L_i) = 1$ for all assignments $\alpha \in \mathsf{Ext}(S)$, that is the literal L_i is fixed and S can be simplified to a set of pseudo-Boolean inequalities S' not containing L_i nor $\overline{L_i}$ such that

$$\mathsf{Ext}(S) = \mathsf{Ext}(S' \cup \{L_i \geq 1\})\ .$$

A similar simplification is possible if several literals need to be fixed. Let L be a set of literals such that S dominates $L_i \geq 1$ for all $L_i \in L$. We can then simplify S to a set of pseudo-Boolean inequalities S' not containing L_i nor $\overline{L_i}$ for all $L_i \in L$ such that

$$\mathsf{Ext}(S) = \mathsf{Ext}(S' \cup \{L \geq |L|\})\ .$$

Note that $L \geq |L|$ dominates $L_i \geq 1$ for all $L_i \in L$. We show how to find some of the fixed literals in L and how to build the simplified set of pseudo-Boolean inequalities S'.

We know that S dominates $L_i \geq 1$ if and only if $S \cup \{\overline{L_i} \geq 1\}$ is unsatisfiable. Obviously, checking whether $S \cup \{\overline{L_i} \geq 1\}$ is unsatisfiable is in general NP-complete. We present an approximation of the unsatisfiability test based on the idea of *unit relaxation* [Hoo88, Hoo89]. We first briefly recall *unit resolution* for classical clauses and then generalize the concept to pseudo-Boolean inequalities.

Classical clauses are pseudo-Boolean inequalities where all coefficients and the right-hand side are 1. A classical clause $L_i \geq 1$ is called *unit clause* and its literal L_i is called *unit literal.* Application of resolution restricted to the case that at least one father is a unit clause is called *unit resolution* or *clausal chaining.* The resolvent of a unit clause $L_i \geq 1$ and a classical clause $L \geq 1$ is defined by

$$\texttt{ures}(L_i, L \geq 1) := \begin{cases} L \setminus \{\overline{L_i}\} \geq 1 & \text{if} \quad \overline{L_i} \in L \\ \top & \text{if} \quad L_i \in L \\ L \geq 1 & \text{otherwise} \end{cases} .$$

Because $\{\texttt{ures}(L_i, L \geq 1), L_i \geq 1\}$ dominates $L \geq 1$ we can replace $L \geq 1$ by $\texttt{ures}(L_i, L \geq 1)$. For a set of classical clauses S we define

$$\texttt{ures}(L_i, S) := \{\texttt{ures}(L_i, L \geq 1) \mid L \geq 1 \in S \text{ and } \texttt{ures}(L_i, L \geq 1) \neq \top\} .$$

Note that there are no tautologies $\top$ in $\texttt{ures}(L_i, S)$. We derive that

$$\mathsf{Ext}(S) = \mathsf{Ext}(\texttt{ures}(L_i, S) \cup \{L_i \geq 1\})$$

if S dominates $L_i \geq 1$. Unit resolution for a set of classical clauses S can be described by the procedure $\texttt{ur}$.

```
ur($S$)
    $U := \emptyset$
    while $\exists L_i \geq 1 \in S$
        $S := \texttt{ures}(L_i, S)$
        $U := U \cup \{L_i\}$
    endwhile
    return $S$
end ur
```

We denote by $\texttt{ul}(S) := U$ the set of unit literals L_i of the unit clauses $L_i \geq 1$ detected after applying $\texttt{ur}(S)$. If $\texttt{ur}(S)$ contains the the empty clause $\Box$, then S is unsatisfiable and we say the *unit relaxation* of S is unsatisfiable. Note that the unit relaxation of a set of classical clauses S is satisfiable if and only if the linear programming relaxation of S is satisfiable [Hoo88]. We know that S is satisfiable if and only if $\texttt{ur}(S)$ is satisfiable. Each satisfying assignment α of S is a satisfying assignment of $\texttt{ur}(S)$ and $\alpha(L_i) = 1$ for all $L_i \in \texttt{ul}(S)$, therefore

$$\mathsf{Ext}(S) = \mathsf{Ext}(\texttt{ur}(S) \cup \{L_i \geq 1 \mid L_i \in \texttt{ul}(S)\}) .$$

For generalizing the concept of unit relaxation to arbitrary linear pseudo-Boolean inequalities we need to know whether a pseudo-Boolean inequality dominates $L_i \geq 1$ for some L_i, that is whether there is a literal L_i that can be fixed.

Lemma 7.1 *A pseudo-Boolean inequality $cL \geq d$ dominates $L_i \geq 1$ if and only if*

$$c_i L_i \in cL \qquad \text{and} \qquad s(c) - c_i < d ,$$

where $s(c)$ denotes the sum over all coefficients of c.

PROOF: Let $c'L'$ be $cL\backslash\{c_iL_i\}$ and let α be in $\mathsf{Ext}(cL \geq d)$. Then $\alpha(c_iL_i)+\alpha(c'L') \geq d$. Note that $s(c) - c_i = s(c')$. Suppose that $s(c') < d$. Since $\alpha(c'L') \leq s(c') < d$ we have $\alpha(c_iL_i) > 0$ and therefore $\alpha(L_i) = 1$. Suppose that $s(c') \geq d$ then each α such that $\alpha(L') = |L'|$ and $\alpha(L_i) = 0$ is in $\mathsf{Ext}(cL \geq d)$. □

If a pseudo-Boolean inequality $cL \geq d$ dominates $L_i \geq 1$ we call L_i *unit literal* w.r.t. $cL \geq d$.

Lemma 7.2 *If a pseudo-Boolean inequality $cL \geq d$ dominates $L_i \geq 1$ where $c_iL_i \in cL$ then $cL \geq d$ dominates $L_j \geq 1$ for all $c_jL_j \in cL$ where $c_j \geq c_i$.*

PROOF: If $s(c) - c_i < d$ then $s(c) - c_j < d$ because $c_j \geq c_i$ and therefore Lemma 7.1 applies. But then $cL \geq d$ dominates $L_j \geq 1$. □

By Lemma 7.2 we see that a pseudo-Boolean inequality $cL \geq d$ dominates $L_i \geq 1$ for some L_i if and only if $cL \geq d$ dominates $L_j \geq 1$ where L_j is a literal having the largest coefficient in $cL \geq d$. We define

$$\texttt{fixed}(cL \geq d) := \begin{cases} L_j & \text{if } s(c) - c_j < d \text{ where } c_jL_j \in cL \text{ and } c_j \geq c_i \text{ for all } c_i \in c, \\ \perp & \text{otherwise.} \end{cases}$$

If there are several possibilities for L_j then any of them can be chosen. We know then that $\texttt{fixed}(cL \geq d) = L_j$ if $cL \geq d$ dominates $L_j \geq 1$. If $\texttt{fixed}(cL \geq d) = \perp$ we know that there is no literal L_j such that $cL \geq d$ dominates $L_j \geq 1$. Given a unit literal L_i we can simplify a pseudo-Boolean inequality $cL \geq d$. We define

$$\texttt{fix}(L_i, cL \geq d) := \begin{cases} \top & \text{if } d - c_i \leq 0 \text{ and } c_iL_i \in cL \\ cL \setminus c_iL_i \geq d - c_i & \text{if } d - c_i > 0 \text{ and } c_iL_i \in cL \\ \Box & \text{if } s(c) - c_i < d \text{ and } c_i\overline{L_i} \in cL \\ cL \setminus c_i\overline{L_i} \geq d & \text{if } s(c) - c_i \geq d \text{ and } c_i\overline{L_i} \in cL \\ cL \geq d & \text{if neither } L_i \text{ nor } \overline{L_i} \text{ in } L; \end{cases}$$

which formally defines the operation of replacing the literal L_i by 1 and bringing the constant c_i to the right-hand side. Special cases arise when the inequality becomes tautologous ($\top$) or unsatisfiable ($\Box$) after fixing.

We derive that

$$\mathsf{Ext}(\{cL \geq d, L_i \geq 1\}) = \mathsf{Ext}(\{\texttt{fix}(L_i, cL \geq d), L_i \geq 1\}) \ . \tag{25}$$

Note that fixing a literal L_i in a pseudo-Boolean inequality $cL \geq d$ may only produce a tautology if $L_i \in L$ and may only produce $\Box$ if $\overline{L_i} \in L$. Given a set of pseudo-Boolean inequalities S we denote by $\texttt{fix}(L_i, S)$ the set of all pseudo-Boolean inequalities $\texttt{fix}(L_i, cL \geq d) \neq \top$ where $cL \geq d$ in S. Note that no tautologies are in $\texttt{fix}(L_i, S)$. We present now the generalization `pbur` (`p`seudo-`b`oolean `u`nit `r`esolution) of the unit resolution procedure `ur` for a set S of linear pseudo-Boolean inequalities.

```
pbur($S$)
    $U := \emptyset$
```

```
        **while** $\exists\, cL \geq d \in S \wedge \texttt{fixed}(cL \geq d) \neq \perp$
            $S := \texttt{fix}(\texttt{fixed}(cL \geq d), S)$
            $U := U \cup \{\texttt{fixed}(cL \geq d)\}$
        **endwhile**
        **return** $S$
    **end** pbur
```

We denote by $\texttt{pbul}(S) := U$ the set of all unit literals $\texttt{fixed}(cL \geq d)$ detected after applying $\texttt{pbur}(S)$. Obviously S is satisfiable if and only if $\texttt{pbur}(S)$ is satisfiable and each satisfying assignment α of S is a satisfying assignment of $\texttt{pbur}(S)$ and $\alpha(L_i) = 1$ for all $L_i \in \texttt{pbul}(S)$. We say that the pseudo-Boolean unit relaxation of S is unsatisfiable if $\Box \in \texttt{pbur}(S)$. If S contains only classical clauses then obviously $\texttt{ur}(S) = \texttt{pbur}(S)$ and therefore $\Box \in \texttt{pbur}(S)$ if and only if $\Box \in \texttt{ur}(S)$. If the pseudo-Boolean unit relaxation of S is unsatisfiable then the linear programming relaxation of S is unsatisfiable. The converse no longer holds. Consider for example

$$\{1 \cdot A + 1 \cdot B + 1 \cdot C \geq 2, 1 \cdot \overline{A} + 1 \cdot \overline{B} + 1 \cdot \overline{C} \geq 2\}\ .$$

Note that because of (25) we have in pbur as invariant of the **while**-loop that

$$\mathsf{Ext}(S_i \cup \{U_i \geq |U_i|\}) = \mathsf{Ext}(S_{i+1} \cup \{U_{i+1} \geq |U_{i+1}|\})\ ,$$

where S_i resp. U_i represents the actual S resp. U in the i-th iteration. So for a set of pseudo-Boolean inequalities S we have

$$\mathsf{Ext}(S) = \mathsf{Ext}(\texttt{pbur}(S) \cup \{\texttt{pbul}(S) \geq |\texttt{pbul}(S)|\})\ . \tag{26}$$

A set of literals that need to be fixed is the set of unit literals $\texttt{pbul}(S)$. Pseudo-Boolean unit resolution is a procedure detecting this set of literals.

