Mathematical Programming Computation (2020) 12:295–335
https://doi.org/10.1007/s12532-020-00180-9

FULL LENGTH PAPER

Split cuts from sparse disjunctions

Ricardo Fukasawa[1] · **Laurent Poirrier**[1] · **Shenghao Yang**[1]

Received: 21 July 2018 / Accepted: 11 February 2020 / Published online: 4 March 2020

Abstract
Split cuts are arguably the most effective class of cutting planes within a branch-and-cut framework for solving general Mixed-Integer Programs (MIP). Sparsity, on the other hand, is a common characteristic of MIP problems, and it is an important part of why the simplex method works so well inside branch-and-cut. In this work, we evaluate the strength of split cuts that exploit sparsity. In particular, we show that restricting ourselves to sparse disjunctions—and furthermore, ones that have small disjunctive coefficients—still leads to a significant portion of the total gap closed with arbitrary split cuts. We also show how to exploit sparsity structure that is implicit in the MIP formulation to produce splits that are sparse yet still effective. Our results indicate that one possibility to produce good split cuts is to try and exploit such structure.

Mathematics Subject Classification 90C11 (Mixed integer programming)

1 Introduction

Cutting planes are fundamental in solving mixed-integer linear programs (MIPs). Over the last 25 years, commercial solvers have accomplished remarkable progress, achieving a machine-independent speed-up of the solution process by more than a factor of 450,000 [10]. General-purpose cutting plane techniques, such as Gomory Mixed Integer (GMI) cuts and Mixed Integer Rounding (MIR) cuts, are arguably the most important contributors to this progress (see, for example, [12]).

To study the impact that a particular family of cuts may have, both theoretically and computationally, a common approach is to consider the *closure* of those cuts. Given

✉ Ricardo Fukasawa
rfukasawa@uwaterloo.ca

Laurent Poirrier
lpoirrier@uwaterloo.ca

Shenghao Yang
s286yang@uwaterloo.ca

[1] Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Canada

a family of cuts, its closure is defined as the intersection of all cuts belonging to the same family that are obtainable from the original MIP formulation. On the theoretical side, topics range from determining the polyhedrality of closures [17] to analyzing their strength [8]. On the computational front, several authors proposed strategies to empirically evaluate the strength of different closures by computing the amount of integrality gap they close: we thus have computational evaluations of the Chvátal closure [22], the split closure [6], the projected Chvátal-Gomory closure [14], the MIR closure [19] and the lift-and-project closure [13].

We focus on the results on the split closure (or equivalently the MIR closure), which consider the class of all split cuts [17], since these are the cuts that are most useful in practice to solve MIPs [12]. Besides efforts on efficient generation of strong split cuts (see for instance [2,18,24,25]), the split closure was shown to be a very tight approximation to the convex hull of all feasible solutions in the corresponding MIP [6,25]. On average it closes more than 75% of the integrality gap on MIPLIB 3.0 [11] instances. The purpose of this work is to determine what will be the effect on this integrality gap if we restrict ourselves to a subset of split cuts defined by its sparsity properties. In the following discussion we motivate such choice of restriction.

Sparsity is a natural condition that helps in the linear algebra routines of the simplex method [28], thus it is a desirable property of cutting planes for MIPs. Indeed, in almost every cut generation procedure described in the articles we mentioned above, specific heuristics are implemented to impose sparsity in the cuts, e.g., introducing a penalty term in the objective of a cut generating problem to make the resulting cut sparser [22], applying a coefficient reduction algorithm to reduce the number of nonzeros the split cut [18], and discarding all dense cuts to ensure that only sparse cuts are added [24]. The effect of sparsity has also recently been noted in a computational study by Walter [29] where it is shown that equivalent, but denser versions of the same cuts negatively affect performance of MIP solvers. Due to all this interest, there has also been some recent work to analyze theoretically the strength of sparse cutting planes [20,21].

While cut sparsity is desirable, it is not entirely clear how to impose a limit on it in a cut separation algorithm for split cuts. On the other hand, seeking sparsity in the split disjunctions may lead to more tractable algorithms. For instance, lift-and-project cuts [4] correspond to splits with the sparsest possible disjunctions, having only one nonzero element. They are a strong subclass of split cuts, and are easy to separate. For this reason, we chose to focus on sparsity of the split disjunctions in this work.

In theory, there is no guarantee that sparse disjunctions yield sparse cuts. Furthermore, there is no guarantee that sparse disjunctions are *useful*. As an example, consider a polyhedron $P := \{x \in \mathbb{R}^n : \frac{1}{3} \leq x_1 + x_2 + \cdots + x_n \leq \frac{1}{2}\}$. The split closure for P is an empty set, since P is included in the split $0 \leq x_1 + \cdots + x_n \leq 1$. However, if we consider only disjunctions with strictly fewer than n nonzeros, the corresponding closure will be P, i.e. there are no such split cuts.

On the other hand, our computational experiments confirm the natural intuition that, for typical sparse formulation, sparse disjunctions yield sparse cuts. Moreover, as mentioned above, lift-and-project cuts [4] are examples of split cuts arising from sparse disjunction, and they have been shown to be useful computationally.

One additional motivation to study the effect of sparsity is the recent result of Bergner et al. [9] where they show that several benchmark instances have an almost

block-diagonal structure called *arrowhead*, that is, a structure with several blocks that are linked only by few linking variables and constraints. This shows that not only are these benchmark instances sparse (on average, MIPLIB 2010 [27] instances only have 1.62% density), but in many cases such sparsity has an identifiable structure that can be exploited. In fact, Bergner et al. [9] show that computing the convex hull of each "block" can, in many instances, lead to better dual bounds than CPLEX can give. Therefore, searching for split cuts based on those blocks seems like a good idea. We note that such a block structure immediately leads to some level of sparsity of the split disjunctions by focusing only on the variables/constraints in each block to generate the splits.

The main contributions of this work can be stated as follows:

- We implement an approximate separation routine based on the work on Balas and Saxena [6] that separates only split cuts whose split disjunctions are sparse and whose split coefficients are small
- We show, empirically, that in spite of those restrictions, the gap closed by this subclass of split cuts is still quite significant (on average 92% of the split closure gap).
- Finally, we consider split cuts computed only from individual blocks of the arrowhead structure of the instances. We show that they also largely preserve the strength of general split cuts, in terms of gap closed.

These results help shed some light into what are important classes of split cuts that we can focus our attention on studying.

In the rest of this paper, we present in more details such results. Section 2 lays out the basic approach of Balas and Saxena [6] for the separation of split cuts, and briefly introduces the automatic decomposition of Bergner et al. [9]. In Sect. 3, we detail the implementation of our split cut separator. In particular, we describe exactly what measures we took to obtain cuts that are numerically stable and effective, while being verifiably valid. Section 4 presents the results of our computational experiments.

2 Background

In this section, we formally present the background necessary to explain our experiments. We start by introducing how to optimize over an approximation of the split closure and then discuss the developments related to the arrowhead decomposition.

2.1 Optimizing over the split closure

Consider a general MIP:

$$\min\{c^\top x : Ax = b,\ x \in \mathbb{Z}^p_+ \times \mathbb{R}^{n-p}_+\} \tag{MIP}$$

where $A \in \mathbb{Q}^{m\times n}$ has full row rank, $c \in \mathbb{Q}^n$ and $b \in \mathbb{Q}^m$. The linear programming relaxation of (MIP) is

$$\min\{c^\top x : x \in P\} \tag{LP}$$

where $P = \{x \in \mathbb{R}^n_+ : Ax = b\}$. For any $(\pi, \pi_0) \in \mathbb{Z}^n \times \mathbb{Z}$ such that $\pi_j = 0$ for $j \geq p+1$, a *split disjunction* is defined as

$$\pi^\top x \leq \pi_0 \ \vee \ \pi^\top x \geq \pi_0 + 1.$$

An inequality $\alpha^\top x \geq \beta$ valid for $P^{(\pi,\pi_0)}$ where

$$P^{(\pi,\pi_0)} = \text{conv}\big(\{x \in P : \pi^\top x \leq \pi_0\} \cup \{x \in P : \pi^\top x \geq \pi_0 + 1\}\big)$$

is called a *split cut* [17].

The problem of finding a violated split is $\mathcal{NP}$-hard in general [15]. Following Farkas' Lemma, a most-violated split cut $\alpha^T x \geq \beta$ for $P^{(\pi,\pi_0)}$ can be found by solving the Cut Generating Linear Program

$$\begin{aligned}
\min \ & \alpha^\top x - \beta \\
\text{s.t. } & \alpha = A^\top y + s - y_0 \pi \\
& \alpha = A^\top z + t + z_0 \pi \\
& \beta = b^\top y - y_0 \pi_0 \\
& \beta = b^\top z + z_0(\pi_0 + 1) \\
& \texttt{normalization condition} \\
& y, z \in \mathbb{R}^m, \ s, t \in \mathbb{R}^n_+, \ y_0, z_0 \in \mathbb{R}_+.
\end{aligned} \tag{CGLP(π, π_0)}$$

A derivation of (CGLP(π, π_0)) and in-depth discussion of the normalization condition were presented in [23]. The following remark on the nonnegativity of the multipliers y and z is useful in simplifying our CGLP.

Remark 1 Suppose (CGLP(π, π_0)) has an optimal solution under some choice of normalization, and let $(\hat{\alpha}, \hat{\beta}, \hat{y}, \hat{z}, \hat{s}, \hat{t}, \hat{y}_0, \hat{z}_0)$ be an optimal solution. Define $\hat{c} \in \mathbb{R}^m$ such that

$$\hat{c}_j := \begin{cases} 0 & \text{if } \hat{y}_j \geq 0 \text{ and } \hat{z}_j \geq 0, \\ -\hat{y}_j & \text{if } \hat{y}_j < 0 \text{ and } \hat{z}_j \geq 0, \\ -\hat{z}_j & \text{if } \hat{y}_j \geq 0 \text{ and } \hat{z}_j < 0, \\ -\hat{y}_j - \hat{z}_j & \text{if } \hat{y}_j < 0 \text{ and } \hat{z}_j < 0. \end{cases}$$

Then

$$\begin{array}{llll} \alpha^* := \hat{\alpha} + A^\top \hat{c}, & \beta^* := \hat{\beta} + b^\top \hat{c}, & y^* := \hat{y} + \hat{c}, & z^* := \hat{z} + \hat{c}, \\ s^* := \hat{s}, & t^* := \hat{t}, & y_0^* := \hat{y}_0, & z_0^* := \hat{z}_0, \end{array}$$

is also an optimal solution (assuming that it, too, satisfies the normalization condition). Moreover, we have that $y^* \geq 0$ and $z^* \geq 0$.

Therefore, we may assume w.l.o.g. in (CGLP(π, π_0)) that all multipliers are non-negative since, as we will see below, our normalization allows it. In all subsequent discussions we assume $y, z \in \mathbb{R}^m_+$.

The *split closure* $\mathcal{C}$ is defined as

$$\mathcal{C} = \bigcap_{\substack{(\pi,\pi_0)\in\mathbb{Z}^n\times\mathbb{Z}:\\ \pi_j=0,\,\forall j\geq p+1}} P^{(\pi,\pi_0)}.$$

Balas and Saxena [6] implemented an iterative procedure that alternates between a Master Problem and a Separation Problem to find

$$\min\{c^\top x : x \in \mathcal{C}\}.$$

At each iteration, the Master Problem is a linear program of the form

$$\min\{c^\top x : x \in P,\ \alpha^t x \geq \beta^t,\ t \in T\} \tag{MP}$$

where $\{\alpha^t x \geq \beta^t : t \in T\}$ is the set of all split cuts generated by the Separation Problem so far. If $\hat{x}$ is an optimal solution to (MP), the Separation Problem then finds a valid cut violated by $\hat{x}$, or proves that $\hat{x} \in \mathcal{C}$. The Separation Problem is a mixed-integer nonlinear program obtained from (CGLP(π, π_0)) with normalization $y_0 + z_0 = 1$, and allowing (π, π_0) to vary over $\mathbb{Z}^n \times \mathbb{Z}$. Formally, the separation problem is stated as:

$$\begin{aligned}
\min\ & \alpha^\top \hat{x} - \beta \\
\text{s.t. } & \alpha = A^\top y + s - y_0\pi \\
& \alpha = A^\top z + t + z_0\pi \\
& \beta = b^\top y - y_0\pi_0 \\
& \beta = b^\top z + z_0(\pi_0 + 1) \\
& 1 = y_0 + z_0 \\
& y, z \in \mathbb{R}^m_+,\ s, t \in \mathbb{R}^n_+,\ y_0, z_0 \in \mathbb{R}_+ \\
& (\pi, \pi_0) \in \mathbb{Z}^n \times \mathbb{Z},\ \pi_j = 0,\ j \geq p+1.
\end{aligned} \tag{SP}$$

In [6], (SP) is shown to be equivalent to a parametric mixed-integer linear program with scalar parameter θ,

$$\min_{0\leq\theta\leq\frac{1}{2}} \text{MILP}(\theta)$$

where MILP(θ) is given by

$$\begin{aligned} \min\ & s^\top \hat{x} - \theta(\pi^\top \hat{x} - \pi_0) \\ \text{s.t. } & A^\top w + s - t - \pi = 0 \\ & b^\top w - \pi_0 = 1 - \theta \\ & w \in \mathbb{R}^m,\ s, t \in \mathbb{R}^n_+ \\ & (\pi, \pi_0) \in \mathbb{Z}^n \times \mathbb{Z},\ \pi_j = 0,\ j \geq p+1. \end{aligned} \tag{MILP(θ)}$$

Therefore, the optimum to (SP) can be approximated from above by solving a finite sequence of problems MILP(θ) with varying values for θ.

