CUTTING PLANES FROM WIDE SPILT DISJUNCTIONS

Pierre Bonami
Andrea Lodi
Andrea Tramontani
Sven Wiese

November 2016

DS4DM-2016-003

POLYTECHNIQUE MONTRÉAL
DÉPARTEMENT DE MATHÉMATIQUES ET GÉNIE INDUSTRIEL
Pavillon André-Aisenstadt
Succursale Centre-Ville C.P. 6079
Montréal - Québec
H3C 3A7 - Canada
Téléphone: 514-340-5121 # 3314

Cutting Planes from Wide Split Disjunctions

Pierre Bonami[1], Andrea Lodi[2], Andrea Tramontani[3], and Sven Wiese[4]

[1] CPLEX Optimization, IBM Spain, pierre.bonami@es.ibm.com
[2] MAGI - École Polytechnique de Montreal, andrea.lodi@polymtl.ca
[3] CPLEX Optimization, IBM Italy, andrea.tramontani@it.ibm.com
[4] DEI - University of Bologna, and OPTIT srl, sven.wiese@unibo.it

Abstract. In this paper, we discuss an extension of split cuts that is based on widening the underlying disjunctions. That the formula for deriving intersection cuts based on splits can be adapted to this case has been known for a decade now. For the first time though, we present applications and computational results. We further discuss extensions of the existing theory with respect to cut strengthening procedures, and present some ideas on how to use the wider disjunctions also in branching.

1 Introduction

As many authors have noted before us, cutting planes are nowadays an essential ingredient in virtually all Mixed Integer Linear Programming (MILP) codes. Several classes of cutting planes are derived from disjunctions whose validity can be easily verified by the integrality requirements of the underlying MILP. For example, split cuts [7] make use of the fact that inside the feasible set, the dot product of the MILP's integer variables with an integral vector never maps to a fractional value, i.e., into the open interval between any two consecutive integers. In this paper, we study situations in which more information is available, and we can exclude the whole interval between two not necessarily consecutive integers from the feasible set.

Our study is motivated by simple observations regarding the modeling techniques used in Constraint Programming (CP). As an illustrative example, consider a sudoku game as in Fig. 1. In CP, we are allowed to work with finite domains, and the domain of the variable $y_{5,5}$, that is to model the number in the 5th row of the 5th column, can simply be written as

$$D(y_{5,5}) = \{1, 2, 5, 6, 8\}.$$

In MILP instead, we would write

$$1 \leq y_{5,5} \leq 8, \quad y_{5,5} \in \mathbb{Z},$$

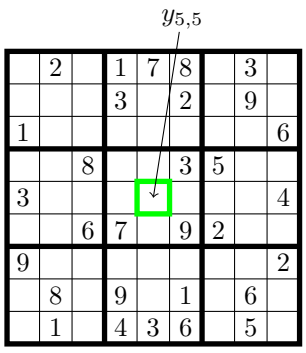


Fig. 1. A 3×3 sudoku

but we have not yet accounted for the constraint $y_{5,5} \notin \{3, 4, 7\}$, which in this direct form is not foreseen in the modeling tools provided by MILP. Of course, all integer programmers would object and (correctly) assert that it is an easy exercise to introduce auxiliary variables that impose this condition. This is what has been done in integer programming for roughly 50 years now and is thus well-proven practice. Yet, it highlights exactly the point we want to make with our admittedly informal example. In order to express the condition that a variable be different from a value in the "interior" of its domain, we have to use auxiliary variables. In CP, such constraints are routinely expressed and, much more important, also exploited algorithmically, e.g., through filtering and propagation. Our aim here is to analyze whether and how we can exploit such explicit representations in MILP.