Example 7.3 *Let S be*

$$\begin{aligned}\{5 \cdot A + 4 \cdot B + 3 \cdot C + 2 \cdot D &\geq 10,\\ 8 \cdot \overline{A} + 6 \cdot D + 4 \cdot E + 3 \cdot F &\geq 8\}\end{aligned}$$

and let us follow a computation of $\texttt{pbur}(S)$. *We obtain first* $\texttt{fixed}(5 \cdot A + 4 \cdot B + 3 \cdot C + 2 \cdot D \geq 10) = A$ *and simplify S to* $\texttt{fix}(\texttt{A}, \texttt{S})$.

$$\begin{aligned}\{4 \cdot B + 3 \cdot C + 2 \cdot D &\geq 5,\\ 6 \cdot D + 4 \cdot E + 3 \cdot F &\geq 8\}\end{aligned}$$

We next have $\texttt{fix}(6 \cdot D + 4 \cdot E + 3 \cdot F \geq 8) = D$ *and obtain* $\texttt{fix}(D, \texttt{fix}(A, S))$.

$$\begin{aligned}\{3 \cdot B + 3 \cdot C &\geq 3,\\ 2 \cdot E + 2 \cdot F &\geq 2\}\end{aligned}$$

Since now for all pseudo-Boolean inequalities fix *returns* $\perp$ *we have*

$$\mathsf{Ext}(S) = \mathsf{Ext}(\{3 \cdot B + 3 \cdot C \geq 3, 2 \cdot E + 2 \cdot F \geq 2, A + D \geq 2\})\ .$$

Transforming the remaining pseudo-Boolean inequalities is trivial and we obtain

$$\mathsf{Ext}(S) = \mathsf{Ext}(\{B + C \geq 1, E + F \geq 1, A + D \geq 2\})\ .$$

When a pseudo-Boolean inequality has been simplified by fixing literals then transforming the simplified pseudo-Boolean inequality is simpler, that is fewer strongest extended clauses are generated.

We know that a set of pseudo-Boolean inequalities S dominates $L_i \geq 1$ for some literal L_i if and only if $(S \cup \{\overline{L_i} \geq 1\})$ is unsatisfiable. We approximate the unsatisfiability test by checking whether the unit relaxation of $S \cup \{\overline{L_i} \geq 1\}$ is unsatisfiable, that is whether $\Box \in \texttt{pbur}(S \cup \{\overline{L_i} \geq 1\})$. We get the procedure `fixing` that detects some of the literals L_i for which $\alpha(L_i) = 1$ for all satisfying assignments α of a set of pseudo-Boolean inequalities S.

```
fixing($S$)
    $K := \emptyset$
    while $\exists L_i : \Box \in \texttt{pbur}(S \cup \{\overline{L_i} \geq 1\})$
        $K := K \cup \texttt{pbul}(S \cup \{L_i \geq 1\})$
        $S := \texttt{pbur}(S \cup \{L_i \geq 1\})$
    endwhile
    return $K$
end fixing
```

Note that S dominates $\texttt{fixing}(S) \geq |\texttt{fixing}(S)|$. Let K be $\texttt{fixing}(S)$ then

$$\mathsf{Ext}(S) = \mathsf{Ext}(\texttt{pbur}(S \cup \{K \geq |K|\}) \cup \{K \geq |K|\}) \ .$$

Note that `fixing` is a stronger procedure for detecting fixed literals of a set of pseudo-Boolean inequalities than pseudo-Boolean unit resolution and we have

$$\texttt{fixing}(S) \supseteq \texttt{pbul}(S) \ .$$

Example 7.4 *Let S be*

$$\begin{array}{rcl} \{5 \cdot A + 4 \cdot B + 3 \cdot C + 2 \cdot D & \geq & 7, \\ 3 \cdot \overline{B} + 3 \cdot D + 2 \cdot \overline{C} & \geq & 4\} \end{array}$$

then none of the two pseudo-Boolean inequalities dominates $L_i \geq 1$ for some literal L_i. Let us apply `fixing` *to S and start with calculating* $\texttt{pbur}(S \cup \{\overline{A} \geq 1\})$*. We first obtain* $\texttt{fix}(\overline{A}, S)$*.*

$$\begin{array}{rcl} \{4 \cdot B + 3 \cdot C + 2 \cdot D & \geq & 7, \\ 3 \cdot \overline{B} + 3 \cdot D + 2 \cdot \overline{C} & \geq & 4\} \end{array}$$

Now $\texttt{fixed}(4 \cdot B + 3 \cdot C + 2 \cdot D \geq 7) = B$ *and we calculate* $\texttt{fix}(B, \texttt{fix}(\overline{A}, S))$*.*

$$\begin{array}{rcl} \{3 \cdot C + 2 \cdot D & \geq & 3, \\ 3 \cdot D + 2 \cdot \overline{C} & \geq & 4\} \end{array}$$

We next derive $\texttt{fixed}(3 \cdot C + 2 \cdot D \geq 3) = C$ *and* $\texttt{fix}(C, \texttt{fix}(B, \texttt{fix}(\overline{A}, S)))$ *gives*

$$\begin{array}{rcl} \{ & & \top, \\ 3 \cdot D & \geq & 4\} \end{array}$$

where $3 \cdot D \geq 4$ is $\Box$. We conclude that $S \cup \{\overline{A} \geq 1\}$ is unsatisfiable. Thus S dominates $A \geq 1$ and we can replace S by $\texttt{fix}(A, S) \cup \{A \geq 1\}$*.*

$$\begin{array}{rcl} \{2 \cdot B + 2 \cdot C + 2 \cdot D & \geq & 2, \\ 3 \cdot \overline{B} + 3 \cdot D + 2 \cdot \overline{C} & \geq & 4, \\ A & \geq & 1\} \ . \end{array}$$

Note that $2 \cdot B + 2 \cdot C + 2 \cdot D \geq 2$ is already normalized. There is no further literal L_i such that $\Box \in S' \cup \{\overline{L_i} \geq 1\}$ and so `fixing` *ends.*

For example on the constraint set of a 0-1 integer optimization problem "air01" found on MIPLIB [BBI92] `fixing` detects that 130 out of the 771 boolean variables have to be fixed. On the other side `fixing` is a rather costly process. Suppose that n variables occur in S then for a complete application of `fixing` we have to compute $2n$ unit resolutions if `fixing` detects no fixed literals.

8 Constrained Simplification

Typically a pseudo-Boolean inequality $cL \geq d$ is part of a set of pseudo-Boolean inequalities S. We present reformulation techniques for a pseudo-Boolean inequality that *constrain* the reformulation of $cL \geq d$ w.r.t. S. In the previous sections we presented a method that reformulated a pseudo-Boolean inequality $cL \geq d$ as a set of extended clauses S' such that

$$\mathsf{Ext}(cL \geq d) = \mathsf{Ext}(S') \ .$$

Since $cL \geq d$ is part of a set of pseudo-Boolean inequalities S it is sufficient if $cL \geq d$ is equivalent to S' with respect to the context S, that is only

$$\mathsf{Ext}(S \cup \{cL \geq d\}) = \mathsf{Ext}(S \cup S')$$

need to hold. Assume that S' can also be a set of pseudo-Boolean inequalities. Then replacing $cL \geq d$ by S' is useful only if S' is in some sense *simpler* than $cL \geq d$.

We present in Sect. 8.1 a method that generates *stronger* pseudo-Boolean inequality $c'L' \geq d'$ from $cL \geq d$, that is $\mathsf{Ext}(c'L' \geq d') \subseteq \mathsf{Ext}(cL \geq d)$, for which

$$\mathsf{Ext}(S \cup \{cL \geq d\}) = \mathsf{Ext}(S \cup \{c'L' \geq d'\})$$

holds. So if $cL \geq d$ is part of the set of pseudo-Boolean inequalities S we can replace $cL \geq d$ by the stronger pseudo-Boolean inequality $c'L' \geq d'$. It is often possible to transform some simple pseudo-Boolean inequalities in S into a small equivalent set of extended clauses which can then be used to strengthen $cL \geq d$. Since extended clauses are computationally easier to handle we restrict ourself mainly to the case that the context S is a set of extended clauses. So the goal is to construct a pseudo-Boolean inequality $c'L' \geq d'$ that dominates $cL \geq d$ but equivalence w.r.t. S, that is $\mathsf{Ext}(S \cup \{cL \geq d\}) = \mathsf{Ext}(S \cup \{c'L' \geq d'\})$, still holds. Transforming such a strengthened pseudo-Boolean inequality $c'L' \geq d'$ instead of $cL \geq d$ generates stronger extended clauses.

In Sect. 8.2 we show how to replace $cL \geq d$ by a set of pseudo-Boolean inequalities S' w.r.t. the context S such that $\mathsf{Ext}(S \cup \{cL \geq d\}) = \mathsf{Ext}(S \cup S')$. The pseudo-Boolean inequalities in S' are simpler in the sense that each of them contains less literals than $cL \geq d$. The method can be applied when the original pseudo-Boolean inequality $cL \geq d$ is too large to be transformed completely. We then hope to transform more easily some of the pseudo-Boolean inequalities in S' and afterwards strengthen the set of extended clauses which again may lead to a strengthening of the remaining pseudo-Boolean inequalities in S'.

We present a variant of the transformation method in Sect 8.3. We assume again that the context of the pseudo-Boolean inequality $cL \geq d$ is a set of extended clauses S. We transform $cL \geq d$ into an equivalent set of extended clauses S' such that $\mathsf{Ext}(S \cup \{cL \geq d\}) = \mathsf{Ext}(S \cup S')$ but the number of extended clauses in S' is reduced or, optionally, the generated extended clauses are stronger than the one of the unconstrained transformation.

The *simplification* of a pseudo-Boolean inequality *constrained* by a context S is common to the methods presented in this section. In the context of constraint logic programming equivalence preserving reformulation of constraints into simpler ones w.r.t. a context is called *relative simplification* [Smo94]; an essential functionality of constraint solvers for concurrent constraint languages.

8.1 Coefficient Reduction

We use the context S for reducing one of the coefficients in the pseudo-Boolean inequality $cL \geq d$. The resulting pseudo-Boolean inequality dominates $cL \geq d$ but is equivalent to $cL \geq d$ w.r.t. the context S.

Let c_iL_i be a product of $cL \geq d$ and let $c'L'$ be $cL \setminus \{c_iL_i\}$. We search for a pseudo-Boolean inequality $c_i'L_i + c'L' \geq d$ where $c_i' < c_i$ and therefore $c_i'L_i + c'L' \geq d$ dominates $cL \geq d$, and

$$\mathsf{Ext}(S \cup \{c_i'L_i + c'L' \geq d\}) = \mathsf{Ext}(S \cup \{cL \geq d\}) \ . \tag{27}$$

Since $c_i'L_i + c'L' \geq d$ dominates $cL \geq d$ we know that $\mathsf{SCP}(\mathsf{SRed}(c_i'L_i + c'L' \geq d))$ dominates $\mathsf{SCP}(\mathsf{SRed}(cL \geq d))$. Because our transformation method generates a set of prime extended clauses we further know that each extended clause in $\mathsf{SCP}(\mathsf{SRed}(cL \geq d))$ is dominated by an extended clause in $\mathsf{SCP}(\mathsf{SRed}(c_i'L_i + c'L' \geq d))$. Transforming $c_i'L_i + c'L' \geq d$ instead of $cL \geq d$ generates therefore only stronger extended clauses, that is every extended clause we obtain by transforming $cL \geq d$ is dominated by an extended clause obtained by transforming $c_i'L_i + c'L' \geq d$.

