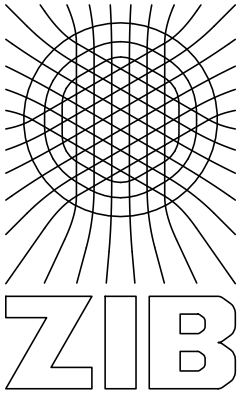
Zuse Institute Berlin

Takustr. 7
14195 Berlin
Germany



JAKOB WITZIG TIMO BERTHOLD STEFAN HEINZ

Computational Aspects of Infeasibility Analysis in Mixed Integer Programming

ZIB Report 19-54 (November 2019)

Zuse Institute Berlin
Takustr. 7
14195 Berlin
Germany

Telephone: +49 30-84185-0
Telefax: +49 30-84185-125

E-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

Computational Aspects of Infeasibility Analysis in Mixed Integer Programming

Jakob Witzig[1], Timo Berthold[2], and Stefan Heinz[3]

[1]Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany
witzig@zib.de

[2]Fair Isaac Germany GmbH, Stubenwald-Allee 19, 64625 Bensheim, Germany
timoberthold@fico.com

[3]Gurobi GmbH, Ulmenstr. 37–39, 60325 Frankfurt am Main, Germany
heinz@gurobi.com

November 6, 2019

Abstract

The analysis of infeasible subproblems plays an important role in solving mixed integer programs (MIPs) and is implemented in most major MIP solvers. There are two fundamentally different concepts to generate valid global constraints from infeasible subproblems. The first is to analyze the sequence of implications, obtained by domain propagation, that led to infeasibility. The result of this analysis is one or more sets of contradicting variable bounds from which so-called conflict constraints can be generated. This concept is called conflict graph analysis and has its origin in solving satisfiability problems and is similarly used in constraint programming. The second concept is to analyze infeasible linear programming (LP) relaxations. Every ray of the dual LP provides a set of multipliers that can be used to generate a single new globally valid linear constraint. This method is called dual proof analysis. The main contribution of this paper is twofold. Firstly, we present three enhancements of dual proof analysis: presolving via variable cancellation, strengthening by applying mixed integer rounding functions, and a filtering mechanism. Further, we provide an intense computational study evaluating the impact of every presented component regarding dual proof analysis. Secondly, this paper presents the first integrated approach to use both conflict graph and dual proof analysis simultaneously within a single MIP solution process. All experiments are carried out on general MIP instances from the standard public test set Miplib 2017; the presented algorithms have been implemented within the non-commercial MIP solver `SCIP` and the commercial MIP solver `FICO Xpress`.

1 Introduction

In the last decades, *mixed integer programming* (MIP) has become one of the most important techniques in Operations Research. The general framework of mixed integer programming was successfully applied to many real world applications, e.g., chip design verification (Achterberg, 2007b), scheduling (Lee et al., 1996; Heinz and Beck, 2012; Schade et al., 2018), supply chain management (You and Grossmann, 2008; Gamrath et al., 2016), and gas transport optimization (Domschke et al., 2011; Hiller et al., 2018) to mention only few of them. A lot of progress has been made in the performance of general MIP solver software; problems that seemed out of scope a decade ago can often be solved in seconds nowadays (Bixby et al., 1999; Achterberg and Wunderling, 2013). Hereby, the development of commercial (e.g., `CPLEX`, `FICO Xpress`, `Gurobi`, `SAS`) and non-commercial (e.g., `CBC`, `SCIP`) MIP solvers was directly stimulated by the theoretical progress in the field. In this paper, we will focus at one specific component that is used in almost every state-of-the-art MIP solver: the analysis of infeasible subproblems. Without loss of generality, we consider MIPs of the form

$$\min\{c^\mathsf{T} x \,|\, Ax \geq b,\ \ell \leq x \leq u,\ x_j \in \mathbb{Z}\ \forall j \in \mathcal{I}\}, \tag{1}$$

with objective coefficient vector $c \in \mathbb{R}^n$, constraint coefficient matrix $A \in \mathbb{R}^{m \times n}$, constraint left-hand side $b \in \mathbb{R}^m$, and variable bounds $\ell, u \in \overline{\mathbb{R}}^n$, where $\overline{\mathbb{R}} := \mathbb{R} \cup \{\pm\infty\}$. Furthermore, let $\mathcal{N} = \{1, \ldots, n\}$ be the index set of all variables let $\mathcal{I} \subseteq \mathcal{N}$ be the set of variables that need to be integral in every feasible solution.

When omitting the integrality requirements, we obtain the *linear program (LP)*

$$\min\{c^\mathsf{T} x \,|\, Ax \geq b,\ \ell \leq x \leq u,\ x \in \mathbb{R}^n\}. \tag{2}$$

The linear program (2) is called *LP relaxation* of (1). The LP relaxation provides a lower bound on the optimal solution value of the MIP (1). This fact is an important ingredient of LP-based branch-and-bound (Dakin, 1965; Land and Doig, 1960), the most commonly used method to solve MIPs. Branch-and-bound is a divide-and-conquer method which splits the search space sequentially into smaller subproblems that are intended to be easier to solve. During this procedure, the solver may encounter infeasible subproblems. Infeasibility can either be detected by contradicting implications, e.g., derived by domain propagation, or by an infeasible LP relaxation.

Modern MIP solvers try to "learn" from infeasible subproblems, e.g., by *conflict analysis* (e.g., Achterberg, 2007a). More precisely, each subproblem can be identified by its local variable bounds, i.e., by local bound changes that come from branching decisions and domain propagation (Savelsbergh, 1994) at the current node and its ancestors. If domain propagation detects infeasibility, one way of conflict learning is traversing the chain of decisions and deductions in a reverse fashion, reconstructing which bound changes led to which other bound changes, and will thereby identify explanations for the infeasibility. If it can be

shown that a small subset of the bound changes suffices to prove infeasibility, a so-called *conflict constraint* is generated that can be exploited in the remainder of the search for domain propagation. As a consequence, other parts of the search tree can be pruned and additional variable bound can be deductions.

The power of conflict analysis arises because branch-and-bound algorithms often repeat a similar search in a slightly different context in another part of the search tree. Conflict constraints address such situations and aim to avoid redundant work.

Conflict analysis for MIP has its origin in solving satisfiability problems (SAT) and goes back to Marques-Silva and Sakallah (1999). Similar ideas are used in constraint programming (e.g., Ginsberg, 1993; Jiang et al., 1994; Stallman and Sussman, 1977). Integrations of these techniques into MIP were independently suggested by Davey et al. (2002); Sandholm and Shields (2006); Achterberg (2007b). In this paper, we mostly follow the concepts and the notation of Achterberg (2007b). Follow-up publications suggested to use conflict information for variable selection in branching (Achterberg and Berthold, 2009; Kılınç Karzan et al., 2009), to tentatively generate conflicts before branching (Berthold et al., 2010; Berthold, 2014; Berthold et al., 2018b), and to analyze infeasibility detected in primal heuristics (Berthold and Gleixner, 2014; Berthold and Hendel, 2015; Witzig and Gleixner, 2019). In addition, instead of simply analyzing infeasibility that is derived more or less coincidentally, methods were introduced to explicitly generate additional conflict information (Dickerson and Sandholm, 2013; Witzig and Gleixner, 2019).

In a preliminary study (Witzig et al., 2017), we examined a vanilla approach of combining conflict graph analysis and dual proof analysis within `SCIP`. The present paper builds upon this work. Our contribution is twofold: We present various enhancements for dual proof analysis and provide an extensive computational study of the described techniques. Overall, the technique presented in this paper improve the solving performance of `SCIP` by 7 % and lead to a tree size reduction of 6.4 % compared to the approach reported in Witzig et al. (2017).

The paper is organized as follows. In Section 2 we give a theoretical overview on the analysis of infeasible subproblems. This covers the concept behind conflict graph analysis and the LP-based theory of dual proof analysis. Section 3 deals with three enhancements on dual proof analysis. We will discuss presolving techniques like nonzero cancellation and the application of mixed integer rounding functions to strengthen the resulting dual proof constraint. Moreover, we present an update mechanism that allows to strengthen certain dual proof constraints during the tree search and a filtering method to only pick the most promising dual proof constraints. An intense computational study of the techniques presented in this paper will be given in Section 4. Next to an individual evaluation, this study contains computational results with the first MIP solvers, to the best of our knowledge, containing both conflict graph and dual proof analysis.

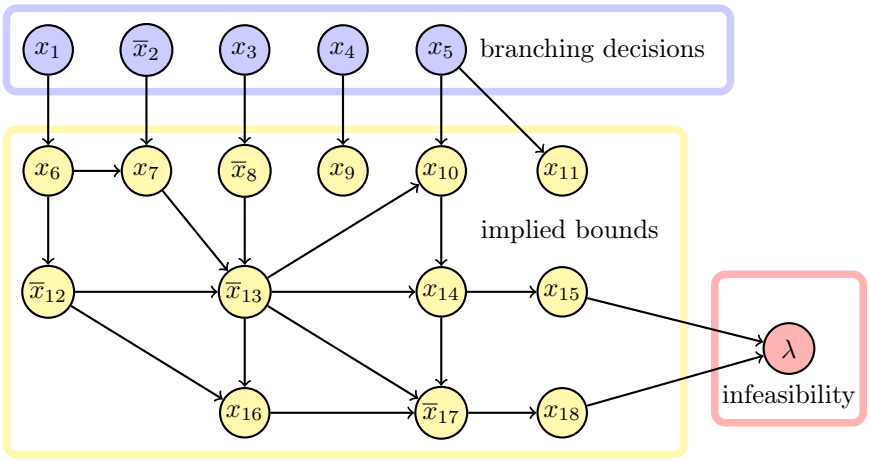


Figure 1: Example of a conflict graph describing all variable bound implications from the branching decisions to the infeasibility λ due to propagation. All shown variables are binary, negations are illustrated with a bar.

2 Infeasibility Analysis for MIP

The analysis of infeasible subproblems is widely established in solving MIPs, today. Most state-of-the-art MIP solvers rely either on an adaption of the conflict graph analysis techniques known from SAT or a pure LP-based approach. Both approaches are strongly connected, as we will argue below.

Assume, we are given an infeasible node of the branch-and-bound tree defined by the subproblem

$$\min\{c^\mathsf{T} x \,|\, Ax \geq b,\ \ell' \leq x \leq u',\ x_j \in \mathbb{Z}\, \forall j \in \mathcal{I}\} \tag{3}$$

with local bounds $\ell \leq \ell' \leq u' \leq u$. In LP-based branch-and-bound, the infeasibility of a subproblem is typically detected by contradicting implications or by an an infeasible LP relaxation. In the following, we describe how both situations can be handled.

