CUT-BASED CONFLICT ANALYSIS IN MIXED INTEGER PROGRAMMING

Gioni Mexi
Zuse Institute Berlin, Germany
mexi@zib.de

Felipe Serrano
COPT GmbH, Berlin, Germany

Timo Berthold
Fair Isaac Deutschland GmbH
TU Berlin, Germany

Ambros Gleixner
HTW Berlin, Germany
Zuse Institute Berlin, Germany

Jakob Nordström
University of Copenhagen, Denmark
Lund University, Sweden

ABSTRACT

For almost two decades, mixed integer programming (MIP) solvers have used graph-based conflict analysis to learn from local infeasibilities during branch-and-bound search. In this paper, we improve MIP conflict analysis by instead using reasoning based on cuts, inspired by the development of conflict-driven solvers for pseudo-Boolean optimization. Phrased in MIP terminology, this type of conflict analysis can be understood as a sequence of linear combinations, integer roundings, and cut generation. We leverage this MIP perspective to design a new conflict analysis algorithm based on mixed integer rounding cuts, which theoretically dominates the state-of-the-art method in pseudo-Boolean optimization using Chvátal-Gomory cuts. Furthermore, we extend this cut-based conflict analysis from pure binary programs to mixed binary programs and—in limited form—to general MIP with also integer-valued variables. We perform an empirical evaluation of cut-based conflict analysis as implemented in the open-source MIP solver SCIP, testing it on a large and diverse set of MIP instances from MIPLIB 2017. Our experimental results indicate that the new algorithm improves the default performance of SCIP in terms of running time, number of nodes in the search tree, and the number of instances solved.

1 Introduction

The use of conflict analysis has a decades-old history in fields like computer-aided verification (Stallman and Sussman 1977) and Boolean satisfiability (SAT) solving (Bayardo Jr. and Schrag 1997, Marques-Silva and Sakallah 1999, Moskewicz et al. 2001). Different sets of researchers have

independently proposed different ways of incorporating such conflict analysis techniques into mixed integer programming (MIP) (Achterberg 2007a, Davey et al. 2002, Sandholm and Shields 2006). We refer the reader to (Witzig et al. 2021) for a recent overview of techniques.

When a SAT or MIP solver encounters an infeasible subproblem during search, the conflict analysis algorithm can be viewed in terms of a *conflict graph*, a directed acyclic graph that captures the sequence of (branching) decisions and (propagated) deductions that led to the infeasibility. Using this graph, a valid constraint can be inferred by identifying a subset of bound changes that separate the decisions from the node where contradiction was reached. Such learned constraints are disjunctive constraints (referred to as clauses or clausal constraints in SAT) that are implied by the original problem and are hence globally valid. SAT conflict analysis can also be described in terms of using the *resolution* proof system (Blake 1937, Davis and Putnam 1960, Davis et al. 1962, Robinson 1965) to produce syntactic derivations of learned constraints from the input formula (Beame et al. 2004). Though this perspective looks quite different, it is equivalent to the graph-based view.

Although the use of conflict analysis in so-called *conflict-driven clause learning (CDCL)* SAT solvers has been hugely successful, one drawback is that from a mathematical point of view the resolution proof system on which it is based is quite weak, and is known to require proofs of exponential length even for simple combinatorial principles (Haken 1985, Urquhart 1987). The requirement to encode the input in conjunctive normal form (CNF) as a collection of disjunctive clauses incurs a further loss in expressive power. It has therefore been studied how to lift SAT-based conflict-driven methods to richer input formats such as *(linear) pseudo-Boolean (PB) constraints*, which translated from SAT to MIP is just another name for 0–1 integer linear programs with integer coefficients. Crucially, here it turns out that the two equivalent ways of describing SAT conflict analysis discussed above generalize in different directions.

Many pseudo-Boolean solvers work by translating the input to CNF, possibly by introducing auxiliary variables, and then perform search and conflict analysis on this representation (Eén and Sörensson 2006, Martins et al. 2014, Joshi et al. 2015, Sakai and Nabeshima 2015). Except for any auxiliary variables, this works the same as the *graph-based conflict analysis* hitherto used in MIP solvers (Achterberg 2007a). Another approach, however, is to adopt the derivation-based view, but to perform the conflict analysis derivation in a proof system adapted to 0–1 linear inequalities. The natural candidate for such a proof system is *cutting planes* (Cook et al. 1987), which has been extensively studied in the area of computational complexity theory. It is a priori not obvious what it would mean to perform conflict analysis in the cutting planes proof system, but methods to do so have been designed in (Dixon and Ginsberg 2002, Chai and Kuehlmann 2005, Sheini and Sakallah 2006) and are currently used in the pseudo-Boolean solvers SAT4J (Le Berre and Parrain 2010) and ROUNDINGSAT (Elffers and Nordström 2018). As a theoretical method of reasoning, such *cut-based* conflict analysis turns out to be exponentially more powerful than the graph-based conflict analysis yielding resolution proofs.

To understand the difference between the two types of conflict analysis just discussed, it is important to note that graph-based conflict analysis does not operate on the linear constraints of the input problem, but instead of clauses extracted from these linear constraints by translating implications in the conflict graph to clausal constraints. In marked contrast, cut-based conflict analysis acts directly on the linear constraints and performs syntactic manipulations on them using derivation rules in the cutting planes proof system. When a conflict is encountered during search in the form of a violated *conflict constraint*, i.e., a constraint that detected infeasibility within the local bounds, the constraint that propagated the last bound change leading to the violation is identified. The goal is then to take a linear combination of this *reason constraint* and the conflict constraint in such a way that the variable

whose bound was propagated is eliminated, but so that the new constraint is still violated (even with the bound change for the propagated variable removed). In order for this to work, the reason constraint might need to be modified before the linear combination between the reason and conflict constraints is generated. The bound change propagated by the reason constraint might have exploited integrality, and if so a linear combination that cancels the propagated variable can become feasible. The way to deal with this in pseudo-Boolean conflict analysis is that a so-called *division* or *saturation* rule is applied on the reason constraint to make the bound propagation tight even when considered over the reals. This is referred to as *reduction* of the reason constraint in pseudo-Boolean terminology, whereas in MIP language this is nothing other than a cut applied to the reason constraint. Since the linear combination is violated by the current set of bound changes, we can repeat this process again, until we derive a constraint for which a termination criterion analogous to the *unique implication point (UIP)* notion used in graph-based conflict analysis applies. The modified reason constraints used in the conflict analysis, together with the conflict constraint and the branching decisions, form an infeasible linear program (LP) even when relaxed to real values, and the learned constraint can be viewed as a Farkas certificate for this LP relaxation.

For a detailed discussion of conflict analysis in mixed integer programming and pseudo-Boolean optimization with their commonalities and differences,, we refer the reader to Sections 1.1 and 1.2 of the conference paper (Mexi et al. 2023) preceding the present work. An in-depth discussion of SAT and pseudo-Boolean conflict analysis can be found in (Buss and Nordström 2021), and for a comprehensive description of MIP solving we refer the reader to, e.g., (Achterberg 2007b).

We remark for completeness that there is a third way of learning from local infeasibilities in MIP called *dual-proof analysis* (Witzig et al. 2021). It is conceptually different from both graph- and cut-based conflict analysis, in that (i) it can only be applied when the infeasibility has been detected via an infeasible LP relaxation, not by propagation, and (ii) the learned *dual-proof constraint* does not depend on the history of bound changes but is constructed in a single step. A notable commonality with cut-based conflict analysis is that dual-proof constraints and cut-based learned constraints are derived by aggregating a subset of the original input constraints, and so both methods operate syntactically on linear constraints. One important difference is that in cut-based conflict analysis, the set of constraints to aggregate and the multipliers to use are computed based on the propagations that have been implied, and also that cuts can be applied to the individual constraints before aggregation. In dual-proof analysis, no cuts are needed since the LP relaxation is already infeasible, and the multipliers are obtained from the dual solution for the LP relaxation. Pseudo-Boolean solving has also been combined with LP solving of relaxed versions of the input problem, and there dual-proof analysis has been combined with pseudo-Boolean conflict analysis (Devriendt et al. 2021).

1.1 Questions Studied in This Work and Our Contributions

Our focus in this work is on how cut-based conflict analysis as described above can be integrated in MIP solvers. A first step was taken in (Mexi et al. 2023) by showing how to apply this method for pure binary programs. In this paper, we go much further by applying cut-based conflict analysis in general MIP solving. We design and implement a new conflict analysis method that is guaranteed to work not only for 0–1 integer linear programs but also in the presence of continuous variables. We also extend this approach to general integer variables, but here the method is not always guaranteed to work when there are integer variable bound changes in the proper interior of the variable's domain.

An important first step is to understand pseudo-Boolean conflict analysis described in MIP terminology. We have already hinted at such a description above, but the key is to reinterpret the reason

reduction algorithm in pseudo-Boolean conflict analysis as finding a cutting plane that separates a fractional point from the feasible region defined by the reason constraint. Armed with this perspective, we consider different reduction algorithms for pure binary programs, and present a new reduction method using mixed integer rounding (MIR) cuts after selectively complementing variables. We study how different reduction algorithms compare, and prove, in particular, that our new MIR-based reduction algorithm provides a stronger reduced reason constraint than the methods used in (Elffers and Nordström 2018) as well in the conference version (Mexi et al. 2023) preceding this paper.

Our second main contribution is for mixed binary programs. We show that a naive extension of cut-based conflict analysis fails for programs with a mix of binary and real-valued variables, even if all branching decisions are made over the binary variables. We then consider a more elaborate reduction algorithm, that utilizes not just the current reason constraint to be reduced but the full history of previous propagations and their reason constraints—in particular, the reasons leading to bound changes for continuous variables—and show how this algorithm produces reduced constraints that guarantee that the conflict analysis will work also in the presence of real-valued variables (as long as no branching decisions are made over these variables).

As our final algorithmic contribution, we consider general mixed integer linear programs with integer-valued variables. Unfortunately, our approach for mixed binary programs does not extend to this setting, and we explain why this is so. Instead, we present a heuristic way of performing conflict analysis also in the presence of integer-valued variables.

We have implemented all of these reduction and conflict analysis methods in the MIP solver SCIP, and present the results of our computational study on MIP instances from MIPLIB 2017 with quantitative data for the performance of different conflict analysis algorithms and also qualitative data about properties of the learned constraints. Our experiments indicate that our proposed conflict analysis algorithm generates useful conflict constraints and improves the default performance of the MIP solver SCIP not only in terms of running time and number of nodes in the search tree, but also in terms of the number of instances solved.