We next describe how to derive a coefficient c_i' such that (27) holds. The key idea is to deduce a lower bound b for the pseudo-Boolean term $c'L'$ w.r.t. $S \cup \{L_i \geq 1\}$.

Proposition 8.1 *If $S \cup \{L_i \geq 1\}$ dominates $c'L' \geq b$ and $c_i > d - b$ then*

$$\mathsf{Ext}(S \cup \{c_iL_i + c'L' \geq d\}) = \mathsf{Ext}(S \cup \{(d-b)L_i + c'L' \geq d\}) \ .$$

We then have reduced the coefficient c_i to $c_i' = d - b < c_i$.

PROOF: The direction $\supseteq$ is obvious since $c_i > d - b$ and therefore $(d-b)L_i + c'L' \geq d$ dominates $c_iL_i + c'L' \geq d$. It remains to show $\subseteq$. Let α be in $\mathsf{Ext}(S \cup \{c_iL_i + c'L' \geq d\})$. We show that then α is in $\mathsf{Ext}(S \cup \{(d-b)L_i + c'L' \geq d\})$ by case analysis on $\alpha(L_i)$.

$\alpha(L_i) = 0$: Since then $\alpha(c_iL_i + c'L') = \alpha((d-b)L_i + c'L') = \alpha(c'L')$ we know that $\alpha(c_iL_i + c'L') \geq d$ if and only if $\alpha((d-b)L_i + c'L') \geq d$.

$\alpha(L_i) = 1$: It is sufficient to show that $\alpha((d-b)L_i + c'L') \geq d$ since α is in $\mathsf{Ext}(S)$. Because $\alpha(L_i) = 1$ we have $\alpha((d-b)L_i + c'L') = (d-b) + \alpha(c'L')$ and it remains to show that $(d-b) + \alpha(c'L') \geq d$ which simplifies to $\alpha(c'L') \geq b$. Since $S \cup \{L_i \geq 1\}$ dominates $c'L' \geq b$ we have $\alpha(c'L') \geq b$ for all assignments $\alpha \in \mathsf{Ext}(S)$ where $\alpha(L_i) = 1$. □

Dietrich et al. [DEC93] present a similar coefficient reduction technique where $\leq$-inequalities are used instead of $\geq$-inequalities. A necessary condition for the application of coefficient reduction is that $S \cup \{L_i \geq 1\}$ dominates $cL \geq d$. Otherwise S does not dominate $c'L' \geq d - c_i$ and therefore for each lower bound b of $c'L'$ w.r.t. $S \cup \{L_i \geq 1\}$ we have $b \leq d - c_i$ which contradicts $c_i > d - b$.

Example 8.2 *Let $cL \geq d$ be*

$$10 \cdot A + 5 \cdot B + 4 \cdot C + 3 \cdot D + 3 \cdot E \geq 10 \ . \tag{28}$$

Good candidates for coefficient reductions are obviously coefficient/literal pairs $c_i L_i$, where the coefficient c_i is relatively large w.r.t. the right-hand side because then the necessary condition, $S \cup \{L_i \geq 1\}$ dominates $cL \geq d$, is more likely to hold. Let us choose $10 \cdot A$ and since the coefficient 10 equals the right-hand side we know that $S \cup \{A \geq 1\}$ dominates (28) for all sets of extended clauses S. We can reduce the coefficient 10 if we find some lower bound $b > 0$ for the pseudo-Boolean term $5 \cdot B + 4 \cdot C + 3 \cdot D + 3 \cdot E$. Suppose that S contains the extended clause $\overline{A} + B + C + F \geq 2$. Since $\{\overline{A} + B + C + F \geq 2, A \geq 1\}$ dominates $B + C \geq 1$ we know that either B or C has to be 1. Therefore we can derive the lower bound $b = 4$.

$$\{\overline{A} + B + C + F \geq 2, A \geq 1\} \text{ dominates } 5 \cdot B + 4 \cdot C + 3 \cdot D + 3 \cdot E \geq 4$$

Following Prop. 8.1 we obtain $c'_i = 6$ and $c'_i L_i + cL \geq d$ is

$$6 \cdot A + 5 \cdot B + 4 \cdot C + 3 \cdot D + 3 \cdot E \geq 10 \ .$$

When transforming both pseudo-Boolean inequalities $c_i L_i + c'L' \geq d$ and $c'_i L_i + c'L' \geq d$ we see that the former generates only extended clauses with right-hand side 1 whereas the latter generates for example the strongest extended clause $A + B + C + D \geq 2$.

In order to obtain an ideal coefficient reduction we need to derive the maximal lower bound $b_{\min}$ of $c'L'$ w.r.t. $S \cup \{L_i \geq 1\}$. Because this involves solving the NP-complete problem

$$\min : c'L' \qquad \text{w.r.t.} \quad S \cup \{L_i \geq 1\}$$

we only approximate the best lower bound $b_{\min}$.

We propose the following procedure to obtain a valid lower bound b. Because the constraint set contains the unit literal L_i we first simplify the set of extended clauses by replacing $S \cup \{L_i \geq 1\}$ by $\texttt{pbur}(S \cup \{L_i \geq 1\})$. Note that $S \cup \{L_i \geq 1\}$ dominates $\texttt{pbur}(S \cup \{L_i \geq 1\})$. We then fix the literals of $\texttt{pbul}(S \cup \{L_i \geq 1\})$ in the objective function. For a set of literals K we define

$$\texttt{o\_reduce}(c'L', K) := \{c_j L_j \mid c_j L_j \in c'L', L_j \notin K \text{ and } \overline{L_j} \notin K\}$$

and the updated objective function then is $\hat{c}\hat{L} := \texttt{o\_reduce}(c'L', \texttt{pbul}(S \cup \{L_i \geq 1\}))$. Summing up the coefficients of the literals that are fixed to 1 gives us the constant $\hat{b}$ which can be added to the lower bound that we obtain from the approximation of the minimum for $\hat{c}\hat{L}$. For a set of literals K we define

$$\texttt{sum\_fixed}(c'L', K) := \sum_{c_j L_j \in c'L' : L_j \in K} c_j$$

and the desired constant then is $\hat{b} := \texttt{sum\_fixed}(c'L', \texttt{pbul}(S \cup \{L_i \geq 1\}))$. The minimum of $c'L'$ w.r.t. $S \cup \{L_i \geq 1\}$ is then the same as the minimum of $\hat{c}\hat{L} + \hat{b}$ w.r.t. $\texttt{pbur}(S \cup \{L_i \geq 1\})$. We next consider the problem of approximating the minimum of $\hat{c}\hat{L}$ w.r.t. the set of extended clauses $\texttt{pbur}(S \cup \{L_i \geq 1\})$, where no extended clause in $\texttt{pbur}(S \cup \{L_i \geq 1\})$ dominates an extended clause of the form $L_i \geq 1$. In other words no more obvious fixings apply.

We derive a set of extended clause S' from the set $\texttt{pbur}(S \cup \{L_i \geq 1\})$ such that S' is dominated by $\texttt{pbur}(S \cup \{L_i \geq 1\})$ and S' contains only literals also occuring in $\hat{L}$. For a set of literals K and a set of extended clauses T we define

$$\texttt{c\_reduce}(T, K) := \{K' \geq \beta' \;\left|\; \begin{array}{l} K' \uplus K'' \geq \beta \in T, \\ \beta' = \beta - |K''| \geq 1, \\ K'' \subseteq K \text{ and } K \cap K' = \emptyset \end{array}\right. \}$$

which is the set of all strict reductions of the extended clauses in T where all literals in K have been eliminated. The set of extended clauses S' then is $\texttt{c\_reduce}(\texttt{pbur}(S \cup \{L_i \geq 1\}), \mathcal{L} \setminus (\hat{L} \cup \overline{\hat{L}}))$.

We now give a greedy like approach to obtain a valid lower bound b of $\hat{c}\hat{L}$ w.r.t. the set of extended clauses S'. For each extended clause $K' \geq \beta'$ in S' we define

$$b_{K' \geq \beta'} := s(\hat{c}') \qquad \text{where} \quad \begin{array}{l} \hat{c}'\hat{L}' \subseteq \hat{c}\hat{L}, \\ \hat{L}' \subseteq K', \\ |\hat{L}'| = \beta' \text{ and} \\ \max(\hat{c}') \leq \min(\hat{c} \setminus \hat{c}') \end{array} .$$

That is $b_{K' \geq \beta'}$ is the sum of the smallest β' coefficients of $\hat{c}\hat{L}$ whose literals are in K'. So for each $\alpha \in \mathsf{Ext}(K' \geq \beta')$ we have $\alpha(\hat{c}\hat{L}) \geq b_{K' \geq \beta'}$ and so a first valid lower bound of $\hat{c}\hat{L}$ is $b_{K' \geq \beta'}$. Let us select some bound $b_{K' \geq \beta'}$. For assuring the bound $b_{K' \geq \beta'}$ we considered all literals occurring in K'. Note that the bound is also valid for the subterm of the objective function containing only the literals of K'. We can therefore safely replace this subterm in the objective function by the bound $b_{K' \geq \beta'}$ and minimize the simpler objective function. Hence a valid lower bound of $\hat{c}\hat{L}$ can be obtained by solving the simpler minimization problem

$$\min : \texttt{o\_reduce}(\hat{c}\hat{L}, K') + b_{K' \geq \beta'} \qquad \text{w.r.t.} \quad S'$$

which can again be solved by the same approach. We obtain the following algorithm calculating a valid lower bound b of $\hat{c}\hat{L}$ w.r.t. S'.

```
lower_bound($\hat{c}\hat{L}, S'$)
    b := 0
    **while** $\hat{c}\hat{L} \neq \emptyset \wedge S' \neq \emptyset$
        $K' \geq \beta'$ := select_clause($\hat{c}\hat{L}, S'$)
        b := b + $b_{K' \geq \beta'}$
        $S'$ := c_reduce($S', K'$)
        $\hat{c}\hat{L}$ := o_reduce($\hat{c}\hat{L}, K'$)
    **endwhile**
    **return** b
**end** lower_bound
```

The procedure `select_clause` selects an extended clause that is used for deriving the current bound. We use the following heuristic. We choose the extended clause $K' \geq \beta'$ that maximizes

$$\frac{b_{K' \geq \beta'}}{\sum_{c_i L_i \in \hat{c}\hat{L} : L_i \in K'} c_i} .$$

The heuristic reflects that the bound is the largest or best w.r.t. the sum of the coefficients it has consumed. The computed lower bound b of $c'L'$ w.r.t. $S \cup \{L_i \geq 1\}$ then is

$$\texttt{sum\_fixed}(c'L', K) + \texttt{lower\_bound}(\texttt{o\_reduce}(c'L', K), \texttt{c\_reduce}(\texttt{pbur}(S \cup \{L_i \geq 1\}), L' \setminus K)) \ ,$$

where $K = \texttt{pbul}(S \cup \{L_i \geq 1\})$.