2.1 Infeasibility due to Domain Propagation

If infeasibility is encountered by domain propagation, modern SAT and MIP solvers construct a directed acyclic graph which represents the logic of how the set of branching decisions led to the detection of infeasibility, see Figure 1. This graph is called a *conflict graph* (or implication graph) (Marques-Silva and Sakallah, 1999). The vertices of the conflict graph represent bound changes of variables and the arcs (v, w) correspond to bound changes implied by propagation, i.e., the bound change corresponding to w is based (besides others) on the bound change represented by v.

In addition to the inner vertices which represent the bound changes from domain propagation, the graph features source vertices for bound changes that correspond to branching decisions and an artificial sink vertex representing the infeasibility. Valid *conflict constraints* can be derived from cuts in the graph that separate the branching decisions from the artificial infeasibility vertex. Based on such a cut, a conflict constraint can be derived consisting of a set of variables with associated bounds, requiring that in each feasible solution at least one of the variables has to take a value outside these bounds. Note that in general, conflict constraints derived from this procedure have no linear representation if general integer or continuous variables are present. Moreover, by using different cuts in the graph, several different conflict constraints might be derived from a single infeasibility.

A well-established strategy for finding good cuts in the conflict graph is to rely on the so-called *First-Unique-Implication-Point* (FUIP) (Zhang et al., 2001). For every level of branching decisions there exists a FUIP. Zhang et al. (2001) have demonstrated that for solving SAT problems using 1-FUIP, i.e., the FUIP of the last branching decision level, is superior to other strategies. In contrast to that, Achterberg (2007a) demonstrated that for solving MIPs considering the FUIPs of all branching decisions and the corresponding conflict constraints along the path to root node and selecting the most promising conflict constraints outperforms using the 1-FUIP solely.

2.2 Infeasibility due to an Infeasible LP Relaxation

During LP-based branch-and-bound the LP relaxation of each subproblem is solved. If the LP relaxation turns out to be infeasible the procedure described in Section 2.1 can not be applied directly. The reason is that the LP does not deduce infeasibility from a single constraint, but from the inequality system (2) as a whole. Naïvely, one would only have to consider all local bound changes as a reason for the infeasibility which would not allow to learn a conflict constraint that can be used for pruning elsewhere in the tree. Although the naïve reason of infeasibility does not provide additional information for the remainder of the search, it can be used as a starting point for conflict graph analysis that might lead to more general conflict constraints. However, it is desirable to identify a small set of variables and bound changes that are sufficient to render the LP infeasible. Such a set of variables and bound changes can be identified by using LP duality theory. Afterwards, the same procedure as for infeasibility due to contradicting variable bounds can be used to derive conflict constraints. Achterberg proposed to pursue the same strategy for analyzing LP infeasibility (see end of Section 2.2.2) as for analyzing infeasibility due to domain propagation, i.e., considering all FUIPs and choosing the most promising conflict constraints Achterberg (2007a). In this paper, we propose to go one step further and to consider the LP-based certificate of infeasibility that encodes all reasons of infeasibility directly, i.e., the so-called dual proof constraint, instead of using this reason of infeasibility as a starting point for conflict graph analysis solely.

2.2.1 An Excursion to Duality Theory

If infeasibility is detected by the LP relaxation, the proof of infeasibility is given by a ray in the dual space. Consider a node of the branch-and-bound tree and the corresponding subproblem of type (3) with local bounds $\ell \leq \ell' \leq u' \leq u$. The *dual LP* of the corresponding LP relaxation of (3) is given by

$$\max\{y^\mathsf{T} b + r^\mathsf{T}\{\ell', u'\} \,|\, y^\mathsf{T} A + r = c,\ y \in \mathbb{R}^n_{\geq 0},\, r \in \mathbb{R}^n\},$$

where

$$r^\mathsf{T}\{\ell', u'\} := \sum_{\substack{j \in \mathcal{N} \\ r_j > 0}} r_j \ell'_j + \sum_{\substack{j \in \mathcal{N} \\ r_j < 0}} r_j u'_j = \sum_{\substack{j \in \mathcal{N} \\ r^\ell_j > 0}} r^\ell_j \ell'_j - \sum_{\substack{j \in \mathcal{N} \\ -r^u_j < 0}} r^u_j u'_j \tag{4}$$

with $r^\ell, r^u \in \mathbb{R}^n_+$ representing the dual variable on the finite bound constraints, i.e., $r := r^\ell - r^u$. Note, since every variable j can only be tight in at most one bound constraint, it follows by complementary slackness that either r^ℓ_j, r^u_j, or none of them will be different to zero but not both at the same time. For every variable x_j it holds that $r_j = c_j - y^\mathsf{T} A_{\cdot j}$, where $A_{\cdot j}$ denotes the j-th column of A. By LP theory, each ray $(y, r) \in \mathbb{R}^{m+n}$ in the dual space that satisfies

$$\begin{aligned} y^\mathsf{T} A + r &= 0 \\ y^\mathsf{T} b + r^\mathsf{T}\{\ell', u'\} &> 0 \end{aligned} \tag{5}$$

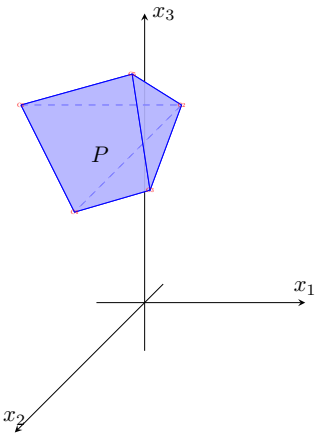
proves infeasibility of (3), which is an immediate consequence of the Lemma of Farkas (1902). Hence, if (2) is infeasible, there exists a solution (y, r) of (5).

2.2.2 Analysis of Infeasible LPs

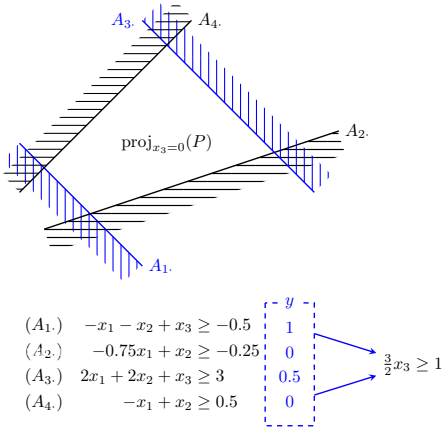
It follows immediately from (5) that infeasibility within the local bounds ℓ' and u' is proven by $0 < y^\mathsf{T} b + r^\mathsf{T}\{\ell', u'\} = y^\mathsf{T} b - y^\mathsf{T} A\{\ell', u'\}$. Therefore, the inequality

$$y^\mathsf{T} A x \geq y^\mathsf{T} b \tag{6}$$

has to be fulfilled by all feasible solutions. Since (6) is a non-negative aggregation of globally valid constraints, it is globally valid, too. In the following, this type of constraint will be called *dual proof constraint*. The dual ray is effectively a list of multipliers on all constraints that are represented in LP relaxation like model constraints (needed by the problem formulation) or additional valid global inequalities, e.g., cutting planes, conflict constraints, and dual proof constraints. Thus, aggregating the constraints according to the multipliers leads to globally valid but redundant constraint. However, with respect to the local bounds ℓ' and u' it leads to a false statement, thereby proving that the set described by the constraints is empty inside the local bounds. The property of proving infeasibility with respect to at least one set of local bounds distinguishes dual proof constraints from arbitrary constraint aggregations and is used as some



a) Polytope P



b) Projection of P. Dual multipliers y and the resulting dual proof constraint.

Figure 2: Illustration of an infeasibility proof. The polytope $P = \{x \in \mathbb{R}^3 \mid Ax \geq b,\ 0 \leq x_j \leq 1 \text{ for } j = 1, 2, 3\}$ representing the convex hull of the global problem $\min\{0^\mathsf{T} x \mid Ax \geq b,\ x \in \{0,1\}^3\}$. After branching in x_3, the subproblem with $x_3 = 0$ can be proven to be infeasible due to an infeasible LP relaxation. The resulting dual proof constraint is $x_3 \geq \frac{2}{3}$ which lead to the global bound deduction $x_3 \geq 1$.

kind of natural "filtering" among the infinitely many different possible constraint aggregations.

In general, infeasibility analysis can also be applied as explained in this paper if locally valid inequalities are present. Here, we assume the corresponding dual multiplier to be 0. The resulting dual proof might not prove local infeasibility anymore. In that case, the analysis of the infeasibility is stopped immediately. A modification of infeasibility analysis that incorporates also locally valid inequalities is described by (Witzig et al., 2019).

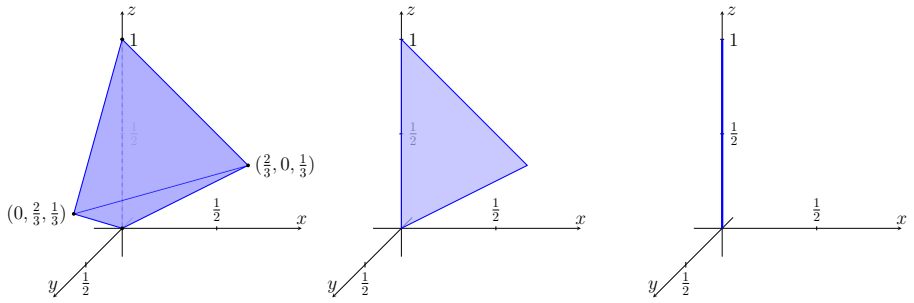
The situation of local infeasibility can be exploited as follows: Either, one starts conflict analysis as described in Section 2.1 from the set of local bounds or one considers only the weights given by the dual ray on the model constraints (Pólik, 2015a,b). If those are aggregated with respect to the dual multipliers, omitting the local bounds, we derive a new globally valid constraint, which can be used for domain propagation. We refer to the latter as *dual proof analysis* and to the outcome of this approach as *dual proof constraint*. For the remainder of the paper, we will refer to the classical conflict analysis as described in Achterberg (2007a) as *conflict graph analysis* in order to better disambiguate the two concepts. We will speak about the analysis of infeasible subproblems if we refer to both concepts together.

An example of an infeasible LP after branching on one variable and the resulting dual proof constraint can be found in Figure 2. In this case, single bound change is derived but in general the resulting constraint can be directly used for domain propagation during the remainder of the search.

In practice, a dual proof constraint of type (6) is expected to be dense, and therefore, it might be worthwhile to search for a sparser infeasibility proof. Therefore, we will discuss in the following how this constraint can be sparsified and strengthened via *dual proof analysis* or used as a starting point for *conflict graph analysis*, which is already described for infeasbility due to domain propagation in Section 2.1.

Dual Proof Analysis Dual proof analysis denotes the modification of a dual proof constraint (6) aiming to sparsify and strengthen the proof of infeasibility. In general, all presolving techniques that are applicable to a single constraint (e.g., Savelsbergh, 1994; Achterberg, 2007b; Achterberg et al., 2016), can be used. In Section 3 we will especially elaborate on presolving reductions that preserve the local proof of infeasibility. The outcome of dual proof analysis is a single linear constraint that can be used in the remainder of the search to derive further bound deductions, to prove infeasibility of other parts of the tree, and as input for the conflict graph analysis.