2.2 Automatic detection of double-bordered block-diagonal structure

The idea of exploiting block-diagonal structure in sparse matrices has been widely discussed in the contexts of numerical linear algebra and mathematical programming. One motivation is that the diagonal blocks usually give rise to small independent subproblems well suited for parallel processing. Applications include solving systems of linear equations arising from a discretization of a continuous domain, LU and QR factorizations, and decomposition-based solution methods for structured (mixed-integer) linear programs. In general, the constraint matrix A of (MIP) does not admit a block-diagonal form, but it can be put into a k-way *double-bordered block-diagonal form*

$$\begin{bmatrix} D^1 & & & F^1 \\ & D^2 & & F^2 \\ & & \ddots & \vdots \\ & & D^k & F^k \\ A^1 & A^2 & \cdots A^k & G \end{bmatrix} \tag{DB-k}$$

for some $k \geq 1$. This is sometimes informally called the arrowhead form. The constraints associated with rows in A^i are called *linking constraints*, and the variables associated with columns in F^i are called *linking variables*.

Given a sparse matrix, Aykanat et al. [3] considered the problem of obtaining a DB-k form by permuting its rows and columns. They reduce the matrix permutation problem to that of graph and hypergraph partitioning. However, even when the number k of blocks is fixed, computational experiments show that the resulting DB-k forms demonstrate significant variability and are very sensitive to input parameters. To cope with this, Bergner et al. [9] proposed to use a proxy measure to automatically detect the "best" DB-k form, for the purpose of applying Dantzig-Wolfe reformulations to general MIPs. Figure 1 shows a few examples of MIPLIB instances, with black dots representing nonzero coefficients of the constraint matrix. The bottom row shows a rearrangement of the columns/rows of the matrix evidencing the DB-k structure.

As mentioned in the Introduction, the motivation to look at the DB-k structure is that [9] showed Dantzig-Wolfe decomposition using DB-k structure was able to improve the dual bounds obtained by CPLEX on several IP instances. This means that computing the convex hull of each block would produce significant improvements. This observation, combined with the fact that split cuts give a "reasonably good" approxi-

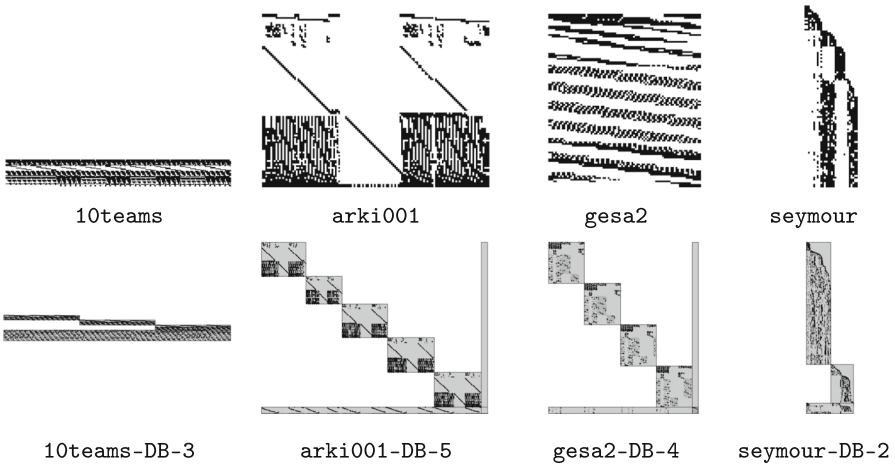


Fig. 1 Original problem structure versus its DB-k forms

mation of the convex hull [6], suggested to use split cuts derived from each block in our experiments. The resulting improvements were probably due to the DB-k decomposition based on the sparsity pattern of the constraint matrix, as evidenced by [7], where the idea of [9] was implemented using different random Dantzig-Wolfe decompositions and results showed that such other decompositions are "rather unlikely" to produce strong dual bounds.

3 Implementation

In this section we outline the computational details of our implementation. We follow the idea of Balas and Saxena [6] to approximate the optimal value of (SP) by solving a sequence of parametric MILPs. Features prefixed by an asterisk (*) were already present in [6]. Note that two of the following features (sparsity constraint and structure constraint) are used only in specific experiments, as indicated later. The others are used in all experiments, but we carry out additional computations in the appendix, to detail their individual benefits.

**Parameter grid* We denote by Θ the set of values of θ for which MILP(θ) will be solved. A uniform parameter grid Θ of points between 0 and 0.5 is created. The initial size of Θ is t, and we increase the number of grid points whenever necessary following the criteria in Algorithm 2.

**Stabilizing objective* To avoid unnecessarily weak cut coefficients (see [6] for a short discussion), we replace $\hat{x}$ in the objective of MILP(θ) with

$$\tilde{x}_j := \max\{\hat{x}_j, \delta\}, \quad \forall\ j,$$

for δ a small positive constant.

**Cut strengthening* Once a feasible solution $(\bar{w}, \bar{s}, \bar{t}, \bar{\pi}, \bar{\pi}_0)$ to MILP($\bar{\theta}$) with a negative objective value is found, we feed $(\bar{\pi}, \bar{\pi}_0)$ to the corresponding *Cut Generating Linear Program* (CGLP($\bar{\pi}, \bar{\pi}_0$)) with normalization

$$e^\top y + e^\top z + e^\top s + e^\top t + y_0 + z_0 = \kappa$$

for a fixed positive constant κ. This normalization is shown in [23] to produce stronger cuts than the normalization $y_0 + z_0 = 1$ used in deriving MILP(θ).

**Cut lifting* We work in the subspace of the variables that are not at one of their bounds in the incumbent solution, and lift the resulting cuts to the full space following the approach described in [5]. Note that for the purpose of determining the variables that are at a bound, we take the original value of the incumbent $\hat{x}$ before the applications of the stabilizing objective described above.

**Set covering* In an effort to impose some orthogonality in the set of split disjunctions, every time a split $(\bar{\pi}, \bar{\pi}_0)$ is found, we solve the set covering problem

$$\min_{z \in \{0,1\}^p} \left\{ \sum_{j=1}^{p} \min\{\hat{x}_j - \lfloor \hat{x}_j \rfloor, \lceil \hat{x}_j \rceil - \hat{x}_j\} z_j : \sum_{j=1}^{p} \mathbb{I}_{[\pi_j \neq 0]} z_j \geq 1, \forall \pi \in \mathcal{S} \right\} \tag{StCvIP($\hat{x}, \mathcal{S}$))}$$

where $\mathcal{S}$ is the set of splits already discovered, and $\mathbb{I}_{[k \neq 0]} = 1$ if $k \neq 0$, $\mathbb{I}_{[k \neq 0]} = 0$ if $k = 0$. Let $\hat{z}$ be an optimal solution to (StCvIP($\hat{x}, \mathcal{S}$)), then we impose $\pi_j = 0$ for all $j \in \{j : \hat{z}_j \neq 0\}$ when solving the next MILP(θ).

Fractionality constraint Split disjunctions (π, π_0) where $\pi^\top \hat{x}$ is too close to either π_0 or $\pi_0 + 1$ usually give rise to weak split cuts. To avoid that, we impose the bounds

$$\sigma \leq \pi^\top \hat{x} - \pi_0 \leq 1 - \sigma \tag{Con1}$$

for a small $\sigma > 0$.

Sparsity constraint To impose the condition that π is sparse with at most M nonzero entries, we introduce binary variables $r \in \{0, 1\}^p$ and constraints

$$-U r_j \leq \pi_j \leq U r_j, \quad \forall\, j = 1, \ldots, p, \quad \text{and} \quad \sum_{j=1}^{p} r_j \leq M, \tag{Con2}$$

where U is an artificial upper bound on the magnitude of the components of π.

Structure constraint Given a DB-k form of the constraint matrix A, to compute split disjunctions whose support lie entirely in a block D^i, we simply impose that:

$$\begin{aligned} \pi_j = s_j = t_j = 0, \quad \forall\, j \notin \mathcal{C}^i \\ w_j = 0, \quad \forall\, j \notin \mathcal{R}^i \end{aligned} \tag{Con3}$$

where $\mathcal{C}^i$ and $\mathcal{R}^i$ are column and row index set of D^i, respectively. In practice, by fixing these variables, we are performing separation for a *relaxation* of the original problem, constructed as follows. All the linking constraints and all the constraints from other blocks are discarded. As a result, variables from other blocks do not appear

anymore. The linking variables are kept but, by fixing the corresponding disjunction coefficients to zero, we are effectively disregarding any integrality constraint on them.

Validity check For every split cut $\alpha^\top x \geq \beta$ generated from CGLP with splits (π, π_0), we provide another certificate for the validity of the cut. Let

$$\hat{\beta}_l := \min_{x \in P}\{\alpha^\top x : \pi^\top x \leq \pi_0\} \quad \text{and} \quad \hat{\beta}_u := \min_{x \in P}\{\alpha^\top x : \pi^\top x \geq \pi_0 + 1\}.$$

Then it should always hold that $\beta \leq \min\{\hat{\beta}_l, \hat{\beta}_u\}$. If the inequality fails to hold, then the cut is invalid and we discard it. This may be the case due to numerical issues within the LP or MIP solver.

Cleaning up cut coefficients To prevent cut coefficients from being too large or too small, once a split cut is returned by CGLP(π, π_0), we scale the cut so that the greatest absolute value of cut coefficients equals 10^4. Furthermore, after scaling we set all cut coefficients whose absolute value is less than 10^{-6} to zero. In general, setting a nonzero cut coefficient to zero may strengthen the cut and make it invalid, but since our tolerance is small, the effect is small as well. Nonetheless, the validity of the cut is always subsequently certified by the independent checker. Note that this scaling process also serves as an implicit dynamism control, i.e., the ratio between the greatest and the smallest absolute value of cut coefficients is no greater than 10^{10}.

The cut generation procedure is summarized in Algorithm 1.

Algorithm 1: Cut Generation$(\hat{x}, \Theta, \gamma, \tau)$

Input: Incumbent solution $\hat{x}$, parameter grid Θ, upper cutoff limit $\gamma < 0$, time limit τ, minimum cut violation $\epsilon > 0$, required properties of split disjunctions (Con1)–(Con3). Polyhedron $P = \{x \in \mathbb{R}^n_+ : Ax = b\}$ describing the constraint set of (LP).

Output: A set $\mathcal{K}$ of split cuts violated by $\hat{x}$.

$\mathcal{K} \leftarrow \emptyset, \mathcal{S} \leftarrow \emptyset$.
for $\theta \in \Theta$ **do**
  Solve StCvIP$(\hat{x}, \mathcal{S})$ and impose partial orthogonality if needed. Add applicable supplementary constraints to MILP(θ), as indicated: fractionality (all experiments), sparsity (second and third experiments), structure (Con1)–(Con3) (third experiment).
  Solve MILP(θ) with time limit τ.
  if *Found a feasible solution* (π, π_0) *to MILP*(θ) *with objective value* $\leq \gamma$. **then**
    Perform cut strengthening to get cut $\alpha^\top x \geq \beta$.
    Perform cut lifting on cut $\alpha^\top x \geq \beta$.
    Perform cut cleaning on cut $\alpha^\top x \geq \beta$.
    if $\alpha^\top \hat{x} - \beta \leq -\epsilon$ **then**
      $\beta_l \leftarrow \min_{x \in P}\{\alpha^\top x : \pi^\top x \leq \pi_0\}$, $\beta_u \leftarrow \min_{x \in P}\{\alpha^\top x : \pi^\top x \geq \pi_0 + 1\}$. $\beta^* \leftarrow \min\{\beta_l, \beta_u\}$.
      if $\beta \leq \beta^*$ **then**
        $\mathcal{K} \leftarrow \mathcal{K} \cup \{\alpha^\top x \geq \beta\}, \mathcal{S} \leftarrow \mathcal{S} \cup \{\pi\}$.
return $\mathcal{K}$.

Time limit on the MIP solver Mixed-integer linear programs are much harder to solve than linear programs in general. As a result, even finding a feasible solution to MILP(θ) can be extremely time-consuming. We observed that this is frequently the case, in particular, when separating a point that is close to the closure we aim to optimize over. Therefore, a deterministic time limit of 800 ticks (roughly 1 s) is set for each MILP(θ) we process. We use CPLEX's *deterministic time* [16] (ticks) so that the results are reproducible and comparable across different machines.

Dynamics At each iteration, if no cut is generated because we could not find a feasible solution to MILP(θ), we increase the time limit to 48,000 ticks (roughly 60 s) and the upper cutoff limit of the objective value. If there is no improvement in the optimal objective value of the Master Problem for a while (see Algorithm 2), we increase the number of grid points and add more cuts per iteration. Furthermore, in order to control the number of cuts presented in the Master Problem, we delete all cuts that are nonbinding in the incumbent solution every five iterations.

Global time limit The whole process is terminated if the entire computation time exceeds a global time limit.

Details of the iterative procedure are described in Algorithm 2. Recall that the algorithm would be an *exact* separation algorithm only if MILP(θ) was solved for all $0 \leq \theta \leq \frac{1}{2}$. Since (like Balas and Saxena [6]) we discretize the interval, we get an (outer) approximation of the split closure. More precisely, the numbers we report are lower bounds on the gap closure corresponding to the first split closure.

Algorithm 2: Overall cut generation loop

1 Initialization.
  Choose initial parameter grid size t, upper objective value cutoff limits $\gamma_1 < \gamma_2 < 0$, deterministic time limits $\tau_1 = 800$ ticks, $\tau_2 = 48{,}000$ ticks. Set iteration counter `Iter` $= 0$. Denote k the number of blocks in a given `DB-`k from; if no decomposition is available, set $k = 1$.
2 `TimeLimit` $\leftarrow \tau_1$, `Cutoff` $\leftarrow \gamma_1$.
3 `Iter` $\leftarrow$ `Iter` + 1. Solve (MP) and obtain optimal solution $\hat{x}$. Denote n the number of consecutive iterations where no improvement in the optimal objective value is made. Delete nonbinding cuts if necessary.
4 **if** $n = 100$ **then return** $\hat{x}$.
5 Update parameter grid size in the current iteration.
  if $0 \leq n \leq 39$ **then** $s \leftarrow 2^{\lfloor 0.1n \rfloor} t$. **else** $s \leftarrow 16t$.
6 Set parameter grid Θ uniformly with $|\Theta| \leftarrow s$.
7 Separation.
  for $j = 1, \ldots, k$ **do** Generate a set $\mathcal{K}^{(j)}$ of cuts following CutGen($\hat{x}$, Θ, `Cutoff`, `TimeLimit`) for block j.
8 **if** $\bigcup_{j=1}^{k} \mathcal{K}^{(j)} \neq \emptyset$ **then** Add cuts to (MP), **go to** 2.
9 **else if** `TimeLimit`$= \tau_1$, `Cutoff`$= \gamma_1$ **then** `TimeLimit` $\leftarrow \tau_2$, **go to** 7.
10 **else if** `TimeLimit`$= \tau_2$, `Cutoff`$= \gamma_1$ **then** `Cutoff` $\leftarrow \gamma_2$, **go to** 7.
11 **else if** `TimeLimit`$= \tau_2$, `Cutoff`$= \gamma_2$ **then return** $\hat{x}$.