In the above case, one could also say $y_{5,5}$ has holes in its domains. For example, it is allowed to take the values 2 and 5, but nothing in between. More formally, this can be expressed by the disjunction

$$y_{5,5} \leq 2 \vee y_{5,5} \geq 5. \tag{1}$$

As stated earlier, in the theory of classical split cuts, the right-hand sides of such two disjunctive terms are always consecutive integers. One special case of split cuts are intersection cuts [4] from split sets, for which a closed form formula exists, and that this formula can be easily extended to disjunctions with non-consecutive right-hand sides has already been proven in [2]. Nevertheless, in more than ten years since, nobody has ever applied it in practice in order to conduct computational results. Our contribution is to revive the aforementioned formula of [2] computationally, and to highlight extensions into different directions. While the authors in [2] use the term *general split disjunctions*, we will call constructs like (1) *wide split disjunctions*, and the resulting cutting planes *wide split cuts*.

The rest of the paper is organized as follows. We provide examples in which the validity of wide split disjunctions can be proven, and show that exploiting this "hole information" algorithmically can be advantageous in sections 2 and 3.1 - 3.2, respectively. In sections 3.3 and 3.4, we provide additional theory and computational evidence on the strength of wide split cuts. Finally, in Sect. 4, we examine the combination of wide split cuts with branching on wide split disjunctions inside a branch-and-cut tree.

2 Validity of Wide Split Disjunctions

For the moment, we assume the existence of an underlying MILP

$$\min \quad c^T x \tag{2}$$

$$\text{s.t.} \quad Ax = b \tag{3}$$

$$x \in \mathbb{R}^{n-p}_+ \times \mathbb{Z}^p_+. \tag{4}$$

Wide split disjunctions are of the form

$$\pi^T x \leq \pi_l \vee \pi^T x \geq \pi_u, \tag{5}$$

where the triple (π^T, π_l, π_u) belongs to the set

$$\Pi := \{(\pi^T, \pi_l, \pi_u) \in \mathbb{Z}^{n+2} \mid \pi_i = 0 \ \ \forall\ i = 1, \ldots, n-p\}.$$

A natural question that arises is the one about the validity of a wide split split disjunction for a MILP. While it is easy to see that any solution to (2)–(4) satisfies any split disjunction with consecutive right-hand sides, in the following called *ordinary* split, this is not necessarily true for wide splits. Nevertheless, we now provide some examples of MILPs in which the validity of wide split disjunctions, that are no ordinary splits, can be certified. A special role in the context of split disjunctions often play so-called simple splits, that is $\pi = \mathbf{e}_i$ for some i, the i-th unit vector. This directly translates to a variable with a non-contiguous domain, i.e., with holes in its domain. In all of the examples we provide below, we end up with simple wide splits. Yet, we believe that for detecting valid wide split disjunctions in general MILPs, it might not be enough to focus on variables with holes only. Therefore, the rest of the paper is outlined for not necessarily simple wide splits.

2.1 Certifying Split Validity by Primal Information

We first give examples in which wide split disjunctions in the form of holes in the domains of variables are implied by the constraints present in an underlying MILP. This happens, e.g., when some modeling tricks with auxiliary binary variables as mentioned in the introduction are applied. The first of these tricks involves big-M constraints.

Consider an integer variable y and the set of constraints,

$$\begin{aligned} l^j - y &\leq (1 - x_j) \cdot (l^j - l^1) \qquad \forall\ j \in [L] \\ y - u^j &\leq (1 - x_j) \cdot (u^L - u^j) \qquad \forall\ j \in [L] \\ \sum_{j=1}^{L} x_j &= 1, \end{aligned}$$

and assume $u^{j-1} < l^j$. In any feasible solution, y will lie in exactly one of the intervals $[l^j, u^j]$, $j \in [L]$. It is easy to check that the above set of constraints implies validity of the simple wide split disjunctions

$$y \leq u^{j-1} \vee y \geq l^j, \quad j = 2, \ldots, L.$$

Clearly, if $u^{j-1} + 1 < l^j$ for some j, there is at least one non-ordinary wide split disjunction. The above constraint structure can be found, e.g., in straightforward MILP formulations for the Traveling Salesman Problem with Multiple Time Windows (TSPMTW) [6].