Conflict Graph Analysis To use the concept of a conflict graph as described in Section 2.1, one needs an initial, preferably small, reason for infeasibility. This can be constructed from an infeasibility proof of type (6): It suffices to consider all local bounds which have a nonzero coefficient in (6). The vertices of the conflict graph which correspond to those local bounds are then connected



a) Convex hull of the global problem.

b) Convex hull of the global problem after conflict graph analysis.

c) Convex hull of the global problem after dual proof analysis.

Figure 3: Illustration of Example 1.

to the artificial sink representing the infeasibility; global bounds are omitted. From thereon, conflict graph analysis can be applied as described in Section 2.1.

To strengthen this procedure, one can sparsify the proof of infeasibility (6) by a heuristic introduced by Achterberg (2007a). The heuristic relaxes some of the local bounds $[\ell', u']$ that appear in the dual proof constraint such that the relaxed local bounds $[\ell'', u'']$ with $\ell < \ell'' \leq \ell' \leq u' \leq u'' < u$ still fulfill

$$y^\mathsf{T} b + r^\mathsf{T}\{\ell'', u''\} > 0.$$

Note, the more bounds can be relaxed, the smaller the initial reason gets and consequently the stronger are the resulting conflict constraints in the end.

Example 1. *A main difference between using a dual proof directly instead of applying conflict graph analysis is the consideration of variables that contribute with their global bound. Consider the following MIP*

$$\begin{aligned} \max\ & x + y + z \\ s.t.\quad & x + y + 2z \leq 2 \\ & x + y - 2z \leq 0 \\ & x + y + z \leq 1 \\ & x, y, z \in \{0, 1\}. \end{aligned}$$

An optimal solution of the LP relaxation is $(0, \frac{2}{3}, \frac{1}{3})$ *with objective value* 1*. After branching on* y *the resulting subproblem with* $y = 1$ *gets infeasible, because its LP relaxation can be proven to be infeasible. A valid infeasibility proof, i.e., an unbounded ray in its dual, is* $(0, -\frac{1}{2}, -1)$*. Following Section 2.2.2, the resulting dual proof constraint is* $1.5x + 1.5y \leq 1$*. Applying conflict graph analysis leads to a single global deduction:* $(y \leq 0)$*, see Figure 3b. In contrast to that, propagating the dual proof constraint directly leads to two global bound changes:* $(x \leq 0) \wedge (y \leq 0)$*, see Figure 3c.* □

3 Three Enhancements for Dual Proof Analysis

Dual proof constraints derived as described in Section 2.2.2 can be used directly in the remainder of the search. However, similar to conflict graph analysis, the dual proof constraint can be modified and strengthened. In this section, we describe presolving and updating steps for strengthening dual proof constraints. Moreover, we discuss how filtering steps can used to decide which dual proof constraint should be applied and which should be rejected.

3.1 Presolving and Strengthening

Presolving plays an important role in solving mixed integer programs (e.g., Brearley et al., 1975; Guignard and Spielberg, 1981; Savelsbergh, 1994; Achterberg, 2007b; Achterberg et al., 2016). In this section, we will discuss presolving techniques that can be applied to a single linear constraint with the aim to preserve the property of proving local infeasibility.

Since we consider MIPs of form $Ax \geq b$, only constraints of type $a^\mathsf{T}x \geq b_0$ are discussed in the remainder of this section. However, in Section 3.2 we will stick to the notation used in the literature and consider constraints of form $a^\mathsf{T}x \leq b_0$. Note, by scaling with -1 each representations can be transformed into the other. Many presolving and strengthening techniques use activity arguments Brearley et al. (1975). Formally, the *maximal activity* of a constraint $a^\mathsf{T}x \geq b_0$ with respect to the (local) bounds ℓ, u is given by

$$\Delta_{\max}(a, \ell, u) = \sum_{j \in \mathcal{N} \,:\, a_j > 0} a_j u_j + \sum_{j \in \mathcal{N} \,:\, a_j < 0} a_j \ell_j.$$

Analogously, the *minimal activity* is given by

$$\Delta_{\min}(a, \ell, u) = \sum_{j \in \mathcal{N} \,:\, a_j > 0} a_j \ell_j + \sum_{j \in \mathcal{N} \,:\, a_j < 0} a_j u_j.$$

Moreover, we will denote the *violation* of a constraint with respect to the (local) bounds ℓ and u by $\nu_{\text{viol}}(a, b_0, \ell, u) = b_0 - \Delta_{\max}(a, \ell, u)$. Hence, a constraint proves infeasibility with respect to ℓ and u if $\nu_{\text{viol}}(a, b_0, \ell, u) > 0$.

3.1.1 Nonzero Cancellation

Nonzero cancellation is a presolving technique to reduce the number of nonzero coefficients in a constraint (Achterberg et al., 2016). One way of doing so is the addition of constraints. More precisely, adding an equality constraint to an inequality constraint will preserve the feasible region of an LP or a MIP. The crucial part of this presolving technique is to choose a pair of constraints and a scaling factor for the equality constraint such that the support of the modified inequality gets reduced. In this section, we present two variants of nonzero cancellation that can be applied to dual proof constraints. Both variants have a linear run time in the size of the support of the modified constraint.

Cancellation with Variable Bound Dual proof constraints might contain variables that contribute to the maximal activity with their global bound. Since the local and global contributions to the maximal activity are equal, e.g., $a_j \ell' = a_j \ell$, those variables can be removed from the infeasibility proof without changing the violation. Let ℓ', u' be the current local bounds of an infeasible subproblem. Then, a new constraint $\phi_B(a)^\mathsf{T} x \geq \psi_B(b_0)$ that still proves local infeasibility can be constructed by

$$\phi_B(a)_j = \begin{cases} a_j, \text{ if } j \notin \mathcal{N}_B \\ 0, \text{ otherwise} \end{cases} \tag{7}$$

$$\psi_B(b_0) = b_0 - \sum_{j \in \mathcal{N}_B} a_j \{\ell_j, u_j\}, \tag{8}$$

where $\mathcal{N}_B := \{j \in \mathcal{N} \mid a_j < 0 \text{ and } \ell' = \ell \text{ or } a_j > 0 \text{ and } u' = u\}$.

Proposition 2. *Let $a^\mathsf{T} x \geq b_0$ be a globally valid constraint and $\phi_B(a)^\mathsf{T} x \geq \psi_B(b_0)$ be constructed as described in (7)– (8) with respect to the local bounds ℓ' and u'. Then, the following holds:*

(i) $\nu_{viol}(a, b_0, \ell, u) = \nu_{viol}(\phi_B(a), \psi_B(b_0), \ell, u)$, *and*

(ii) $\tilde{\mathcal{X}} := \{(\ell', u') \in \mathbb{R}^n \times \mathbb{R}^n \mid \ell \leq \ell' \leq u' \leq u,\ \Delta_{\max}(\phi_B(a), \ell', u') < \psi_B(b_0)\} \subseteq \mathcal{X} := \{(\ell', u') \in \mathbb{R}^n \times \mathbb{R}^n \mid \ell \leq \ell' \leq u' \leq u,\ \Delta_{\max}(a, \ell', u') < b_0\}$.

Proof. (i) This follows immediately by construction.

(ii) We show, that for every subproblem defined by $(\ell', u') \in \tilde{\mathcal{X}}$ it follows that $a^\mathsf{T}\{\ell', u'\} < b_0$. In other words, every tuple of bound vectors that is proven to be infeasible after applying (7) and (8) is proven to be infeasible by the original infeasibility proof, too. With the following observation

$$\sum_{\substack{j \in \mathcal{N}_B \\ \phi_B(a)_j = 0}} a_j \{\ell', u'\} < \sum_{\substack{j \in \mathcal{N}_B \\ \phi_B(a)_j = 0}} a_j \{\ell, u\},$$

where we use the short notation as introduced in (4), it follows immediately that

$$\begin{aligned} &\Delta_{\max}(\phi_B(a), \ell', u') + \sum_{\substack{j \in \mathcal{N}_B \\ \phi_B(a)_j = 0}} a_j \{\ell', u'\} < \psi_B(b_0) + \sum_{\substack{j \in \mathcal{N}_B \\ \phi_B(a)_j = 0}} a_j \{\ell_j, u_j\} \\ \iff \quad & \Delta_{\max}(a, \ell', u') < b_0, \end{aligned}$$

where ℓ', u', ℓ, u are chosen to maximize every expression.

□

Proposition 2 shows that using variable bounds to cancel nonzero coefficients in a dual proof constraint leads to a weaker infeasibility proof in general. Note that in LP-based branch-and-bound, as it is implemented in most of the modern

MIP solvers, branching is performed on integer variables only. Therefore, we suggest to apply the nonzero cancellation procedure as described in this section to continuous variables only. In our implementation we pursue a slightly more conservative strategy which only removes continuous variables that contribute with a global bound to the maximal activity with respect to the current local bounds, i.e., depending on the sign of the coefficient in the dual proof it holds $\ell' = \ell$ and $u' = u$, respectively. We apply this cancellation as long as $\text{supp}(a) > 0.15 \cdot |\mathcal{N}|$, where $\text{supp}(a) := \{j \in \mathcal{N} \mid a_j \neq 0\}$ denotes the support of $a^\mathsf{T} x \geq b$. This threshold is identically to the default threshold used to decide whether a conflict constraint, i.e., an infeasibility certificate derived from conflict graph analysis, should be accepted.

For the special case of all $j \in \text{supp}(a) \cap \mathcal{I}$ are binary and all continuous variables contribute with a global bound, i.e., $j \in \text{supp}(a) \cap (\mathcal{N} \setminus \mathcal{I}) : j \in \mathcal{N}_B$, we apply a more sophisticated strategy. Consider a maximal set of binary variables $\mathcal{I}^{\max} \subset (\text{supp}(a) \cap \mathcal{I})$ which is not sufficient to prove infeasibility with respect to the local bounds ℓ' and u', i.e.,

$$\Delta_{\max}(a|_{\mathcal{I}^{\max}}, \ell', u') + \Delta_{\max}(a|_{\mathcal{N}\setminus\mathcal{I}^{\max}}, \ell, u) \geq b$$

but for any binary variable x_j with $j \notin \mathcal{I}^{\max}$ it holds

$$\Delta_{\max}(a|_{\mathcal{I}^{\max}\cup j}, \ell', u') + \Delta_{\max}(a|_{\mathcal{N}\setminus(\mathcal{I}^{\max}\cup j)}, \ell, u) < b \quad \forall j \in (\text{supp}(a) \cap \mathcal{I}) \setminus \mathcal{I}^{\max}.$$

If such a set exists, all continuous variables can be canceled with their global bound without weakening the dual proof if they are never sufficient to prove infeasibility, i.e.,

$$\Delta_{\max}(a|_{\mathcal{I}^{\max}}, \ell', u') + \Delta_{\max}(a|_{\mathcal{I}\setminus\mathcal{I}^{\max}}, \ell, u) + \Delta_{\min}(a|_{\mathcal{N}\setminus\mathcal{I}}, \ell, u) \geq b.$$

In that case the assignment of one additional binary variable x_j with $j \notin \mathcal{I}^{\max}$ to its current local bound is always needed to prove infeasibility, independently from the assignment of the continuous variables.