Example 8.3 *Let $cL \geq d$ be*

$$5 \cdot A + 4 \cdot B + 4 \cdot C + 4 \cdot D + 3 \cdot E + 2 \cdot F \geq 11 \tag{29}$$

and let the set of extended clauses S be

$$\{\overline{A} + B \geq 1, C + D + G \geq 2, D + E + F \geq 1\} \ .$$

Let us reduce the the coefficient 5 of the literal A. We apply pseudo-Boolean unit resolution to $S \cup \{A \geq 1\}$ and obtain

$$\begin{aligned} \texttt{pbur}(S \cup \{A \geq 1\}) &= \{C + D + G \geq 2, D + E + F \geq 1\} \text{ and} \\ \texttt{pbul}(S \cup \{A \geq 1\}) &= \{A, B\} \ . \end{aligned}$$

We next reduce $\texttt{pbur}(S \cup \{A \geq 1\})$ such that only the literals of the objective function remain, that is we eliminate all literals in $\mathcal{L} \setminus \{C, D, E, F, \overline{C}, \overline{D}, \overline{E}, \overline{F}\}$ and obtain

$$\{C + D \geq 1, D + E + F \geq 1\} \ .$$

We update the objective function $c'L'$ by calculating

$$\begin{gathered} \texttt{sum\_fixed}(4 \cdot B + 4 \cdot C + 4 \cdot D + 3 \cdot E + 2 \cdot F, \{A, B\}) = 4 \\ \texttt{o\_reduce}(cL, \{A, B\}) = 4 \cdot C + 4 \cdot D + 3 \cdot E + 2 \cdot F \end{gathered}$$

and then call $\texttt{lower\_bound}$. Calculating the $b_{K' \geq \beta'}$ yields

$$b_{C+D \geq 1} = 4 \qquad b_{D+E+F \geq 1} = 2$$

and since $4/8 > 2/9$ we select $C + D \geq 1$. Updating the objective function and the extended clause set gives

$$\begin{aligned} \texttt{o\_reduce}(4 \cdot C + 4 \cdot D + 3 \cdot E + 2 \cdot F, \{C, D\}) &= 3 \cdot E + 2 \cdot F \\ \texttt{c\_reduce}(\{C + D \geq 1, D + E + F \geq 1\}, \{C, D\}) &= \emptyset \end{aligned}$$

and hence $\texttt{lower\_bound}$ stops and returns 4. The overall bound b then is $4 + 4 = 8$ and therefore $S \cup \{A \geq 1\}$ dominates

$$4 \cdot B + 4 \cdot C + 4 \cdot D + 3 \cdot E + 2 \cdot F \geq 8 \ .$$

By Prop. 8.1 we have $5 = c_i > d - b = 11 - 8 = 3$ and can so reduce the coefficient 5 of A to 3. We finally obtain the stronger pseudo-Boolean inequality

$$4 \cdot B + 4 \cdot C + 4 \cdot D + 3 \cdot A + 3 \cdot E + 2 \cdot F \geq 11$$

which is equivalent to (29) w.r.t. to the context S.

The presented method for deriving a lower bound of $c'L'$ w.r.t. $S \cup \{L_i \geq 1\}$ yields good bounds when the set of extended clauses S is sparse or if the set of extended clauses S is nearly prime. Note that any method can be used to obtain a valid lower bound and the better the approximation of $b_{\min}$ the more likely it is that coefficient reduction applies.

8.2 Constrained Strict Reductions

We next consider a method that replaces a pseudo-Boolean inequality $cL \geq d$ by a set of pseudo-Boolean inequalities S' such that $cL \geq d$ is equivalent to S' w.r.t. a set of extended clauses S.

$$\mathsf{Ext}(S \cup \{I\}) = \mathsf{Ext}(S \cup S')$$

In the previous section we focused on deriving lower bounds for a pseudo-Boolean subterm of the original pseudo-Boolean inequality and so strengthened the pseudo-Boolean inequality and therefore the extended clauses generated by the transformation method. Application of coefficient reduction is always an improvement. Conversely to the method of Sect. 8.1 we allow here that S' is weaker than $cL \geq d$, that is $\mathsf{Ext}(S') \supseteq \mathsf{Ext}(cL \geq d)$. The set S' is simpler because the pseudo-Boolean inequalities in S' contain less literals than $cL \geq d$. Unfortunately we may loose strong extended clauses when transforming the pseudo-Boolean inequalities in S' instead of $cL \geq d$. On the other side we possibly reduce the number of generated extended clauses. The idea is that some of the pseudo-Boolean inequalities in S' are easier to transform than $cL \geq d$. Combining the obtained strongest extended clauses with S may strengthen S which again may lead to further simplifications.

When transforming a pseudo-Boolean inequality $cL \geq d$ we need to calculate all strict reductions $\mathsf{SRed}(cL \geq d)$. We show how to reduce the number of necessary strict reductions, that is we *constrain* the generation of strict reductions and so reduce the number of generated extended clauses.

Proposition 8.4 *Let $cL \geq d$ be a pseudo-Boolean inequality and let $L' \geq \beta$ be an extended clause where $L' \subseteq L$. Let S' be the set of pseudo-Boolean inequalities*

$$\left\{\hat{c}\hat{L} \geq \hat{d} \;\middle|\; \begin{array}{l} \hat{c}\hat{L} = cL \setminus c''L'', \\ \hat{d} = d - s(c''), \\ c''L'' \subseteq cL, \\ L'' \subseteq L' \text{ and } |L''| = \beta \end{array} \right\}$$

then $\mathsf{Ext}(\{L' \geq \beta, cL \geq d\}) = \mathsf{Ext}(\{L' \geq \beta\} \cup S')$. Note that S' is the set of all strict reductions of $cL \geq d$, where the set of literals L'' for all $L'' \subseteq L'$ and $|L''| = \beta$ has been eliminated.

PROOF: Since $S' \subseteq \mathsf{SRed}(cL \geq d)$ we know that $cL \geq d$ dominates S'. It remains to show that every satisfying assignment α of $\{L' \geq \beta\} \cup S'$ is a satisfying assignment of $cL \geq d$. Let α be in $\mathsf{Ext}(\{L' \geq \beta\} \cup S')$. Let $L''' \subseteq L'$ be the set of literals such that for each $L_i \in L'''$ we have $\alpha(L_i) = 1$. Because $L' \geq \beta$ we know that $|L'''| \geq \beta$. Let L'' be a subset of L''' such that $|L''| = \beta$. We select the strict reduction $\hat{c}\hat{L} \geq d - s(c'')$ from S' where L'' has been eliminated. We denote by $s(c'')$ the sum over the coefficients c_i for all $c_iL_i \in c''L''$. Because α is a solution of S' we know that $\alpha(\hat{c}\hat{L}) \geq d - s(c'')$ and therefore $\alpha(\hat{c}\hat{L}) + s(c'') \geq d$. Because $\alpha(\hat{c}\hat{L}) + s(c'') = \alpha(cL)$ we have $\alpha(cL) \geq d$. □

Example 8.5 *Let $cL \geq$ be*

$$c_1 \cdot A + c_2 \cdot B + c_3 \cdot C + cL \geq d$$

and suppose that the extended clause $A + B + C \geq 2$ is dominated by an extended clause in the context S. Note that c_1, c_2, c_3 can be every coefficients of $cL \geq d$, not necessarily the largest ones. We can then replace $cL \geq d$ by $S' =$

$$\begin{aligned}\{c_1 \cdot A + cL &\geq d - (c_2 + c_3),\\ c_2 \cdot B + cL &\geq d - (c_1 + c_3),\\ c_3 \cdot C + cL &\geq d - (c_1 + c_2)\}\end{aligned}$$

according to Prop. 8.4 and have $\mathsf{Ext}(S \cup \{cL \geq d\}) = \mathsf{Ext}(S \cup S')$.

The use of Prop. 8.4 is only practicable if the number of pseudo-Boolean inequalities it generates is not too high. Note that Prop. 8.4 generates $\binom{|L'|}{\beta}$ extended clauses. So most reasonably it is applicable if β is close to $|L'|$ since then the number of pseudo-Boolean inequalities is small and the number of deleted literals is high. In Example 8.5 we used an extended clause, where $|L'| - \beta = 1$ and so $\binom{|L'|}{\beta} = |L'|$. Extended clauses of the form $L' \geq |L'| - 1$ are also called *clique inequalities* or *special ordered set constraints* [NW88] and frequently occur in practical pseudo-Boolean constraint sets.

One problem of the application of Prop. 8.4 is that we may loose strong extended clauses.

Example 8.6 [a] *Let $cL \geq d$ be*

$$5 \cdot A + 4 \cdot B + 3 \cdot C \geq 6$$

then $cL \geq d$ is equivalent to $A+B+C \geq 2$. Applying Prop. 8.4 on $cL \geq d$ with the extended clause $A + B \geq 1$ gives the set of pseudo-Boolean inequalities

$$\{5 \cdot A + 3 \cdot C \geq 2, 4 \cdot B + 3 \cdot C \geq 1\} \tag{30}$$

equivalent to $cL \geq d$ w.r.t. $\{A + B \geq 1\}$. Transforming the pseudo-Boolean inequalities in (30) yields

$$\{A + C \geq 1, B + C \geq 1\} \ .$$

Now Prop. 8.4 assures that

$$\mathsf{Ext}(\{A + C \geq 1, B + C \geq 1, A + B \geq 1\}) = \mathsf{Ext}(\{A + B + C \geq 2\})$$

but the better *formulation of $cL \geq d$ is obviously $\{A + B + C \geq 2\}$. Indeed in [Bar94] we describe how to obtain this better formulation if only $\{A+C \geq 1, B+C \geq 1\}$ and $A+B \geq 1$ are given.*

[b] *Let $cL \geq d$ be*

$$d \cdot A + c_i \cdot B + cL \geq d$$

then we know that all generated extended clauses have degree 1. Suppose that there is a clique inequality in S dominating $A+B \geq 1$. Then there is no strict reduction on A, because eliminating A would give a tautology. Because of Prop. 8.4 we know that $cL \geq d$ is equivalent to

$$(d - c_i) \cdot A + cL \geq d - c_i$$

and we are sure that we do not miss a strongest extended clause. So applications of Prop. 8.4 where all but one[3] *of the generated strict reductions are dominated by the context are safe.*

Another problem is that transforming the pseudo-Boolean inequalities in the set S' of Prop. 8.4 may involve redundant computation. While transforming the pseudo-Boolean inequalities of S' in Example 8.5 we fully transform 3 times the strict reduction $cL \geq d - (c_1 + c_2 + c_3)$. We avoid the redundant computation by using Prop. 8.4 directly in the transformation routine.

Example 8.7 *Let $cL \geq d$ be*

$$c_1 \cdot A + c_2 \cdot B + c_3 \cdot C + cL \geq d$$

as in Example 8.5 and suppose that $c_1 \geq c_2 \geq c_3$. We assume that the extended clause

$$A + B + C \geq 2$$

is dominated by an extended clause in the context. Because of Prop. 8.4 we know that $cL \geq d$ is equivalent to the set of pseudo-Boolean inequalities $S' =$

$$\begin{aligned} \{c_1 \cdot A + cL &\geq d - (c_2 + c_3), \\ c_2 \cdot B + cL &\geq d - (c_1 + c_3), \\ c_3 \cdot C + cL &\geq d - (c_1 + c_2)\} \ . \end{aligned}$$

We now transform $c_1 \cdot A + cL \geq d - (c_2 + c_3)$ completely with the standard transformation method but we start the transformation of $c_2 \cdot B + cL \geq d - (c_1 + c_3)$ with the forbidden elimination on B since all extended clauses generated when B is eliminated are already obtained by the full transformation of $c_1 \cdot A + cL \geq d - (c_2 + c_3)$. Likewise we forbid elimination of C when transforming $c_3 \cdot C + cL \geq d - (c_1 + c_2)$ and so avoid the redundant transformation of $cL \geq d - (c_1 + c_2 + c_3)$.