A GUB-link constraint is characterized by a pair of equations

$$y = \sum_{j=1}^{L} \lambda_j x_j, \quad \sum_{j=1}^{L} x_j = 1. \tag{6}$$

Here, the wide split disjunctions

$$y \leq \lambda_{j-1} \vee y \geq \lambda_j, \quad \forall\ j = 2, \ldots, L$$

are valid. Clearly, if the λ_i are non-consecutive integers, there are non-ordinary split disjunctions. Such structures can be found in time-indexed MILP formulations for scheduling problems, see, e.g., [8].

2.2 Certifying Split Validity by Dual Information

In the previous section, we deduced the validity of non-contiguous domains in a MILP by primal information: every (primal) feasible solution was assured to lie in these domains. Now instead, we allow that non-contiguous domains, that not every primal feasible solution satisfies, are imposed. In the following MILP, however, we will see that for every solution that is excluded in this way, there is at least one other feasible solution with identical objective value. The resulting problem and the original MILP are thus equivalent in the sense that they have the same optimal objective value. Hence, we can characterize these domains to be derived from dual information.

The MILP we are going to analyze is a formulation for the so-called Lazy Bureaucrat Problem (LBP), and strictly speaking, it is a pure Integer Linear Program (ILP). The LBP can be seen as a lazy counterpart of the classical Knapsack Problem [12]. Similar to therein, we are given a set of items $i \in [n]$ with non-negative profits p_i and non-negative weights w_i, both of which we assume to be integral, and a knapsack with capacity C. The objective is to pack a subset of items into the knapsack such that:

- the profit of all packed items is minimized,
- their weight does not exceed the capacity,
- but adding any non-packed item would exceed it.

The task is thus to find a so-called maximal packing with minimum profit. In [9] are proposed several ILP formulations for the LBP. The most competing one, with the critical item i_c and the critical weight w_c defined as therein, is

$$\min \quad \sum_{i=1}^{n} p_i x_i \tag{7}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} w_i x_i \leq C \tag{8}$$

$$\sum_{i=1}^{n} w_i x_i + z \geq C + 1 \tag{9}$$

$$z \leq w_c - (w_c - w_i)(1 - x_i) \qquad \forall\ i \in [i_c] \tag{10}$$

$$(x, z) \in \{0,1\}^n \times \mathbb{Z}_+ \tag{11}$$

The variable z models the weight of the smallest item left out of the packing, and it is easy to construct examples with feasible solutions with $z \notin \{w_i \mid i \leq i_c\}$. Yet, we have the following simple result.

Lemma 1. *For every feasible solution of* (7)–(11)*, there is another feasible solution with the same cost and*

$$z \in \{w_i \mid i \leq i_c\}. \tag{12}$$

Proof. Let $(\bar{x}, \bar{z})$ be feasible to (7)–(11) and denote the corresponding packing by $S := \{i \in [n] \mid \bar{x}_i = 1\}$. Further, let $\tilde{i} := \min\{i \in [n] \mid i \notin S\}$. Clearly, $\tilde{i} \leq i_c$, and from (10) we get $\bar{z} \leq w_{\tilde{i}}$. Because increasing the value of $\bar{z}$ does not violate (9), $(\bar{x}, w_{\tilde{i}})$ is feasible, satisfies (12), and since the objective function (7) depends on x only, has the same cost as $(\bar{x}, \bar{z})$. □

In the end, if the weights are non-consecutive integers, we can consider a version of the problem with valid wide split disjunctions.