Cancellation with Variable Bound Constraints Next to variable bounds that are explicitly given by the problem formulations, modern MIP solvers detect and use so-called *variable bound constraints* (Gamrath et al., 2019).

Definition 3 (Variable Bound Constraint (Gamrath et al., 2019))**.** *Let* $b^l, b^u \in \overline{\mathbb{R}}, c \in \mathbb{R}$ *with* $c \neq 0$*. A* variable bound constraint *has the form*

$$b^l \leq x + cy \leq b^u,$$

where x *is a continuous or integer variable and* y *a binary or integer variable.*

If $b^l = -\infty$ and b^u finite, the constraint gives an upper bound on x, while a lower bound on x is given if b^l is finite and $b^u = \infty$. A typical example of variable upper bound constraints are so-called big-M constraints $x \leq My$, where M is a (usually) very large constant and y is binary. Other typical use-cases are precedence constraints in start time variables in scheduling.

Let $\mathcal{N}_V \subseteq \text{supp}(a)$ be the index of variable for which a compatible variable bound constraint exists. A variable bound constraint on variable x_j with $a_j > 0$ is compatible if it defines an upper bound. Analogously, a variable lower bound constraint on x_j is compatible if $a_j < 0$. Moreover, let $\mathcal{V}$ be an arbitrary set of compatible variable bound constraints with respect to a containing for every j at least one variable bound constraint

$$b_j^l \leq x_j + c_{jk} y_k \leq b_j^u$$

such that y_k is binary or general integer, respectively, if x_j is general integer or continuous, respectively. Since y_k defines a bound on x_j, we identify variable bound constraint in $\mathcal{V}$ by the tuple (j, k).

Given the sets $\mathcal{N}_V$ and $\mathcal{V}$, one can construct an alternative proof $\phi_V(a)^\mathsf{T} x \geq \psi_V(b_0)$, where

$$\phi_V(a)_j = \begin{cases} 0, & \text{if } j \in \mathcal{N}_V \\ a_j - \sum\limits_{\substack{k \neq j \\ (k,j) \in \mathcal{V}}} a_k c_{kj}, & \text{if } j \notin \mathcal{N}_V \end{cases} \tag{9}$$

$$\psi_V(b_0) = b_0 - \sum_{\substack{j \in \mathcal{N}_V \\ (j,k) \in \mathcal{V}}} a_j \{b_j^l, b_j^u\}. \tag{10}$$

Proposition 4. *Let $a^\mathsf{T} x \geq b_0$ be a globally valid dual proof constraint and $\phi_V(a)^\mathsf{T} x \geq \psi_V(b_0)$ be a modified constraint following (9)– (10) with respect to the local bounds ℓ' and u'. For every $j \in \mathcal{N}_V$ with $(j,k) \in \mathcal{V}$ and $b_j^l = \ell'_j + c_{jk}\{\ell'_k, u'_k\}$ or $u'_j + c_{jk}\{\ell'_k, u'_k\} = b_j^u$ it holds*

$$\nu_{viol}(a, b_0, \ell', u') = \nu_{viol}(\phi_V(a), \psi_V(b_0), \ell', u').$$

The proof of the proposition follows by construction because only variable bound constraints that are satisfied with equality with respect to the local bounds are used for substitution.

By construction, the substitution of variables with variable bound constraints does not need to lead to a smaller support in general. However, it can improve the capability of propagating the modified infeasibility proof, as the following example shows.

Example 5. *Let $x_1 + x_2 - \frac{1}{2} y \geq 1$, $x_1, x_2 \in \{0,1\}$ with $\ell_j = \ell'_j = 0, u_j = u'_j = 1$, $y \in \mathbb{Z}$ with $0 \leq y \leq 4$. Moreover, let $y - 2x_1 \geq 0$ be a variable lower bound constraint of y. Applying (9) and (10) leads to*

$$\left.\begin{array}{rcrcrcl} & x_1 & + & x_2 & - & 0.5y & \geq & 1 \\ - & x_1 & & & + & 0.5y & \geq & 0 \end{array}\right\} \quad \Longrightarrow \quad x_2 \geq 1.$$

Let $x_1 + x_2 - z \geq 1$, $x_1, x_2 \in \{0,1\}$ with $\ell_j = \ell'_j = 0, u_j = u'_j = 1$, $z \in \mathbb{R}$ with $0 \leq z \leq M$ and $M > 2$. Moreover, let $z - Mw \geq 0$ be a variable lower bound

constraint of z with $w \in \{0,1\}$. Applying (9) and (10) leads to

$$\left.\begin{array}{rcrcrcrcl} x_1 & + & x_2 & - & z & & & \geq & 1 \\ & & & + & z & - & Mw & \geq & 0 \end{array}\right\} \implies w = 0.$$

□

In our implementation we make use of both described nonzero cancellation strategies. Firstly, we try to replace continuous and general integer variables with a variable bound constraint that is tight with respect to the local bounds such that no additional nonzero coefficient is introduced. Afterwards, we cancel continuous variables that contribute to the activity of the dual proof with their global bound following the procedure and case distinction described above.

3.1.2 Updating Procedure

During the tree search subproblems are pruned due to infeasibility or because of the corresponding LP is proven to exceed the solution value of the currently best known solution. This threshold is called the *cutoff bound*. In general, an LP that is proven to exceed the cutoff bound can be transformed into an infeasible LP by adding a constraint

$$c^\mathsf{T} x \leq c^\mathsf{T} x^\star, \tag{11}$$

where $x^\star$ denotes the currently best known solution. In the following, we will denote this solution as *incumbent solution*.

A valid proof that the LP relaxation exceeds the cutoff bound is given by every dual feasible solution (y, r) fulfilling

$$y^\mathsf{T} b + r^\mathsf{T}\{\ell', u'\} > c^\mathsf{T} x^\star \iff y^\mathsf{T} b + (c - y^\mathsf{T} A)^\mathsf{T}\{\ell', u'\} > c^\mathsf{T} x^\star, \tag{12}$$

where r_j denotes the reduced costs of x_j for all $j \in \mathcal{N}$. A globally valid dual proof constraint can be derived from (12):

$$(y^\mathsf{T} A - c)^\mathsf{T} x \geq y^\mathsf{T} b - c^\mathsf{T} x^\star. \tag{13}$$

This dual proof constraint has to be fulfilled by all improving solutions. Constraint (13) is globally valid since it is a convex combination of all rows $A_{i\cdot}$ and the cutoff constraint (11) scaled by -1.

Proposition 6. *Let $x^\star$ be the incumbent solution and $(y^\mathsf{T} A - c)x \geq y^\mathsf{T} b - c^\mathsf{T} x^\star$ an infeasibility proof for a boundexceeding LP with respect to local bounds ℓ' and u'. For every feasible solution $\bar{x}^\star$ with $c^\mathsf{T}\bar{x}^\star < c^\mathsf{T} x^\star$ the dual proof constraint can be strengthened to $(y^\mathsf{T} A - c)x \geq y^\mathsf{T} b - c^\mathsf{T}\bar{x}^\star$ by tightening the right-hand side. Afterwards, the strengthened constraint*

(i) still proves infeasibility with respect to ℓ' and u' and

(ii) is globally valid.

Proof. (i) $y^\mathsf{T} b + r^\mathsf{T}\{\ell', u'\} > c^\mathsf{T} x^\star > c^\mathsf{T}\bar{x}^\star$

(ii) By construction $(y^\mathsf{T} A - c)x \geq y^\mathsf{T} b - c^\mathsf{T}\bar{x}^\star$ is a convex combination of all rows $A_{i\cdot}$ and the strengthened constraint (11) scaled by -1. Hence, the dual proof constraint is globally valid.

□

In our implementation we use this updating scheme to strengthen dual infeasibility proofs derived from boundexceeding LPs each time a new incumbent solution is found.

3.2 Mixed-Integer Rounding

Applying *mixed integer rounding* (MIR) cuts (Nemhauser and Wolsey, 1988, 1990) was proven to be very successful (e.g., Bixby et al., 2004; Bonami et al., 2008) when generating strong cutting planes for mixed integer programming. It has been shown (Nemhauser and Wolsey, 1988) that Gomory's mixed integer cuts (Gomory, 1960) are implied by MIR cuts. The family of mixed integer rounding cuts has a wide range and covers structured as well as unstructured MIPs. In this section, we will focus on *complemented MIR* (Marchand and Wolsey, 2001) inequalities and we will discuss how this well-established type of mixed integer rounding function can be used to strengthen dual proofs.

3.2.1 c-MIR Inequalities

The MIR procedure strengthens a single linear inequality by rounding the coefficients of integer variables.

Theorem 7 (Cornuéjols (2008))**.** *Consider a mixed integer set defined by a single inequality*

$$\mathcal{S} := \{(x, z) \in \mathbb{Z}^n_+ \times \mathbb{R}^p_+ \mid a^\mathsf{T} x + g^\mathsf{T} y \leq b\}.$$

Let $f_0 = b - \lfloor b \rfloor$ *and* $f_j = a_j - \lfloor a_j \rfloor$*. Then the inequality*

$$\sum_{j=1}^{n} \left(\lfloor a_j \rfloor + \frac{(f_j - f_0)^+}{1 - f_0} \right) x_j + \frac{1}{1 - f_0} \sum_{j=1\,:\; g_j<0}^{p} g_j y_j \leq \lfloor b \rfloor \tag{14}$$

is a valid inequality for $conv(S)$*.*

For the proof of this theorem we refer to Cornuéjols (2008). Inequality (14) is called *MIR inequality*. Note, Theorem 7 defines the MIR inequality for nonnegative variables only. However, every variable with at least one finite bound can be shifted into the nonnegative space by complementing with one of its (finite) bounds. Complementing variables can be used to strengthen the MIR inequality, too. A variant of the MIR inequality combined with a special scaling parameter was proposed by Marchand and Wolsey (2001).

Theorem 8 (Marchand and Wolsey (2001)). *Consider a mixed integer set defined by a single inequality*

$$\mathcal{S} := \{(x, y) \in \mathbb{Z}^n_+ \times \mathbb{R}_+ \mid a^\mathsf{T} x + y \leq b,\ x_j \leq u_j\, \forall j \in \mathcal{I}\}.$$

Let (T, C) be a partition of $\mathcal{I}$ and $\delta > 0$. A complemented-MIR *(c-MIR) for S associated with (T, C) and $\delta > 0$ is obtained by complementing variables in C and dividing by δ before generating the MIR inequality.*

$$\sum_{j \in T} G\left(\frac{a_j}{\delta}\right) x_j + \sum_{j \in C} G\left(\frac{-a_j}{\delta}\right)(u_j - x_j) \leq \lfloor \beta_0 \rfloor + \frac{y}{\delta(1 - f_0)},$$

where $\beta_0 = (b - \sum_{j \in C} a_j u_j)/\delta$, $f_0 = \beta_0 - \lfloor \beta_0 \rfloor$, and $G(d) = \left(\lfloor d \rfloor + \frac{(f_d - f_0)^+}{1 - f_0}\right)$.