4 Computational experiments

In this section we first discuss the practical setup for our experiments, then present our computational results. We implemented our code in C, with IBM ILOG CPLEX 12.7.1 as black-box MIP and LP solver. The computations were conducted on an assortment of machines with `x86_64` architecture CPUs. In order to ensure reproducibility, all machines used the same single-threaded binary code, and all time limits made use of CPLEX's *deterministic time* feature [16], aside from the global time limit.

4.1 Choice of model parameter values

The values of various model parameters used in the computation are summarized in Table 1. An asterisk (*) indicates that the parameter does not apply to all experiments. We also present below a brief motivation for our choices.

In their experimental analysis, Balas and Saxena [6] noted that the split disjunctions they computed generally featured two interesting characteristics. Although not being intentionally restricted,

(i) most split disjunctions had a support of size between 10 and 20, irrespective of the size of the problem; and
(ii) most split disjunctions did not have very large coefficients, with the average coefficient size per iteration typically being less than 5.

We chose the sparsity parameter of $M = 10$ to reflect the lower end of that spectrum. When attempting to limit the size of the split coefficients, we chose bounds $U = 1$ (i.e., $-1 \leq \pi_j \leq 1$, for all $1 \leq j \leq p$) since these would be the simplest splits obtainable. When only sparsity constraints were enforced, we set $U = 100$ (i.e., $-100 \leq \pi_j \leq 100$, for all $1 \leq j \leq p$) to allow for splits with somewhat larger coefficients.

At each iteration, the initial parameter grid size depends on whether a DB-k form of the constraint matrix is supplied or not. If no DB-k form is given, we set $t = 80$,

Table 1 Model parameter values used in computation

Measure	Parameter	Value
Maximum number of nonzero components in π (*)	M	10
Bounds on $\|\pi_j\|$, $1 \leq j \leq p$ (*)	U	1 or 100
Initial number of grid points (without DB-k form)	t	80
Initial number of grid points (with DB-k form)	t	20
Normalization constant	κ	10^4
Minimum nonzero objective coefficient	δ	10^{-4}
Upper cutoff limits of objective value	γ_1, γ_2	$-10^{-3}, -10^{-5}$
Minimum cut violation	ϵ	10^{-6}
Fractionality bound	σ	0.025

Table 2 Gap closed as a percentage of the best known gap closure (from [6,19])

Relative gap closed (%)	# Instances
$> 100\%$	9
$\geq 99\%$	20
$\geq 90\%$	26
$\geq 50\%$	29
$< 1\%$	25

and 80 MILP(θ) are processed; if a decomposition is given, then we set $t = 20$ for each of the k blocks, and therefore $20k$ MILP(θ) are processed in total.

For the fractionality bound σ on the set of split disjunctions, a natural value could be, for example, the integrality tolerance 10^{-6}. Although more split cuts may be obtained by using such a loose bound, we impose a rather strict 0.025 bound instead. In practice, this led to more gap closed per iteration on average and more gap closed overall within our time limits. Adding a fractionality bound also helped preventing MILP(θ) from yielding unviolated split disjunctions due to numerical errors.

4.2 Computational results

4.2.1 First experiment: how does our implementation compare with the best available results?

To check whether our implementation was reasonable, we first tested it on the MIPLIB 3.0 [11] instances, in a configuration where it approximates a straightforward split cut separator, i.e., without any sparsity or structure constraints on the split disjunctions. Artificial lower and upper bounds ± 100 are applied on the disjunction coefficients ($U = 100$), which allows for a reasonably large subset of all disjunctions to be considered. The entire computation time for each instance is limited to 24 h, including the time taken to check cut validity.[1] At termination, we measure the final percentage of integrality gap closed, which is then compared with the best of the bounds given in [6] and [19]. For each instance, we look at that percentage of gap closure divided by the best of the analogous results in [6] and [19]. We do this comparison on the 57 instances where the best known gaps are strictly positive. Table 2 shows the number of instances that fall within various categories based on this ratio. In particular, on 9 instances, we closed more gap than the best available result, and on 26 instances we closed at least 90% relative gap.

On the other hand, we closed less than 1% relative gap on 25 instances. As shown in Fig. 2 where each dot represents an individual instance, those are generally the instances that have the most integer variables. While there are many plausible explanations for our poor performance in this large set of instances, an important one is that the parameters in our code were not fine-tuned for this experiment, but rather for the experiments considering sparsity. When changing the values of parameters such

[1] Note that neither [6] nor [19] have any cut validity procedure and also that [6] has no time limit on their experiments

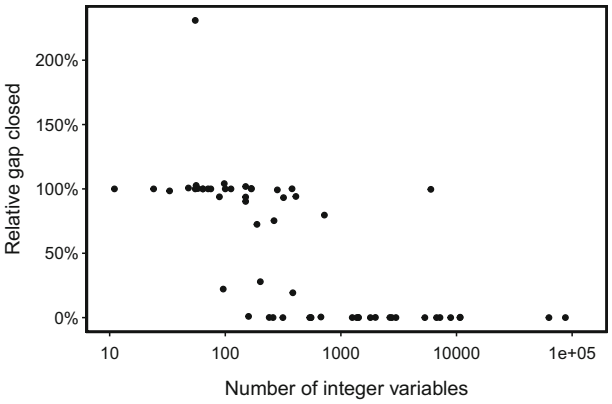


Fig. 2 Gap closed as a percentage of the best known gap closure (from [6,19]), versus number of integer variables

as the number $|\Theta|$ of grid points and the fractionality bound σ, we were able to close significantly more gap on these 25 instances. The purpose of this experiment was to determine if our implementation was comparable with earlier ones, which seems to be the case to some extent (more convincing evidence is provided by subsequent experiments).

While we use the same algorithm as Balas and Saxena [6] in theory, the large number of implementation choices one has to make in practice can result in wide variations on individual instances. Part of our objective with this paper is to more thoroughly document one such set of choices. On the other hand, we show in the appendix that a modified, but equivalent, implementation of the separation routine can lead to significantly more gap closed, which is more consistent with the results reported by Balas and Saxena [6]. However, since devising a better ad-hoc separation routine for arbitrary split cuts was not the goal of this work, we proceed with our original implementation in all subsequent experiments.

4.2.2 Second experiment: how does sparsity help?

In this section, we evaluate the relative strength of split cuts (i) whose split disjunctions are sparse and (ii) whose split coefficients are also small. We ran our implementation again on the MIPLIB 3.0 instances, first with the additional sparsity constraint obtained by setting $M = 10$. Note that the "structure constraint" described in Sect. 3 is not used yet in this experiment. Then, we additionally considered ± 1 bounds on the disjunction coefficients (that is, setting $U = 1$). As was the case earlier, a time limit of 24 h was set for all computations. Table 3 shows the details of our results. The first column of the table shows the best gap given in [6,19], followed by results obtained with arbitrary disjunctions, sparse disjunctions, and sparse disjunctions with ± 1 bound, respectively.

In each setting, column *% time checking* shows the percentage of the total computation time that was spent checking cut validity, and column *NR* shows the number of rounds of cutting planes that were added. Observe that on a few large instances, the time spent on checking took most of the computation time. For example, in comput-

Table 3 Gap closed for the (i) full split closure, (ii) sparse split cuts only, and (ii) sparse ±1 split cuts only

Best	Instance	$M = +\infty$, $U = 100$					$M = 10$, $U = 100$					$M = 10$, $U = 1$				
gap [6,19]		Gap closed	# Cuts binding	Time (s)	% Time checking	NR	Gap closed	# Cuts binding	Time (s)	% Time checking	NR	Gap closed	# Cuts binding	Time (s)	% Time checking	NR
100.00	10teams	0.00	4098	86,400.00	27.18	136	66.67	1378	86,400.00	40.95	350	81.00	1105	86,400.00	24.64	326
100.00	air03	0.31	480	86,400.00	98.08	854	100.00	390	4050.00	90.30	35	100.00	324	192.00	85.31	11
91.23	air04	0.00	4323	86,400.00	96.87	68	36.24	283	86,400.00	89.18	92	57.38	539	86,400.00	95.14	95
61.98	air05	0.03	428	86,400.00	97.91	459	29.74	345	86,400.00	76.36	179	62.03	376	86,400.00	62.10	898
83.95	arki001	0.00	2602	86,400.00	95.07	53	64.87	218	86,400.00	0.26	378	42.41	224	86,400.00	0.51	402
99.60	bell3a	99.64	81	71,493.00	0.03	1226	75.50	100	5865.00	0.03	64	75.73	81	649.00	0.17	51
92.95	bell5	93.22	174	74,604.00	0.01	201	91.16	79	86,400.00	0.00	42	92.57	222	1549.00	0.03	14
46.52	blend2	35.03	453	10,135.00	2.01	1850	39.71	76	11,970.00	0.03	128	45.56	72	2230.00	0.12	84
65.17	cap6000	64.95	362	86,400.00	0.69	908	63.83	61	3419.00	1.40	70	64.69	119	4452.00	0.81	128
0.22	dano3mip	0.00	504	86,400.00	95.09	21	0.00	364	86,400.00	46.03	20	0.09	106	86,400.00	58.45	21
8.20	danoint	8.42	466	86,400.00	8.25	1285	7.04	228	86,400.00	2.23	450	7.86	362	86,400.00	3.21	538
100.00	dcmulti	100.00	1098	6486.00	6.32	1130	99.80	250	86,400.00	0.03	91	100.00	617	483.00	2.53	36
100.00	egout	100.00	281	597.00	0.40	60	100.00	223	721.00	0.11	24	98.64	231	17.00	2.35	11
19.08	fast0507	0.00	1437	86,400.00	89.44	32	0.00	437	86,400.00	97.03	13	0.10	307	86,400.00	86.42	10
99.68	fiber	0.04	442	86,400.00	38.34	44,801	24.00	248	86,400.00	0.07	662	62.64	276	86,400.00	0.05	731
99.75	fixnet6	99.84	536	86,400.00	0.16	767	99.74	562	86,400.00	0.04	272	99.84	337	86,400.00	0.02	213
100.00	flugpl	100.00	158	200.00	0.05	5	100.00	158	195.00	0.05	5	98.49	124	16.00	0.62	6
100.00	gen	90.28	510	45,719.00	11.26	4181	93.16	233	86,400.00	0.11	203	100.20	716	570.00	6.54	30
99.70	gesa2	93.84	251	86,400.00	0.86	2551	88.21	244	86,400.00	0.05	281	99.92	249	86,400.00	0.03	117
99.97	gesa2_o	79.62	347	86,400.00	2.78	6376	77.53	309	86,400.00	0.12	645	94.00	243	86,400.00	0.05	236

Table 3 continued

Best gap [6,19]	Instance	$M = +\infty$, $U = 100$				NR	$M = 10$, $U = 100$				NR	$M = 10$, $U = 1$				NR
		Gap closed	# Cuts binding	Time (s)	% Time checking		Gap closed	# Cuts binding	Time (s)	% Time checking		Gap closed	# Cuts binding	Time (s)	% Time checking	
95.81	gesa3	18.49	342	86,400.00	5.42	9451	94.14	253	86,400.00	0.18	587	95.99	293	86,400.00	0.03	99
95.20	gesa3_o	0.45	482	19,419.00	12.80	1735	93.44	333	86,400.00	0.26	1026	95.97	281	86,400.00	0.03	79
98.38	gt2	71.29	2155	86,400.00	1.05	6423	93.26	91	86,400.00	0.01	388	98.89	147	86,400.00	0.01	681
58.48	harp2	0.02	142	812.00	17.06	239	12.93	90	10,534.00	0.05	168	43.09	172	86,400.00	0.04	1207
100.00	khb05250	100.00	308	703.00	0.65	31	100.00	386	1298.00	0.43	29	100.00	451	29.00	14.83	15
95.20	l152lav	0.14	562	86,400.00	79.93	8093	27.48	173	86,400.00	4.88	923	43.17	180	86,400.00	2.29	796
93.75	lseu	87.97	104	86,400.00	0.02	596	69.82	68	86,400.00	0.00	84	75.21	78	86,400.00	0.00	80
14.02	mas74	14.28	225	86,400.00	1.02	4464	10.13	45	86,400.00	0.00	61	11.81	49	86,400.00	0.00	91
26.52	mas76	24.84	123	86,400.00	0.45	2110	12.77	74	86,400.00	0.00	80	14.94	52	86,400.00	0.00	76
51.70	misc03	0.49	468	86,400.00	12.09	11,856	50.66	156	86,400.00	0.07	224	51.42	173	86,400.00	0.01	72
100.00	misc06	100.00	279	160.00	8.31	14	100.00	325	298.00	4.26	14	100.00	168	116.00	11.29	11
20.11	misc07	0.01	1714	86,400.00	16.11	1852	15.20	102	86,400.00	0.65	460	14.32	257	86,400.00	0.09	178
100.00	mitre	0.00	2148	86,400.00	5.38	56	0.06	1797	86,400.00	0.43	106	17.20	2109	86,400.00	0.63	285
36.16	mkc	0.00	6424	86,400.00	9.21	100	17.61	690	86,400.00	1.55	1332	54.26	684	86,400.00	1.75	4225
99.98	mod008	93.19	375	86,400.00	1.22	10,721	52.95	65	86,400.00	0.00	51	55.48	99	86,400.00	0.00	76
100.00	mod010	0.09	2690	86,400.00	46.52	2640	72.43	239	86,400.00	6.33	1029	100.00	384	6554.00	10.65	168
72.44	mod011	16.06	4487	86,400.00	9.82	167	74.25	985	86,400.00	1.18	853	79.42	1030	86,400.00	1.08	1235
92.18	modglob	95.97	191	55,564.00	0.03	215	96.52	181	68,848.00	0.02	262	93.66	201	86,400.00	0.01	136
100.00	nw04	0.03	421	86,400.00	95.72	374	24.18	265	86,400.00	99.55	43	77.03	329	86,400.00	25.05	484
87.42	p0033	86.06	50	86,400.00	0.00	91	79.89	53	54,836.00	0.00	65	83.13	135	79,386.00	0.00	56
74.93	p0201	20.91	431	86,400.00	14.75	27,833	65.46	193	86,400.00	0.11	804	70.71	167	86,400.00	0.04	250
99.99	p0282	99.22	125	86,400.00	0.10	871	98.52	125	86,400.00	0.01	180	98.34	142	86,400.00	0.01	175
99.42	p0548	0.00	6407	27,863.00	1.14	200	92.39	346	86,400.00	0.02	288	93.74	332	86,400.00	0.01	218