3 Cut Derivation and Computation

In this section, we first show how to algebraically derive wide split cuts and give some examples. We denote the points of the feasible region of (2)–(4), that in addition satisfy the wide split disjunction (5), by $\mathcal{F}$. Also, we denote the feasible region of the LP relaxation of (2)–(4) by P, i.e., $P = \{x \in \mathbb{R}^n \mid Ax = b, x \geq 0\}$. Furthermore, we assume that all $x \in \mathcal{F}$ satisfy the wide split disjunction (5). Thus, we set

$$P_l^{(\pi,\pi_l)} := \{x \in P \mid \pi^T x \leq \pi_l\}, \quad P_u^{(\pi,\pi_u)} := \{x \in P \mid \pi^T x \geq \pi_u\},$$

and $P^{(\pi,\pi_l,\pi_u)} := \text{conv}(P_l^{(\pi,\pi_l)} \cup P_u^{(\pi,\pi_u)})$. By definition, a wide split cut is any linear inequality that is valid for $P^{(\pi,\pi_l,\pi_u)}$. Since $\mathcal{F} \subseteq P^{(\pi,\pi_l,\pi_u)}$, any wide split cut is also valid for $\mathcal{F}$. We pick up on what has been shown in [2], i.e., how to derive wide split cuts as intersection cuts, but we note that it is also possible to derive wide split cuts in a lift-and-project fashion, see [14, Sect. 4.2.2].

3.1 Intersection Cuts from Wide Split Disjunctions

We assume that the LP relaxation of (2)–(4) has been solved to the point $\hat{x}$ by means of the simplex method and that we are given the optimal simplex tableau $T = T(B)$ of P,

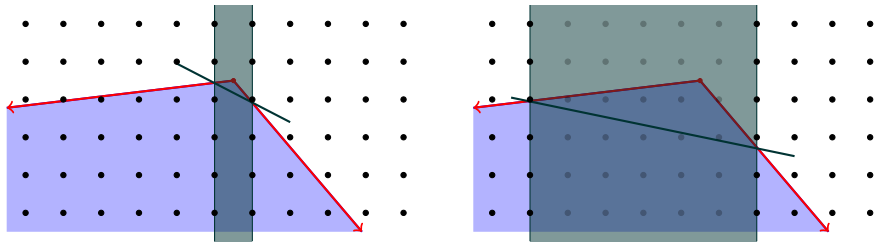
$$x_i = \hat{x}_i \; + \sum_{j \in N} r_j^i x_j, \qquad i \in B \tag{13}$$

$$x_i \geq 0, \qquad i \in [n], \tag{14}$$

where the index set of variables is partitioned into basic and non-basic variables, $[n] = B \cup N$, respectively. We state the aforementioned result of [2].

Proposition 1. *Assume that $\hat{x}$ violates the wide split disjunction (π^T, π_l, π_u), i.e., $\pi_l < \pi^T\hat{x} < \pi_u$, and $\forall\ j \in N$, define $f_j := \pi_j + \sum_{i \in B} \pi_i r_j^i$. A valid inequality for $P^{(\pi,\pi_l,\pi_u)}$ is then given by*

$$\sum_{j \in N} \max\left\{\frac{-f_j}{\pi^T\hat{x} - \pi_l}, \frac{f_j}{\pi_u - \pi^T\hat{x}}\right\} x_j \geq 1. \tag{15}$$



(a) Ordinary split cut (b) Wide split cut

Fig. 2. Intersection cuts in the plane

The valid inequality of Proposition 1 is clearly a wide split cut, and it is easy to check that it is violated by $\hat{x}$. If in (13)–(14) one relaxes the non-negativity on the basic variables, then the resulting set can be shown to be a translated polyhedral cone, often denoted by $P(B)$. Geometrically, (15) can be obtained by computing the intersection points of the extreme rays of $P(B)$ with the boundary of the split set $S := \{x \in \mathbb{R}^n \mid \pi_l \leq \pi^T x \leq \pi_u\}$. Figures 2 (a) and (b) depict an example of $P(B)$, and the split sets with corresponding cuts of an ordinary split disjunction and a wide split disjunction, respectively, in a two-dimensional basic space. We now give some algebraic examples of Proposition 1.