In the context of cutting planes, the MIR inequality is usually built from a nonnegative linear combination of valid inequalities. A heuristic how to derive a proper nonnegative linear combination that separates a given point (x, y) was proposed in Marchand and Wolsey (2001), where (x, y) is valid for the LP relaxation but violates the integrality conditions. Using the dual ray (y, r) yields a nonnegative linear combination which is valid for the global problem, see 2.2.1. Thus, the MIR procedure can be applied to the resulting proof constraint.

3.2.2 Applying c-MIR to Dual Proofs

In the following, we will define the partition (T, C) of $\mathcal{I}$, such that we always choose the closest bound (Marchand and Wolsey, 2001). Formally, if $\ell \leq x \leq u$ is the given reference point it follows $C := \{j \in \mathcal{I} \mid x_j > (u_j - \ell_j)/2\}$ and $T := \mathcal{I} \setminus C$.

In contrast to the classical separation procedure, where an LP-valid point should be separated from the convex hull, no such point is given when analyzing a subproblem with an infeasible LP relaxation. Due to this fact, there exists a certain degree of freedom when choosing a reference point that is used for the flow-cover separation or complementing the MIR inequality. The (arbitrary) reference point is used to compute the efficacy of the infeasibility proof before and after applying the c-MIR procedure. We aim to strengthen the initial proof of infeasibility such that it propagates well in the remainder of the search. Since we aim at using globally valid proofs, the global bounds need to be used for complementation.

The question remains which reference point to use for complementation. The LP solution of the root node or the current incumbent solution are not related to the current subproblem. In contrast to that, a natural choice that is related to the bounds contributing to the activity seem to be $\hat{x}$ with

$$\hat{x}_j = \begin{cases} u_j & \text{if } a_j < 0, \\ \ell_j & \text{if } a_j \geq 0. \end{cases}$$

However, always using the bounds contributing to the minimal activity according to the respective signs for complementation might in general lead to weak local proofs, as can be seen from the following lemma.

Lemma 9. *Let $a^\mathsf{T}x \leq b$ be a proof of local infeasibility with respect to ℓ' and u' and $\ell' \leq \tilde{x} \leq u'$ be a reference point minimizing the activity of $a^\mathsf{T}x$. Moreover, let $\delta = 1$ and (T, C) and (T', C') be two partitions of $\mathcal{I}$ such that $C' \subset C$ and $|C \setminus C'| = 1$. Further, let $k \in C \setminus C'$ with $a_k < 0$. Assume $(1 - f_k)u_k > u'_k$, $G(a_k) = \lfloor a_k \rfloor$, and $G(-a_k) = \lfloor -a_k \rfloor$, i.e., the fractionality of the coefficient is smaller than the fractionality of the right-hand side in both cases. Then, complementing with (T', C') yields a stronger local infeasibility proof with respect to $\tilde{x}$.*

Proof. Let $R(x) := \sum_{j \in T} G(a_j)x_j + \sum_{j \in C'} G(-a_j)(u_j - x_j)$ be the common part of both inequalities after applying the c-MIR procedure

$$R(x) + G(a_k)x_k \leq \lfloor \beta_0 \rfloor \tag{15}$$

$$\text{and} \qquad R(x) + G(-a_k)(u_k - x_k) \leq \lfloor \beta_0 \rfloor - a_k u_k. \tag{16}$$

We show that (15) has a smaller slack than (16) with respect to $\tilde{x}$ under the assumption that $(1 - f_k)u_k > u'_k$.

$$R(\tilde{x}) + G(a_k)\tilde{x}_k + s_1 = \lfloor \beta_0 \rfloor$$

$$\text{and} \qquad R(\tilde{x}) + G(-a_k)(u_k - \tilde{x}_k) + s_2 = \lfloor \beta_0 \rfloor - a_k u_k.$$

Assume $s_1 < s_2$:

$$\begin{aligned}
& G(a_k)\tilde{x}_k + s_1 - G(-a_k)(u_k - \tilde{x}_k) - s_2 = a_k u_k \\
\Leftrightarrow \quad & G(a_k)u'_k + s_1 - G(-a_k)(u_k - u'_k) - s_2 = a_k u_k \\
\Leftrightarrow \quad & G(a_k)u'_k - G(-a_k)(u_k - u'_k) > a_k u_k \\
\Leftrightarrow \quad & (\lfloor a_k \rfloor + \lfloor -a_k \rfloor)u'_k - \lfloor -a_k \rfloor u_k > a_k u_k \\
\Leftrightarrow \quad & (\lfloor a_k \rfloor - \lceil a_k \rceil)u'_k + \lceil a_k \rceil u_k > a_k u_k \\
\Leftrightarrow \quad & -u'_k + \lceil a_k \rceil u_k > a_k u_k \\
\Leftrightarrow \quad & (1 - f_k)u_k > u'_k
\end{aligned}$$

□

Proposition 10. *For arbitrary but finite lower bounds $\ell_j \neq 0$ Lemma 9 generalizes to*

$$u'_k < (1 - f_k)u_k + f_k \ell_k.$$

The proof of the above proposition follows the proof of Lemma 9 whereby inequality (15) changes to $R(x) + G(a_k)(x_k - \ell_k) \leq \lfloor \beta_0 \rfloor - a_k \ell_k$. Based on the

above proposition we propose to use the reference point $\bar{x}$ with

$$\bar{x}_j = \begin{cases} u_j & \text{if } a_j > 0 \text{ and } \ell'_j \geq \frac{u_j+\ell_j}{2}, \\ u_j & \text{if } a_j < 0 \text{ and } u'_j \geq \frac{u_j+\ell_j}{2}, \\ \ell_j & \text{if } a_j > 0 \text{ and } \ell'_j < \frac{u_j+\ell_j}{2}, \\ \ell_j & \text{if } a_j < 0 \text{ and } u'_j < \frac{u_j+\ell_j}{2}, \end{cases}$$

instead of using the global upper or lower bound depending on the sign of the respective coefficients, when applying the c-MIR procedure to dual infeasibility proofs. This heuristic relaxes the result of Lemma 9 and Proposition 10 by dropping the dependency on the fractionality of the coefficients.

3.3 Filtering

Conflict graph analysis and dual proof analysis share the initial reason of infeasibility $y^\mathsf{T} b + r^\mathsf{T}\{\ell, u\} > 0$ as a common starting point (see Section 2). Both techniques follow different paths such that the resulting constraints are of very different nature. Conflict constraints derived from analyzing the conflict graph rely on combinatorial arguments. Consequently, these constraints are not challenging with respect to their numerical properties and expected to be short. Here, the shorter the conflict constraint is the stronger it is (Iwama, 1997). This gets easily clear when looking at a conflict constraint from the SAT perspective, i.e., interpreting it as set of clauses which cannot be all satisfied at the same time. Thus, adding an additional clause weakens the proof. In contrast to that, constraints derived from dual proof analysis purely rely on the numerical conditions of the constraints of the MIP and the properties of the dual proof derived by solving the infeasible LP relaxation. Therefore, these proofs can be expected to be numerically more challenging and more dense than those derived from analyzing the conflict graph. In the case of dual proofs, "sparser is better" does not hold anymore. Therefore, it is essential to pick the most promising infeasibility proofs derived from dual proof analysis. In contrast to conflict constraints derived by conflict graph analysis, dual proof constraints rely on the actual coefficients of the initial proof because these coefficients are needed to define the actual constraint. We filter out dual proof constraints that might lead to numerically unstable propagation steps. This is done by restricting the maximal absolute range of all coefficients. For a given constraint $a^\mathsf{T} x \geq b$ we consider the minimal and maximal absolute coefficient which will be denoted by

$$a_{\min} := \min\{|a_j| \mid j \in \operatorname{supp}(a)\} \quad \text{and} \quad a_{\max} := \max\{|a_j| \mid j \in \operatorname{supp}(a)\}.$$

In our implementation we discard all proof constraints for which $a_{\max}/a_{\min}$ exceeds a threshold of 1e+8.

Dual infeasibility proofs derived by aggregating constraints $A_{i\cdot}$ weighted by a dual ray (y, r) are often dense; but it is a priori not clear whether sparse,

i.e., short, dual proofs are to prefer. The reason why longer proof constraints are assessed to be inferior against shorter proof constraints is twofold and hold for general constraints as well. On the one hand, it is to expect that fewer bound changes on the support of a short constraint are necessary to derive new deductions. Therefore, constraints with a small support are expected to propagate "earlier" than constraints with a large support. Moreover, there are also technical reasons to prefer constraints with a small support. Firstly, dense constraints consume more memory. Secondly, certain types of constraints are much more computational costly to propagate than others. For example, consider a general linear constraint involving arbitrary variables and coefficients–which is most likely the case for a dual proof. Propagating a general linear constraint $a^\mathsf{T} x \geq b$, e.g., by activity based bound tightening (Brearley et al., 1975), might be computational costly with $\mathcal{O}(|\operatorname{supp}\{a\}|)$. In contrast to that, set covering constraints which contain binary variables only are very efficient to propagate in $\mathcal{O}(1)$ when using the 2-literal watching scheme (Moskewicz et al., 2001).

To not systematically abolish dense dual proof we propose a dynamic threshold on the size of its support to decide whether the proof should be accepted and maintained in the remainder of the search or immediately rejected. To this end, we consider the average density of all model constraints $A_{i\cdot}$ which will be denoted by

$$\operatorname{supp}_\emptyset(A) := \frac{\sum_{A_{i\cdot} \in A} |\operatorname{supp}(A_{i\cdot})|}{|A|}.$$

Further, let $\mathcal{C}$ be the set of all dual proofs currently maintained and $a^\mathsf{T} x \geq b$ a new dual proof. We accept the constraint if,

$$\operatorname{supp}_\emptyset(\mathcal{C} \cup \{a^\mathsf{T} x \geq b\}) \leq \max\{\alpha \operatorname{supp}_\emptyset(A),\ \beta|\mathcal{N}|\},$$

with $\alpha, \beta > 0$. By this, we do not restrict the density of a single constraint but rather the average density over all maintained dual proofs. Since dense dual proof constraints are expected to cover a larger variety of reasons of infeasibility but are computational costly during domain propagation, we try to balance both properties. With this strategy we aim to adjust the threshold dynamically depending on the actual density of all accepted dual proof constraints. Thus, if sufficiently many sparse dual proof constraints are maintained at the current point in time, we are willing to spent some more effort on dense dual proof constraints as long as the average expected effort does not exceed the dynamic threshold. Note that dual proof constraints with a support of size one are immediately transformed into bound change and therefore not included in the average density.