1.2 Organization of This Paper

After a review of preliminaries in Section 2, we formalize our MIP interpretation of pseudo-Boolean cut-based conflict analysis in Section 3. Section 4 presents a collection of different reduction algorithms, and Section 5 studies dominance relationships between them. We extend cut-based conflict analysis to mixed binary programs in Section 6, and discuss general MIP problems with integer variables in Section 7. The results from our computational evaluation are presented in Section 8, after which we provide some concluding remarks and discuss future research directions in Section 9.

2 Preliminaries and Notation

In this section we provide the necessary notation and definitions used throughout the paper. We consider *mixed integer programs* (MIPs) of the form

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & c^\top x \\ \text{s.t.} \quad & Ax \geq b, \\ & \ell_j \leq x_j \leq u_j \quad \text{for all } j \in \{1,\dots,n\}, \\ & x_j \in \mathbb{Z} \quad \text{for all } j \in \mathcal{I}, \end{aligned} \tag{1}$$

where $m, n \in \mathbb{Z}_{\geq 0}$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, $\ell, u \in (\mathbb{R} \cup \{\pm\infty\})^n$, and $\mathcal{I} \subseteq \mathcal{N} := \{1, \ldots, n\}$. A single constraint hence takes the form $\sum_j a_{ij} x_j \geq b_i$, where $a_i^\top \in \mathbb{R}^n$ is the i-th row of the matrix A, and $b_i \in \mathbb{R}$ the corresponding entry of the vector b. For the sake of readability, we omit the index i when it is not essential to the context and define a generic constraint as $C : \sum_j a_j x_j \geq b$. By slight abuse of notation, we use C also to denote the half-space $\{x \in \mathbb{R}^n : \sum_j a_j x_j \geq b\}$ defined by the constraint C.

Depending on the set $\mathcal{I}$ and the bound vectors ℓ and u, we distinguish the following special cases of MIPs. We call (1) a *mixed binary program* (MBP) if $\ell_j = 0$ and $u_j = 1$ for all $j \in \mathcal{I}$. If additionally $\mathcal{I} = \{1, \ldots, n\}$, then we call (1) a *pure binary program* (BP).

2.1 States and Activities

When processing a node in the branch-and-bound tree, we may encounter infeasibilities. To apply conflict analysis we need information on all bound changes of variables that were applied by branching or propagation between the root node and the current node. At each node p, let $q = 0, 1, 2, \ldots$ index the sequence of bound changes on some variable at this node. By $\rho_{p,q}$ we then refer to the entire state of the problem after applying all bound changes on the unique path from the root node to node p, including bound changes $0, 1, \ldots, q$ at node p. At a node p, the first state $\rho_{p,0}$ always represents the problem state after the single branching decision taken at this node. Between two subsequent states $\rho_{p,q}$ and $\rho_{p,q+1}$, only one bound change on a single variable is recorded, and therefore either the lower or the upper bound vector differs in exactly one entry.

In conflict analysis, we typically only consider the states on the unique path from the root node to the current node. In this case, we can use the so-called *decision level* as node index p, which gives the number of branching decisions up to and including the current state. Then the states on a single path can be ordered lexicographically by their indices via

$$(p, q) \preccurlyeq (p', q') :\Leftrightarrow p < p' \vee (p = p' \wedge q \leq q').$$

In the following, we often omit the node index p and the bound change index q if the specific positions are not crucial for the argument we are trying to convey. For each state ρ, we can query several relevant pieces of information such as

- the vector $\ell^\rho \in (\mathbb{R} \cup \{-\infty\})^n$ of *local lower bounds* of all variables,
- the vector $u^\rho \in (\mathbb{R} \cup \{\infty\})^n$ of *local upper bounds* of all variables,
- the *variable index* $\text{varidx}(\rho) \in \mathcal{N}$ of the corresponding bound change, and
- the *reason constraint* $\text{reason}(\rho)$ from which the bound change was derived.

For a generic constraint $C : \sum_j a_j x_j \geq b$, we denote the *maximal activity* of C under state ρ as $\text{act}^{\max}(C, \rho) := \sum_j \max\{a_j \cdot \ell_j^\rho, a_j \cdot u_j^\rho\}$. Similarly, we define the *minimal activity* of C under ρ as $\text{act}^{\min}(C, \rho) := \sum_j \min\{a_j \cdot \ell_j^\rho, a_j \cdot u_j^\rho\}$. We call the constraint C *infeasible* under some state if its maximal activity is less than the right-hand side, i.e., if the respective inequality cannot hold for any solution within the local bounds. Otherwise, we call C *feasible*. We are particularly interested in the fact whether the bound changes on a variable affect the current maximal activity of a constraint. We call a variable x_j *relaxable* for constraint C under state ρ if $\text{act}^{\max}(C, \rho)$ remains unchanged when we replace its local bounds ℓ_j^ρ and u_j^ρ by its global bounds ℓ_j and u_j, respectively. By definition, this is the case if and only if $a_j = 0$ or $a_j > 0 \wedge u_j^\rho = u_j$ or $a_j < 0 \wedge \ell_j^\rho = \ell_j$. Otherwise, the variable is called *non-relaxable*. Note that these terms provide a generalization from the PB to the MIP setting.

In the literature on pseudo-Boolean optimization, where there are only binary variables, relaxable and non-relaxable variables correspond to non-falsified and falsified literals, respectively.

2.2 Propagation of Linear Constraints

Beyond identifying trivial infeasibility, the maximal activity of a constraint under some state ρ can also be used to identify whether variable bounds can be tightened. A commonly used propagator for linear constraints is the bound strengthening technique going back to (Brearley et al. 1975). If $\text{act}^{\max}(C, \rho) < \infty$, then we can rewrite constraint C as

$$a_r x_r \geq b - \sum_{j \neq r} a_j x_j \geq b - \max_{x \in [\ell^\rho, u^\rho]} \sum_{j \neq r} a_j x_j \tag{2}$$

For $a_r > 0$ we obtain

$$a_r x_r \geq b - \text{act}^{\max}(C, \rho) + a_r u_r^\rho$$

and dividing by a_r gives

$$x_r \geq u_r^\rho + \frac{b - \text{act}^{\max}(C, \rho)}{a_r} =: \tilde{\ell}_r, \tag{3}$$

which can be used to tighten the lower bound of x_r if $\tilde{\ell}_r > \ell_r^\rho$. In a similar fashion, if $a_r < 0$, we can deduce the upper bound $x_r \leq \tilde{u}_r := \ell_r^\rho + \frac{b - \text{act}^{\max}(C,\rho)}{a_r}$ if $\tilde{u}_r < u_r^\rho$. For completeness, note that if $\text{act}^{\max}(C, \rho) = \infty$ because $a_r > 0 \wedge u_r^\rho = \infty$ or $a_r < 0 \wedge \ell_r^\rho = -\infty$, but $\max_{x \in [\ell^\rho, u^\rho]} \sum_{j \neq r} a_j x_j < \infty$, then (2) can be used directly to derive valid bounds.

For integer variables, i.e., for $r \in \mathcal{I}$, the derived bounds can be rounded to $\lceil \tilde{\ell}_r \rceil$ and $\lfloor \tilde{u}_r \rfloor$, respectively. As will become clear in Section 3, we are particularly interested in propagations where this second step is redundant because either the variable x_r is continuous or the propagated bounds are already integral, i.e., $\tilde{\ell}_r \in \mathbb{Z}$ and $\tilde{u}_r \in \mathbb{Z}$, respectively. We refer to such propagations as *tight*.

2.3 Coefficient Tightening and Cutting Planes

The following techniques from the PB and MIP literature to strengthen linear inequalities are key ingredients of our conflict analysis algorithms. *Coefficient tightening* (Brearley et al. 1975) seeks to tighten the coefficients of integer variables in C to derive a more restrictive constraint while preserving all feasible integer solutions.

Definition 1 (Coefficient Tightening)**.** *Let* $C : \sum_{j \in J \cup K} a_j x_j \geq b$*,* $x \in \mathbb{Z}^J_{\geq 0} \times \mathbb{R}^K_{\geq 0}$*, where* $J \subseteq \mathcal{I}, K \subseteq \mathcal{N} \setminus \mathcal{I}$*, and* $a_j > 0$ *for all* $j \in J$*. Further, let* $\text{act}^{min}(C)$ *denote the minimal activity of the constraint under the global bounds. Then* coefficient tightening *yields the constraint*

$$\sum_{j \in J} \min\{a_j, \tilde{b}\} x_j + \sum_{k \in K} a_k x_k \geq b - \sum_{j \in J} \max\{0, a_j - \tilde{b}\} \ell_j, \tag{4}$$

where $\tilde{b} = b - \text{act}^{min}(C)$*. The general case for* $a_j \in \mathbb{R} \setminus \{0\}, j \in J$ *is analogous.*

Another well known cut from the IP literature which is already used in the context of pseudo-Boolean conflict analysis is the *Chvátal-Gomory cut* (Chvátal 1973).

Definition 2 (Chvátal-Gomory Cut)**.** *Let* $C : \sum_{j\in J} a_j x_j \geq b$ *with* $x \in \mathbb{Z}^J_{\geq 0}$*,* $J \subseteq \mathcal{I}$*. The* Chvátal-Gomory (CG) Cut *of* C *is given by the constraint*

$$\sum_{j\in J} \lceil a_j \rceil \, x_j \geq \lceil b \rceil \, . \tag{5}$$

To see why (5) is valid for $\{x \in \mathbb{Z}^J_{\geq 0} \, : \, \sum_{j\in J} a_j x_j \geq b\}$, we can think of it as two steps: rounding up coefficients on the left-hand side relaxes the constraint and is hence valid; the validity of rounding up the right-hand side follows from the integrality of the left-hand side.

Next, we recall the *Mixed Integer Rounding (MIR) cut* (Marchand and Wolsey 2001).

Definition 3 (Mixed Integer Rounding Cut)**.** *Let* $C : \sum_{j\in J\cup K} a_j x_j \geq b$ *with* $x \in \mathbb{Z}^J_{\geq 0} \times \mathbb{R}^K_{\geq 0}$*, where* $J \subseteq \mathcal{I}, K \subseteq \mathcal{N} \setminus \mathcal{I}$*. The* Mixed Integer Rounding (MIR) Cut *of* C *is given by*

$$\sum_{\substack{j\in K:\\ a_j>0}} \frac{a_j}{f(b)} x_j + \sum_{j\in J} \left(\lfloor a_j \rfloor + \min \left\{ 1, \frac{f(a_j)}{f(b)} \right\} \right) x_j \geq \lceil b \rceil \, , \tag{6}$$

where $f(r) = r - \lfloor r \rfloor$ *is the fractional part of* r*.*

The proof that (6) is valid for $\{x \in \mathbb{Z}^J_{\geq 0} \times \mathbb{R}^K_{\geq 0} \, : \, \sum_j a_j x_j \geq b\}$ can found in (Marchand and Wolsey 2001).