8.3 Constrained Transformation

We present a variant of the transformation method that restricts the number of generated extended clauses while transforming a pseudo-Boolean inequality $cL \geq d$ w.r.t. a set of extended clauses S. Let E be the set of non redundant extended clauses generated by the transformation of $cL \geq d$ then *constrained transformation* generates a smaller set $E' \subseteq E$ such that

$$\mathsf{Ext}(S \cup \{cL \geq d\}) = \mathsf{Ext}(S \cup E') \ .$$

While transforming a pseudo-Boolean inequality $cL \geq d$ we need to compute all strict reductions $\mathsf{SRed}(cL \geq d)$. When proving the equivalence of $\mathsf{SCP}(\mathsf{SRed}(cL \geq d))$ and $cL \geq d$ in Theorem 3.6 we show that for each non satisfying assignment α of $cL \geq d$ we find an extended clause $L' \geq \beta$ in $\mathsf{SCP}(\mathsf{SRed}(I))$ such that α is not a satisfying assignment of $L' \geq \beta$. If α is not a satisfying assignment of the context S then we need not find an extended clause in $\mathsf{SCP}(\mathsf{SRed}(cL \geq d))$ for which α is not a satisfying assignment. Hence the corresponding generated extended clause need

[3]If all generated strict reductions are dominated by the context then of course the whole pseudo-Boolean inequality is dominated by the context and can be ignored.

not be added for preserving equivalence. We can so restrict the number of needed strict reductions and therefore the number of generated extended clauses. We first define a set of strict reductions of $cL \geq d$ taking into account the context S.

Definition 8.8 Let $cL \geq d$ be a pseudo-Boolean inequality, let S be a set of extended clauses and $c'L' \geq d'$ be a strict reduction of $cL \geq d$. The set of literals $L \backslash L'$ is the set of literals that have been eliminated from $cL \geq d$ in order to obtain $c'L' \geq d'$. We denote by $\mathsf{SRed}(cL \geq d)_{|S} \subseteq \mathsf{SRed}(cL \geq d)$ the set of strict reductions of $cL \geq d$ such that

$$c'L' \geq d' \in \mathsf{SRed}(cL \geq d)_{|S} \quad \text{iff} \quad S \cup \{L \setminus L' \geq |L \setminus L'|\} \text{ is satisfiable} \ .$$

We now show that the set $\mathsf{SCP}(\mathsf{SRed}(cL \geq d)_{|S})$ of strongest extended clauses obtained from the smaller set of strict reductions is sufficient for the equivalence with $cL \geq d$ w.r.t. a set S of extended clauses.

Theorem 8.9 *Let $cL \geq d$ be a pseudo-Boolean inequality and let S a set of extended clauses, then*

$$\mathsf{Ext}(S \cup \{cL \geq d\}) = \mathsf{Ext}(S \cup \mathsf{SCP}(\mathsf{SRed}(cL \geq d)_{|S})) \ .$$

PROOF:

- $\mathsf{Ext}(S \cup \{cL \geq d\}) \subseteq \mathsf{Ext}(S \cup \mathsf{SCP}(\mathsf{SRed}(cL \geq d))_{|S})$: as in Theorem 3.6.
- $\mathsf{Ext}(S \cup \{cL \geq d\}) \supseteq \mathsf{Ext}(S \cup \mathsf{SCP}(\mathsf{SRed}(cL \geq d))_{|S})$:
  We show that if $\alpha \notin \mathsf{Ext}(S \cup \{cL \geq d\})$ then $\alpha \notin \mathsf{Ext}(S \cup \mathsf{SCP}(\mathsf{SRed}(cL \geq d)_{|S}))$ from which the theorem follows. Suppose that $\alpha \notin \mathsf{Ext}(S \cup \{cL \geq d\})$. If $\alpha \notin \mathsf{Ext}(S)$ then obviously $\alpha \notin \mathsf{Ext}(S \cup \mathsf{SCP}(\mathsf{SRed}(cL \geq d)_{|S}))$.
  Suppose that $\alpha \in \mathsf{Ext}(S)$ but $\alpha \notin \mathsf{Ext}(cL \geq d)$. We show that in this case there is a strict reduction in $\mathsf{SRed}(cL \geq d)_{|S}$ such that α is not a satisfying assignment of this reduction, hence $\alpha \notin \mathsf{Ext}(\mathsf{SCP}(\mathsf{SRed}(cL \geq d)_{|S}))$ and thus $\alpha \notin \mathsf{Ext}(S \cup \mathsf{SCP}(\mathsf{SRed}(cL \geq d)_{|S}))$. Since $\alpha \notin \mathsf{Ext}(cL \geq d)$ we know that $\alpha(cL) = d' < d$. Let Y be the set of literals L_i where $\alpha(L_i) = 1$ and $Z = L \backslash Y$, then $\sum_{i:L_i \in Y} c_i = d' < d$. Let $c'L' \geq d'$ be $\sum_{i:L_i \in Z} c_i L_i \geq d - d'$. Because $d' < d$ we know that $d - d' \geq 1$ and therefore $c'L' \geq d'$ is a strict reduction of $cL \geq d$, obtained by eliminating the literals that are mapped to zero by α. We know furthermore that $Y = L \setminus L'$, hence $\alpha(L \setminus L') \geq |L \setminus L'|$. Because $\alpha \in \mathsf{Ext}(S)$ we know that $\alpha \in (S \cup \{L \setminus L' \geq |L \setminus L'|\})$ and therefore $c'L' \geq d'$ in $\mathsf{SRed}(cL \geq d)_{|S}$. Let $\mathsf{SCP}(c'L' \geq d')$ be $L' \geq \beta$. We know that $\beta \geq 1$ because $d - d' \geq 1$. Because $\alpha(L') = 0$ we derive that $\alpha \notin \mathsf{Ext}(\mathsf{SCP}(c'L' \geq d'))$ and therefore $\alpha \notin \mathsf{Ext}(\mathsf{SCP}(\mathsf{SRed}(cL \geq d)))$, which establish the theorem. □

The proof of Theorem 8.9 is similar to the proof of Theorem 3.6. The main difference is that we additionally show that each strict reduction we need is in the restricted set of strict reductions. Theorem 3.6 can be seen as an instance of Theorem 8.9, where the set of extended clauses S is the empty set.

An implementation of the constrained transformation algorithm is now straightforward. For each strict reduction $c'L' \geq d'$ of $cL \geq d$ we maintain a set of extended clauses $c'L' \geq d'_S :=$

$S \cup \{L \setminus L' \geq |L \setminus L'|\}$. By Theorem 8.9 we can eliminate the strict reduction $c'L' \geq d'$ from the transformation process if $c'L' \geq d'_S$ is unsatisfiable. Note that if $c'L' \geq d'_S$ is unsatisfiable then for each strict reduction $c''L'' \geq d''$ of $c'L' \geq d'$ the corresponding set of extended clauses $\{S \cup \{L \setminus L'' \geq |L \setminus L''|\}$ is unsatisfiable. Care must be taken when applying the strong redundancy criteria presented in Sect. 4. Suppose that a strict reduction $c'L' \geq d'$ is eliminated because $c'L' \geq d'_S$ is unsatisfiable. Suppose furthermore that $\mathsf{SCP}(c'L' \geq d')$ dominates an extended clause $\mathsf{SCP}(c''L'' \geq d'')$ where $c''L'' \geq d''$ is a strict reduction of $c'L' \geq d'$ and there is no other non redundant extended clause dominating $\mathsf{SCP}(c'L' \geq d')$. In that case we have to add $\mathsf{SCP}(c''L'' \geq d'')$ or $\mathsf{SCP}(c'L' \geq d')$ for preserving equivalence. The simplest way is to add $\mathsf{SCP}(c'L' \geq d')$ always if $c'L' \geq d'_S$ is unsatisfiable and then safely forbid further strict reductions of $c'L' \geq d'$, that is we not recursively transform $c'L' \geq d'$. When later inserting the transformed set of extended clauses into a larger set, redundant extended clauses can be deleted.

In the constrained transformation algorithm we need to solve again a satisfiability problem for the set of extended clauses $S \cup \{L \setminus L' \geq |L \setminus L'|\}$. Because this is an NP-complete problem we relax the problem again to checking whether the unit relaxation of $S \cup \{L \setminus L' \geq |L \setminus L'|\}$ is unsatisfiable. That is, at each node we check whether $\square \in \mathtt{pbur}(S \cup \{L \setminus L' \geq |L \setminus L'|\})$. Note that all literals in $L \setminus L'$ are unit literals and therefore unsatisfiability of $S \cup \{L \setminus L' \geq |L \setminus L'|\}$ is likely to be detected by unit resolution.

Example 8.10 *Let $cL \geq d$ be*

$$6 \cdot A + 5 \cdot B + 4 \cdot C + 3 \cdot D + 2 \cdot E + F \geq 13$$

and suppose that the extended clause $\overline{B} + \overline{C} \geq 1$ is dominated by an extended clause in the context S. Hence $S \cup \{B + C \geq 2\}$ is unsatisfiable which is detected by pseudo-Boolean unit resolution. The full transformation of $cL \geq d$ generates 5 extended clauses. One strict reduction of $cL \geq d$ is

$$6 \cdot A + 3 \cdot D + 2 \cdot E + F \geq 4 \tag{31}$$

from which the two non redundant extended clauses

$$A + E + F \geq 1 \qquad A + D \geq 1$$

are derived. Since the set of extended clauses $6 \cdot A + 3 \cdot D + 2 \cdot E + F \geq 4_{|S}$ is unsatisfiable, and therefore $\mathsf{SRed}(6 \cdot A + 3 \cdot D + 2 \cdot E + F \geq 4)_{|S} = \emptyset$, we need not consider $6 \cdot A + 3 \cdot D + 2 \cdot E + F \geq 4$ and constrained transformation generates only 3 extended clauses. Note that $\mathsf{SCP}(6 \cdot A + 3 \cdot D + 2 \cdot E + F \geq 4)$ is redundant and dominated by another extended clause from the remaining 3 and so we need not include $\mathsf{SCP}(6 \cdot A + 3 \cdot D + 2 \cdot E + F \geq 4)$.

Instead of deleting the strict reduction $c'L' \geq d'$ if $c'L' \geq d'_{|S}$ is unsatisfiable we can use that fact for strengthening the generated extended clause. We explain the idea first on an example. Let $cL \geq d$ be

$$c_1 \cdot A + c_2 \cdot B + cL \geq d \ .$$

and suppose that $S \cup \{A + B \geq 2\}$ is unsatisfiable. Hence the strict reduction

$$cL \geq d - (c_1 + c_2) \ . \tag{32}$$

need not be considered. However because $S \cup \{A + B \geq 2\}$ is unsatisfiable we know that $\overline{A} + \overline{B} \geq 1$ is a valid extended clause w.r.t. S. So we can derive an upper bound for the pseudo-Boolean term $c_1 \cdot A + c_2 \cdot B$, namely $\{\overline{A} + \overline{B} \geq 1\}$ dominates $c_1 \cdot A + c_2 \cdot B \leq \max(c_1, c_2)$. Therefore we derive the stronger pseudo-Boolean inequality

$$cL \geq d - \max(c_1, c_2) \ . \tag{33}$$

We miss no extended clause if we replace the strict reduction (32) by (33) but may only generate stronger extended clauses. Note that we derive here an *upper bound* for a pseudo-Boolean term and use the bound in order to strengthen the generated extended clauses.