Table 3 continued

Best gap [6,19]	Instance	$M = +\infty$, $U = 100$				NR	$M = 10$, $U = 100$				NR	$M = 10$, $U = 1$				NR
		Gap closed	# Cuts binding	Time (s)	% Time checking		Gap closed	# Cuts binding	Time (s)	% Time checking		Gap closed	# Cuts binding	Time (s)	% Time checking	
99.90	p2756	0.00	6412	42,414.00	2.07	200	85.50	430	86,400.00	0.03	786	86.50	458	86,400.00	0.07	515
0.00	pk1	0.00	6220	86,400.00	0.13	85	0.00	290	86,400.00	0.00	24	0.00	520	86,400.00	0.01	26
97.03	pp08a	97.02	187	86,400.00	0.06	739	97.01	175	86,400.00	0.00	74	97.03	171	86,400.00	0.00	44
95.81	pp08aCUTS	95.79	168	86,400.00	0.25	474	95.75	178	86,400.00	0.01	68	95.82	214	86,400.00	0.00	68
77.51	qiu	78.05	350	86,400.00	4.15	613	78.03	404	86,400.00	3.10	397	78.04	332	86,400.00	0.76	426
100.00	qnet1	0.08	1016	86,400.00	20.94	3620	75.20	239	86,400.00	0.15	1211	100.00	293	26,562.00	0.13	145
100.00	qnet1_o	0.08	611	86,400.00	22.86	6186	91.55	199	86,400.00	0.02	875	100.00	216	7454.00	0.07	109
23.40	rentacar	54.03	326	78,331.00	9.61	234	37.59	264	86,400.00	7.10	111	6.93	265	24,555.00	17.77	83
100.00	rgn	100.00	438	50,849.00	0.10	1152	73.96	181	86,400.00	0.01	711	75.15	795	86,400.00	0.01	168
70.70	rout	0.00	6411	66,795.00	5.69	200	40.56	254	86,400.00	0.29	908	58.40	268	86,400.00	0.26	1193
89.74	set1ch	0.10	387	86,400.00	2.45	10,831	89.76	410	32,800.00	0.02	90	89.75	342	24,640.00	0.01	73
61.52	seymour	0.00	2835	86,400.00	41.13	43	0.02	313	86,400.00	26.98	160	34.97	590	86,400.00	20.78	392
0.00	stein27	0.00	156	32,007.00	0.00	23	0.00	169	8522.00	0.01	14	0.00	125	11,471.00	0.01	14
0.00	stein45	0.00	317	86,400.00	0.39	248	0.00	3467	86,400.00	0.37	90	0.00	4782	86,400.00	0.26	115
33.93	swath	0.00	4509	86,400.00	70.41	127	9.04	255	86,400.00	11.75	342	22.91	265	86,400.00	3.56	359
100.00	vpm1	100.00	400	17,217.00	0.25	494	96.09	170	86,400.00	0.01	243	100.00	488	897.00	0.28	31
81.05	vpm2	81.32	165	86,400.00	0.10	581	81.18	176	86,400.00	0.00	57	81.43	217	86,400.00	0.00	63
75.17	Average	40.02	1343	69,063	19.84	3049	59.38	347	71,069	10.24	320	66.30	415	62,237	9.02	308

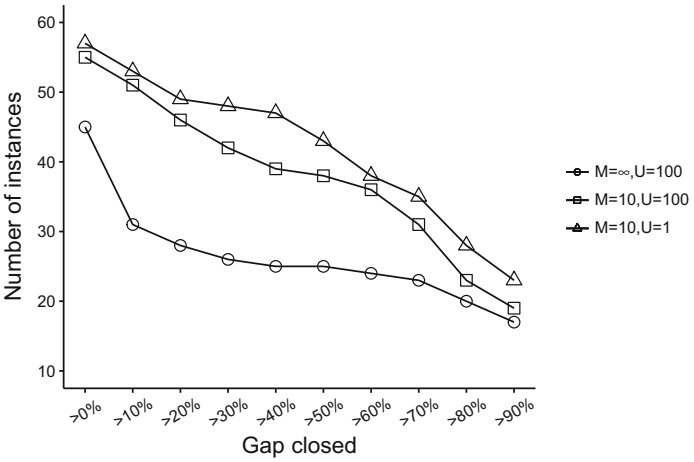


Fig. 3 Number of MIPLIB 3.0 instances versus gap closure in three different settings

ing the gap closed by sparse disjunctions with ± 1 bounds on the instance `air04`, of the 24 h spent, only 5% contributed to the actual computation. The remaining 95% was all dedicated to the verification of cut validity. We should thus expect that the bound obtainable on these large instances should be greater than the result shown in Table 3, had we chosen a longer time limit. Nonetheless, by restricting ourselves to split disjunctions with at most 10 nonzero coefficients, we still obtained significantly better results in terms of relative gap closed on instances that have a large number of integer variables, as opposed to the poor performance we observed with arbitrary disjunctions. To help visualize the impact of adding this additional sparsity constraint, we plot in Fig. 3 the number of MIPLIB 3.0 instances on which at least $x\%$ integrality gap was closed, for $x \in \{10, 20, \ldots, 90\}$, for each of the three computational settings. This outcome could be surprising at first since, in theory, this new experiment involves restricting the set of potential cuts. However, recall that we are solving a hard problem for each separation, under strict time limits. With the sparsity constraints added, we get a weaker set of cuts, but each computation becomes faster, yielding better results overall.

Besides allowing for more gap closed in less time, another related interesting effect to observe is the sparsity of the cuts produced. Observing that sparse disjunctions do not necessarily lead to sparse cuts, Fig. 4 compares the densities of cuts (i.e., proportion of cut coefficients that are nonzero) obtained from different sets of split disjunctions. For each of the 60 MIPLIB 3.0 instances, we computed the average cut density by considering all the cuts that were used to obtain the results in Table 3. This resulted in 60 average cut densities for each set of split disjunctions. We then plot the distribution of these average cut densities in Fig. 4. The horizontal lines in the figures represent the range of densities (with outliers omitted), the rectangles represent the 25–75 percentile interval and the solid vertical line represents the median. We consider as "outliers" cuts that are extremely dense, as determined by the following. Let r be the difference in density between the 25th and 75th percentile. Any cut with density of more than

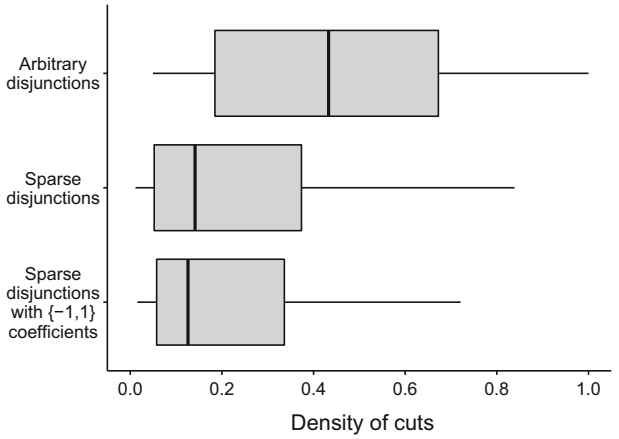


Fig. 4 Distribution of cut densities with different experimental settings

$1.5r$ above the 75 percentile is considered an outlier. Observe that sparse disjunctions did indeed lead to sparser split cuts in general: While the median density was 0.433 with arbitrary disjunctions, it dropped to 0.141 with sparse ones, and 0.126 with sparse ± 1 disjunctions.

While the results above tend to indicate that the sparsity of the cuts follows the sparsity of the splits, Table 9 shows that this is not always the case. In that table, for every split cut, we compute the percentage of its nonzero cut coefficients that came from nonzero split coefficients (corr_n). In other words, if split (π, π_o) generated cut $\alpha^T x \leq \beta$, we compute $\frac{|\{j:\alpha_j \neq 0 \text{ and } \pi_j \neq 0\}|}{|\{j:\alpha_j \neq 0\}|}$. For the purposes of that table, we ran the experiments using $M = 10$ and $U = 1$. Also, we only counted integer variables in the percentage above. The results show that, for some instances, as much as 97% of the cut coefficients come from terms with zero split coefficient, which indicate that sparsity of the cuts does not follow the sparsity of the splits. Nonetheless, on average, that number is 47%. While still high, notice that the experiments on that table were done based on splits of at most 10 nonzero elements. This implies that, on average, the cuts generated have at most 20 nonzero coefficients, so while sparsity of the cuts does not follow directly the sparsity of the splits, the two seem correlated, but further investigation is needed to corroborate this.

Moreover, in the case of block decompositions, most of the cut coefficients did come from the *block* used for separation: This is shown, also in Table 9, by the percentage of nonzero cut coefficients that came from inside the current block (corr_b), i.e., for a block with column indices $\mathcal{C}^i$, we compute $\frac{|\{j:\alpha_j \neq 0 \text{ and } j \in \mathcal{C}^i\}|}{|\{j:\alpha_j \neq 0\}|}$. On average, that percentage, varies from 94% for DB-2 to 92% for DB-5. Note that in Table 9, *NA* corresponds to instances without a corresponding DB-k decomposition and *NC* correspond to no cuts being found.

To better evaluate the strength of the split cuts in the most restricted experiment (i.e., disjunctions with at most 10 nonzero ± 1 coefficients), we also run the experiments for a very long time (one week), and recomputed the gap closed on MIPLIB 3.0

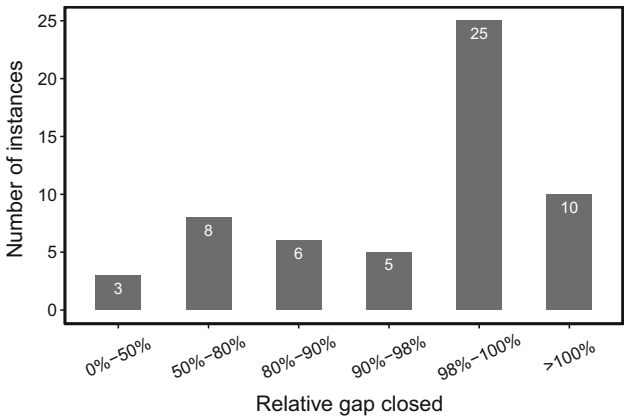


Fig. 5 Distribution of gap closed with time limit of one week ($U = 1$, $M = 10$)

instances. The resulting average integrality gap is 69.0%, accounting for 92% of the 75.2% average for the best in [6] and [19]. Figure 5 shows a breakdown of the 57 instances whose best gap is strictly positive, according to the relative gap closed in this case. Surprisingly, we lost almost nothing (at most 2%) on more than half of MIPLIB 3.0 instances. Furthermore, we closed at least 90% relative gap on more than two thirds of the instances.

Our conclusion from this experiment is twofold. First, split cuts based on sparse disjunctions with small coefficients are almost as strong as general split cuts. Secondly, they tend to be sparser.

4.2.3 Third experiment: how does structured sparsity help?

Problem-specific DB-k forms provide a natural way to exploit sparsity. The potential advantages of generating split disjunctions whose support lies entirely within individual blocks are to produce split cuts that are both sparse and mutually orthogonal—two vital characteristics that make a cut effective. Moreover, working with small blocks in a DB-k decomposition may potentially reduce the computational time required to find a violated cut. In principle, there is no reason to expect the support of useful disjunctions to be, in general, limited to a single block, and restricting ourselves to such a narrow class of cuts can result in a much weaker cut family. However, Bergner et al. [9] show that computing the convex hull of these blocks can lead to significant bound improvements, so using split cuts from those blocks may also lead to significant bound improvements. The experiments in this section were designed to try and quantify these improvements and the tradeoffs of using blocks.

We use GCG 2.1.1 [26] as a black-box tool to generate the required DB-k forms on MIPLIB 3.0 instances, and then implement our model with the additional structure constraint (Con3) on the disjunctions, as described in Sect. 3. Furthermore, for comparison purposes,

– we have kept the sparsity parameter $M = 10$ and coefficient bound $U = 1$ on split disjunctions;

– for each instance with a given decomposition, we adhered to that decomposition in all iterations, i.e., we didn't change the structural requirement on disjunctions from one iteration to another;
– we ignored all linking constraints and linking variables by setting the corresponding multipliers to zero;
– the time limit was set to one week.

Table 4 shows the final gap closed by restricting split disjunctions with the structures given by DB-k forms for $k = 2, 3, 4, 5$ (GAPk). The first four columns of Table 4 are the result from previous section, with the same one week time limit, obtained by using disjunctions with $M = 10$ and $U = 1$ but no structure constraint (GAPnodb). The last column represents the highest gap closed between all DB-k forms. We removed from the table three instances where the gap closed without DB-k was zero (`pk1`, `stein27`, `stein45`) and eight instances where no DB-k form was found for any $k \in \{2, 3, 4, 5\}$ (`air03`, `cap6000`, `mas74`, `mas76`, `mod008`, `nw04`, `p0033`, `rentacar`).