2.4 Weakening and Complementation

Two operations on constraints that are used in later sections are *weakening* and *complementation* of variables. Weakening is a valid operation since it simply adds a multiple of the globally valid bound constraints $x_j \leq u_j \Leftrightarrow -x_j \geq -u_j$, or $x_j \geq \ell_j$ to C. Weakening a variable x_s in a constraint $C : \sum_j a_j x_j \geq b$ is defined as: weaken$(C, \, x_s) := \sum_{j\neq s} a_j x_j \geq b - \max\{a_s u_s, a_s \ell_s\}$. Weakening entails a loss of information. However, as we will see, it is a necessary operation in some reduction algorithms. Note that whenever weakening is applied on relaxable variables at the current state, it does not change the bound propagations of the remaining variables in the constraint.

The second operation that we use in the reduction is complementation of variables. Complementation is a valid operation that does not entail a loss of information since it simply replaces a variable x_s by its complement $\overline{x}_s = u_s - x_s$. Complementing a variable x_s in the general constraint C from above is defined as: complement$(C, \, x_s) := \sum_{j\neq s} a_j x_j - a_s \overline{x}_s \geq b - a_s u_s$.

3 The Simple Case: Conflict Analysis under Tight Propagations

Conflict analysis is applicable whenever a locally infeasible constraint is detected at a node of the branch-and-bound search. In MIP solvers, this constraint may be detected directly during propagation or as trivially infeasible aggregation of constraints resulting from an infeasible LP relaxation (Achterberg 2007a). This can occur after many propagations on the current node, and the main goal of conflict analysis is to obtain a constraint that would have identified the infeasibility earlier. Such a constraint would have propagated in an earlier node, preventing us from visiting this infeasible node in the first place.The technique for constructing such a constraint is as follows.

Given the constraint that found the infeasibility, called *conflict constraint*, and the state history, we can find the last constraint that propagated a variable at a state ρ which contributed to the infeasibility of the conflict constraint, called *reason constraint*. The conflict and reason constraints form a system

of *two* constraints that cannot be satisfied simultaneously at a state $\tilde{\rho}$ before ρ. Now, the goal is to obtain a *single* globally valid constraint that is infeasible under $\tilde{\rho}$. If we have a general recipe to achieve this, this process can be repeated, using the learned constraint as the new conflict constraint. A common criterion is to iterate this step until a so-called *first unique implication point* (FUIP) is reached, i.e., until the learned constraint would propagate some variable at a state in the previous decision level. Note that the learned constraint may also be infeasible under the global bounds, in which case we have proven that the problem is globally infeasible. To summarize, the fundamental problem of conflict analysis is: Given two constraints C_{confl} and C_{reason}, a local domain L, and a global domain G, such that the system $\{C_{\text{confl}}, C_{\text{reason}}, L\}$ is infeasible, find a constraint C_{learn} that is valid for $\{C_{\text{confl}}, C_{\text{reason}}, G\}$ such that $\{C_{\text{learn}}, L\}$ is infeasible.

In SAT, this fundamental problem is easy to solve. Indeed, constraints in SAT are of the form $\sum_{i\in J_0} x_i + \sum_{j\in J_1} \bar{x}_j \geq 1$, where $\bar{x}_j$ is the negation of x_j, i.e., $\bar{x}_j = 1 - x_j$. The only way such a constraint can propagate is that all variables but one are fixed to 0. Without loss of generality, let us assume that x_r is the variable propagated by $C_{\text{reason}} : x_r + \sum_{i\in J_0} x_i + \sum_{j\in J_1} \bar{x}_j \geq 1$ in the local domain L. Then, in L it must be that $x_i = 0$ for all $i \in J_0$ and $x_j = 1$ for all $j \in J_1$. Furthermore, since C_{confl} is infeasible after the bound tightening $x_r = 1$, it must be of the form $\bar{x}_r + \sum_{i\in J'_0} x_i + \sum_{j\in J'_1} \bar{x}_j \geq 1$ with $x_i = 0$ for all $i \in J'_0$ and $x_j = 1$ for all $j \in J'_1$ in L. The learned constraint C_{learn} is then obtained by the so-called *resolution rule*, which is nothing else than taking the sum of both constraints,

$$\sum_{i\in J_0} x_i + \sum_{i\in J'_0} x_i + \sum_{j\in J_1} \bar{x}_j + \sum_{j\in J'_1} \bar{x}_j \geq 1. \tag{7}$$

and applying coefficient tightening to (7) which yields

$$\sum_{i\in J_0\cup J'_0} x_i + \sum_{j\in J_1\cup J'_1} \bar{x}_j \geq 1. \tag{8}$$

This constraint is clearly valid in any global domain G and infeasible in L and also corresponds to the constraint obtained by resolution in graph-based conflict analysis (Achterberg 2007a). Note that the above argument didn't use integrality information.

Remark 1. *We can interpret the above technique as trying to eliminate x_r from the system $\{C_{\text{confl}}, C_{\text{reason}}, L\}$ by Fourier-Motzkin elimination. Indeed, the system is still infeasible even when we relax the domain of x_r to $(-\infty, \infty)$. Then, by Fourier-Motzkin elimination (Williams 1976), projecting out x_r yields exactly* (7) *Note that this immediately proves the infeasibility of* (7) *in L, since the projection of the empty set is the empty set.*

The idea above can be generalized to linear constraints. The linear combination of the constraints that eliminates a given variable followed by coefficient tightening is what is known as *generalized resolution* (Hooker 1988, 1992). In the context of conflict analysis, the learned constraint is also called the *resolvent*. As the following example shows, the naive application of generalized resolution does not necessarily guarantee that the resolvent is infeasible in the local domain.

Example 1. *Consider the two pure binary constraints $C_r : x_1 + x_2 + 2x_3 \geq 2$, $C_c : x_1 - 2x_3 + x_4 + x_5 \geq 1$, $L = \{0\} \times \{0, 1\}^4$, and $G = \{0, 1\}^5$. Under L, constraint C_r propagates $x_3 \geq 0.5$. Since x_3 is binary, we conclude that $x_3 \geq 1$. Then, C_c is clearly infeasible. Eliminating x_3, by adding the two constraints and applying coefficient tightening, we obtain the constraint $2x_1 + x_2 + x_4 + x_5 \geq 3$ which is globally valid, but not infeasible under L.*

A notable difference between the SAT case and Example 1 is that in Example 1 we had to use integrality information because C_{reason} did not propagate x_3 *tightly*, i.e., to an integer value. Indeed, this is the only reason why generalized resolution can fail. The following proposition shows that whenever no integrality information is used during the propagation of C_{reason}, the resolvent of C_{reason} and C_{confl} that eliminates x_r remains infeasible.

Proposition 1. *Let $C_{\text{reason}} : a_r x_r + \sum_{j \neq r} a_j x_j \geq b$ be a constraint propagating a variable x_r in the state ρ tightly. Further, assume that $C_{\text{confl}} : a'_r x_r + \sum_{j \neq r} a'_j x_j \geq b'$ becomes infeasible in the state ρ. Then the resolvent of C_{reason} and C_{confl} that eliminates x_r remains infeasible in the state ρ.*

Proof. Since a_r and a'_r are both non-zero and have opposite signs, without loss of generality, we assume that $a_r = 1$ and $a'_r = -1$. Then, the constraint C_{reason} propagates a lower bound β on the variable x_r, namely

$$x_r \geq \min_{x \in [\ell^\rho, u^\rho]} \{b - \sum_{j \neq r} a_j x_j\} = b - \max_{x \in [\ell^\rho, u^\rho]} \sum_{j \neq r} a_j x_j =: \beta.$$

Now, since C_{confl} is infeasible in the state ρ, we have that

$$-\beta + \max_{x \in [\ell^\rho, u^\rho]} \sum_{j \neq r} a'_j x_j < b'$$

The right-hand side of the resolvent $C_{\text{res}} := C_{\text{reason}} + C_{\text{confl}}$ is $b + b'$. However, its maximum activity is

$$\begin{aligned} \max_{x \in [\ell^\rho, u^\rho]} \sum_{j \neq r} (a_j + a'_j) x_j &\leq \max_{x \in [\ell^\rho, u^\rho]} \sum_{j \neq r} a_j x_j + \max_{x \in [\ell^\rho, u^\rho]} \sum_{j \neq r} a'_j x_j \\ &= b - \beta + \max_{x \in [\ell^\rho, u^\rho]} \sum_{j \neq r} a'_j x_j < b + b', \end{aligned}$$

which shows that C_{res} is infeasible in the state ρ. □

To obtain a less syntactic, more geometric perspective, let us view the lower bound change on a variable x_r from propagation of $C_{\text{reason}} : a_r x_r + \sum_{j \neq r} a_j x_j \geq b$, $a_r > 0$ as the solution of the one-row linear program

$$\min\{x_r \,:\, a_r x_r + \sum_{j \neq r} a_j x_j \geq b,\, x_j \in [\ell_j^\rho, u_j^\rho]\}. \tag{9}$$

Non-tight propagation can occur if and only if the optimum of (9) over the current state is attained at a non-integer vertex, as illustrated in Figure 1 for the reason constraint of Example 1. By contrast, for SAT constraints the feasible region defined by the reason constraint and the global domain does not contain non-integer vertices. This geometric perspective also shows a clear path forward to make conflict analysis work: by strengthening the propagating reason constraint C_{reason} in order to cut off the non-integer vertex from the feasible region $C_{\text{reason}} \cap G$. Note that for pure binary programs the reason is a knapsack constraint, and hence it is sufficient to study cuts for the knapsack polytope (Hojny et al. 2020). In the PB literature, the techniques to achieve such cuts are called *reduction techniques*. In the next section, we first present the cut-based conflict analysis algorithm and then proceed to discuss various reduction techniques.