Lemma 8.11 *Let $cL \geq d$ be a pseudo-Boolean inequality and let $c'L' \geq d'$ be a strict reduction of $cL \geq d$. Let S be a set of extended clauses such that $c'L' \geq d'_S$ is unsatisfiable. Let $c''L''$ be $cL \setminus c'L'$ then*

$$\mathsf{Ext}(S \cup \{cL \geq d\}) = \mathsf{Ext}(S \cup (\mathsf{SCP}(\mathsf{SRed}(cL \geq d)_{|S})) \cup \{c'L' \geq d' + \min(c'')\}) \ .$$

PROOF: The direction $\supseteq$ is immediate by Theorem 8.9. It remains to show $\subseteq$, that is $S \cup \{cL \geq d\}$ dominates $c'L' \geq d' + \min(c'')$. Let $\alpha \in \mathsf{Ext}(S \cup \{cL \geq d\})$ then we have $\alpha(cL) \geq d$ and therefore $\alpha(c'L') + \alpha(c''L'') \geq d$. Since $c'L' \geq d'_S = S \cup \{L'' \geq |L''|\}$ is unsatisfiable we know furthermore that S dominates $\overline{L''} \geq 1$, that is at least one literal of L'' is mapped to 0 by α. We conclude that $\alpha(c''L'') \leq s(c'') - \min(c'')$ and since $\alpha(c'L') + \alpha(c''L'') \geq d$ we have $\alpha(c'L') + s(c'') - \min(c'') \geq d$. Because $d' = d - s(c'')$ we derive $\alpha(c'L') + s(c'') - \min(c'') \geq d' + s(c'')$ which simplifies to $\alpha(c'L') \geq d' + \min(c'')$. □

With Lemma 8.11 we can replace the strict reduction $c'L' \geq d'$ by the stronger pseudo-Boolean inequality $c'L' \geq d' + \min(c'')$ while preserving equivalence w.r.t. to the context S.

Example 8.12 *Let $cL \geq d$ again be*

$$6 \cdot A + 5 \cdot B + 4 \cdot C + 3 \cdot D + 2 \cdot E + F \geq 13$$

and suppose that $S \cup \{B + C \geq 2\}$ is unsatisfiable. A non redundant extended clause dominated by $cL \geq d$ is $A + E + F \geq 1$. While eliminating B and C we detect unsatisfiability of $S \cup \{B + C \geq 2\}$ and can replace the reduction

$$6 \cdot A + 3 \cdot D + 2 \cdot E + F \geq 4$$

by

$$6 \cdot A + 3 \cdot D + 2 \cdot E + F \geq 8$$

from which we derive the non redundant extended clause $A + E \geq 1$ dominating $A + E + F \geq 1$. Note that $6 \cdot A + 3 \cdot D + 2 \cdot E + F \geq 8$ dominates $A \geq 1$ which is obtained from the strongest extended clause of the strict reduction obtained when further eliminating $\{D, E, F\}$. So in this case we replace a non redundant extended clause by a stronger one but we may also derive new extended clauses that do not dominate other generated extended clause.

We propose to use Lemma 8.11 in the constrained transformation algorithm and to consider no further strict reductions of the stronger strict reduction $c'L' \geq d' + \min(c'')$ but only add $\mathsf{SCP}(c'L' \geq d' + \min(c''))$[4] to the set of extended clauses. Note that we get the stronger extended clause $\mathsf{SCP}(c'L' \geq d' + \min(c''))$ with minimal computational effort and we have to add $\mathsf{SCP}(c'L' \geq d')$ in any case in order to apply the redundancy criteria presented in Sect. 4.

Obviously, all results still hold if we use instead of a set of extended clauses as context a set of pseudo-Boolean inequalities for constraining the transformation. We presented the constrained transformation for extended clauses because we can strengthen the set of extended clauses using the deductive system "Generalized Resolution" [Hoo92, Bar94]. For such a strengthened set of extended clauses, unsatisfiability of $c'L' \geq d'_S$ is typically more often detected by pseudo-Boolean unit resolution and therefore the set of generated extended clauses is even smaller. The presentation of the strengthening methods [Bar94] is not in the scope of this paper.

9 Applications

We present two applications of the transformation algorithm besides its use for the constraint solver of CLP($\mathcal{PB}$). We show how to enhance the symbolic deduction method "Generalized Resolution" [Hoo92] and we relate our method to traditional linear 0-1 integer programming [NW88].

9.1 Generalized Resolution

Hooker [Hoo92] defined the *resolvent* of two extended clauses as the classical resolvent of two classical clauses, each of them dominated by one of the extended clauses. We enhance the notion of resolvents between extended clauses and show how to efficiently implement the inference rule with help of the presented transformation method.

Example 9.1 *Let*

$$A + B + \overline{D} + E + \overline{F} + \overline{G} \geq 4 \tag{34}$$

$$\overline{A} + \overline{C} + \overline{D} + \overline{E} + F + G \geq 3 \tag{35}$$

be two extended clauses. The classical clause $A + B + \overline{D} \geq 1$ is dominated by (34) and $\overline{A} + \overline{C} + \overline{D} + \overline{E} \geq 1$ is dominated by (35). From these two classical clauses we obtain the classical resolvent $B + \overline{C} + \overline{D} + \overline{E} \geq 1$ by resolving on A.

It is well known [Hoo88] that the resolvent of two classical clauses can be obtained by linear combination of the two classical clauses and the valid bounds $L_i \geq 0$ and $-L_i \geq -1$ for all literals L_i, and integer rounding. So resolvents are also cutting plane inequalities. It is easy to see that resolvents of extended clauses can also be obtained by linear combination and rounding since classical clauses dominated by extended clause are strict reductions that are obtained by adding the valid bound $-L_i \geq -1$ for all literals L_i that are eliminated. We show that transforming the linear

[4]Or better the non redundant extended clause dominating $\mathsf{SCP}(c'L' \geq d' + \min(c''))$ on the path H_{k+1} of Prop. 4.8.

pseudo-Boolean inequality obtained by summing up two extended clauses generates all extended clauses that can be derived from the two extended clauses; among them all possible resolvents.

The *sum* of two pseudo-Boolean inequalities $cL \geq d$ and $c'L' \geq d'$ is the pseudo-Boolean inequality $cL + c'L' \geq d + d'$. We assume as usual that the sum of two pseudo-Boolean inequalities is brought into pseudo-Boolean normal form.

Proposition 9.2 *Each resolvent of two extended clauses $L \geq d$ and $L' \geq d'$ is dominated by an extended clause in $\mathsf{SCP}(\mathsf{SRed}(cL + c'L' \geq d + d'))$.*

PROOF: We first extract the literals that occur positive in one extended clause and negative the other. Let $L_R + K = L$ and $L'_R + \overline{K} = L'$, where $K = \{L_i \mid L_i \in L \text{ and } \overline{L_i} \in L'\}$ and $\overline{K} = \{\overline{L_i} \mid L_i \in K\}$. The general representation[5] of each classical clause that is dominated by $L \geq d$ is $L_C + K_C \geq 1$, where $|K_C| = 1$ and $|L_R \setminus L_C| + |K \setminus K_C| \leq d - 1$. Similarly the general representation of each classical clause that is dominated by $L' \geq d'$ is $L'_C + K'_C \geq 1$, where $|K'_C| = 1$ and $|L'_R \setminus L'_C| + |K' \setminus K'_C| \leq d' - 1$. We next extract the literals that occur in both L_C and L'_C. Let $P \cup Q = L_C$ and $P \cup Q' = L'_C$ where $L_C \cap L'_C = P$ and $Q \cap Q' = \emptyset$. So the general representation of a resolvent of $L \geq d$ and $L' \geq d'$ is

$$P \cup Q \cup Q' \geq 1 \ .$$

Let us now investigate the sum of $L \geq d$ and $L' \geq d'$. We know that $L + L' \geq d + d'$ is equivalent to

$$L_R + L'_R + |K| \geq d + d'(\geq |L_R \setminus L_C| + |L'_R \setminus L'_C| + 2 + 2 \cdot (|K| - 1))$$

from which we derive

$$L_R + L'_R \geq d + d' - |K|(\geq |L_R \setminus L_C| + |L'_R \setminus L'_C| + |K|) \ .$$

A valid strict reduction of $L + L' \geq d + d'$ is

$$L_C + L'_C \geq d + d' - |K| - |L_R \setminus L_C| - |L'_R \setminus L'_C|(\geq |K| \geq 1)$$

obtained by eliminating the literals in $(L_R \setminus L_C) \cup (L'_R \setminus L'_C)$. We rewrite L_C and L'_C and obtain

$$2 \cdot P + Q + Q' \geq d + d' - |K| - |L_R \setminus L_C| - |L'_R \setminus L'_C|(\geq |K| \geq 1)$$

We apply the cutting plane operation of Sect. 3 and obtain

$$P + Q + Q' \geq \left\lceil \frac{d + d' - |K| - |L_R \setminus L_C| - |L'_R \setminus L'_C|}{2} \right\rceil (\geq \left\lceil \frac{|K|}{2} \right\rceil \geq \left\lceil \frac{1}{2} \right\rceil = 1)$$

which obviously implies $P + Q + Q' \geq 1$. Because each extended clause obtained by linear combination and rounding is dominated by a strongest extended clause the proposition is established. □

[5]Note that we consider only classical clauses that do not introduce new literals, because their resolvent is dominated by either one of the fathers, or by the resolvent of the valid classical clauses where we do not introduce new literals.

By Prop. 9.2 we can replace the original resolution rule of "Generalized Resolution" by a rule generating the equivalent set of extended clauses of the pseudo-Boolean inequality $L + L' \geq d + d'$ while preserving completeness.

Example 9.3 *[Hoo92] Let*

$$A + B + \overline{D} + E + \overline{F} + \overline{G} \geq 4$$
$$\overline{A} + \overline{C} + \overline{D} + \overline{E} + F + G \geq 3$$

be two extended clauses. With the enhanced resolution rule of Hooker [Hoo92] we may generate 12 different resolvents, where

$$B + \overline{C} + \overline{D} + \overline{E} + F + \overline{G} \geq 2$$

is one of them. The sum of the two extended clauses is (already normalized) $2 \cdot \overline{D} + B + \overline{C} \geq 3$. *After transformation we obtain the two extended clauses*

$$\overline{D} + \overline{C} + B \geq 2 \qquad \overline{D} \geq 1 \ ,$$

where each of the 12 resolvents is dominated by one of them.

So using the presented transformation method we generate fewer and stronger extended clauses than with the original resolution method. Note that there are no further non redundant resolution steps since resolution between the extended clauses in $\mathsf{SCP}(\mathsf{SRed}(L + L' \geq d + d'))$ are not possible since no literals occur negatively and positively in them. Additionally there is no literal occurring in $L \geq d$ or $L' \geq d'$ and negatively in an extended clause in $\mathsf{SCP}(\mathsf{SRed}(L+L' \geq d+d'))$. So the only possible resolution steps are the ones between $L \geq d$ and $L' \geq d'$. But they are all dominated by an extended clause in $\mathsf{SCP}(\mathsf{SRed}(L + L' \geq d + d'))$ because of Prop. 9.2. We just want to mention that we generate not only resolvents but also *diagonal sums* [Hoo92], generated by the other rule of "Generalized Resolution". Note that the sum of $A+B \geq 1$ and $C+D \geq 1$ is $A+B+C+D \geq 2$, which is already an extended clause. Since there is no other extended clause dominating $A+B+C+D \geq 2$ that is derivable from $A + B \geq 1$ and $C + D \geq 1$ the strongest extended clause $A + B + C + D \geq 2$ is non redundant. In fact we can show that $\{L \geq d, L' \geq d'\} \cup \mathsf{SCP}(\mathsf{SRed}(L + L' \geq d + d'))$ is prime.