4 Computational Study

In this section we present an intense computational study on general MIP problems investigating the computational impact of the different ways of analyzing

infeasibility in MIP presented in this paper. To evaluate the individual impact of all the different technique presented in this paper, we will analyze each of them in detail.

In Section 4.1, we compare the impact of conflict graph analysis and dual proof analysis. Moreover, we present computational results where both techniques were simultaneously within the state-of-the-art MIP solvers `SCIP` and `FICO Xpress`. To the best of our knowledge, these are the first implementations of a combined approach within a single solver. In Section 4.2– 4.4 we present individual computational results for all enhancement techniques presented in Section 3.

All experiments of Section 4.2 were performed with the academic MIP solver `SCIP` 5.1.0 using `SoPlex` 3.1.1 as LP solver (Gleixner et al., 2017). The experiments in Section 4.1 were conducted with `SCIP` and `FICO Xpress` 8.6.3 FICO Xpress Optimizer. To evaluate the generated data the *interactive performance evaluation tool* (IPET) (Hendel, 2019) was used. The `SCIP` experiments were run on a cluster of identical machines equipped with Intel Xeon E5-2690 CPUs with 2.6 GHz and 128 GB of RAM. The `FICO Xpress` experiments were run on a cluster of identical machines equipped with Intel Xeon E5-2640 CPUs with 2.4 GHz and 64 GB of RAM. A time limit of 7200 seconds was set in either case.

As test set we used the newly released benchmark set of MIPLIB 2017[1].

To account for the effect of performance variability (Danna, 2008; Lodi and Tramontani, 2013) all `SCIP` experiments were performed with three different global random seeds; `FICO Xpress` experiments were run on three different permutations of the problem. Determinism is preserved because `SCIP` and `FICO Xpress` use pseudo-random number generators only. Every pair of MIP problem and seed/permutation is treated as an individual observation, effectively resulting in a test set of 720 instances. We will use the term "instance" when referring to a problem-seed combination or a specific permutation of a problem.

In the following, we present aggregated results for every experiment containing the number of solved instances ($\mathbf{S}$) and the absolute and relative solving times in seconds ($\mathbf{T}$ and $\mathbf{T_Q}$) and number of explored branch-and-bound nodes ($\mathbf{N}$ and $\mathbf{N_Q}$). Absolute numbers are always given for the baseline setting, whereas relative numbers are shown for all other settings. To aggregate the individual observations a shifted geometric mean (Achterberg, 2007b) is used, where a shift of 1 and 100 is used for solving time and nodes, respectively.

Relative solving times of setting s are defined by the quotient t_s/t_b, where t_s is the absolute solving time of setting s and t_b is the absolute solving time of setting b that is used as a baseline. An analogous definition holds for explored branch-and-bound nodes. Thus, numbers less then one implies that the setting s is superior and a number greater than one implies that it is inferior to the baseline setting b. Note, that a relative solving time t_s/t_b corresponds to a

[1] `http://miplib.zib.de`

Table 1: LP-based infeasibility analysis parameters of the different settings. Note, infeasibility due to propagation remains enabled in all settings.

Parameter	Description	Type	Value			
			`nolpinf`	`confgraph`	`dualproof`	`combined`
`useboundlp`	Analyze bound exceeding LPs.	char	`o`	`c`	`d`	`b`
`useinflp`	Analyze infeasible LPs.	char	`o`	`c`	`d`	`b`

speedup factor of t_b/t_s.

Besides the results for the benchmark set of MIPLIB 2017[2], that are denoted by *all*, the tables state the impact on instances that are *affected*. An instance is called affected, if it can be solved by at least one setting within the time limit and the number of tree nodes differs between settings. Further, the subset of affected instances is grouped into a hierarchy of increasingly harder classes "$\geq k$". Class "$\geq k$" contains all instances for which at least one setting needs at least k seconds and can be solved by at least one setting within the time limit. As explained by Achterberg and Wunderling (2013), this excludes instances that are "easy" for all settings in an unbiased manner. Detailed tables with instance-wise computational results regarding `SCIP` can be found in the electronic supplement ().

4.1 General Overview

To analyze the overall computational impact of infeasibility analysis of infeasible LPs or bound exceeding LPs within `SCIP`, we consider four different configurations: disabled infeasibility analysis (`nolpinf`), conflict graph analysis (`confgraph`) or dual proof analysis (`dualproof`) solely, and using a combination of both techniques (`combined`). Note, all four settings only differ in the way how infeasible and bound exceeding LP relaxations are analyzed. Infeasibility due to propagation remains unchanged, see Table 1 for the different configurations of infeasibility analysis considered in this paper. In the following, `nolpinf` is used as a baseline.

Both settings using dual proof analysis (`dualproof` and `combined`) were using all techniques described in Section 3. Aggregated results on all four settings are shown in Table 2.

For every LP relaxation considered for infeasibility analysis, at most 10 conflict constraints and at most 2 dual proof constraints are stored. As suggested by Achterberg (2007a), we store the 10 most promising conflict constraints generated by All-FUIP. When dual proof analysis is enabled, we store the modified and strengthened dual proof and the result of applying c-MIR to that proof constraints, if the procedure succeeds (see Section 4.3).

To maintain all conflict constraints and dual proofs we use a pool-based approach (Witzig et al., 2017). Here, the maximal number of conflict constraints

[2]Excluding instances where at least one settings finished with numerical violations.

Table 2: Aggregated computational results on MIPLIB 2017 benchmark. Relative changes by at least 5% are highlighted in bold and blue (**improvement**) or italic and red (*deterioration*).

		nolpinf			dualproof			confgraph			combined		
		S	**T**	**N**	**S**	**T$_Q$**	**N$_Q$**	**S**	**T$_Q$**	**N$_Q$**	**S**	**T$_Q$**	**N$_Q$**
all	711	324	1482	7314	317	0.998	0.975	339	**0.916**	**0.876**	342	**0.897**	**0.851**
affected	283	254	532	9885	247	0.993	**0.943**	269	**0.799**	**0.707**	272	**0.759**	**0.677**
≥10	274	245	620	11046	238	0.991	**0.941**	260	**0.795**	**0.702**	263	**0.753**	**0.671**
≥100	228	199	1113	18866	192	0.977	**0.932**	214	**0.760**	**0.659**	217	**0.699**	**0.620**
≥1000	151	122	2387	49751	115	1.009	0.967	137	**0.697**	**0.575**	140	**0.628**	**0.545**

maintained at the same time when using `nolpinf`, `confgraph`, and `combined` was limited to 10.000. When using `dualproof` and `combined`, at most 100 dual proofs of infeasible and 75 dual proofs of bound exceeding LPs are maintained simultaneously. These pool sizes turned out to have the best trade-off between additional information and time spent in evaluating these constraints. A similar observation regarding the performance of dual proof analysis within `Gurobi` was made by Achterberg et al. (2016). If one of the pools reached its limit, the constraint that did not lead to new bound deductions for the longest time is removed before adding the new conflict or dual proof constraint. This procedure corresponds to an aging scheme as it is used in SAT (Moskewicz et al., 2001).

Our computational results indicate that 80 % of the instances that can be solved by at least one of the four settings are affected by analyzing infeasible and bound exceeding LP relaxations.

All three variants of LP infeasibility analysis are superior to `SCIP` without any of these techniques. Over all subsets of instances shown in Table 2 we observe a clear ordering. `combined` is superior to `confgraph` and `confgraph` is superior to `nolpinf` with respect to both solving time and tree size. At the same time `dualproof` is clearly inferior to both `confgraph` and `combined` but superior to `nolpinf` regarding tree size. Surprisingly, `dualproof` has hardly any impact regarding the solving time compared to `nolpinf`. However, the reduction of the tree size on affected instances by using dual proof analysis solely is 6.7 %. Here, our observations differ from those reported by Achterberg et al. (2016), where dual proof analysis leads to a reduction of solving time by 6 % on affected instances. The disparity in the observations may be caused by different test sets and a different implementation in `Gurobi`, which is tuned to relay solely on dual proof analysis.

The version that combines both conflict graph analysis and dual proof analysis (`combined`) solves 18 additional instances compared to `nolpinf` and 3 more instances compared to `confgraph`. In our experiments the impact on the tree size by applying conflict analysis is consistently larger than the impact on the overall solving time. This observation is to expect because every additional constraints derived from either of the presented techniques and considered in the remainder of the search increases the time spent at every node of the search tree, i.e., during domain propagation.

Table 3: Aggregated computational results on MIPLIB 2017 benchmark for FICO Xpress. Relative changes by at least 5% are highlighted in bold and blue (**improvement**) or italic and red (*deterioration*).

		nolpinf			confgraph			combined		
		S	**T**	**N**	**S**	**T_Q**	**N_Q**	**S**	**T_Q**	**N_Q**
all	693	515	77.5	7024	530	**0.926**	**0.876**	529	**0.909**	**0.856**
affected	350	330	225	70842	344	**0.847**	**0.784**	343	**0.833**	**0.772**
≥10	306	286	394	90997	300	**0.832**	**0.779**	299	**0.815**	**0.763**
≥100	224	204	948	137229	218	**0.780**	**0.728**	217	**0.767**	**0.719**
≥1000	118	98	2794	287999	112	**0.706**	**0.648**	111	**0.699**	**0.642**

On affected instances we observed a success rate of 83 % for `combined`, i.e., the portion of analyzed infeasible and bound exceeding LP relaxations from which at least one conflict constraint or dual proof could be derived. `dualproof` and `confgraph` yield a success rate of 67 % and 34 %, respectively. The small success rate of `confgraph` is due a very strict limit on the size of the support to accept a conflict constraint. Due to the combinatorial nature of conflict constraints it is known that shorter is always better. Moreover, on the set of affected instances bound exceeding LP relaxations were analyzed 3.3 to 4.4 times as often as infeasible LP relaxations.

On instances that are known to be infeasible `confgraph` and `combined` perform best while reducing the solving time by roughly 65 %. In contrast to that, `dualproof` performs like `nolpinf`. Hence, we conclude that on infeasible instances conflict graph analysis is superior to dual proof analysis. Note, MIPLIB 2017 contains only 8 infeasible instances. Thus, our observation is based on a small test set of 24 instances. On the other hand, on the set of nontrivial feasibility instances[3] `confgraph` performs worse compared to the set of affected instances. Here, `confgraph` improves the solving time by 10.9 % (affected: 20.1 %) only. In contrast to that, `combined` improves the solving time by 24.5 % (affected: 24.1 %) and `dualproof` leads to slight slowdown of 2.3 %. Consequently, we conclude that the combination of both conflict graph analysis and dual proof analysis is particularly beneficial on feasibility instances.