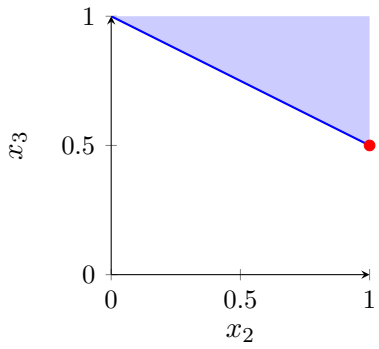

Figure 1: Inequality $x_1 + x_2 + 2x_3 \geq 2$ on the face of the polytope with $x_1 = 0$. The fractional vertex is $(0, 1, 0.5)$.

4 Reduction Techniques for Pure Binary Programs

Algorithm 1 shows the base algorithm for all variants of cut-based conflict analysis considered in this paper. The algorithm is initialized with an infeasible state $\rho_{p,q}$ and a conflicting constraint C_{confl} in $\rho_{p,q}$. First, the learned conflict constraint C_{learn} is set to the conflict constraint C_{confl}. In each iteration, the state $\rho_{p,q}$ is set to the smallest state, with respect to the lexicographic order, such that the learned conflict constraint is infeasible, and we extract the variable x_r whose bound was changed in $\rho_{p,q}$. If the bound change was due to propagation of a constraint, then we extract the reason constraint C_{reason} that propagated x_r. In line 7 we "reduce" the reason constraint such that the resolvent, i.e., linear combination, of C_{learn} and the reduced reason C_{reason} (Line 8) that cancels x_r remains infeasible. The learned conflict constraint is set to the resolvent, which can then be strengthened in Line 9 by, e.g., applying coefficient tightening as in SAT resolution conflict analysis. We continue this process until we reach an FUIP (C_{learn} is *asserting*) or C_{learn} proves global infeasibility. The conflict

Algorithm 1: Cut-Based Conflict Analysis for 0-1 IP

Input : initial conflict constraint C_{confl}, infeasible state $\rho_{p,q}$
Output : learned conflict constraint C_{learn}

1 $C_{\text{learn}} \leftarrow C_{\text{confl}}$
2 **while** C_{learn} not asserting **and** $C_{\text{learn}} \neq \bot$ **do**
3 $\quad (p, q) \leftarrow \min\{(\tilde{p}, \tilde{q}) \mid C_{\text{learn}} \text{ is infeasible in } \rho_{\tilde{p},\tilde{q}}\}$
4 $\quad r \leftarrow \text{varidx}(\rho_{p,q})$
5 $\quad$ **if** x_r propagated **then**
6 $\quad\quad C_{\text{reason}} \leftarrow \text{reason}(\rho_{p,q})$
7 $\quad\quad C_{\text{reason}} \leftarrow \text{reduce}(C_{\text{reason}}, C_{\text{learn}}, \rho_{p,q})$
8 $\quad\quad C_{\text{learn}} \leftarrow \text{resolve}(C_{\text{learn}}, C_{\text{reason}}, x_r)$
9 $\quad\quad C_{\text{learn}} \leftarrow \text{strengthen}(C_{\text{learn}})$
10 **return** C_{learn}

analysis algorithm can fail if the resolvent becomes feasible. As explained in the previous section, this can occur only if the reason constraint C_{reason} does not propagate tightly. In the following, we present various reduction algorithms for the reason constraint C_{reason} that ensure that this constraint propagates tightly enough: the SAT-like *clausal-based reduction* Achterberg (2007a), which reduces the reason constraint to a clause responsible for the propagation; the *coefficient tightening-based reduction*, which is a generalization of (Chai and Kuehlmann 2005); and a stronger version of the *MIR-based reduction* presented in (Mexi et al. 2023).

As we mentioned at the end of last section, the goal of reduction algorithms is to obtain a constraint that propagates a variable tightly from a reason constraint that did not propagate tightly. Therefore, reduction algorithms only look at a single given constraint and the current domain in a state ρ. Hence, in this section we work under the following assumption.

Assumption 1. *The reason constraint is of the form* $C_{\text{reason}} : x_r + \sum_{j \in J} a_j x_j \geq b$, *where* $r \notin J \subseteq \mathcal{I}$ *and* $a_j \geq 0$ *for all* $j \in J$. *Furthermore,* C_{reason} *propagates* x_r *in* ρ *non-tightly, i.e.,* $b - \max \sum_{j \in J} a_j x_j \notin \mathbb{Z}$.

This assumption can be made without loss of generality. Indeed, from a general reason constraint we can first complement any variable with a negative coefficient. Afterwards, we can divide by the coefficient of the variable x_r to obtain the aforementioned form. Obviously, the information in both constraints is exactly the same.

4.1 Coefficient Tightening-based Reduction

Algorithm 2 summarizes the coefficient tightening-based reduction. Similar to the implementation in a PB solver (Chai and Kuehlmann 2005), in each iteration, the algorithm picks a relaxable variable in the reason constraint different from the variable we are resolving on and weakens it. Then it applies coefficient tightening to the resulting constraint. After each iteration, the algorithm checks if the resolvent of the reason constraint and conflict constraint is infeasible. In this case, the algorithm terminates and returns the reduced reason constraint. A proof that the reduction algorithm produces a reason constraint that propagates strongly enough, in that the reduced reason is guaranteed to propagate tightly after all relaxable variables have been weakened, can be found in (Mexi et al. 2023).

Algorithm 2: Coefficient Tightening-based Reduction Algorithm

Input : conflict constraint C_{confl}, reason constraint C_{reason}, variable to resolve x_r, state ρ
Output : reduced reason C_{reason}

while resolve$(C_{\text{reason}}, C_{\text{confl}}, x_r)$ *is feasible in* ρ **do**
  $x_s \leftarrow$ relaxable variable in $C_{\text{reason}} \backslash \{x_r\}$
  $C_{\text{reason}} \leftarrow$ weaken(C_{reason}, x_s)
  $C_{\text{reason}} \leftarrow$ coefTight(C_{reason})
return C_{reason}

4.2 cMIR-based Reduction

For pure binary constraints, a very competitive alternative to coefficient tightening in the reduction algorithm is based on Chvátal-Gomory cuts (Elffers and Nordström 2018). In this reduction, relaxable variables with fractional coefficients in the reason constraint are weakened before applying Chvátal-Gomory rounding as in (2). In (Mexi et al. 2023) we show that applying the more general MIR cut instead yields a reduced reason constraint that is at least as strong.

Next, we present a further improved reduction technique also based on the MIR formula (3). The improvement comes from the fact that weakening becomes obsolete after complementing relaxable variables. We call this reduction *cMIR-based reduction*. In Section 5 we show one of the main results of this paper, which is the dominance of cMIR-based reduction over both Chvátal-Gomory and MIR-based reduction from (Mexi et al. 2023). But first, we show that the reduced reason constraint from the cMIR-based reduction propagates x_r tightly.

Proposition 2. *Let C_{reason} be as in Assumption 1. Complementing all variables in $P = \{j \in J : u_j^\rho = 1\}$ and applying MIR gives the reduced reason constraint*

$$C_{\text{cMIR}} : x_r + \sum_{j \notin P} \psi(a_j) x_j - \sum_{j \in P} \psi(-a_j) x_j \geq 1 - \sum_{j \in P} \psi(-a_j) \tag{10}$$

with

$$\psi(a) = \lfloor a \rfloor + \min \left\{ 1, \frac{f(a)}{f(b - \sum_{j \in P} a_j)} \right\}.$$

C_{cMIR} propagates x_r tightly.

Proof. Complementing the variables in P yields

$$x_r + \sum_{j \notin P} a_j x_j - \sum_{j \in P} a_j \bar{x}_j \geq b - \sum_{j \in P} a_j =: \tilde{b}. \tag{11}$$

Note that $0 < \tilde{b} < 1$ by Assumption 1Indeed, since C_{reason} propagates non-tightly, we have that $\min\{b - \sum_{j \in J} a_j x_j\}$ is between 0 and 1. After applying MIR to the complemented constraint we obtain

$$C_{\text{cMIR}} : x_r + \sum_{j \notin P} \psi(a_j) x_j + \sum_{j \in P} \psi(-a_j) \bar{x}_j \geq 1 \tag{12}$$

where

$$\psi(a) = \lfloor a \rfloor + \min \left\{ 1, \frac{f(a)}{f(\tilde{b})} \right\}.$$

Complementing back gives us (9). Finally, we can use the fact that for $j \notin P$, x_j is fixed to 0 and that $\psi(-a_j) \leq 0$ to show that C_{cMIR} propagates x_r tightly. The propagated bound is

$$\begin{aligned} &1 - \sum_{j \in P} \psi(-a_j) - \max \Big\{ \sum_{j \notin P} \psi(a_j) x_j - \sum_{j \in P} \psi(-a_j) x_j \Big\} \\ &\quad = 1 - \sum_{j \in P} \psi(-a_j) - \Big(\sum_{j \notin P} \psi(a_j) \cdot 0 - \sum_{j \in P} \psi(-a_j) \cdot 1 \Big) = 1. \end{aligned}$$

□

Remark 2. *The reduced constraint C_{cMIR} is precisely the Gomory mixed integer cut (Gomory 1960) generated from the optimal tableau obtained by solving the one-row LP* (9)*.*

5 Dominance Relationships

A natural goal is to find a reduction technique that yields the strongest possible reduced constraint to use in the resolution step of conflict analysis. In this section, we discuss dominance relationships between the different reduction techniques in the following sense.

Definition 4. *A constraint C'' **dominates** a constraint C' if any $\tilde{x} \in [\ell, u]$ that satisfies C'' also satisfies C'. In other words, the set of feasible points defined by the variables bounds and C'' is a subset of the set of feasible points defined by the variables bounds and C'.*

The main goal of this section is to compare the cMIR-based reduction from Section 4.2 with the wMIR-based reduction from (Mexi et al. 2023), which we recall next. Let $C_{\text{reason}} : x_r + \sum_j a_j x_j \geq b$

be as in Assumption 1, propagating x_r non-tightly. Then the reduced constraint

$$C_{\text{wMIR}} : x_r + \sum_{j \in P_Z} a_j x_j + \sum_{j \notin P} \psi_w(a_j) x_j \geq \left\lceil b - \sum_{j \in P_W} a_j \right\rceil \tag{13}$$

propagates x_r tightly, where $P = \{j : u_j = 1\}$, $P_W = \{j \in P : a_j \notin \mathbb{Z}\}$, $P_Z = \{j \in P : a_j \in \mathbb{Z}\}$, and

$$\psi_w(a) = \lfloor a \rfloor + \min \left\{ 1, \frac{f(a)}{f(b - \sum_{j \in P_W} a_j)} \right\}.$$

The above formula comes from weakening the variables in P_W and then applying the MIR cut to the resulting constraint. In what follows we show that this is not the strongest possible reduction.