Note that the set of split cuts we used to obtain the results on Table 4 is extremely restrictive: (i) the corresponding disjunctions have at most 10 nonzero coefficients which are either 1 or -1, and (ii) the cuts are obtained by aggregating only rows and columns that belong to a single block in a DB-k form. Despite being so selective, these cuts close a significant amount of gap in most cases. In fact, of the 49 instances left, the average gap closed without DB-k is 75% and the best gap closed among all DB-k is 57%.

The above averages indicate that, while using DB-k forms lets us close a significant amount of gap in absolute terms, the results are comparatively weaker than for the full split closure in many cases. This suggests that using DB-k decompositions may not always pay off. To try and discard bad decompositions, we filtered the results in Table 4. The results are summarized in Fig. 6. For a given DB-k decomposition and a value of ρ, we first removed from the DB-k results the ones obtained from a decomposition where either the percentage of linking constraints or variables were above ρ percent. Then, for each remaining instance, we computed the relative gap closed (RGAP) as:

$$\frac{\text{GAP}k}{\text{GAPnodb}} \tag{RGAP}$$

The following statistics are shown in Fig. 6 for the instances remaining after filtering:

– Average (bold line)
– 10-th percentile (dashed line)
– Median (solid line)
– 25-75th percentile (shaded region)

Note that, while in principle (RGAP) should be always at most 100%, due to time limits, it is possible that the result from (GAPnodb) is not as high as it should be, resulting in (RGAP) above 100%. The results in Fig. 6 show that eliminating DB-k forms with a high number of linking variables or constraints is indeed a good indicator to filter out results where split cuts from DB-k form do not close too much gap.

Table 4 Gap closed for the full split closure, and for sparse ±1 split cuts for DB-k only

Instance	Without DB-k				DB-2				DB-3			
	Gap closed	# Cuts binding	Time (h)	% Time checking	Gap closed	# Cuts binding	Time (h)	% Time checking	Gap closed	# Cuts binding	Time (h)	% Time checking
10teams	93.76	1245	168.00	14.50	96.99	1096	168.00	6.97	100.00	807	168.00	7.93
air04	89.57	525	168.00	67.36	45.61	171	110.02	7.45	NA	NA	NA	NA
air05	63.53	370	168.00	16.50	0.00	0	1.01	0.00	NA	NA	NA	NA
arki001	42.61	236	32.54	0.24	33.03	145	8.76	0.13	33.38	171	42.03	0.05
bell3a	75.73	81	0.08	0.20	70.74	83	0.00	10.00	70.74	64	0.00	10.00
bell5	92.57	222	0.29	0.03	92.01	90	0.11	0.05	91.94	84	0.00	2.22
blend2	45.56	72	0.86	0.12	19.79	14	0.00	0.00	19.79	16	0.00	3.33
cap6000	64.69	119	2.12	0.65	0.00	0	0.00	0.00	0.00	0	0.00	0.00
dano3mip	0.24	492	168.00	36.16	0.21	205	168.00	26.47	0.25	267	168.00	25.20
danoint	8.98	393	168.00	0.74	0.77	72	3.68	0.80	0.80	63	0.32	4.28
dcmulti	99.84	309	11.71	0.01	95.57	196	17.34	0.01	73.24	138	2.09	0.01
egout	98.64	230	0.00	2.22	95.63	132	0.01	0.70	92.14	90	0.00	2.00
fast0507	7.50	401	168.00	61.66	0.00	0	0.01	0.00	NA	NA	NA	NA
fiber	72.99	263	168.00	0.02	65.50	200	168.00	0.01	73.18	177	168.00	0.01
fixnet6	99.86	342	168.00	0.01	84.26	274	1.54	0.08	83.56	319	0.08	0.49
flugpl	98.49	124	0.00	0.59	90.16	18	0.00	0.00	65.03	13	0.00	0.00
gen	100.20	378	0.08	6.95	100.20	225	0.12	3.08	100.20	203	0.54	0.47
gesa2	100.00	277	36.31	0.01	99.97	192	20.38	0.03	99.95	214	8.28	0.02
gesa2_o	99.99	274	37.79	0.01	44.60	125	19.68	0.00	38.67	166	2.50	0.02
gesa3	96.01	292	54.24	0.01	95.87	190	30.06	0.01	95.87	119	28.00	0.01
gesa3_o	96.07	242	122.70	0.01	95.85	169	66.91	0.01	96.08	184	119.44	0.00
gt2	98.89	175	75.47	0.01	0.00	0	0.00	NA	0.00	0	0.00	NA
harp2	45.21	161	70.59	0.02	0.00	0	0.00	0.00	0.00	0	0.00	0.00

Table 4 continued

Instance	Without DB-k				DB-2				DB-3			
	Gap closed	# Cuts binding	Time (h)	% Time checking	Gap closed	# Cuts binding	Time (h)	% Time checking	Gap closed	# Cuts binding	Time (h)	% Time checking
khb05250	100.00	451	0.01	12.50	53.48	19	0.00	2.00	53.48	19	0.00	1.67
l152lav	48.93	199	168.00	0.33	0.24	23	0.84	0.04	NA	NA	NA	NA
lseu	76.16	103	65.68	0.00	52.62	64	0.00	1.00	19.94	52	0.00	2.00
misc03	51.49	216	57.13	0.01	0.00	6	0.00	NA	0.00	0	0.00	NA
misc06	100.00	168	0.02	11.82	94.18	45	0.00	13.33	26.55	27	0.00	5.00
misc07	16.62	322	168.00	0.01	0.00	54	0.25	0.40	0.00	0	0.00	0.00
mitre	23.52	2195	168.00	0.28	55.23	2098	168.00	0.63	63.34	3758	168.00	0.46
mkc	65.11	942	168.00	1.27	62.90	467	168.00	0.35	69.08	198	168.00	0.22
mod010	100.00	384	2.78	9.36	0.00	0	2.41	0.00	NA	NA	NA	NA
mod011	85.35	1115	168.00	0.78	89.12	1129	168.00	0.92	92.51	1256	168.00	0.85
modglob	95.49	164	132.90	0.00	93.08	161	4.30	0.02	85.58	117	0.68	0.05
p0201	71.31	162	168.00	0.01	54.14	93	78.58	0.00	50.00	103	10.39	0.01
p0282	98.45	132	149.50	0.00	3.51	79	0.01	0.26	83.88	52	0.00	0.59
p0548	96.37	369	168.00	0.00	90.20	278	114.73	0.00	89.83	287	167.93	0.00
p2756	87.29	467	119.63	0.01	88.29	408	62.77	0.01	86.99	427	17.61	0.02
pp08a	97.04	184	134.58	0.00	95.53	188	0.08	0.10	94.28	169	0.03	0.53
pp08aCUTS	95.82	196	83.59	0.00	93.73	159	0.43	0.10	92.80	186	0.03	0.98
qiu	78.09	419	168.00	0.24	78.09	262	168.00	0.15	0.00	0	0.16	0.00
qnet1	100.00	230	2.61	0.16	2.44	23	2.14	0.00	2.45	42	1.30	0.01
qnet1_o	100.00	369	3.51	0.05	20.96	9	0.00	NA	20.96	9	0.00	NA
rgn	75.15	616	107.75	0.00	68.40	187	2.18	0.03	48.20	115	0.01	1.74
rout	68.99	293	168.00	0.08	42.00	319	107.82	0.05	44.14	393	33.88	0.08

Table 4 continued

Instance	Without DB-k				DB-2				DB-3			
	Gap closed	# Cuts binding	Time (h)	% Time checking	Gap closed	# Cuts binding	Time (h)	% Time checking	Gap closed	# Cuts binding	Time (h)	% Time checking
set1ch	89.75	342	6.41	0.01	89.73	332	0.17	0.40	89.73	349	0.26	0.54
seymour	57.96	1039	168.00	18.67	35.94	439	168.00	21.80	53.89	708	168.00	23.00
swath	28.93	305	168.00	0.54	19.08	107	0.05	10.53	16.65	73	0.06	3.39
vpm1	100.00	488	0.11	0.34	78.18	189	0.01	1.90	34.55	115	0.00	2.22
vpm2	81.46	217	22.41	0.00	73.85	135	0.92	0.02	74.59	157	0.16	0.21
Average	75.60	386	90.51	5.29	53.23	217	40.03	2.34	53.96	260	35.82	2.37

Instance	DB-4				DB-5				Best
	Gap closed	# Cuts binding	Time (h)	%Time checking	Gap closed	# Cuts binding	Time (h)	% time checking	gap (DB-k)
10teams	100.00	1415	168.00	3.32	100.00	1295	168.00	1.17	100.00
air04	NA	NA	NA	NA	NA	NA	NA	NA	45.61
air05	NA	NA	NA	NA	NA	NA	NA	NA	0.00
arki001	32.31	173	0.77	0.94	32.31	186	2.89	0.66	33.38
bell3a	70.74	59	0.00	0.00	70.66	56	0.00	10.00	70.74
bell5	88.99	53	0.01	0.57	86.83	58	0.00	1.25	92.01
blend2	19.79	17	0.00	0.00	19.79	13	0.00	0.00	19.79
cap6000	0.00	0	0.00	0.00	0.00	0	0.00	0.00	0.00
dano3mip	0.22	214	168.00	27.97	0.25	246	168.00	36.12	0.25
danoint	0.30	30	0.05	6.95	0.72	83	0.05	37.81	0.80
dcmulti	69.56	137	5.56	0.02	60.28	130	1.22	0.02	95.57
egout	86.56	98	0.00	3.33	93.86	93	0.00	2.31	95.63
fast0507	NA	NA	NA	NA	NA	NA	NA	NA	0.00

Table 4 continued

Instance	DB-4				DB-5				Best
	Gap closed	# Cuts binding	Time (h)	%Time checking	Gap closed	# Cuts binding	Time (h)	% time checking	gap (DB-k)
fiber	24.22	90	48.54	0.01	15.77	52	13.48	0.01	73.18
fixnet6	84.43	366	0.05	1.05	84.86	413	0.03	3.80	84.86
flugpl	7.05	4	0.00	0.00	45.83	4	0.00	0.00	90.16
gen	100.20	380	0.04	9.47	100.20	392	0.02	9.19	100.20
gesa2	99.95	253	0.43	0.42	99.94	209	21.69	0.01	99.97
gesa2_o	38.59	161	10.73	0.01	38.60	155	4.63	0.01	44.60
gesa3	95.84	206	47.65	0.00	96.03	290	32.57	0.01	96.03
gesa3_o	46.56	238	2.59	0.04	60.21	208	8.91	0.01	96.08
gt2	0.00	0	0.00	0.00	0.00	0	0.00	0.00	0.00
harp2	0.00	0	0.00	0.00	0.00	0	0.00	0.00	0.00
khb05250	53.48	19	0.00	1.67	53.48	19	0.00	1.11	53.48
l152lav	NA	NA	NA	NA	NA	NA	NA	NA	0.24
lseu	38.58	37	0.00	0.00	4.21	5	0.00	0.00	52.62
misc03	0.00	0	0.00	NA	0.00	0	0.00	NA	0.00
misc06	68.14	39	0.00	10.00	97.51	52	0.00	9.29	97.51
misc07	0.00	0	0.00	NA	NA	NA	NA	NA	0.00
mitre	64.02	2756	168.00	0.59	65.93	2160	168.00	0.56	65.93
mkc	69.84	543	168.00	0.20	72.91	353	168.00	0.19	72.91
mod010	NA	NA	NA	NA	NA	NA	NA	NA	0.00
mod011	89.01	1388	168.00	1.03	90.61	1217	168.00	1.00	92.51
modglob	85.87	128	0.10	0.38	83.64	119	0.05	0.65	93.08
p0201	0.00	0	0.00	NA	0.00	0	0.00	NA	54.14

Table 4 continued

Instance	DB-4				DB-5				Best
	Gap closed	# Cuts binding	Time (h)	%Time checking	Gap closed	# Cuts binding	Time (h)	% time checking	gap (DB-k)
p0282	1.52	10	0.00	0.00	83.87	35	0.01	0.80	83.88
p0548	89.73	313	168.00	0.00	88.20	344	146.88	0.00	90.20
p2756	85.17	540	13.44	0.02	86.04	478	17.91	0.01	88.29
pp08a	95.59	184	0.02	0.57	95.18	201	0.01	2.07	95.59
pp08aCUTS	93.37	214	0.02	3.11	89.06	185	0.02	2.00	93.73
qiu	78.09	946	75.38	0.31	0.00	0	0.16	0.00	78.09
qnet1	2.45	42	3.48	0.00	1.90	29	2.33	0.00	2.45
qnet1_o	20.96	6	0.00	NA	20.96	6	0.00	0.00	20.96
rgn	35.86	116	0.00	3.75	NA	NA	NA	NA	68.40
rout	41.75	282	33.04	0.05	71.42	513	164.45	0.03	71.42
set1ch	89.73	452	0.15	0.89	89.73	427	0.08	1.07	89.73
seymour	54.63	481	168.00	23.02	60.48	918	168.00	24.04	60.48
swath	11.26	87	0.01	31.59	12.78	120	0.01	31.11	19.08
vpm1	34.55	101	0.03	0.18	42.42	85	0.00	2.86	78.18
vpm2	57.52	132	0.02	0.55	66.38	182	0.01	1.60	74.59
Average	49.48	282	31.51	3.22	53.09	264	33.15	4.41	56.73