Example 1. Consider the MILP with GUB-Link constraints

$$\begin{aligned}
\min \quad & -x_2 && (16)\\
\text{s.t.} \quad & x_1 + x_2 \leq 6 && (17)\\
& -x_1 + x_2 \leq 0 && (18)\\
& x_1 = x_3 + 2x_4 + 4x_5 + 8x_6 && (19)\\
& 1 = x_3 + x_4 + x_5 + x_6 && (20)\\
& (x_1, x_2, x_3, x_4, x_5, x_6) \in \mathbb{Z}^6_+. && (21)
\end{aligned}$$

Note that variables x_3, x_4, x_5, x_6 are implicitly binary constrained due to (20). Therefore, it is easy to see that $x_1 \leq 2 \vee x_1 \geq 4$ is a valid (simple) wide split disjunction. Introducing slack variables s_1 and s_2 for (17) and (18), respectively, one can check that the basic row of x_1 in the optimal simplex tableau is $x_1 = 3 - \frac{1}{2}s_1 + \frac{1}{2}s_2$, showing that the wide split disjunction is violated. This equation can be used together with (15) to calculate the wide split cut $s_1 + s_2 \geq 2$, that, after substituting slacks, is equivalent to $x_2 \leq 2$. Thus, the wide split cut is a facet of the integer hull of the MILP in the (x_1, x_2)-space. Clearly, in the optimal solution of the LP relaxation, there will be two fractional binary variables, one between x_3 and x_4, and one between x_5 and x_6. Assume that $0 < \hat{x}_4 < 1$ and $0 < \hat{x}_5 < 1$. The two intersection cuts corresponding to these two violated split disjunctions can be checked to be $x_2 - 3x_3 - 4x_6 \leq 2$ and $x_2 - x_3 - 6x_6 \leq 2$. Both cut off the optimal LP-solution, but are dominated by the wide split cut. □

The validity of the wide split disjunctions in the above examples can be thought of as being obtained by dual and primal information, respectively. Note that in both cases, the wide split cut dominates the cut derived from the ordinary split disjunction. It is easy to prove this as a general result, see [14, Prop. 4.7].

Constraints (20)–(21) in Example 1 above define the GUB-Link (6) as introduced in Section 2.1, thus any wide split disjunction on variable x_1 could be interpreted as a disjunction on the binary variables x_3, x_4, x_5, x_6. More generally, given the GUB-link (6) and $L = \{1, \ldots, \ell\}$, let W be the (possibly) wide split disjunction $y \leq \lambda_\ell \vee y \geq \lambda_{\ell+1}$, and S be the ordinary split disjunction $\sum_{j \in L} x_j \leq 0 \vee \sum_{j \in L} x_j \geq 1$. The relationship between the cuts obtained by those disjunctions is settled by the two following lemmata.

Lemma 2. *Let $C(D)$ denote the split closure, i.e., the set of all split inequalities without integer lifting (lift-and-project cuts) that can be obtained from a split disjunction D (see, e.g., [2]). Then, $C(S)$ dominates $C(W)$, i.e., $C(S) \subseteq C(W)$.*

Proof. Let P denote the polyhedron of the LP relaxation of system (2)–(4). Define the four polyhedra $W_0 := P \cap y \leq \lambda_\ell$, $W_1 := P \cap y \geq \lambda_{\ell+1}$, $S_0 := P \cap \sum_{j \in L} x_j \leq 0$, and $S_1 := P \cap \sum_{j \in L} x_j \geq 1$. The closure $C(S)$ is simply $C(S) = conv(S_0 \cup S_1)$, while $C(W) = conv(W_0 \cup W_1)$. On the polyhedron P, it holds that *(i)* $\sum_{j \in L} x_j \geq 1$ implies $y \leq \lambda_\ell$, therefore $S_1 \subseteq W_0$, and *(ii)* $\sum_{j \in L} x_j \leq 0$ implies $y \geq \lambda_{\ell+1}$, therefore $S_0 \subseteq W_1$. Thus, $C(S) \subseteq C(W)$. □