Proposition 3. *Let ρ be the current state and $C_{\text{reason}} : x_r + \sum_{j \in J} a_j x_j \geq b$ be as in Assumption 1, then C_{cMIR} (10) dominates C_{wMIR} (13).*

Proof. Let $P = \{j \in J : u_j^\rho = 1\}$, $P_W = \{j \in P : a_j \notin \mathbb{Z}\}$, and $P_Z = \{j \in P : a_j \in \mathbb{Z}\}$. The constraint C_{cMIR} is given by

$$x_r + \sum_{j \notin P} \psi_c(a_j) x_j - \sum_{j \in P} \psi_c(-a_j) x_j \geq 1 - \sum_{j \in P} \psi_c(-a_j),$$

where

$$\psi_c(a) = \lfloor a \rfloor + \min \left\{ 1, \frac{f(a)}{f(b - \sum_{j \in P} a_j)} \right\}.$$

This can be rewritten as

$$x_r + \sum_{j \notin P} \psi_c(a_j) x_j - \sum_{j \in P_Z} \psi_c(-a_j) x_j - \sum_{j \in P_W} \psi_c(-a_j) x_j \geq 1 - \sum_{j \in P} \psi_c(-a_j)$$

For $j \in P_Z$, we have $\psi_c(-a_j) = -a_j$. Therefore, we can rewrite C_{cMIR} as

$$x_r + \sum_{j \notin P} \psi_c(a_j) x_j + \sum_{j \in P_Z} a_j x_j - \sum_{j \in P_W} \psi_c(-a_j) x_j \geq 1 - \sum_{j \in P_W} \psi_c(-a_j) + \sum_{j \in P_Z} a_j.$$

This is equivalent to

$$x_r + \sum_{j \notin P} \psi_c(a_j) x_j + \sum_{j \in P_Z} a_j x_j + \sum_{j \in P_W} \psi_c(-a_j) \bar{x}_j \geq 1 + \sum_{j \in P_Z} a_j.$$

Given that for $j \in P_W, \psi_c(-a_j) \leq 0$, the constraint

$$x_r + \sum_{j \notin P} \psi_c(a_j) x_j + \sum_{j \in P_Z} a_j x_j \geq 1 + \sum_{j \in P_Z} a_j, \tag{14}$$

is dominated by C_{cMIR}. Hence, it suffices to show that C_{wMIR} is equivalent to (14). To this end, notice that C_{wMIR} is given by

$$x_r + \sum_{j \in P_Z} a_j x_j + \sum_{j \notin P} \psi_w(a_j) x_j \geq \left\lceil b - \sum_{j \in P_W} a_j \right\rceil$$

with

$$\psi_w(a) = \lfloor a \rfloor + \min\left\{1, \frac{f(a)}{f(b - \sum_{j \in P_W} a_j)}\right\}.$$

However, the fractional parts of $b - \sum_{j \in P_W} a_j$ and $b - \sum_{j \in P} a_j$ are equal because their difference $\sum_{j \in P_Z} a_j \in \mathbb{Z}$. That is, $f(b - \sum_{j \in P_W} a_j) = f(b - \sum_{j \in P} a_j)$. This implies that $\psi_w = \psi_c$. Finally,

$$\left\lceil b - \sum_{j \in P_W} a_j \right\rceil = \left\lceil b - \sum_{j \in P} a_j + \sum_{j \in P_Z} a_j \right\rceil = \left\lceil b - \sum_{j \in P} a_j \right\rceil + \sum_{j \in P_Z} a_j = 1 + \sum_{j \in P_Z} a_j.$$

Thus, C_{wMIR} is equivalent to (14) and, therefore, dominated by C_{cMIR}. □

For completeness, we recall that the wMIR-based reduction dominates the Chvátal-Gomory-based reduction as described in (Mexi et al. 2023). Hence, Proposition 3 shows that the cMIR-based reduction also dominates the Chvátal-Gomory-based reduction. From Gocht et al. (2019) it follows that the Chvátal-Gomory and coefficient tightening reduction algorithms are incomparable. Similar arguments can be used to show the same result for the MIR reduction from Proposition 2 and the coefficient tightening reduction from Algorithm 2. Details can be found in (Mexi et al. 2023, Gocht et al. 2019).

6 Conflict Analysis for Mixed Binary Programs

Our next goal is to extend cut-based conflict analysis to the class of mixed binary programs. In Section 4 we have seen that for pure binary programs it is always possible to reduce the reason constraint to guarantee that in each iteration of conflict analysis the linear combination of the current conflict constraint and the reduced reason constraint remains infeasible under the local bounds, and hence the conflict analysis invariant is preserved. The key property of the reduced reason constraint is that it can be made to propagate the resolved variable tightly even when considered over the reals. In the case of MBPs, continuous variables are always propagated tightly. Hence, from Proposition 1 it follows that whenever resolving a continuous variable, the linear combination of the current conflict constraint and the reason constraint that propagated the variable we are resolving on remains infeasible under the local bounds.

However, as shown in detail in the following example, the mere presence of non-relaxable continuous variables can lead to a situation where resolving a binary variable x_r is impossible without using additional problem information.

Example 2. *Consider the following system of constraints:*

$$C_1 : -2x_1 - 4y_1 - 2y_2 \geq -3, \quad C_2 : 20x_1 + 5y_1 - y_2 \geq 4, \quad C_3 : -20x_1 + 5y_1 - 10y_2 \geq -16,$$
$$C_4 : -y_2 - x_2 \geq 0, \quad C_5 : y_2 - x_3 \geq 0$$

where x_1, x_2 and x_3 are binary variables, y_1 is a continuous variable with global bounds $[0, 1]$, and y_2 is a continuous variable with global bounds $[-1, 1]$. After branching on x_2, in the subproblem $x_2 = 0$ we can deduce $y_2 \leq 0$ from C_4. Next, from C_5 we can deduce $x_3 = 0$ and $y_2 \geq 0$. The constraint C_1 propagates $y_1 \leq 3/4$ which leads to C_2 propagating $x_1 \geq 1$. Under the local bounds, the constraint C_3 is infeasible. Eliminating x_1 with C_2 being the reason constraint and C_3 the conflict, we obtain $C_2 + C_3 : 10y_1 - 11y_2 \geq -12$ which is not infeasible under the local bounds. Next we show that it is not possible to find any globally valid inequality that is

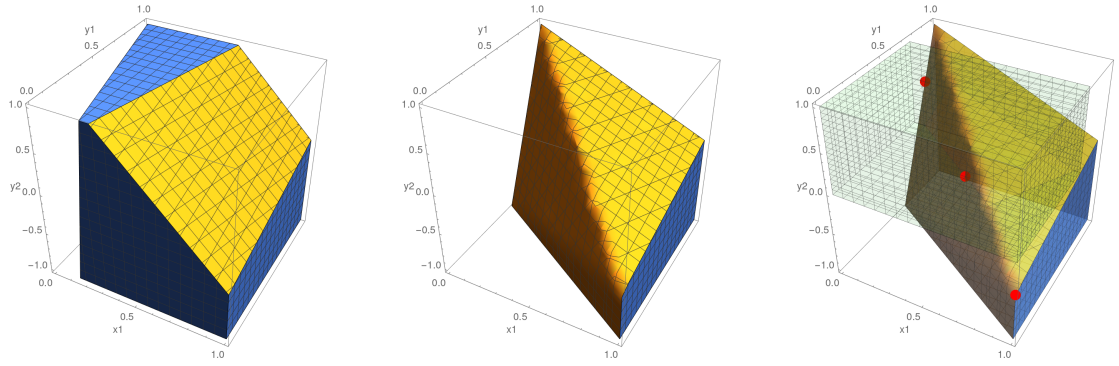

Figure 2: Left: Region defined by C_2, C_3, and the global domain; Middle: Mixed-integer hull; Right: Mixed-integer hull intersected with the local domain.

violated in the local bounds by considering only the constraints C_2 and C_3. Consider the two points $p_1 = (1, 0, 0, 0, -0.4)$ and $p_2 = (0, 0, 0, 0.88, 0.4)$. Both points are integer feasible for the constraints C_2 and C_3. Since any globally valid inequality for the system C_2 and C_3 must satisfy the two points, it must also satisfy any convex combination $p^\lambda = \lambda p_1 + (1 - \lambda)p_2$ for $\lambda \in [0, 1]$. However, the point $p^{0.5} = (0.5, 0, 0, 0.44, 0)$ lies in the local domain of the variables and hence cannot be separated by any globally valid inequality. Figure 2 shows the non-empty intersection of the mixed integer hull with the local domain.

In the following section we propose a new reason reduction algorithm for MBP that does not only consider the reason and conflict constraint, but also constraints that propagated non-relaxable continuous variables.

6.1 Reduction Algorithm for Mixed Binary Programs

The goal of the reduction algorithm is to remove non-relaxable continuous variables from the reason constraint while preserving the propagation of the binary variable x_r. The next proposition shows that after resolving a non-relaxable continuous variable from the reason constraint, the new reason constraint still propagates the binary variable x_r.