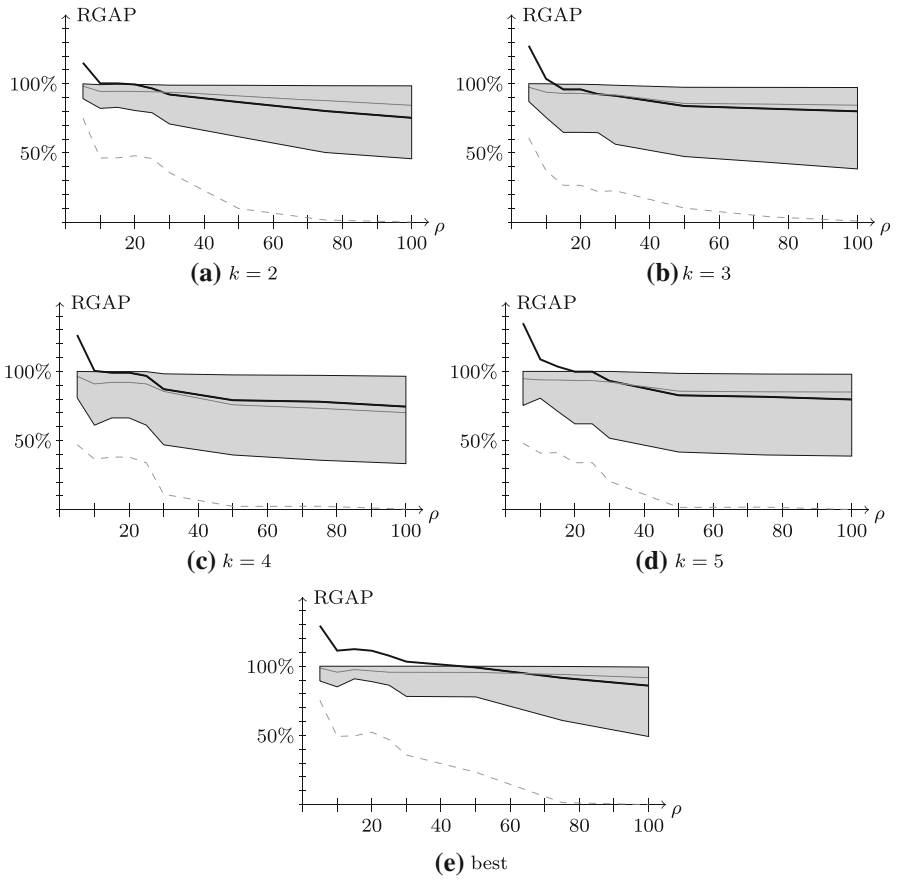


Fig. 6 Distribution of relative gap closed for DB-k forms for several values of ρ

The above results show that the gap loss is not too big when restricting ourselves to split cuts from DB-k form. We now try to understand how structured sparsity helps to produce more effective cuts. Table 5 shows the average support size in the first 100 disjunctions (of a given type) obtained by our implementation, and the corresponding average cut density, for instances `10teams`, `mkc`, and `seymour`. We picked `10teams` as an extreme example where, without utilizing a DB-2 structure, highly sparse disjunctions (8.3 nonzero entries, which accounts for only 5% of the 1800 integer variables) have produced almost completely dense cuts. Instance `mkc` and `seymour` were picked because they represent reasonably large instances that are also in MIPLIB 2003. We observe that, as expected, exploiting the DB-2 structure yields sparser cuts. Furthermore, the last row of Table 5 shows that disjunctions with arbitrarily many nonzero entries that are much denser still lead to sparse cuts when exploiting problem structure.

In Fig. 7 we compare the distributions of average cut densities on the 40 MIPLIB 3.0 instances whose DB-2 forms have at most 50% linking variables or constraints.

Table 5 Disjunction and cut density for three example instances

Instance	Disjunction type	Average support size of disjunctions (#)	Average cut density (%)
10teams	$M = 10,\ U = 1$	8.3	95.5
10teams	$M = 10,\ U = 1$, with DB-2	8.2	54.6
mkc	$M = 10,\ U = 1$	9.4	12.6
mkc	$M = 10,\ U = 1$, with DB-2	9.2	3.8
seymour	$M = 10,\ U = 1$	9.8	9.6
seymour	$M = 10,\ U = 1$, with DB-2	8.2	3.7
seymour	$M = +\infty,\ U = 1$, with DB-2	255.2	10.3

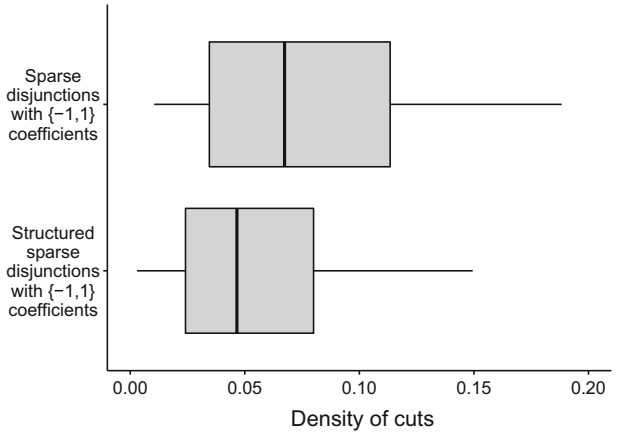
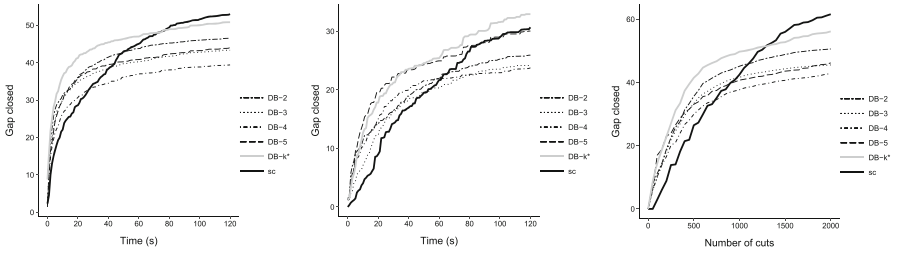


Fig. 7 Distribution of cut densities for DB-2

We picked DB-2 as a candidate for comparison because this is the simplest DB-k decompostion, having just 2 blocks. Other decompositions that contain more blocks all demonstrate a similar pattern. The 50% threshold was applied so that instances whose DB-k forms have a high number of linking variables or constraints are excluded from comparison. As discussed earlier, split cuts based on these decompositions are unlikely to close much gap, regardless of how sparse they are. The cut densities in category "Sparse disjunctions with $\{-1,1\}$ coefficients" are computed based on the cuts obtained in the previous section, and the cut densities under "Structured sparse disjunctions" are computed based on the results with DB-2 forms. As seen in Fig. 7, block structures lead to the sparsest cuts: Even comparing with the disjunctions ($M = 10, U = 1$) that previously led to the sparsest cuts, it further decreased the median of cut densities from 0.067 to 0.047, and the 75 percentile from 0.115 to 0.080.

Finally, we illustrate one more potential advantage of using split cuts based on DB-k forms. Figure 8 shows the evolution of the average gap closed in terms of runtime of our cut procedure and in terms of number of cuts added in our cut procedure. It can be seen that the split cuts obtained by using DB-k forms converge faster to a gap closer to



(a) Average gap closed over time

(b) Average gap closed over time for large instances

(c) Average gap closed per cuts added

Fig. 8 Evolution of average gap closed

the final gap both in terms of time (Fig. 8a) and in terms of number of cuts (Fig. 8c). The gain in terms of time is even more pronounced if we focus on large instances, that is, instances with at least 1000 variables, among which at least 50 are integer (Fig. 8b). The grey lines labelled "DB-k*" in Fig. 8 represent the average gap closed had we chosen for each instance the DB-k form that closes the most gap after adding up to 500 cuts. While it is hard to completely attribute these gains to a few factors only, we note that the most apparent difference between these cuts and those generated earlier is their higher degree of both sparsity and orthogonality.

4.2.4 Results on MIPLIB 2003 instances

Our final set of experiments was to run our code on larger instances than were previously available in the literature. For this purpose, we ran our code on MIPLIB 2003 [1] instances. However, since these instances are typically larger than the ones available in MIPLIB 3.0, we were able to run our code only using the parameters $M = 10$ and $U = 1$ and imposing a time limit of two weeks. Table 6 shows the results for those instances that are in MIPLIB 2003 but not in MIPLIB 3.0. Since there are no previous split closure numbers for those instances, we compare against the lift-and-project results of Bonami [13]. Compared to lift-and-project, significantly more gap can still be closed with the split closure approximation that does not exploit DB-k structure. Also, note that, even though the average results for DB-k based cuts are not as good, there are some instances where these results are significantly better than any of the other approaches, closing as much as 100% of the gap.

5 Conclusion

The main motivation for this work was to search for subsets of split cuts with promising computational properties. Our approach was to develop a tool that can empirically answer the following question: How much can we restrict the set of split cuts that we separate over, while retaining enough of the strength of the first split closure? While our tool is rather general (it can be seen as a continuation to Balas and Saxena's separation algorithm [6]), the specific restrictions that we explore aim at two desirable characteristics: First, we want sparse cuts, because they are beneficial to the linear

Table 6 Gap closed in MIPLIB2003 instances

L&P	Str. L&P	Instance	Without DB-k				DB-2				DB-3			
gap [13]	gap [13]		Gap closed	# Cuts binding	Time (h)	% Time checking	Gap closed	# Cuts binding	Time (h)	% Time checking	Gap closed	# Cuts binding	Time (h)	% Time checking
78.76	78.76	a1c1s1	93.54	398	336.00	0.03	92.55	680	99.08	0.03	90.65	649	147.50	0.03
42.41	43.27	aflow30a	65.09	245	336.00	0.00	3.43	19	0.67	0.00	0.00	0	0.05	0.00
34.29	35.87	aflow40b	52.56	335	336.00	0.03	0.00	0	0.97	0.00	0.00	0	0.06	0.00
1.09	1.77	atlanta-ip	0.00	204	336.00	7.99	NA	NA	NA	NA	NA	NA	NA	NA
0.00	0.13	glass4	0.00	204	2.27	0.22	0.00	70	0.26	0.35	0.00	84	0.15	1.35
NA	NA	manna81	82.80	1944	336.00	0.03	96.84	1813	336.00	0.06	88.27	1613	336.00	0.15
44.88	45.15	momentum1	40.76	425	336.00	39.62	41.22	568	336.00	54.63	28.91	317	336.00	75.16
41.47	41.84	momentum2	65.68	824	336.00	64.68	26.28	200	25.55	70.23	14.01	278	43.26	51.01
42.23	44.65	msc98-ip	0.00	0	0.07	0.00	0.00	0	1.28	0.00	0.00	0	2.54	4.00
56.47	100.00	mzzv11	63.35	648	336.00	78.10	46.42	364	336.00	83.22	57.73	426	336.00	84.87
87.73	100.00	mzzv42z	79.92	612	336.00	77.71	43.22	400	336.00	84.71	91.73	483	336.00	86.99
22.73	22.71	net12	5.97	862	336.00	46.87	2.87	543	336.00	37.02	6.61	672	336.00	31.85
36.88	77.09	nsrand-ipx	65.38	1075	336.00	0.10	61.08	1057	336.00	0.09	50.25	843	336.00	0.06
0.19	26.32	opt1217	4.89	387	336.00	0.42	0.00	0	0.01	0.00	0.00	0	0.00	0.00
10.29	10.83	protfold	37.41	2172	336.00	24.98	1.68	1055	336.00	29.50	1.14	1130	336.00	29.43
0.00	0.00	rd-rplusc-21	0.00	203	22.59	1.52	NA	NA	NA	NA	NA	NA	NA	NA
16.31	55.90	roll3000	37.90	504	291.02	0.57	NA	NA	NA	NA	NA	NA	NA	NA
42.06	59.91	sp97ar	65.04	533	336.00	0.54	47.62	410	336.00	1.29	41.47	355	336.00	1.15
26.99	42.45	timtab1	86.10	265	23.70	0.01	54.71	169	13.34	0.00	40.56	151	4.18	0.01

Table 6 continued

L&P gap [13]	Str. L&P gap [13]	Instance	Without DB-k Gap closed	# Cuts binding	Time (h)	% Time checking	DB-2 Gap closed	# Cuts binding	Time (h)	% Time checking	DB-3 Gap closed	# Cuts binding	Time (h)	% Time checking
20.98	40.18	timtab2	84.67	430	283.14	0.01	55.39	297	21.38	0.02	46.24	279	23.24	0.01
64.12	64.12	tr12-30	88.05	680	30.95	0.01	85.60	676	5.99	0.05	84.60	699	3.75	0.06
31.90	42.43	average	48.53	617	255.13	16.35	31.38	396	136.03	17.20	30.58	380	138.70	17.43

L&P gap [13]	Str. L&P gap [13]	Instance	DB-4 Gap closed	# Cuts binding	Time (h)	% Time checking	DB-5 Gap closed	# Cuts binding	Time (h)	% Time checking	Best gap (DB-k)
78.76	78.76	a1c1s1	90.85	689	39.81	0.08	88.69	688	19.18	0.19	92.55
42.41	43.27	aflow30a	0.00	0	0.01	0.00	0.00	0	0.00	0.00	3.43
34.29	35.87	aflow40b	0.00	0	0.25	0.00	0.00	0	0.00	0.00	0.00
1.09	1.77	atlanta-ip	NA	NA	NA	NA	NA	NA	NA	NA	NA
0.00	0.13	glass4	0.00	57	0.07	1.00	0.00	87	0.21	0.71	0.00
NA	NA	manna81	100.00	1974	62.90	0.24	100.00	1796	34.64	0.29	100.00
44.88	45.15	momentum1	37.06	487	336.00	76.85	34.83	540	336.00	49.24	41.22
41.47	41.84	momentum2	27.59	478	30.57	64.42	17.92	523	22.55	65.76	27.59
42.23	44.65	msc98-ip	0.00	0	1.80	21.10	0.00	0	1.94	37.73	0.00
56.47	100.00	mzzv11	56.55	587	336.00	86.50	65.17	602	336.00	89.08	65.17
87.73	100.00	mzzv42z	71.99	739	336.00	84.86	91.18	827	336.00	80.45	91.73
22.73	22.71	net12	4.49	628	336.00	38.61	5.77	741	336.00	33.96	6.61
36.88	77.09	nsrand-ipx	52.82	907	336.00	0.10	49.61	878	336.00	0.08	61.08
0.19	26.32	opt1217	0.00	0	0.00	0.00	0.00	0	0.00	0.00	0.00
10.29	10.83	protfold	1.19	1273	336.00	32.47	1.29	1118	336.00	25.03	1.68

Table 6 continued

L&P gap [13]	Str. L&P gap [13]	Instance	DB-4				DB-5				Best gap (DB-k)
			Gap closed	# Cuts binding	Time (h)	% Time checking	Gap closed	# Cuts binding	Time (h)	% Time checking	
0.00	0.00	rd-rplusc-21	NA	NA	NA	NA	NA	NA	NA	NA	NA
16.31	55.90	roll3000	NA	NA	NA	NA	NA	NA	NA	NA	NA
42.06	59.91	sp97ar	41.12	318	336.00	1.40	25.91	267	336.00	1.58	47.62
26.99	42.45	timtab1	34.08	158	2.56	0.01	29.17	183	0.07	0.22	54.71
20.98	40.18	timtab2	31.92	275	26.04	0.01	31.40	270	5.42	0.01	55.39
64.12	64.12	tr12-30	85.01	679	0.64	0.29	84.04	692	0.75	0.23	85.60
31.90	42.43	average	30.22	440	119.84	19.43	29.76	439	116.04	18.31	34.97

algebra that underlies MIP solution methods. Secondly, we want cuts that are computed from different parts of the constraint matrix, and involve varied subsets of the variables. The latter point corresponds to generating cuts that are (approximately) mutually orthogonal, to as high a degree as possible, and it has been observed [6,25] to be favorable in getting tighter relaxations with fewer cuts.