9.2 0-1 Integer Programming

Extended cover inequalities of a linear 0-1 inequality, well known in 0-1 integer programming, can be related to the extended clauses generated by our transformation method. In a preprocessing phase of a linear programming based 0-1 optimization algorithm, branch-and-bound, valid extended cover inequalities can be added to the constraint set in order to strength the linear programming relaxation of the problem [NW88].

We show how to use the transformation method for generating these extended cover inequalities and how to restrict the transformation method in order to produce *strong* extended cover inequalities or extended cover inequalities over a special subset of variables. We test the method on a subset of 0-1 integer programming problems and report computational results.

9.2.1 Extended Clauses and Extended Cover Inequalities

Preprocessing 0-1 integer programming problems and applying constraint generation techniques have made large scale 0-1 problems solvable in reasonable time [CJP83, JKS85]. The wildly used branch-and-bound method for solving 0-1 integer optimization problems solves the linear programming (LP) relaxation of the problem and then branches on variables with fractional values in the current LP-solution. In a preprocessing phase valid 0-1 inequalities are generated, so called strong cuts, that better approximate the integral solution of the problem and solving the reformulated problem therefore typically require less branch and bound cycles.

The constraint set of a linear 0-1 integer programming problem consists of a set of linear 0-1 inequalities of the form

$$c_1 \cdot X_1 + \cdots + c_n \cdot X_n \leq d \tag{36}$$

which is equivalent to

$$-c_1 \cdot X_1 - \cdots - c_n \cdot X_n \geq -d$$

and can then be brought into pseudo-Boolean normal form.

An important constraint generation technique is presented in [NW88]. It is based mainly on the notion of *minimal covers* or *minimal dependent sets.*

Definition 9.4 [NW88] A set $C \subseteq \{1, \ldots, n\}$ is a minimal dependent set of a linear 0-1 inequality (36) if

$$\sum_{i \in C \setminus \{j\}} c_i \leq d < \sum_{i \in C} c_i$$

for all $j \in C$.

Proposition 9.5 *[NW88] If C is a minimal dependent set for a linear 0-1 inequality (36) then*

$$\sum_{i \in E(C)} X_i \leq |C| - 1 \ , \tag{37}$$

where

$$E(C) = C \cup \{k \mid 1 \leq k \leq n \text{ and } c_k \geq c_j \text{ for all } j \in C\}$$

is a valid inequality w.r.t. (36). The set of indices $E(C)$ is called the extension *of C and we call (37) an* extended cover inequality.

Let us write (37) in pseudo-Boolean normal form.

$$\sum_{i \in E(C)} \overline{X_i} \geq |E(C) \setminus C| + 1 \tag{38}$$

We know that

$$\sum_{i \in E(C)} \overline{X_i} \geq |E(C) \setminus C| + 2$$

is not a valid inequality, because otherwise

$$\sum_{i \in E(C)} X_i \leq |C| - 2$$

p0033	# Ineqs	# Clauses	# Nodes	PrePro	Solve	LP-Sol.	Optimum
Original	11	4	9685	0.00	59.89	2520.57	3089
Normal	11	4	1289	0.00	6.47	2819.36	3089
Simple	2	80	124	0.04	1.42	2846.00	3089
Full	0	8710	118	0.90	237.30	2846.00	3089

Figure 4: Computational Results for "p0033"

is valid which is not the case since C is minimal. Hence (38) is the strongest extended clause containing exactly the literals X_i for all $i \in E(C)$. We can therefore use our transformation procedure to systematically construct the set of all strongest non redundant extended cover inequalities.

In a preprocessing phase for 0-1 integer programming extended cover inequalities are generated and *added* to the original constraint set. Because of our equivalence result we can replace the original inequality by its set of equivalent extended clauses.

Example 9.6 *Let us look again at Example 5.1 on page 20.*

[**a**] *By writing the extended clauses as "$\leq$-inequalities" we see that the two extended cover inequalities*

$$\begin{aligned} X_1 + X_2 + X_3 + X_4 + X_5 &\leq 3 \\ X_1 + X_2 + X_3 &\leq 2 \end{aligned}$$

are equivalent to the linear 0-1 inequality

$$79 \cdot X_1 + 53 \cdot X_2 + 53 \cdot X_3 + 45 \cdot X_4 + 45 \cdot X_5 \leq 178 \ .$$

[**b**] *The single extended cover inequality*

$$X_1 + X_6 + X_7 + X_9 \leq 1$$

is equivalent to the linear 0-1 inequality

$$774 \cdot X_1 + 76 \cdot X_2 + 22 \cdot X_3 + 42 \cdot X_4 + 21 \cdot X_5 + 760 \cdot X_6 + 818 \cdot X_7 + 62 \cdot X_8 + 785 \cdot X_9 \leq 1500 \ .$$

So for Example 9.6[**a**],[**b**] a full transformation is suitable. For Example 5.1[**c**] the number of extended clauses is too large and we need to restrict the number of extended clauses we generate. A trivial, but powerful, approach is to transform only *simple* pseudo-Boolean inequalities.

For example the complete constraint set of a 0-1 integer programming problem [NW88, page465], also available in MIPLIB [BBI92] as problem "p0033", consists of 15 inequalities where 4 of them are already extended clauses. From the remaining 11 inequalities 9 are *simple*, that is they are equivalent to only a few extended clauses and just 2 inequalities produce more than 4000 extended clauses. So a naive approach is to transform the 13 simple inequalities completely and leave the 2 *hard* inequalities unchanged. We ran the commercial mixed integer solver CPLEX 2.1 on the original and the preprocessed problem. The results are reported in Fig. 4. In the columns we denote by "Ineqs" the number of inequalities with coefficients > 1 and by "Clauses" the number of extended clauses. “Nodes” is the number of nodes the branch and bound method of CPLEX needed in order to find the optimal solution and prove optimality. “PrePro” is the time in user cpu seconds on a SPARC-10 used by the Prolog implementation of the transformation method in order to preprocess the problem. “Solve” is the time in user cpu seconds CPLEX needed to solve the problem. By “LP-Sol.” we denote the optimal solution of the linear programming relaxation of the problem and by “Optimum” the integer optimal solution. The rows describe 4 different formulations of “p0033”. In “Original” we solve the original problem as found in MIPLIB. In “Normal” we solve the problem after bringing each inequality into pseudo-Boolean normal form.

Bringing an inequality into pseudo-Boolean normal form includes coefficient reduction as presented in [CJP83] and for “p0033” we obtain a speedup of factor 9. This is reflected by the better optimal solution of the linear programming relaxation of the problem and the number of linear programs solved (“Nodes”). Transforming the *simple* inequalities completely generates a larger problem (82 versus 15 constraints) but we obtain a speedup of factor 4 which is again due to a better linear programming relaxation. Note that the time consumed by the preprocessing phase is very small w.r.t. the “Solve”-time. A complete transformation of all inequalities shows to be too clumsy. The size of the problem (8710 versus 15 inequalities) slows down the linear programming solver although even fewer nodes are needed. Using constrained transformation as presented in Sect. 8.3, that is we transform each inequality completely where the context is the set of all remaining inequalities, generates only 4484 extended clauses. So when a complete transformation is required constrained transformation is advantageous. Since complete transformation of a constraint set of a 0-1 integer optimization problem is not suitable in the context of preprocessing we no longer consider constrained transformation. For preprocessing 0-1 integer programming problems we need to make our transformation method more flexible.

9.2.2 Restrictions

By restricting the number of generated extended clauses we can use the transformation method as generator of extended cover inequalities, that is strongest extended clauses, in the preprocessing phase for solving 0-1 integer optimization problems. We show how to focus on extended clauses that are sufficiently strong and restrict the number extended clauses by focusing on a set of variables.

We first give a measure of extended clauses describing their *quality* and show how to adopt the transformation method in order to obtain only extended clauses better than or equal to a given quality border. The *weakness* w of an extended clause $L \geq d$ is defined by

$$w(L \geq d) := |L| - d \ .$$

Note that $w(L \geq d) \leq w(L' \geq d')$ is a necessary condition for $L \geq d$ dominates $L' \geq d'$, which is obvious from Lemma 2.5. Ordering extended clauses w.r.t. their negative weakness gives us therefore an approximation of the domination relation. We say that $L \geq d$ is *stronger* than $L' \geq d'$ if $w(L \geq d) \leq w(L' \geq d')$.

Given a pseudo-Boolean inequality we can easily determine the minimal weakness of all extended clauses the transformation method generates.

Lemma 9.7 *Let $cL \geq d$ be a pseudo-Boolean inequality in normal form and β be such that*

$$\sum_{i=1}^{\beta} c_{n-i-1} < d \leq \sum_{i=1}^{\beta+1} c_{n-i-1} \tag{39}$$

then $\beta' = |L| - \beta - 1$ is the smallest weakness of all extended clauses generated by the transformation method.

PROOF: We know that

$$\sum_{i=1}^{n-\beta} c_i L_i \geq d - \sum_{i=1}^{\beta} c_{n-i-1}$$

is a valid strict reduction of $cL \geq d$. The right-hand side of its corresponding strongest extended clause is 1 because of (39) and therefore its weakness is $n - \beta - 1 = \beta'$. We next show that there is no extended clause with a smaller weakness. Let us recall (13) which states that the degree of the extended clauses along a path are either equal or exactly one smaller. On the other side the number of literals in the extended clauses along a path are always one smaller. So we conclude that the weakness of the extended clauses along a path are either equal or decrease. In order to determine the minimal weakness it is therefore sufficient to consider only the leaves of the graph, that is all extended clauses where the right hand side equals 1. So we need to investigate all strict reductions

$$\sum_{i \in M} c_i L_i \geq d - \sum_{i \in N \setminus M} , \tag{40}$$

where $M \subseteq N$ such that $\sum_{i \in M} c_i < d$ and $c_{\max(N \setminus M)} + \sum_{i \in M} c_i \geq d$. The weakness of these extended clauses is $|M| - 1$ and we search for an M with a minimal cardinality. Obviously $\{1, \ldots, \beta\}$ is an M with minimal cardinality since $\sum_{i=1}^{\beta} c_i \geq \sum_{i \in M} c_i$ for all M satisfying (40). □

We can easily adopt our transformation method to generating only extended clauses having a weakness less than or equal to a given bound by applying Lemma 9.7 to every strict reduction we construct and abandon further reductions if the minimal weakness is greater than the bound. So by Lemma 9.7 we can filter out the generation of extended clauses that do not have a weakness less or equal the bound. Note that we avoid unnecessary computation of branches that do not lead to extended clauses that are strong enough.