Proposition 4. *Let $C_{\text{cont}} : a_c x_c + a_r x_r + \sum_{j \notin \{r,c\}} a_j x_j \geq b$ be a MBP constraint propagating a continuous variable $x_c \in [\ell_c, u_c]$ in the state ρ_1. Further, assume that the bound change on the continuous variable is non-relaxable for a constraint $C_{\text{reason}} : a'_c x_c + a'_r x_r + \sum_{j \notin \{r,c\}} a'_j x_j \geq b'$ that propagates a variable x_r in a later state ρ_2. Then, the linear combination of C_{reason} and C_{cont} that eliminates x_c propagates x_r at least as strong in the sense that it gives the same or a better bound as C_{reason} in ρ_2.*

Proof. Since the bound change implied by C_{cont} on x_c is non-relaxable for C_{reason}, the coefficients a_c and a'_c must have opposite sign, and we can assume without loss of generality that $a_c = 1$ and $a'_c = -1$. The bound change of x_r implied by C_{reason} follows from

$$a'_r x_r \geq \min_{\rho_2}\{b' + x_c - \sum_{j \notin \{r,c\}} a'_j x_j\} = \ell_c^{\rho_2} + \min_{\rho_2}\{b' - \sum_{j \notin \{r,c\}} a'_j x_j\}.$$

By assumption, $\ell_c^{\rho_2} = \ell_c^{\rho_1}$ is the result of propagating C_{cont} in state ρ_1 and is given by

$$x_c \geq \min_{\rho_1}\{b - a_r x_r - \sum_{j \neq \{r,c\}} a_j x_j\} = \ell_c^{\rho_2}.$$

Therefore, the bound change on x_r from propagating C_{reason} in state ρ_2 is the one implied by

$$a'_r x_r \geq \min_{\rho_1}\{b - a_r x_r - \sum_{j \neq \{r,c\}} a_j x_j\} + \min_{\rho_2}\{b' - \sum_{j \neq \{r,c\}} a'_j x_j\}. \tag{15}$$

On the other hand, the linear combination of C_{reason} and C_{cont} eliminating x_c is given by

$$(a_r + a'_r)x_r + \sum_{j \neq \{r,c\}} (a_j + a'_j)x_j \geq b + b'. \tag{16}$$

Hence, the bound change on x_r from propagating this last constraint in state ρ_2 is the one implied by

$$\begin{aligned} a'_r x_r &\geq b + b' - a_r x_r - \sum_{j \neq \{r,c\}} (a_j + a'_j)x_j \\ &= b - a_r x_r - \sum_{j \neq \{r,c\}} a_j x_j + b' - \sum_{j \neq \{r,c\}} a'_j x_j \\ &\geq \min_{\rho_2}\{b - a_r x_r - \sum_{j \neq \{r,c\}} a_j x_j\} + \min_{\rho_2}\{b' - \sum_{j \neq \{r,c\}} a'_j x_j\}. \end{aligned} \tag{17}$$

Comparing the right-hand sides of (15) and (17), we see that the linear combination (16) implies a bound at least as strong as C_{reason}.

□

Remark 3. *Notice that in the proof above, the bound implied by the linear combination of C_{reason} and C_{cont} can be substantially stronger than the one implied by the reason constraint. Indeed, the aggregated constraint can even show infeasibility at state ρ_2. A particularly interesting example of this situation is when the sign of the coefficient of x_r in the aggregated contraint ($a_r + a'_r$) is different from the one in the reason constraint (a'_r). To see this, assume that $a'_r > 0$ (and that x_r is binary for simplicity). Since the sign changes in the aggregated constraint, it holds that $a_r < -a'_r < 0$. This means that $\ell_c^{\rho_2}$ reduces to $\min_{\rho_1}\{b - \sum_{j \neq \{r,c\}} a_j x_j\}$. Furthermore, since the reason constraint propagates x_r, we have that the right-hand side of* (15) *is positive, i.e.,*

$$\kappa := \min_{\rho_1}\{b - \sum_{j \neq \{r,c\}} a_j x_j\} + \min_{\rho_2}\{b' - \sum_{j \neq \{r,c\}} a'_j x_j\} > 0.$$

This gives rise to the contradiction

$$0 \geq (a_r + a'_r)x_r \geq b + b' - \sum_{j \neq \{r,c\}} (a_j + a'_j)x_j \geq \kappa > 0.$$

This contradiction proves infeasibility independently of the bound change on x_r in state ρ_2. Hence, the aggregated constraint (16) *is already infeasible at the state preceding ρ_2.*

Algorithm 3 shows the reduction algorithm for MBP. It takes as input the reason constraint C_{reason} propagating in state $\rho_{\hat{p},\hat{q}}$, the conflict constraint C_{confl} infeasible in state $\rho_{\bar{p},\bar{q}}$, and the binary variable to resolve x_r. Then it iteratively resolves all non-relaxable continuous variables from the reason constraint until no such variable exists. The set of non-relaxable continuous variables at a state $\rho_{p,q}$

is denoted by $\text{nr}(C_{\text{reason}}, \rho_{p,q})$. The algorithm may terminate early if the reason constraint becomes infeasible under the local bounds (see Remark 3 in Appendix D). Finally, after the main loop, we apply the reduction algorithm as in the binary case. The algorithm always terminates since the number of states is finite and it explores the states in a monotonically decreasing order with respect to the lexicographical order of the indices (p, q) and the non-relaxable continuous variables appearing in the reason constraint.

Algorithm 3: Reduction Algorithm for Mixed Binary Programs

Input : reason constraint C_{reason} propagating in state $\rho_{\hat{p},\hat{q}}$, conflict constraint C_{confl}, binary variable to resolve x_r
Output : reduced reason C_{reason} or an earlier conflict C_{confl}

$(\tilde{p}, \tilde{q}) \leftarrow (\hat{p}, \hat{q})$
while $\text{nr}(C_{\text{reason}}, \rho_{\tilde{p},\tilde{q}}) \neq \varnothing$ **do**
  $(\tilde{p}, \tilde{q}) \leftarrow \max\{(p, q) \mid \text{varidx}(\rho_{p,q}) \in \text{nr}(C_{\text{reason}}, \rho_{p,q}) \cap K,\ (p, q) \prec (\tilde{p}, \tilde{q})\}$
  $c \leftarrow \text{varidx}(\rho_{\tilde{p},\tilde{q}})$
  $C_{\text{cont}} \leftarrow \text{reason}(\rho_{\tilde{p},\tilde{q}})$
  $C_{\text{reason}} \leftarrow \text{resolve}(C_{\text{reason}}, C_{\text{cont}}, x_c)$
  if C_{reason} *is infeasible in the predecessor state of* $\rho_{\hat{p},\hat{q}}$ **then**
    $C_{\text{confl}} \leftarrow C_{\text{reason}}$
    return C_{confl}
$C_{\text{reason}} \leftarrow \text{reduce}(C_{\text{reason}}, C_{\text{confl}}, x_r, \rho_{\hat{p},\hat{q}})$
return C_{reason}

Example 3. *In Example 2 we show that no globally valid inequality can be constructed that is violated in the local bounds by using only the reason constraint C_2, the conflict constraint C_3 and the global bounds of the variables. Next we show that, after resolving all non-relaxable continuous variables from the reason constraint C_2, the new reason constraint contains only binary variables and propagates x_1:*

1. *Resolving y_1 by adding $1.25 \cdot C_1$ to C_2 gives the new $C_{\text{reason}} : 17.5x_1 - 3.5y_2 \geq 0.25$.*
2. *Resolving y_2 by adding $3.5 \cdot C_5$ to C_{reason} gives the new $C_{\text{reason}} : 17.5x_1 - 3.5x_3 \geq 0.25$.*

This constraint contains only binary variables, propagates x_1 to 1 non-tightly, and hence can be reduced as shown in Section 4.

7 Conflict Analysis for Mixed Integer Programs

Next, we briefly discuss the limitations of the conflict analysis algorithm under the presence of general integer variables. Again, the goal is to resolve a bound change on a variable by some suitable reduction and aggregation of constraints. As expressed in Proposition 1, whenever the propagation of the resolved variable is tight, we can simply aggregate the reason for the bound change to the current conflict and the resolvent remains infeasible. However, in the presence of general integer variables, our reduction algorithms are not guaranteed to find a reduced reason that propagates tightly.

Non-tight propagation occurs because of the existence of non-integer vertices for the linear relaxation of C_{reason} over the local domain L. The goal then is to find a cut that separates the non-integer vertex from the feasible region $C_{\text{reason}} \cap G$, where G is the box defined by the global bounds. This is not possible if the general integer variable contributes with a non-global bound to the propagation of the reason constraint. Figure 3 gives an example that illustrates such a situation.

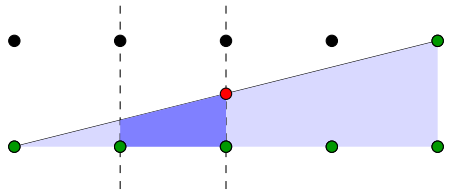

Figure 3: Example where it is impossible to find a linear cut that separates the non-integer vertex (2,0.5) from the feasible region.

One approach to addressing the resolution of general integer variables is proposed in (Jovanovic and de Moura 2013). Here the authors allow only fixings of variables to their current lower or upper bound. While this strategy restricts the branching heuristic, it also avoids scenarios like the one illustrated in Figure 3, allowing reduction algorithms to effectively eliminate non-integer vertices.

Another work that deals with general MIP is (Achterberg 2007a), which extends the graph-based conflict analysis typically used for SAT problems to the general MIP domain. In the MIP context, conflict graph nodes represent bound changes rather than variable fixings. Unlike in SAT, where conflicts derived from the graph are disjunctions of literals and can be represented as linear constraints, conflicts involving non-binary variables are disjunctions of general bound changes, which cannot be captured by a single linear constraint. Consequently, MIP solvers utilize these conflicts solely for propagation and not as cuts.

In our implementation, we consistently attempt to resolve variable bound changes by aggregating the reason and conflict constraints. Preliminary experiments show that in cases where integer variables do not contribute with a global bound to the propagation of the reason constraint, resolution still succeeds in 72% of cases, meaning the resulting resolvent remains infeasible even without reducing the reason constraint. In the remaining 28% of cases, we attempt to separate the non-integer vertex from the feasible region heuristically, by applying the general cMIR procedure (Marchand and Wolsey 2001). However, this approach is only successful about 3% of the time.

8 Experimental Evaluation

The focus of this section is to evaluate the performance of the different conflict analysis algorithms in a MIP solver. Our experiments are designed to answer the following questions:

- How do different conflict analysis algorithms perform when being applied to the same set of infeasibilities, not only in terms of performance but also regarding the characteristics of the generated constraints, e.g., constraint type, size and potential for propagation?
- What is the performance impact of cut-based conflict analysis in the MIP solver SCIP, and how does it integrate with other conflict analysis algorithms?

All techniques discussed in this paper have been implemented in the open-source MIP solver SCIP 9.0 (Bolusani et al. 2024). We have uploaded the code and results to a public GitHub repository hosted by the INFORMS Journal on Computing (Mexi et al. 2024). The experiments were conducted on a cluster equipped with Intel Xeon Gold 5122 CPUs @ 3.60GHz and 96GB of RAM. Each run was performed on a single thread with a time limit of two hours. To mitigate the effects of performance variability (Lodi and Tramontani 2013) and to ensure a fair comparison of the different conflict analysis algorithms, we used a large and diverse set of test instances, namely the MIPLIB 2017 benchmark set (Gleixner et al. 2021), with five permutations of each individual model. After

excluding all models that are not solvable by any setting for any of the five seeds, we obtained a total of 795 measurements per run. For the remainder of this paper, we will refer to the combination of a model and a permutation as an instance.