Our experiments show that explicitly enforcing sparsity of the split *disjunctions*, and bounding the magnitude of their coefficients, yields one such promising family of split cuts. We observe that the resulting cuts themselves are sparse too, which was expected but not a priori obvious. More surprisingly, even in an extreme setting where we only allow 10 nonzero disjunction coefficients with values ± 1, we obtain cuts that are 91% as effective as all split cuts together (in terms of gap closed, and compared to the best known results for the split closure [6,19]). Note, for context, that were we to only allow one nonzero coefficient, we would obtain the lift-and-project closure of Balas, Ceria and Cornuéjols [4].

Next, in the same spirit of restricting the split disjunctions available to us, we exploit problem structure to impose static constraints on how cuts are generated. Specifically, we start by computing block decompositions of our problems. Then, we force our split cut generator to use, for each cut, only constraints and variables from a single block. In a second series of experiments, we test this approach with arrowhead decompositions [9,26] of the constraint matrices, while keeping the same limitations on the disjunctions as before. In this even more restricted setting, we observe a significant degradation of the average gap closure. However, we demonstrate that it is easy to determine a priori which instances will benefit from block decompositions, and which will not. With a very simple rule based on the number the linking constraints and variables, we are able to isolate the instances that are most suited for this technique. By using decompositions only when appropriate, we get a subset of instances on which, due to time limits, we close even more gap than without decomposition. Moreover, as a general rule, we observe that this setting lets us cut much more gap *per cut* on average. We attribute this desirable feature to the orthogonality of the cuts generated.

Overall, our results suggest that there exist small subsets of split cuts that exhibit advantageous properties, and that are yet to be exploited.

Acknowledgements We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), [funding reference number RGPIN-2018-04335 and RGPIN-2014-05623].

Appendix

Effect of heuristic features in computation

In order to examine the effect of some features and modifications we introduced to our separation routine, we ran the code on MIPLIB 3.0 instances with $M = +\infty$ and $U = 100$. We set the global time limit to an hour, and disabled the following features one at a time:

– cut strengthening (`cut_str_off`);

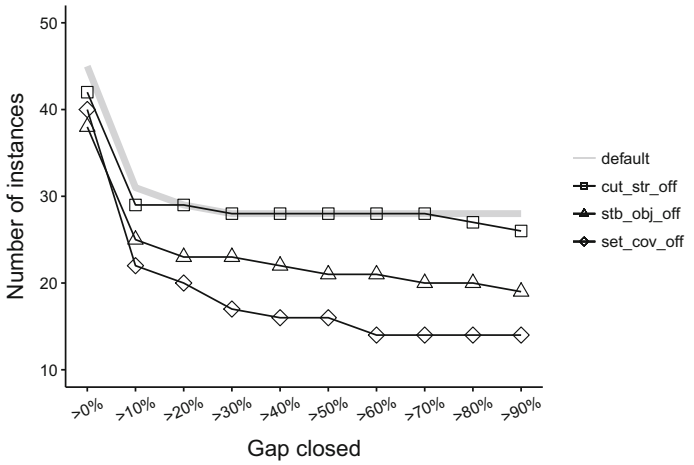


Fig. 9 Effect of heuristic features

Table 7 Average gap closed under different settings

Setting	Average gap closed
`default`	31.04%
`cut_str_off`	28.35%
`stb_obj_off`	21.32%
`set_cov_off`	12.41%

– stabilizing objective (`stb_obj_off`);
– set covering (`set_cov_off`).

We then plotted, for each scenario, the number of instances on which at least $x\%$ integrality gap was closed, for $x \in \{10, 20, \ldots, 90\}$. As shown in Fig. 9, the additional features indeed helped to obtain better results.

Among the three heuristics compared, using set covering to impose partial orthogonality between split cuts plays the most important role. The advantage of stabilizing objective is also evident. Note that, these results were obtained with a bound $U = 100$ already imposed on the disjunction coefficients. We thus expect to observe a more dramatic gain from stabilized objective, had we chosen a larger bound U to begin with. Table 7 shows the average gap closed in each computational setting. Note that we were able to close significantly more integrality gap when all features were used (`default`).

Gap closed for the full split closure under a modified separation routine

Our original implementation for the separation of arbitrary split cuts (with bounds $U = 100$ on disjunction coefficients) essentially finds a feasible solution to MILP(θ) whose objective value is less than a preset cut violation threshold $\epsilon < 0$. In theory, the same result can be achieved by adding the requirement that the objective value be less

Table 8 Gap closed for the full split closure under the original and the modified MILP(θ)

Instance	Original MILP(θ)				Modified MILP(θ)				Modified MILP(θ) + GMI			
	Gap closed	# Cuts binding	Time (s)	% Time checking	Gap closed	# Cuts binding	Time (s)	% Time checking	Gap closed	# Cuts binding	Time (s)	% Time checking
10teams	0.00	4098	86,400.00	27.18	34.92	1035	86,400.00	8.21	34.34	900	86,400.00	11.58
air03	0.31	480	86,400.00	98.08	100.00	56	2.00	70.00	100.00	32	0.00	0.00
air04	0.00	4323	86,400.00	96.87	79.48	345	86,400.00	66.43	86.28	523	86,400.00	52.43
air05	0.03	428	86,400.00	97.91	58.13	394	86,400.00	28.77	60.30	418	86,400.00	22.36
arki001	0.00	2602	86,400.00	95.07	63.25	189	86,400.00	0.14	60.79	175	86,400.00	0.07
bell3a	99.64	81	71,493.00	0.03	86.45	74	86,400.00	0.01	89.56	69	86,400.00	0.01
bell5	93.22	174	74,604.00	0.01	92.70	85	1833.00	0.01	92.94	123	4469.00	0.01
blend2	35.03	453	10,135.00	2.01	22.38	60	111.00	0.54	22.40	52	93.00	0.43
cap6000	64.95	362	86,400.00	0.69	46.67	49	427.00	1.48	46.67	50	929.00	0.46
dano3mip	0.00	504	86,400.00	95.09	0.11	241	86,400.00	46.26	0.18	263	86,400.00	37.92
danoint	8.42	466	86,400.00	8.25	7.44	327	86,400.00	0.70	7.64	324	86,400.00	0.49
dcmulti	100.00	1098	6486.00	6.32	99.91	294	73,163.00	0.01	99.86	317	16,439.00	0.01
egout	100.00	281	597.00	0.40	100.00	206	782.00	0.04	100.00	209	228.00	0.04
fast0507	0.00	1437	86,400.00	89.44	10.38	269	86,400.00	24.13	8.74	376	86,400.00	56.54
fiber	0.04	442	86,400.00	38.34	55.44	318	86,400.00	0.18	81.99	211	86,400.00	0.14
fixnet6	99.84	536	86,400.00	0.16	99.79	639	86,400.00	0.04	99.81	592	86,400.00	0.03
flugpl	100.00	158	200.00	0.05	100.00	81	684.00	0.01	100.00	65	304.00	0.03
gen	90.28	510	45,719.00	11.26	100.00	260	14,206.00	0.14	100.00	206	20,457.00	0.06
gesa2	93.84	251	86,400.00	0.86	85.23	184	9170.00	0.11	99.93	252	42,938.00	0.01
gesa2_o	79.62	347	86,400.00	2.78	90.66	188	86,400.00	0.01	94.05	267	33,991.00	0.02
gesa3	18.49	342	86,400.00	5.42	95.98	304	86,400.00	0.02	95.94	302	86,400.00	0.02

Table 8 continued

Instance	Original MILP(θ)				Modified MILP(θ)				Modified MILP(θ) + GMI			
	Gap closed	# Cuts binding	Time (s)	% Time checking	Gap closed	# Cuts binding	Time (s)	% Time checking	Gap closed	# Cuts binding	Time (s)	% Time checking
gesa3_o	0.45	482	19,419.00	12.80	96.03	308	86,400.00	0.01	96.03	316	86,400.00	0.02
gt2	71.29	2155	86,400.00	1.05	97.79	153	86,400.00	0.01	93.87	89	11,401.00	0.01
harp2	0.02	142	812.00	17.06	26.59	146	44,365.00	0.02	24.98	132	43,398.00	0.02
khb05250	100.00	308	703.00	0.65	100.00	293	695.00	0.45	100.00	295	345.00	0.61
l152lav	0.14	562	86,400.00	79.93	30.98	182	86,400.00	1.63	34.51	156	86,400.00	0.98
lseu	87.97	104	86,400.00	0.02	83.27	89	86,400.00	0.00	83.54	73	86,400.00	0.00
mas74	14.28	225	86,400.00	1.02	7.70	26	413.00	0.44	7.73	30	365.00	0.44
mas76	24.84	123	86,400.00	0.45	3.87	26	31.00	1.29	5.50	35	35.00	1.43
misc03	0.49	468	86,400.00	12.09	51.54	173	55,599.00	0.01	51.01	179	86,400.00	0.01
misc06	100.00	279	160.00	8.31	100.00	251	106.00	2.74	100.00	213	194.00	1.75
misc07	0.01	1714	86,400.00	16.11	15.65	271	86,400.00	0.03	16.34	241	86,400.00	0.02
mitre	0.00	2148	86,400.00	5.38	100.00	4907	83,744.00	0.60	100.00	8706	22,268.00	0.68
mkc	0.00	6424	86,400.00	9.21	52.52	893	86,400.00	0.20	63.11	1011	86,400.00	0.22
mod008	93.19	375	86,400.00	1.22	64.82	142	86,400.00	0.02	62.42	127	86,400.00	0.01
mod010	0.09	2690	86,400.00	46.52	95.83	170	86,400.00	5.23	100.00	550	5262.00	17.35
mod011	16.06	4487	86,400.00	9.82	60.17	851	22,199.00	1.41	62.73	789	86,400.00	1.24
modglob	95.97	191	55,564.00	0.03	84.63	144	643.00	0.20	78.05	142	701.00	0.24
nw04	0.03	421	86,400.00	95.72	42.72	258	86,400.00	96.11	45.36	283	86,400.00	95.38
p0033	86.06	50	86,400.00	0.00	84.61	63	86,400.00	0.00	84.84	59	86,400.00	0.00
p0201	20.91	431	86,400.00	14.75	71.22	169	86,400.00	0.04	72.16	152	86,400.00	0.02
p0282	99.22	125	86,400.00	0.10	97.91	123	86,400.00	0.00	97.77	136	86,400.00	0.01
p0548	0.00	6407	27,863.00	1.14	94.96	288	86,400.00	0.01	93.16	304	86,400.00	0.01
p2756	0.00	6412	42,414.00	2.07	84.20	438	86,400.00	0.01	84.46	418	80,290.00	0.01

Table 8 continued

Instance	Original MILP(θ)				Modified MILP(θ)				Modified MILP(θ) + GMI			
	Gap closed	# Cuts binding	Time (s)	% Time checking	Gap closed	# Cuts binding	Time (s)	% Time checking	Gap closed	# Cuts binding	Time (s)	% Time checking
pk1	0.00	6220	86,400.00	0.13	0.00	277	86,400.00	0.00	0.00	518	86,400.00	0.00
pp08a	97.02	187	86,400.00	0.06	97.05	211	64,310.00	0.00	97.04	211	45,023.00	0.00
pp08aCUTS	95.79	168	86,400.00	0.25	95.74	217	86,400.00	0.00	95.80	190	68,985.00	0.00
qiu	78.05	350	86,400.00	4.15	78.03	391	86,400.00	0.44	78.07	390	86,400.00	0.34
qnet1	0.08	1016	86,400.00	20.94	99.39	181	86,400.00	0.04	99.70	139	86,400.00	0.02
qnet1_o	0.08	611	86,400.00	22.86	100.00	192	17,996.00	0.03	99.86	190	86,400.00	0.03
rentacar	54.03	326	78,331.00	9.61	49.15	396	86,400.00	6.86	46.65	386	86,400.00	7.72
rgn	100.00	438	50,849.00	0.10	89.80	139	86,400.00	0.01	85.33	142	86,400.00	0.02
rout	0.00	6411	66,795.00	5.69	30.96	338	86,400.00	1.47	36.06	290	86,400.00	1.69
set1ch	0.10	387	86,400.00	2.45	89.68	318	86,400.00	0.00	89.17	304	54,874.00	0.00
seymour	0.00	2835	86,400.00	41.13	59.34	676	86,400.00	10.54	59.75	738	86,400.00	10.80
stein27	0.00	156	32,007.00	0.00	0.00	120	20,893.00	0.00	0.00	124	27,105.00	0.00
stein45	0.00	317	86,400.00	0.39	0.00	4725	86,400.00	0.26	0.00	2049	86,400.00	0.13
swath	0.00	4509	86,400.00	70.41	10.66	315	86,400.00	1.28	24.39	335	86,400.00	1.02
vpm1	100.00	400	17,217.00	0.25	100.00	353	18,258.00	0.01	98.55	297	86,400.00	0.00
vpm2	81.32	165	86,400.00	0.10	81.22	239	86,400.00	0.00	81.39	251	86,031.00	0.00
Average	40.02	1343	69,063	19.84	65.95	418	61,881	6.31	67.18	445	59,835	5.41