Computational Impact of Infeasibility Analysis in FICO Xpress. Table 2 shows the overall computational impact of using or a combination of conflict graph and dual proof analysis (`combined`), neither of the two (`nolpinf`), or only conflict graph analysis (`confgraph`) within the state-of-the-art commercial MIP solver `FICO Xpress`. Using only dual proof analysis is not easily possible within the current `FICO Xpress` implementation.

Independent of the actual experiments, there are some principal differences between the `SCIP` results and those of `FICO Xpress`. We observed a significantly larger number of memouts. This has three major reasons. Firstly, the machines

[3]The set of instances with a nontrivial objective function for which the gap between the final dual bound at the root node and the best known solution is below 0.1 %.

on which FICO Xpress was run only have half the RAM compared to those of SCIP. Secondly, FICO Xpress was run with 20 threads, which leads to 64 MIP tasks being created and therefore 64 copies of the problem being taken, see Berthold et al. (2018a). Thirdly, the node throughput on hard problems was much larger, see the "$\geq k$" row. This leads to a much larger search tree being created (and partly is a consequence of the larger thread number). Instances for which at least one variant of FICO Xpress hit the memory limit were removed, which is the reason for the smaller number of instances in the testbed.

FICO Xpress solved significantly more instances than SCIP. This increases the number of affected instances. At the same time, FICO Xpress solved a larger share of instances at the root node. This decreases the number of affected instances. As a consequence, the set of affected instances differs a lot between the two solvers, although the number (330 versus 283) is only slightly different.

Nevertheless, our observations show quite similar tendencies. As for SCIP, conflict graph analysis gives a clear performance boost to the solver and a combined approach of conflict graph and dual proof analysis gives the best performance. Altogether, we observe a speed-up of 9%, when applying infeasibility analysis techniques in FICO Xpress. On affected instances, the speed-up is roughly 17%, and goes up to 30% on the hardest models. Those number are smaller than the corresponding numbers for SCIP (10% on all, 24% on affected, 37% on the hardest). Next to the different sets of affected instances, there is also a technical explanation for this deviation: The balancing between local cutting and infeasibility analysis differs between the two solvers. FICO Xpress separates significantly more cuts that are only locally valid, which constrains the applicability and the impact of infeasibility analysis. As a consequence, only 65% of the solved instances are affected (compared to 80% for SCIP).

As for SCIP, the reductions in the number of branch-and-bound nodes are larger than the time reductions and by using infesibility analysis techniques, more instances can be solved. However, in marked difference, using dual proof analysis on top of conflict graph analysis does not lead to more solved instances, but to one instance less being solved. Nevertheless, the consistently better running times of the combined setting lead us to the conclusion that an approach that conducts both techniques is preferable. Consequently, this is the default setting of FICO Xpress.

4.2 Presolving and Strengthening Techniques

To evaluate the computational impact within SCIP of the nonzero cancellation techniques described in Section 3.1.1 and the strengthening of dual proof constraints based on the current incumbent solution described in Section 3.1.2, we disable both features and compare it to combined. In the following, we will refer to both variable cancellation strategies by cancellation, whereas the individual procedures using global variables bound and variable bound constraints, respectively, will be called bound-cancellation and vbound-cancellation, respectively. Moreover, we will refer to the strengthening step by update.

Table 4: Aggregated computational results on instances affected by applying all cancellation techniques. Relative changes by at least 5% are highlighted in bold and blue (**improvement**) or italic and red (*deterioration*).

		`cancellation` disabled			`combined`		
		S	**T**	**N**	**S**	**T$_Q$**	**N$_Q$**
all	714	345	1334	6349	344	0.996	0.996
affected	146	146	596	26147	145	0.983	0.981
≥10	144	144	634	27527	143	0.983	0.978
≥100	119	119	1135	47143	118	0.973	0.961
≥1000	78	78	2313	144897	77	0.965	0.958

Variable Cancellation. `cancellation` affects 146 instances that can be solved by at least one setting, see Table 4. On these instances, our computational experiments indicate a slight performance improvement with respect to solving time of up to 3.5 % and a reduction on the tree size by up to 4.2 % gained by activating `cancellation`. The observed improvements gained by `cancellation` are small but consistent over all increasingly hard groups of affected instances. In our experiments we observed a speed up (slowdown) of at least 5 % on 36 % (24 %) of the the affected instances when `cancellation` is enabled. The number dual proof constraints rejected due to restrictions on the size of the support decreased by 73.6 % when `cancellation` is enabled.

Within `cancellation`, `bound-cancellation` and `vbound-cancellation` were successfully applied on roughly 3 % and 7 % of all analyzed infeasible LP relaxations.

When disabling `bound-cancellation` solely 76 instances are affected. Note, in our implementation nonzero variable cancellation with their global variable bounds is applied to continuous variables only. On the set of affected instances, using `bound-cancellation` improves the solving time by up to 6.2 %. The reduction of the tree size is up to 7.7 %.

Compared to `bound-cancellation`, deactivating `vbound-cancellation` affects 115 instances. When `vbound-cancellation` is enabled, the impact with respect to solving time is between 2.2 % (affected) and 4.1 % ("≥1000"). Although we observe a minor reduction of solving time on the set of affected instances, `vbound-cancellation` reduces the tree size by 4.6 % to 8.1 %. Since `vbound-cancellation` substitutes continuous variables with variable bound constraints defined by general integer or binary variables, it is to expect that the time spent for domain propagation slightly increases for this constraints. However, the modified dual proof constraints propagate better, i.e., lead to more additional variable bound deductions, since less continuous variables are involved.

In our implementation, `vbound-cancellation` is called before `bound-cancellation`. Consequently, every success of `vbound-cancellation` might have an immediate impact on the success rate of `bound-cancellation`. In our computational study, we observe that the success rate with respect to applied cancellations of the latter reduced by 15 % when `vbound-cancellation` is called

Table 5: Aggregated computational results on instances affected by updating dual bound exceeding proofs. Relative changes by at least 5% are highlighted in bold and blue (**improvement**) or italic and red (*deterioration*).

		`update` disabled			`combined`		
		S	**T**	**N**	**S**	**T$_Q$**	**N$_Q$**
all	715	347	1328	6415	345	1.001	0.986
affected	161	160	566	25898	158	1.005	0.990
≥10	160	159	580	26124	157	1.005	0.990
≥100	136	135	932	42217	133	1.003	0.981
≥1000	80	79	2151	165943	77	1.044	1.032

first.

Updating Dual Proofs of Bound Exceeding LPs. Dual proof constraints derived from LP relaxations exceeding the current cutoff bound can be strengthened whenever a new incumbent solution is found. We will refer to this strengthening step by `update`. Enabling or disabling `update` affects 161 instances in our test set, see Table 5. The reduction of the solving time and tree size on the complete set of affected is neutral. On the hardest instances `update` leads to a slowdown of 4.4 %. During the whole solving process `update` was applied to 35 % of all accepted dual proof constraints derived from a bound exceeding LP relaxation. In our experiments we observed a speed up (slowdown) of at least 5 % on 31 % (29 %) of the affected instances when `update` is enabled.

In contrast to the previously discussed strengthening techniques, `update` explicitly incorporate the objective function and the cutoff bound. Thus, it is to expect that this kind of dual proof constraints especially works well if the primal bound incrementally improve over time. On the other hand, if either a near optimal solution is found right in the beginning or all solutions are found at the very end of the tree search, `update` might have a noticeable impact an the overall solving process. Consequently, the impact of this strengthening technique strongly depends on the progress on the primal side. The better the solutions found early in the solving process, the less impact by activating `update` is to expect.

4.3 Mixed Integer Rounding

In order to generate an alternative proof of infeasibility we separate a c-MIR (see Section 3.2.1) inequality based in the dual proof constraint after applying the techniques discussed in Section 3.1. In the following, we will refer to this procedure as `mir-procedure`. To decide whether the additionally separated inequality should be accepted for further considerations its efficacy

$$\frac{b - a^{\mathsf{T}}\bar{x}}{\max\{0.000001, \|a\|_2\}}$$

Table 6: Aggregated computational results on instances affected by applying the c-MIR procedure. Relative changes by at least 5% are highlighted in bold and blue (**improvement**) or italic and red (*deterioration*).

		`mir-procedure` disabled			`combined`		
		S	**T**	**N**	**S**	**T$_Q$**	**N$_Q$**
all	715	346	1335	6373	345	0.996	0.992
affected	70	69	1048	81118	68	**0.932**	0.950
$\geq$10	70	69	1048	81118	68	**0.932**	0.950
$\geq$100	63	62	1518	114782	61	**0.895**	**0.933**
$\geq$1000	50	49	2240	166041	48	**0.874**	**0.931**

with respect to the reference point $\bar{x}$ discussed in Section 3.2.2 is taken into account. A positive efficacy gives the normalized violation, otherwise the normalized redundancy. Thus, a larger efficacy is preferred and we only accept the separated inequality if the corresponding efficacy is larger than the efficacy of the dual proof constraint from which it was derived.

In our computational study `mir-procedure` affects 70 instances, while reducing the solving time and tree size by 6.8 % and 5 %, respectively. Our computational results indicate, that harder the instances are the more benefit is gained by applying `mir-procedure`. On the set of affected instances, `mir-procedure` is successfully applied to 12 % of all analyzed infeasible LP relaxations. In our experiments we observed a speed up (slowdown) of at least 5 % on 25 % (25 %) of the affected instances when `mir-procedure` is enabled.

4.4 Filtering

To distinguish between dual proof constraints that are expected to be promising in the remainder of the search and those that might lead to numerical troubles or an increase of the time spent during propagation, we apply a filtering step as described in Section 3.3. In the following, we refer to this step by `filtering`. Both filtering due to numerical conditions and size of the support are applied after all previously discussed techniques are applied.

Our computational study indicates that `filtering` has the most significant impact among all presented techniques, see Table 7. `filtering` affects 135 instances and leads to a reduction of solving time and tree size by 4.5 % and 4.6 %, respectively, on these instances. The harder instances are the larger is the impact of `filtering`. On the group of hardest instances, the solving time can be reduced by 8.9 %. Moreover, it leads to 7 additionally solved instances. In our experiments we observed a speed up (slowdown) of at least 5 % on 35 % (33 %) of the affected instances when `filtering` is enabled.

By applying `filtering`, 34 % of the dual proof constraints were rejected, where less than 1 % were rejected due to numerical properties, i.e., the ratio of the maximal and minimal nonzero coefficient in absolute value were larger than 1e+8. Thus, the most important filtering criterion is the size of the support. As described in Section 3.3, we consider the ratio of the average number of nonzeros

Table 7: Aggregated computational results on instances affected by filtering dual proofs. Relative changes by at least 5% are highlighted in bold and blue (**improvement**) or italic and red (*deterioration*).

		`filtering` disabled			`combined`		
		S	**T**	**N**	**S**	**T$_Q$**	**N$_Q$**
all	715	338	1342	6378	345	0.991	0.992
affected	135	127	539	16806	134	0.955	0.954
$\geq$10	134	126	555	16932	133	0.955	0.954
$\geq$100	107	99	1117	34141	106	**0.933**	**0.916**
$\geq$1000	69	61	2581	101951	68	**0.911**	**0.899**

of all maintained dual proof constraints and all model constraints.