8.1 Implementation details

In our experiments, we do not use the Chvátal-Gomory-based reduction since it is dominated by the CMIR-CA reduction (and does not generalize to problems with continuous variables). Moreover, as demonstrated in Proposition 3, opting for complementation over weakening results in more robust reasoning constraints. Therefore, we consider the CMIR-CA reduction solely with complementation.

For the coefficient tightening-based reduction, we employ a single-sweep approach for variable weakening, improving the reduction algorithm's speed by avoiding repeated activity computations and applying only one cut per iteration. As observed in (Mexi et al. 2023), this does usually not lead to a loss of information since all, or almost all, variables have to be weakened before applying coefficient tightening to obtain a tightly propagating reason constraint.

Finally, we decided to mitigate the weak points of cut-based conflict analysis: dealing with general integer variables or propagations that are not explained by a single linear inequality. If cut-based conflict analysis fails to generate a conflict in such cases, we fall back to graph-based conflict analysis. In a preliminary experiment on MIPLIB 2017 we measured this fallback mechanism to occur in approximately 19% of the conflict analysis calls.

8.2 Comparison of Conflict Analysis Algorithms in Isolation

In our first experiment, we compare the following three conflict analysis algorithms:

- GRAPH-CA : Graph-based conflict analysis algorithm (Achterberg 2007a).
- COEFT-CA : Cut-based conflict analysis using the coefficient tightening-based reduction.
- CMIR-CA : Cut-based conflict analysis using the cMIR-based reduction.

When aiming to compare the effect of different conflict analysis algorithms throughout a MIP solve, there is a major caveat: Usually, the first few conflict analysis calls already cause the solution path of the solve to diverge significantly (demonstrating that the analysis had an impact). Thus, the vast majority of conflict analysis calls will be on a completely different set of infeasibilities, and it is hard to impossible to say whether observations made about conflicts differing in their characteristics are structural or mostly a side effect of the path divergence.

Therefore, to answer the first question from the beginning of this section, we carefully designed an experiment that allows us to compare different conflict analysis algorithms by different conflicts from the exact same infeasibilities throughout a complete tree search. To achieve this, we split the generation and the exploitation of the conflicts into two separate runs.

In the first run, whenever an infeasibility is found, we generate a single conflict from the 1-FUIP; however, we do not use the conflicts for propagation, do not apply bound changes, consider them for the branching decision, or allow any other interaction with the solving process. They are collected and "ignored". We call this the *conflict generation run*. Thus, we will traverse an identical search tree, independent of the conflict analysis algorithm used, and each algorithm will be applied to the same set of infeasibilities.

In the second run, we add the conflicts from the conflict generation run to the problem (right from the beginning of the search) and allow propagation and other interactions with the solution process. We refer to this run as the *conflict exploitation run*. This approach enables us to draw conclusions about the impact of conflicts generated from the same information. It even allows for a one-to-one comparison of conflicts from different algorithms.

In the conflict generation run, we set the same time limit as in the conflict exploitation run. However, we try to avoid hitting this time limit, since the point at which a wall-clock time limit is hit, is non-deterministic and we may observe a different number of conflict calls between two runs, if one of them hit the time limit slightly earlier/later. Therefore, as additional deterministic working limits, we impose a limit of 50,000 nodes and a limit of 5,000 conflict analysis calls. This also avoids the conflict exploitation run using unnaturally many conflicts since SCIP typically limits the number of conflicts that are handled at the same time to a few thousand. With these settings, only 74 of the instances hit the time limit and were consequently discarded from the results below. The rest of the instances are either solved, or hit the deterministic node or conflict analysis calls limit. Moreover, we discard a few instances for which the settings reported numerically inconsistent results. This gives us a total of 704 instances for the comparison. While SCIP usually discards conflicts from GRAPH-CA that are larger than a certain threshold, we deactivated this restriction for this experiment so as not to skew the results.

Table 1: Comparison of the different conflict analysis algorithms.

Setting	opt	time(s)	lin.confs	confs	avg.length	used(%)	bdchgs
GRAPH-CA	553	433.0	-	96.7	27.5	31.2	4.9
COEFT-CA	562	437.8	58.7	99.5	59.6	44.9	24.4
CMIR-CA	562	436.8	59.0	99.4	59.7	45.4	24.2

Table 1 summarizes the results of the experiment for the three different conflict analysis algorithms. The table shows the number of instances (inst), the number of instances solved to optimality (opt), the average solving time in seconds (time(s)), the average number of conflicts generated (confs), the average number of non-zeros in conflicts (avg.length), the percentage of conflicts that were used in propagation (used(%)), and the average number of bound changes applied (per node) by the conflicts (bdchgs).

For the two cut-based variants, we additionally show the number of linear conflicts generated(lin. confs); recall that for general integer variables or certain propagators, we might fall back to graph-based conflict analysis. The confs column, in these cases, considers both cut-based and graph-based conflicts.

Our first observation is that both COEFT-CA and CMIR-CA perform very similarly, which aligns nicely with the results for pure binary programming by (Mexi et al. 2023). When compared to GRAPH-CA , both solve eight more instances to optimality and the average time is comparable. In the following, we will concentrate on comparing CMIR-CA to GRAPH-CA ; all observations and conclusions would be the same or very similar for a COEFT-CA to GRAPH-CA comparison.

Two of the most noticeable differences in Table 1 are the large discrepancy in the number of propagations and the percentage of conflicts actually used in propagation. Both are significantly larger for the cut-based variants, which is a favorable result. When analyzing our results, we found three different reasons for this behavior.

Firstly, CMIR-CA conflicts are quite different from GRAPH-CA conflicts. On the one hand, GRAPH-CA conflicts are always logic clauses or bound disjunctions that only propagate one single bound

change when all other variables in the clause are fixed to a value not satisfying the clause. On the other hand, CMIR-CA conflicts might propagate multiple bound changes even when most variables are not fixed. For example, for the instance `neos-957323` with seed 0, CMIR-CA generated the following set packing conflict constraint:

$$x_{1130} + x_{1131} + x_{1132} + x_{1133} + x_{1134} + x_{1135} + x_{1150} + x_{1153} + x_{1156} \leq 1,$$

which propagates all remaining variables to zero whenever one of the variables in the constraint is fixed to one.

Secondly, both CMIR-CA and COEFT-CA conflicts are, on average, twice as long as those from GRAPH-CA . As an extreme example, consider the instance `nw04` with seed 0. This instance has 87 482 variables, 36 constraints, and 636 666 non-zeros, leading to an average of 17 685 non-zeros per constraint. Conflict constraints generated from CMIR-CA have an average length of 18 160 non-zeros, thus very similar to the model constraints. At the same time, the GRAPH-CA conflicts have an average length of "only" 243. While clausal conflicts tend to be weaker the longer they are, conflicts from the cut-based approach can get stronger the longer they are, see the example above or the rich literature on lifting cutting planes. Consequently, we observed a total of 864 bound changes from cut-based conflict constraints for instance `nw04` and only 26 propagations from GRAPH-CA conflicts.

Thirdly, there is another peculiar situation relating to general integer and continuous variables. Take as an example the instance `gen-ip002` with seed 1. For this instance all variables involved in conflict constraints are general integers with lower bound 0 and an upper bound of 140 or less. Conflicts from GRAPH-CA are clauses and are therefore restricted to one particular bound change per variable that they can propagate, in our example, those was typically tightening the upper bound to a particular single-digit value. Conflicts from cut-based conflict analysis, however, can propagate arbitrary bound changes, in particular, the same conflict can tighten the upper (or lower) bound of the same variable several times during the same dive in the tree search or even at the same node (if other propagators tightened bounds of other variables in the conflict in the meantime). The difference in structure, while having a similarly positive impact on performance, indicates that there might be potential for the two strategies to complement each other when applied in combination. Hence, our next experiment addresses a proper integration of cut-based conflict analysis into default SCIP.

8.3 MIP Performance

In this experiment, we activate cut-based conflict analysis as an additional conflict analysis method within SCIP. This means that conflicts from both cut- and graph-based conflict analysis, are generated at each infeasibility. For the cut-based conflict analysis, we only use the cMIR-based reduction. As seen in the previous section CMIR-CA and COEFT-CA perform similarly, however in an extended experiment with a longer time limit of 6 hours and 10 different random seeds, the cMIR-based reduction was on average 8% faster on "affected" instances and required 8% fewer nodes on those solved to optimality.

SCIP discards dense conflicts and regularly removes conflicts that haven't propagated in a while based on an aging strategy. We use the same length and age limits also for conflicts from cut-based conflict analysis. Preliminary experiments showed a slowdown of 4-5% if we do not limit the length of conflict constraints. On the other hand, the number of explored nodes decreases by 2% on average.

Table 2 shows the performance of SCIP with and without the CMIR-CA on MIPLIB2017. One instance was excluded due to numerical issues, leaving 794 instances in total. The table is divided

Table 2: Performance comparison of SCIP vs SCIP +CMIR-CA .

		SCIP			SCIP + CMIR-CA			relative	
Subset	instances	solved	time	nodes	solved	time	nodes	time	nodes
all	794	636	445.9	-	644	424.2	-	0.95	-
affected	492	463	454.1	-	471	419.2	-	0.92	-
[10,tilim]	598	569	406.0	-	577	380.2	-	0.94	-
[100,tilim]	443	414	987.7	-	422	890.7	-	0.90	-
[1000,tilim]	243	214	2594.4	-	222	2217.8	-	0.85	-
diff-timeouts	50	21	5934.9	-	29	4180.8	-	0.70	-
both-solved	615	615	201.2	2412	615	194.1	2283	0.96	0.95
aff.-both-solved	442	442	339.4	5413	442	323.0	5022	0.95	0.93

into different subsets of instances. The first row shows the performance on all 794 instances, and the second row on all 492 instances that are "affected" by conflict analysis, i.e., where the execution path of the two settings differs. The next three rows show the performance on instances where at least one of the two settings needs at least 10, 100, or 1000 seconds to solve the instance, respectively. The next row "diff-timeouts" shows the performance on instances where only one of the two settings is able to solve the instance to optimality. SCIP solved 21 instances that SCIP + CMIR-CA did not, while SCIP + CMIR-CA solved 29 instances that SCIP did not. In a post-hoc analysis with a time limit of 6 hours, there is only a single instance that SCIP solved in under 2 hours that SCIP + CMIR-CA still could not solve after 6 hours. Notably, there are 9 instances that SCIP failed to solve within 6 hours, while SCIP + CMIR-CA had already solved them in under 2 hours. The last two rows show the performance on instances that are solved to optimality by both settings and the subset of those instances that are affected by conflict analysis "aff.-both-solved". We included these rows because they allow for a comparison of tree sizes.