Table 9 Percentage of nonzero cut coefficients, **before** lifting, that (i) overlap with nonzero split coefficients (corr_n) (ii) overlap with the block where nonzero split coefficients come from (corr_b)

Instance	sc	DB-2		DB-3		DB-4		DB-5	
	corr_n	corr_n	corr_b	corr_n	corr_b	corr_n	corr_b	corr_n	corr_b
10teams	3.62	11.05	75.76	12.96	73.58	13.02	63.31	13.46	69.29
air03	26.39	NA	NA	NA	NA	NA	NA	NA	NA
air04	4.43	22.21	84.43	NA	NA	NA	NA	NA	NA
air05	3.28	NC	NC	NA	NA	NA	NA	NA	NA
arki001	46.02	43.21	95.35	39.70	91.63	43.72	93.30	38.70	91.94
bell3a	92.06	98.05	99.88	97.01	100.00	98.26	100.00	98.63	100.00
bell5	95.51	96.05	98.98	89.58	97.54	89.60	93.54	97.17	97.97
blend2	100.00	87.38	96.90	92.28	92.28	72.31	82.53	86.21	91.38
cap6000	60.89	NC	NC	NC	NC	NC	NC	NC	NC
dano3mip	6.98	6.49	100.00	5.77	100.00	6.43	100.00	5.04	98.65
danoint	52.06	47.41	71.12	43.84	61.77	37.22	49.43	22.22	37.26
dcmulti	81.92	83.99	99.87	74.27	99.39	72.78	98.89	67.81	100.00
egout	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
fast0507	3.41	NC	NC	NA	NA	NA	NA	NA	NA
fiber	51.42	64.08	98.44	66.41	98.03	69.13	91.77	75.87	94.89
fixnet6	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
flugpl	79.75	76.50	96.36	79.73	90.53	87.50	87.50	87.50	87.50
gen	81.21	84.00	98.53	83.86	99.45	84.94	98.12	84.45	98.98
gesa2	65.85	67.53	99.82	67.87	99.43	66.46	98.46	67.59	98.84
gesa2_o	65.32	80.29	95.73	78.34	94.16	75.05	91.24	77.92	94.15
gesa3	58.29	63.28	97.13	62.93	96.19	64.07	94.46	63.32	95.17
gesa3_o	57.02	61.17	98.87	58.77	96.27	79.53	99.54	71.24	94.87
gt2	59.32	NC	NC	NC	NC	NC	NC	NC	NC

Table 9 continued

Instance	sc	DB-2		DB-3		DB-4		DB-5	
	corr_n	corr_n	corr_b	corr_n	corr_b	corr_n	corr_b	corr_n	corr_b
harp2	38.27	NC	NC	NC	NC	NC	NC	NC	NC
khb05250	99.37	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
l152lav	6.11	37.27	92.15	NA	NA	NA	NA	NA	NA
lseu	76.81	90.68	98.66	88.33	97.14	97.97	99.01	100.00	100.00
mas74	25.37	NA	NA	NA	NA	NA	NA	NA	NA
mas76	28.13	NA	NA	NA	NA	NA	NA	NA	NA
misc03	30.77	41.94	68.52	NC	NC	NC	NC	NC	NC
misc06	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
misc07	22.78	54.06	83.85	NC	NC	NC	NC	NA	NA
mitre	48.88	51.62	100.00	59.42	100.00	61.39	100.00	65.35	100.00
mkc	29.72	31.36	92.76	29.00	86.74	23.77	83.69	24.96	82.03
mod008	32.20	NA	NA	NA	NA	NA	NA	NA	NA
mod010	15.59	NC	NC	NA	NA	NA	NA	NA	NA
mod011	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
modglob	92.92	97.72	100.00	98.38	100.00	95.32	98.96	95.95	100.00
nw04	6.46	NA	NA	NA	NA	NA	NA	NA	NA
p0033	75.32	NA	NA	NA	NA	NA	NA	NA	NA
p0201	25.06	38.05	99.41	44.14	98.83	NC	NC	NC	NC
p0282	45.02	68.78	91.95	49.89	66.67	75.83	75.83	52.57	63.95
p0548	65.16	79.59	98.59	78.96	98.53	78.51	96.88	78.21	97.50
p2756	81.68	88.23	100.00	88.48	99.93	88.92	99.49	88.06	99.84
pk1	21.24	NA	NA	NA	NA	NA	NA	NA	NA
pp08a	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Table 9 continued

Instance	sc	DB-2		DB-3		DB-4		DB-5	
	corr_n	corr_n	corr_b	corr_n	corr_b	corr_n	corr_b	corr_n	corr_b
pp08aCUTS	71.92	76.05	95.54	72.64	94.16	73.67	93.36	70.28	89.60
qiu	28.58	13.61	90.52	NC	NC	14.54	86.44	NC	NC
qnet1	36.15	59.22	85.41	47.98	68.49	54.41	78.23	50.03	75.37
qnet1_o	51.75	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
rentacar	97.96	NA	NA	NA	NA	NA	NA	NA	NA
rgn	68.77	84.81	99.58	82.39	97.51	78.09	97.34	NA	NA
rout	25.24	31.51	95.37	32.09	93.71	31.44	93.79	32.96	96.27
set1ch	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
seymour	19.49	25.56	92.31	34.10	94.53	37.66	92.64	39.83	95.17
stein27	33.38	51.28	68.07	28.09	62.72	NC	NC	NC	NC
stein45	12.32	26.90	64.18	19.43	48.19	16.73	43.00	31.81	57.85
swath	26.48	82.88	96.70	77.81	98.74	71.91	92.79	69.12	86.98
vpm1	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
vpm2	71.05	73.17	100.00	72.50	99.30	71.30	96.51	69.31	98.33
Average	53.41	67.33	93.93	69.68	92.57	70.79	91.75	71.73	91.94
Median	51.91	74.61	98.56	77.81	98.53	75.44	97.11	76.90	98.15

than ϵ, $s^\top \hat{x} - \theta(\pi^\top \hat{x} - \pi_0) < \epsilon$, as a constraint in MILP(θ), and then finding a feasible solution. Furthermore, in order to encourage some sparsity in split disjunctions, we can introduce a new objective function into MILP(θ), thus obtaining the following modified MILP(θ):

$$\begin{aligned} \min \quad & \sum_{j=1}^{p} r_j \\ \text{s.t. } & s^\top \hat{x} - \theta(\pi^\top \hat{x} - \pi_0) < \epsilon \\ & -Ur_j \le \pi_j \le Ur_j, \quad j = 1, 2, \ldots, p \\ & \texttt{plus original constraints of MILP}(\theta) \end{aligned}$$

Although, in theory, this modified formulation of MILP(θ) should produce the same results, provided that all the other computational parameters/heuristics are set to be the same, in practice the modified MILP(θ) could lead to very different results. In particular, we ran the modified separation routine with $U = 100$ as an approximation to the full split closure, and compare the results with those obtained from our original implementation. We set a global time limit to 24 h. As shown in Table 8, with the modified formulation of MILP(θ), we were able to close much more integrality gap on average within the same time limit.

Effect of split sparsity pattern on cut sparsity pattern

Table 9 presents the results discussed in Sect. 4 on how sparsity of the cuts and splits are related. Results of this table are for $M = 10$ and $U = 1$ parameters only.

References

1. Achterberg, T., Koch, T., Martin, A.: MIPLIB 2003. Oper. Res. Lett. **34**(4), 361–372 (2006). https://doi.org/10.1016/j.orl.2005.07.009
2. Andersen, K., Weismantel, R.: Zero-coefficient cuts. In: Eisenbrand, F., Shepherd, F.B. (eds.) Integer Programming and Combinatorial Optimization, pp. 57–70. Springer, Berlin (2010)
3. Aykanat, C., Pinar, A., Çatalyürek, U.: Permuting sparse rectangular matrices into block-diagonal form. SIAM J. Sci. Comput. **25**(6), 1860–1879 (2004). https://doi.org/10.1137/S1064827502401953
4. Balas, E., Ceria, S., Cornuéjols, G.: A lift-and-project cutting plane algorithm for mixed 0–1 programs. Math. Program. **58**(1–3), 295–324 (1993)
5. Balas, E., Ceria, S., Cornuéjols, G.: Mixed 0–1 programming by lift-and-project in a branch-and-cut framework. Manag. Sci. **42**(9), 1229–1246 (1996). https://doi.org/10.1287/mnsc.42.9.1229
6. Balas, E., Saxena, A.: Optimizing over the split closure. Math. Program. **113**(2), 219–240 (2008). https://doi.org/10.1007/s10107-006-0049-5
7. Bastubbe, M., Lübbecke, M.E., Witt, J.T.: A computational investigation on the strength of Dantzig–Wolfe reformulations. In: D'Angelo, G. (ed.) 17th International Symposium on Experimental Algorithms (SEA 2018), *Leibniz International Proceedings in Informatics (LIPIcs)*, vol. 103, pp. 11:1–11:12. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2018). https://doi.org/10.4230/LIPIcs.SEA.2018.11. http://drops.dagstuhl.de/opus/volltexte/2018/8946
8. Basu, A., Bonami, P., Cornuéjols, G., Margot, F.: On the relative strength of split, triangle and quadrilateral cuts. Math. Program. **126**(2), 281–314 (2011)

9. Bergner, M., Caprara, A., Ceselli, A., Furini, F., Lübbecke, M.E., Malaguti, E., Traversi, E.: Automatic Dantzig–Wolfe reformulation of mixed integer programs. Math. Program. **149**(1), 391–424 (2015). https://doi.org/10.1007/s10107-014-0761-5
10. Bixby, R.E.: A brief history of linear and mixed-integer programming computation. Documenta Mathematica, 107–121 (2012)
11. Bixby, R.E., Ceria, S., McZeal, C.M., Savelsbergh, M.W.P.: An updated mixed integer programming library: MIPLIB 3.0. Optima **58**, 12–15 (1998)
12. Bixby, R.E., Rothberg, E.: Progress in computational mixed integer programming: a look back from the other side of the tipping point. Ann. Oper. Res. **149**(02), 37–41 (2007)
13. Bonami, P.: On optimizing over lift-and-project closures. Math. Program. Comput. **4**(2), 151–179 (2012). https://doi.org/10.1007/s12532-012-0037-0
14. Bonami, P., Cornuéjols, G., Dash, S., Fischetti, M., Lodi, A.: Projected Chvátal–Gomory cuts for mixed integer linear programs. Math. Program. **113**(2), 241–257 (2008). https://doi.org/10.1007/s10107-006-0051-y
15. Caprara, A., Letchford, A.N.: On the separation of split cuts and related inequalities. Math. Program. **94**(2), 279–294 (2003). https://doi.org/10.1007/s10107-002-0320-3
16. Center, I.K.: Deterministic time limit. https://www.ibm.com/support/knowledgecenter/SSSA5P_12.6.3/ilog.odms.cplex.help/CPLEX/Parameters/topics/DetTiLim.html. Accessed 18 Jan 2019
17. Cook, W.J., Kannan, R., Schrijver, A.: Chvátal closures for mixed integer programs. Math. Program. **47**, 155–174 (1990)
18. Cornuéjols, G., Nannicini, G.: Practical strategies for generating rank-1 split cuts in mixed-integer linear programming. Math. Program. Comput **3**(4), 281–318 (2011). https://doi.org/10.1007/s12532-011-0028-6
19. Dash, S., Günlük, O., Lodi, A.: MIR closures of polyhedral sets. Math. Program. **121**(1), 33–60 (2010). https://doi.org/10.1007/s10107-008-0225-x
20. Dey, S.S., Molinaro, M., Wang, Q.: Approximating polyhedra with sparse inequalities. Math. Program. **154**(1), 329–352 (2015). https://doi.org/10.1007/s10107-015-0925-y
21. Dey, S.S., Molinaro, M., Wang, Q.: Analysis of sparse cutting planes for sparse MILPs with applications to stochastic MILPs. Math. Oper. Res. **43**(1), 304–332 (2018). https://doi.org/10.1287/moor.2017.0866
22. Fischetti, M., Lodi, A.: Optimizing over the first Chvátal closure. Math. Program. **110**(1), 3–20 (2007). https://doi.org/10.1007/s10107-006-0054-8
23. Fischetti, M., Lodi, A., Tramontani, A.: On the separation of disjunctive cuts. Math. Program. **128**(1), 205–230 (2011). https://doi.org/10.1007/s10107-009-0300-y
24. Fischetti, M., Salvagnin, D.: A relax-and-cut framework for Gomory mixed-integer cuts. Math. Program. Comput. **3**(2), 79–102 (2011). https://doi.org/10.1007/s12532-011-0024-x
25. Fischetti, M., Salvagnin, D.: Approximating the split closure. INFORMS J. Comput. **25**(4), 808–819 (2013). https://doi.org/10.1287/ijoc.1120.0543
26. Gamrath, G., Lübbecke, M.E.: Experiments with a generic dantzig-wolfe decomposition for integer programs. In: Festa, P. (ed.) Experimental Algorithms, pp. 239–252. Springer, Berlin (2010)
27. Koch, T., Achterberg, T., Andersen, E., Bastert, O., Berthold, T., Bixby, R.E., Danna, E., Gamrath, G., Gleixner, A.M., Heinz, S., Lodi, A., Mittelmann, H., Ralphs, T., Salvagnin, D., Steffy, D.E., Wolter, K.: MIPLIB 2010. Math. Program. Comput. **3**(2), 103–163 (2011)
28. Suhl, U.H., Suhl, L.M.: Computing sparse LU factorizations for large-scale linear programming bases. ORSA J. Comput. **2**(4), 325–335 (1990)
29. Walter, M.: Sparsity of lift-and-project cutting planes. In: Helber, S., Breitner, M., Rösch, D., Schön, C., Graf von der Schulenburg, J.M., Sibbertsen, P., Steinbach, M., Weber, S., Wolter, A. (eds.) Operations Research Proceedings 2012, pp. 9–14. Springer, Cham (2014)