A very popular and strong class of extended clauses that can so be generated are extended clauses with weakness 1. They are also called *cliques* or *special ordered set constraints* and have proven to be useful for preprocessing 0-1 integer programming problems. With the method presented here in combination with Lemma 9.7 we can generate extended clauses having an arbitrary weakness limit and generalize so existing methods that generate only cliques.

Another possibility is to generate only extended clauses from a pseudo-Boolean inequality containing a special subset of variables. Suppose that we have a clique inequality for some variables and have no extended clauses containing the remaining variables. So we are interested in generating extended clauses where these set of remaining variables V occurs. Another interesting set of variables is the one with large coefficients in the objective function, since they are mainly responsible for the resulting objective function value.

Name	# Ineqs	# Clauses	# Nodes	PrePro	Solve	LP-Sol.	Opt.
p0033-Original	11	4	9685	0.00	59.89	2520.57	3089
p0033-Normal	11	4	1289	0.00	6.47	2819.36	3089
p0033-Simple	2	80	124	0.04	1.42	2846.00	3089
p0040-Original	23	20	96	0.00	0.55	61796.54	62027
p0040-Normal	23	20	59	0.00	0.32	61829.08	62027
p0040-Simple	0	24	0	0.03	0.05	62027.00	62027
p0201-Original	33	100	5238	0.00	167.95	6875.00	7615
p0201-Normal	33	100	2775	0.00	90.42	7125.00	7615
p0201-Simple F	30	106	2273	0.33	84.83	7125.00	7615
p0201-WL(9) F	30	112	1258	0.43	47.47	7125.00	7615
p0548-Original	89[1]	63	&	0.00	&	315.25	8691
p0548-Normal	89	63	&	0.02	&	4568.75	8691
p0548-Simple F	52	508	&	0.54	&	6617.47	8691
p0548-WL(1) CR F	52	672	&	8.10	&	7306.68	8691
p2756-Original	378[2]	352	&	0.00	&	2688.75	3124
p2756-Normal	378	352	&	2.35	&	2701.14	3124
p2756-WL(1) CR F	343	2096	&	256.50	&	2701.75	3124
lseu-Original	11	17	55526	0.00	791.60	834.68	1120
lseu-Normal	11	17	28551	0.00	465.65	944.75	1120
lseu-Simple	7	22	18032	0.00	275.02	985.54	1120

[1] originally 176 but 10 inequalities are tautologies and 14 become tautologous after fixing

[2] originally 755 but 6 inequalities are tautologies and 19 become tautologous after fixing

& out of memory

Figure 5: Some Computational Results

We obtain valid extended clauses containing a special set V of variables if we transform a strict reduction of the pseudo-Boolean inequality, where we have eliminated in advance all literals containing the variables that are not in V. Another reason to transform a strict reduction and not the whole pseudo-Boolean inequality is given if the pseudo-Boolean inequality contains some relatively large and some relatively small coefficients. The generated strongest extended clauses typically contain the literals with the largest coefficients and eliminating literals with small coefficients in advance focus on these strong extended clauses. Note that we can obtain strong extended clauses also by restricting the weakness as showed at the beginning of this section. By eliminating literals in advance we reduce the size of the search tree of the transformation method but in exchange only approximate the strongest extended clauses. The approach can be used if the pseudo-Boolean inequalities are too large to be transformed completely.

9.2.3 Computational Results

We have tested the preprocessing approach on some pure 0-1 integer programming problems found in MIPLIB [BBI92]. For the format of the figures see Sect. 9.2.1, page 44. Additionally we denote by "F" whether variables have been fixed using the `fixing` procedure presented in Sect. 7 and with "WL(w)" that for the non simple inequalities the generation of extended clauses has been limited to the maximal weakness w.

We obtain promising results on the smaller examples of the "pxxxx" problem set [CJP83] since these are sparse problems and considering each inequality on its own is promising under this premise. On dense problems we can not expect similar improvements. We denote by "CR" whether coefficient reduction, as presented in Sect. 8.1 has been successfully applied to problem. Coefficient reduction is tried on the literal with the largest coefficient in each pseudo-Boolean inequality that can not be simply transformed. The set of extended clauses that is used for the coefficient reduction is the set of extended clauses generated by the transformation of all simple pseudo-Boolean inequalities. Coefficient reduction was not very successful in the preprocessing phase[6].

The presented techniques can be used in a preprocessing phase when solving 0-1 integer programming problems. They have shown to be useful on sparse problems. Significant speedups can be obtained when the optimal solution of the linear programming relaxation of the partially transformed problem is better than in the original problem. In the context of 0-1 integer optimization we generalized the generation of clique inequalities (extended clauses with weakness 1) to the generation of extended clauses with an arbitrary weakness limit. The transformation method can also be used as a cut generator in a branch-and-bound algorithm. When searching an extended clause violating the fractional solution of the current linear programming relaxation we can extract just the needed extended clause. With the given redundancy criteria we assure that we select the best possible extended clause. Assume for example that the variables A, B have the fractional value 0.5 in the current LP-solution and that a pseudo-Boolean inequality $cL \geq d$ of the constraint set dominates the non redundant extended clause $A + B + C \geq 2$, that is $A + B + C \geq 2$ is in $\mathsf{SCP}(\mathsf{SRed}(cL \geq d))$. A naive approach is to just add the extended clause $A + B \geq 1$ which is also in $\mathsf{SCP}(\mathsf{SRed}(cL \geq d))$. Because of the redundancy criteria we choose $A + B + C \geq 2$ and so forbid further fractional solutions, among them for example $B = C = 0.5$.

Note that the main feature of the transformation method is that on the obtained set of extended clauses the deductive system "Generalized Resolution" [Hoo92] can be applied. So information coming from several pseudo-Boolean inequalities are used and may lead to stronger extended clauses. In [Bar94] we investigate the strengthening of extended clauses by using a restricted version of "Generalized Resolution". The output of the transformed and strengthened constraint set can be stored in a table and extended clauses serving as cuts can then be selected by need from the branch-and-cut algorithm. This is subject to further research.

10 Conclusion

The nice properties of extended clauses — that the domination relation is easily decidable and 0-1 solutions spaces can be represented more compactly than with classical clauses — suggests to use them as basic formulas of the constraint solver for CLP($\mathcal{PB}$), a constraint logic programming language over pseudo-Boolean constraints. We have presented a method for transforming an arbitrary linear pseudo-Boolean inequality into a minimal set of extended clauses while preserving the 0-1 solution space. Although we may generate an exponential number of extended clauses, the

[6]We feel that coefficient reduction would be more successful when applied on the pseudo-Boolean inequalities constructed while exploring the branch and bound tree of CPLEX, since then some literals are fixed and so better bounds may be obtained for the simpler pseudo-Boolean inequalities.

method generates significantly fewer extended clauses than a transformation into classical clauses. With the presented transformation method we can deal with pseudo-Boolean inequalities arising in practical applications. This has been achieved by developing specialized strong redundancy criteria that efficiently ensure that the set of generated extended clauses is minimal. The compact representation of pseudo-Boolean inequalities and extended clauses speeds up the transformation process, and so allows to process even very large pseudo-Boolean inequalities with special structure. Direct manipulation of the compact representations of extended clauses is the subject of further research. Simplification methods that take into account that pseudo-Boolean inequalities are typically part of a set of constraints are presented and the transformation method is generalized to constrained transformation.

Reformulation of a linear 0-1 inequality into a set of extended clauses is motivated by the idea of solving 0-1 problems with the deductive method "Generalized Resolution" [Hoo92] in the context of constraint logic programming. We feel that the use of extended clauses for representing 0-1 solution spaces arising from combinatorial problems is suitable. Future research consists of adapting "Generalized Resolution" for extended clauses as a constraint solver for pseudo-Boolean constraints.

References

[Bar93] P. Barth. Linear 0-1 inequalities and extended clauses. In *Logic Programming and Automated Reasoning : international conference LPAR '93; St. Petersburg, Russia; proceedings*, July 1993.

[Bar94] P. Barth. Simplifying clausal satisfiability problems. submitted, 1994.

[BB93] P. Barth and A. Bockmayr. Solving 0-1 problems in CLP($\mathcal{PB}$). In *Proc. 9th Conf. Artificial Intelligence for Applications (CAIA), Orlando*. IEEE, 1993.

[BBI92] R.E. Bixby, E.A. Boyd, and R. Indovina. MIPLIB: A Test Set of Mixed-Integer Programming Problems. *SIAM News*, 25(16), 1992.

[BM84] E. Balas and J. B. Mazzola. Nonlinear 0-1 programming: I. Linearization techniques. *Mathematical Programming*, 30:1–21, 1984.

[Boc93] A. Bockmayr. Logic programming with pseudo-Boolean constraints. In F. Benhamou and A. Colmerauer, editors, *Constraint Logic Programming - Selected Research*, chapter 18, pages 327 – 350. MIT Press, 1993.

[CD93] P. Codognet and D. Diaz. Boolean constraint solving using clp(FD). In D. Miller, editor, *Logic Programming. Proceedings of the 1993 international symposium*, 1993.

[CJP83] H. Crowder, E.L. Johnson, and M. Padberg. Solving large-scale zero-one linear programming problems. *Operations Research*, 31(5):803–834, September 1983.

[DEC93] B.L. Dietrich, L.F. Escudero, and F. Chance. Efficient reformulation for 0-1 pograms - methods and compuational results. *Discrete Applied Mathematics*, 42:147–175, 1993.

[GG80] D. Granot and F. Granot. Generalized covering relaxation for 0-1 programs. *Operations Research*, 28:1442–1450, 1980.

[GH71] F. Granot and P. L. Hammer. On the use of boolean functions in 0-1 programming. *Methods of Operations Research*, 12:154–184, 1971.

[Hoo88] J. N. Hooker. A quantitative approach to logical inference. *Decision Support Systems*, 4:45 – 69, 1988.

[Hoo89] J. N. Hooker. Input proofs and rank one cutting planes. *ORSA Journal for Computing*, 1:137 – 145, 1989.

[Hoo92] J. N. Hooker. Generalized resolution for 0-1 linear inequalities. *Annals of Mathematics and Artificial Intelligence*, 6:271–286, 1992.

[HR68] P.L. Hammer and S. Rudeanu. *Boolean Methods in Operations Research and Related Areas*. Springer-Verlag, 1968.

[Jac92] Peter Jackson. Computing prime implicates incrementally. In Deepak Kapur, editor, *11th International Conference on Automated Deduction*, pages 253–267, Saratoga Springs, NY, June 1992. Springer LNAI 607.

[JKS85] E.L. Johnson, M. Kostreva, and U.H. Suhl. Solving 0-1 integer programming problems arising from large scale planning models. *Operations Research*, 33(4):803–819, July 1985.

[JL87] J. Jaffar and J.-L. Lassez. Constraint logic programming. In *Proc. 14th ACM Symp. Principles of Programming Languages*, Munich, 1987.

[NW88] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* Series in Discrete Mathematics and Optimization. Wiley–Interscience, 1988.

[Rob65] J.A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, 1965.

[Smo94] Gert Smolka. A calculus for higher-order concurrent constraint programming with deep guards. Research Report RR-94-03, Deutsches Forschungszentrum für Künstliche Intelligenz, Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany, February 1994.

[SR90] V. A. Saraswat and M. Rinard. Concurrent constraint programming. In *17th Annual ACM Symp. Principles of Programming Languages, San Francisco*, 1990.

[VH89] P. Van Hentenryck. *Constraint satisfaction in logic programming.* MIT Press, 1989.

Index