5 Conclusion and Outlook

In this paper, we studied the combination of conflict graph analysis and dual proof analysis for infeasible and bound exceeding LP relaxations within a single solver. Our computational results indicate that both improve the performance of MIP solvers, with conflict graph analysis being the more powerful technique. A combined approach performed best, for both solvers, `SCIP` and `FICO Xpress`, that we used for our experiments.

Furthermore, we introduced three enhancements for dual proof analysis: presolving via variable cancellation, strengthening by applying mixed integer rounding functions, and a filtering mechanism. All of those led to clear performance improvements. A method to update bound-exceeding proofs, however, did not benefit the solver.

We conclude that infeasibility analysis plays an important role for the solution of mixed integer programming problems and that a combination of different techniques is worthwhile. The generalization of the presented methods towards the field of mixed integer nonlinear programming provides an interesting line for future research. First results in this direction have recently been published in Witzig et al. (2019).

Acknowledgments

The work for this article has been conducted within the Research Campus Modal funded by the German Federal Ministry of Education and Research (fund number 05M14ZAM).

References

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

T. Achterberg. Constraint integer programming, 2007b.

T. Achterberg and T. Berthold. Hybrid branching. In W.-J. van Hoeve and J. N. Hooker, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 6th International Conference, CPAIOR 2009*, volume 5547 of *Lecture Notes in Computer Science*, pages 309–311. Springer Berlin Heidelberg, May 2009.

T. Achterberg and R. Wunderling. Mixed integer programming: Analyzing 12 years of progress. In *Facets of combinatorial optimization*, pages 449–481. Springer, 2013.

T. Achterberg, R. E. Bixby, Z. Gu, E. Rothberg, and D. Weninger. Presolve reductions in mixed integer programming. Technical Report 16-44, ZIB, Takustr. 7, 14195 Berlin, 2016.

T. Berthold. *Heuristic algorithms in global MINLP solvers.* PhD thesis, 2014.

T. Berthold and A. M. Gleixner. Undercover: a primal MINLP heuristic exploring a largest sub-MIP. *Mathematical Programming*, 144(1–2):315–346, 2014.

T. Berthold and G. Hendel. Shift-And-Propagate. *Journal of Heuristics*, 21(1): 73–106, 2015.

T. Berthold, T. Feydy, and P. J. Stuckey. Rapid learning for binary programs. In A. Lodi, M. Milano, and P. Toth, editors, *Proc. of CPAIOR 2010*, volume 6140 of *LNCS*, pages 51–55. Springer Berlin Heidelberg, June 2010.

T. Berthold, J. Farmer, S. Heinz, and M. Perregaard. Parallelization of the FICO Xpress-Optimizer. *Optimization Methods and Software*, 33(3):518–529, 2018a.

T. Berthold, P. J. Stuckey, and J. Witzig. Local rapid learning for integer programs. Technical Report 18-56, ZIB, Takustr. 7, 14195 Berlin, 2018b.

E. R. Bixby, M. Fenelon, Z. Gu, E. Rothberg, and R. Wunderling. MIP: Theory and practice—closing the gap. In *IFIP Conference on System Modeling and Optimization*, pages 19–49. Springer, 1999.

R. E. Bixby, M. Fenelon, Z. Gu, E. Rothberg, and R. Wunderling. Mixed-integer programming: A progress report. In *The sharpest cut: the impact of Manfred Padberg and his work*, pages 309–325. SIAM, 2004.

P. Bonami, G. Cornuéjols, S. Dash, M. Fischetti, and A. Lodi. Projected Chvátal–Gomory cuts for mixed integer linear programs. *Mathematical Programming*, 113(2):241–257, 2008.

A. Brearley, G. Mitra, and H. Williams. Analysis of mathematical programming problems prior to applying the simplex algorithm. *Mathematical Programming*, 8(1):54–83, 1975.

G. Cornuéjols. Valid inequalities for mixed integer linear programs. *Mathematical Programming*, 112(1):3–44, 2008.

R. J. Dakin. A tree-search algorithm for mixed integer programming problems. *The Computer Journal*, 8(3):250–255, 1965.

E. Danna. Performance variability in mixed integer programming. In *Workshop on Mixed Integer Programming, Columbia University, New York*, volume 20, 2008.

B. Davey, N. Boland, and P. J. Stuckey. Efficient intelligent backtracking using linear programming. *INFORMS Journal of Computing*, 14(4):373–386, 2002.

J. P. Dickerson and T. Sandholm. Throwing darts: Random sampling helps tree search when the number of short certificates is moderate. In *Sixth Annual Symposium on Combinatorial Search*, 2013.

P. Domschke, B. Geißler, O. Kolb, J. Lang, A. Martin, and A. Morsi. Combination of nonlinear and linear optimization of transient gas networks. *INFORMS Journal on Computing*, 23(4):605–617, 2011.

J. Farkas. Theorie der einfachen ungleichungen. *Journal für die reine und angewandte Mathematik*, 124:1–27, 1902. URL `http://eudml.org/doc/149129`.

FICO Xpress Optimizer. `https://www.fico.com/de/products/fico-xpress-optimization`.

G. Gamrath, A. Gleixner, T. Koch, M. Miltenberger, D. Kniasew, D. Schlögel, A. Martin, and D. Weninger. Tackling industrial-scale supply chain problems by mixed-integer programming. Technical Report 16-45, ZIB, Takustr. 7, 14195 Berlin, 2016.

G. Gamrath, T. Berthold, S. Heinz, and M. Winkler. Structure-driven fix-and-propagate heuristics for mixed integer programming. *Mathematical Programming Computation*, Apr 2019. ISSN 1867-2957. doi: 10.1007/s12532-019-00159-1. URL `https://doi.org/10.1007/s12532-019-00159-1`.

M. L. Ginsberg. Dynamic backtracking. *Journal of Artificial Intelligence Research*, 1:25–46, 1993.

A. Gleixner, L. Eifler, T. Gally, G. Gamrath, P. Gemander, R. L. Gottwald, G. Hendel, C. Hojny, T. Koch, M. Miltenberger, B. Müller, M. E. Pfetsch, C. Puchert, D. Rehfeldt, F. Schlösser, F. Serrano, Y. Shinano, J. M. Viernickel, S. Vigerske, D. Weninger, J. T. Witt, and J. Witzig. The SCIP Optimization Suite 5.0. Technical Report 17-61, ZIB, Takustr. 7, 14195 Berlin, 2017.

R. Gomory. An algorithm for the mixed integer problem. Technical report, RAND CORP SANTA MONICA CA, 1960.

M. Guignard and K. Spielberg. Logical reduction methods in zero-one programming—minimal preferred variables. *Operations Research*, 29(1):49–74, 1981.

S. Heinz and J. C. Beck. Reconsidering mixed integer programming and MIP-based hybrids for scheduling. In *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*, pages 211–227. Springer, 2012.

G. Hendel. IPET interactive performance evaluation tools. `https://github.com/GregorCH/ipet`, 2019.

B. Hiller, T. Koch, L. Schewe, R. Schwarz, and J. Schweiger. A system to evaluate gas network capacities: Concepts and implementation. *European Journal of Operational Research*, 270(3):797 – 808, 2018.

K. Iwama. Complexity of finding short resolution proofs. In I. Prívara and P. Ružička, editors, *Mathematical Foundations of Computer Science 1997*, pages 309–318, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg. ISBN 978-3-540-69547-9.

Y. Jiang, T. Richards, and B. Richards. No-good backmarking with min-conflict repair in constraint satisfaction and optimization. In *PPCP*, volume 94, pages 2–4. Citeseer, 1994.

F. Kılınç Karzan, G. L. Nemhauser, and M. W. P. Savelsbergh. Information-based branching schemes for binary linear mixed-integer programs. *Mathematical Programming Computation*, 1(4):249–293, 2009.

A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

H. Lee, J. M. Pinto, I. E. Grossmann, and S. Park. Mixed-integer linear programming model for refinery short-term scheduling of crude oil unloading with inventory management. *Industrial & Engineering Chemistry Research*, 35(5):1630–1641, 1996.

A. Lodi and A. Tramontani. Performance variability in mixed-integer programming. In *Theory Driven by Influential Applications*, pages 1–12. INFORMS, 2013.

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

J. P. Marques-Silva and K. Sakallah. GRASP: A search algorithm for propositional satisfiability. *Computers, IEEE Transactions on*, 48(5):506–521, 1999.

M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient sat solver. In *Proceedings of the 38th annual Design Automation Conference*, pages 530–535. ACM, 2001.

G. L. Nemhauser and L. A. Wolsey. Integer programming and combinatorial optimization. *Wiley, Chichester. GL Nemhauser, MWP Savelsbergh, GS Sigismondi (1992). Constraint Classification for Mixed Integer Programming Formulations. COAL Bulletin*, 20:8–12, 1988.

G. L. Nemhauser and L. A. Wolsey. A recursive procedure to generate all cuts for 0–1 mixed integer programs. *Mathematical Programming*, 46(1-3):379–390, 1990.

I. Pólik. (re)using dual information in MILP. In *INFORMS Computing Society conference*, Richmond, VA, 2015a.

I. Pólik. Some more ways to use dual information in MILP. In *International Symposium on Mathematical Programming*, Pittsburgh, PA, 2015b.

T. Sandholm and R. Shields. Nogood learning for mixed integer programming. In *Workshop on Hybrid Methods and Branching Rules in Combinatorial Optimization, Montréal*, 2006.

M. W. Savelsbergh. Preprocessing and probing techniques for mixed integer programming problems. *ORSA Journal on Computing*, 6(4):445–454, 1994.

S. Schade, T. Schlechte, and J. Witzig. Structure-based decomposition for pattern-detection for railway timetables. In *Operations Research Proceedings 2017*, pages 715 – 721, 2018. doi: 10.1007/978-3-319-89920-6_95.

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

J. Witzig and A. Gleixner. Conflict-driven heuristics for mixed integer programming. 2019.

J. Witzig, T. Berthold, and S. Heinz. Experiments with conflict analysis in mixed integer programming. In *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 211–220. Springer, 2017.

J. Witzig, T. Berthold, and S. Heinz. A status report on conflict analysis in mixed integer nonlinear programming. In L.-M. Rousseau and K. Stergiou, editors, *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 84–94, Cham, 2019. Springer, Springer International Publishing. ISBN 978-3-030-19212-9.

F. You and I. E. Grossmann. Mixed-integer nonlinear programming models and algorithms for large-scale supply chain design with stochastic inventory management. *Industrial & Engineering Chemistry Research*, 47(20):7802–7817, 2008.

L. Zhang, C. F. Madigan, M. H. Moskewicz, and S. Malik. Efficient conflict driven learning in a boolean satisfiability solver. In *Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design*, pages 279–285. IEEE Press, 2001.