The results show that using CMIR-CA leads to a notable improvement in the performance of SCIP. Namely, it increases the number of solved instances from 636 to 644 and decreases the overall running time by 5%. On affected instances, we observed a runtime decrease of 8%. As shown in the rows [10,tilim], [100,tilim], and [1000,tilim], the performance improvement is more pronounced on harder instances, with a speedup of up to 15% on instances that need at least 1000 seconds to solve.

The performance improvement is also reflected in the number of nodes. The number of nodes can only be reliably compared on instances that are solved to optimality by both settings. In this case, the number of nodes is reduced by 5% overall and by 7% on affected instances.

9 Conclusion

In this work, we extended the cut-based conflict analysis algorithm, which originates from pseudo-Boolean solving, to MIP. Framed in MIP terminology, conflict analysis can be understood as a sequence of linear combinations, integer roundings, and cut generation. This MIP perspective allowed us to interpret the reason reduction subroutine as the separation of a non-integer vertex from the feasible region defined by the reason constraint and global bounds. Hence, our presentation helped to bridge the gap between pseudo-Boolean and MIP views on conflict analysis, connecting the study of cuts for the knapsack polytope with the reason reduction for pure binary problems. We then proposed a new reduction algorithm for pure binary programs using mixed integer rounding cuts

and proved that it is stronger than the division-based reduction algorithm (Elffers and Nordström 2018) and our previous work (Mexi et al. 2023).

On our path to generalize cut-based conflict analysis to mixed integer problems in full generality we encountered both positive and negative results. First, we showed that the cut-based approach cannot be directly applied to mixed binary programs without considering additional problem information. To address this limitation, we designed a new reduction algorithm that not only utilizes two constraints (reason and conflict) in each iteration of conflict analysis but also incorporates constraints whose propagation is responsible for tightened bounds of continuous variables. We demonstrate that in the presence of general integer variables conflict analysis cannot be guaranteed to work as long as we refrain from lifting the problem into higher dimension. However, our empirical results show that these cases occur only rarely in practice, rendering our cut-based conflict analysis a fruitful approach also for general integer variables.

Our computational study, first helped us to understand that the learned constraints generated by the cut-based conflict analysis differ structurally from those learned by clausal reasoning and propagate variable bounds more frequently on average. Finally, we observed that the cut-based conflict analysis is able to significantly enhance the out-of-the-box performance of the MIP solver SCIP, solving more instances, reducing the running time and the size of the search tree over a large and diverse set of MIP instances from MIPLIB 2017 by at least 5%. This result constitutes a substantial performance improvement for a single technique in the mature field of MIP solving.

References

Tobias Achterberg. Conflict analysis in mixed integer programming. *Discrete Optimization*, 4(1):4–20, March 2007a.

Tobias Achterberg. *Constraint Integer Programming*. PhD thesis, Technische Universität Berlin, 2007b.

Roberto J. Bayardo Jr. and Robert Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI '97)*, pages 203–208, July 1997.

Paul Beame, Henry Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *Journal of Artificial Intelligence Research*, 22:319–351, December 2004. Preliminary version in *IJCAI '03*.

Archie Blake. *Canonical Expressions in Boolean Algebra*. PhD thesis, University of Chicago, 1937.

Suresh Bolusani, Mathieu Besançon, Ksenia Bestuzheva, Antonia Chmiela, João Dionísio, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Mohammed Ghannam, Ambros Gleixner, Christoph Graczyk, Katrin Halbig, Ivo Hedtke, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Dominik Kamp, Thorsten Koch, Kevin Kofler, Jurgen Lentz, Julian Manns, Gioni Mexi, Erik Mühmer, Marc E. Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Mark Turner, Stefan Vigerske, Dieter Weninger, and Lixing Xu. The SCIP Optimization Suite 9.0. Technical report, Optimization Online, February 2024. URL `https://optimization-online.org/2024/02/the-scip-optimization-suite-9-0/`.

AL Brearley, Gautam Mitra, and H Paul Williams. Analysis of mathematical programming problems prior to applying the simplex algorithm. *Mathematical programming*, 8:54–83, 1975.

Samuel R. Buss and Jakob Nordström. Proof complexity and SAT solving. In Armin Biere, Marijn J. H. Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, chapter 7, pages 233–350. IOS Press, 2nd edition, February 2021.

Donald Chai and Andreas Kuehlmann. A fast pseudo-Boolean constraint solver. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(3):305–317, March 2005. Preliminary version in *DAC '03*.

Vasek Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete mathematics*, 4(4): 305–337, 1973.

William Cook, Collette Rene Coullard, and György Turán. On the complexity of cutting-plane proofs. *Discrete Applied Mathematics*, 18(1):25–38, November 1987.

Bruce Davey, Natashia Boland, and Peter J Stuckey. Efficient intelligent backtracking using linear programming. *INFORMS Journal on Computing*, 14(4):373–386, 2002.

Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3): 201–215, 1960.

Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, July 1962.

Jo Devriendt, Ambros Gleixner, and Jakob Nordström. Learn to relax: Integrating 0-1 integer linear programming with pseudo-Boolean conflict-driven search. *Constraints*, 26(1–4):26–55, October 2021. Preliminary version in *CPAIOR '20*.

Heidi E. Dixon and Matthew L. Ginsberg. Inference methods for a pseudo-Boolean satisfiability solver. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI '02)*, pages 635–640, July 2002.

Niklas Eén and Niklas Sörensson. Translating pseudo-Boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2(1-4):1–26, March 2006.

Jan Elffers and Jakob Nordström. Divide and conquer: Towards faster pseudo-Boolean solving. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI '18)*, pages 1291–1299, July 2018.

Ambros Gleixner, Gregor Hendel, Gerald Gamrath, Tobias Achterberg, Michael Bastubbe, Timo Berthold, Philipp M. Christophel, Kati Jarck, Thorsten Koch, Jeff Linderoth, Marco Lübbecke, Hans D. Mittelmann, Derya Ozyurt, Ted K. Ralphs, Domenico Salvagnin, and Yuji Shinano. MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library. *Mathematical Programming Computation*, 13:443–490, 2021. doi: 10.1007/s12532-020-00194-3.

Stephan Gocht, Jakob Nordström, and Amir Yehudayoff. On division versus saturation in pseudo-Boolean solving. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI '19)*, pages 1711–1718, August 2019.

Ralph E. Gomory. An algorithm for the mixed integer problem. Technical Report P-1885, The RAND Corporation, June 1960.

Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39(2-3):297–308, August 1985.

Christopher Hojny, Tristan Gally, Oliver Habeck, Hendrik Lüthen, Frederic Matter, Marc E Pfetsch, and Andreas Schmitt. Knapsack polytopes: a survey. *Annals of Operations Research*, 292:469–517, 2020.

John N. Hooker. Generalized resolution and cutting planes. *Annals of Operations Research*, 12(1):217–239, December 1988.

John N. Hooker. Generalized resolution for 0-1 linear inequalities. *Annals of Mathematics and Artificial Intelligence*, 6(1):271–286, March 1992.

Saurabh Joshi, Ruben Martins, and Vasco M. Manquinho. Generalized totalizer encoding for pseudo-Boolean constraints. In *Proceedings of the 21st International Conference on Principles and Practice of Constraint Programming (CP '15)*, volume 9255 of *Lecture Notes in Computer Science*, pages 200–209. Springer, August-September 2015.

Dejan Jovanovic and Leonardo de Moura. Cutting to the chase solving linear integer arithmetic. *Journal of Automated Reasoning*, 51(1):79–108, June 2013. Preliminary version in *CADE-23*.

Daniel Le Berre and Anne Parrain. The Sat4j library, release 2.2. *Journal on Satisfiability, Boolean Modeling and Computation*, 7:59–64, July 2010.

Andrea Lodi and Andrea Tramontani. Performance variability in mixed-integer programming. In *Theory driven by influential applications*, pages 1–12. INFORMS, 2013.

Hugues Marchand and Laurence A Wolsey. Aggregation and mixed integer rounding to solve mips. *Operations research*, 49(3):363–371, 2001.

João P. Marques-Silva and Karem A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, May 1999. Preliminary version in *ICCAD '96*.

Ruben Martins, Vasco M. Manquinho, and Inês Lynce. Open-WBO: A modular MaxSAT solver. In *Proceedings of the 17th International Conference on Theory and Applications of Satisfiability Testing (SAT '14)*, volume 8561 of *Lecture Notes in Computer Science*, pages 438–445. Springer, July 2014.

G. Mexi, F. Serrano, T. Berthold, A. Gleixner, and J. Nordström. Cut-based Conflict Analysis in Mixed Integer Programming, 2024. Available for download at `https://github.com/INFORMSJoC/2024.0999`.

Gioni Mexi, Timo Berthold, Ambros Gleixner, and Jakob Nordström. Improving conflict analysis in MIP solvers by pseudo-Boolean reasoning. In *Proceedings of the 29th International Conference on Principles and Practice of Constraint Programming (CP '23)*, volume 280 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 27:1–26:19, August 2023.

Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th Design Automation Conference (DAC '01)*, pages 530–535, June 2001.

John Alan Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1): 23–41, January 1965.

Masahiko Sakai and Hidetomo Nabeshima. Construction of an ROBDD for a PB-constraint in band form and related techniques for PB-solvers. *IEICE Transactions on Information and Systems*, 98-D(6):1121–1127, June 2015.

Tuomas Sandholm and Robert Shields. Nogood learning for mixed integer programming. In *Workshop on Hybrid Methods and Branching Rules in Combinatorial Optimization, Montréal*, volume 20, pages 21–22, 2006.

Hossein M. Sheini and Karem A. Sakallah. Pueblo: A hybrid pseudo-Boolean SAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 2(1-4):165–189, March 2006. Preliminary version in *DATE '05*.

Richard M Stallman and Gerald J Sussman. Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *Artificial intelligence*, 9(2):135–196, 1977.

Alasdair Urquhart. Hard examples for resolution. *Journal of the ACM*, 34(1):209–219, January 1987.

H. P. Williams. Fourier-Motzkin elimination extension to integer programming problems. *Journal of Combinatorial Theory, Series A*, 21(1):118–123, July 1976.

Jakob Witzig, Timo Berthold, and Stefan Heinz. Computational aspects of infeasibility analysis in mixed integer programming. *Mathematical Programming Computation*, 13(4):753–785, 2021.