Lemma 3. *The dominance in Lemma 2 only applies to closures. In particular, given an optimal tableau T, the wide split cut associated with T and W is not dominated by the split cut associated with T and S, and it could even happen that the reverse is true (i.e., the cut from W dominates the cut from S).*

Proof. In Example 1, assuming again that x_4 and x_5 are basic in the optimal simplex solution, the ordinary split $x_3 + x_4 \leq 0 \vee x_3 + x_4 \geq 1$ is violated. Such a split can be constructed by collecting all the binary variables with coefficients less or equal than π_l in the GUB-Link. With Proposition 1 one can show that the intersection cut from the above binarization split is $x_2 - x_3 - 4x_6 \leq 2$. This cut is still weaker than the wide split cut: point $(3, 3, \frac{1}{5}, \frac{3}{5}, 0, \frac{1}{5})$ satisfies it. □

3.2 Computation

We now present the results of computational experiments with wide split cuts applied to the LBP introduced in Sect. 2.2. Recall that the valid wide split disjunctions in this case are simple ones, i.e., we have holes in the domains of integer variables. We are particularly interested in testing the computational advantage of wide split cuts over corresponding cuts from ordinary splits. Therefore, we include intersection cuts from simple, ordinary splits in our experiments. However, we do not only include such cuts corresponding to fractional values that fall into a hole, but also to fractional values that do not, and in particular corresponding to all integer and binary variables in the model. This will give us more flexibility in testing different cut separation strategies.

In general, we perform several rounds of separation. In a single round, we solve the LP relaxation of the underlying MILP to optimality, separate cuts according to the chosen strategy, add all separated cuts to the model and solve again. The strategies we test are

- w/o (without wide splits): For each basic fractional binary or integer variable, we compute an intersection cut from an ordinary split.
- w (with wide splits): For each basic integer variable, we compute a wide split cut if its value lies in a hole, or otherwise an intersection cut from an ordinary split, if its value is fractional. In addition, for each basic fractional binary variable, we compute an intersection cut from an ordinary split.
- o (only wide splits): For each basic integer variable, we compute a wide split cut if its value lies in a hole.

The procedure has been coded in C/C++ with CPLEX 12.6.1 as LP solver. Throughout the following, we usually report the percentage of the initial dual gap that is closed by the separated cuts. Also, the total number of cuts that have been generated is sometimes shown. In order to create test instances, we used *class 4* of the knapsack instance generator presented in [11] and available at [13]. As in [9], this led to a total of 54 instances with different combinations of the parameters. Table 1 shows an excerpt (and mean values) of those 25 out of the 54 instances in which we can close significantly more gap with strategy w than with w/o in 10 rounds of separation. That means that the separation of wide split cuts on top of ordinary split cuts is highly advantageous. Interestingly, on these instances the separation of only wide intersection cuts (strategy o) already leads to close significantly more gap than with strategy w/o, and almost reaches the one of strategy w. A (positive) side effect of of wide split cuts is the reduction of the number of cuts that are separated in total. Apart from closing more gap, strategy w also decreases this number significantly with respect to strategy w/o. This is a desirable effect since the number of constraints in an LP, to which separated cutting planes have to be added in order to benefit from closing additional gap, influences the computational effort when solving it. Remarkably, the significant gap closed by strategy o requires a small number of separated cuts. We note that the side effect of reducing the number of cuts also persists in instances that do not benefit significantly from the separation of wide split cuts in terms of the gap closed. In particular, in 14 out of the remaining 29 instances, the same gap can be closed with a significantly lower number of

	% gap closed			#cuts		
instance	w/o	w	o	w/o	w	o
10000-4-100-75	9.21	12.53	11.73	139	62	5
10000-4-10-25	32.97	34.28	28.40	46	27	2
10000-4-10-50	79.12	100.00	100.00	1	1	1
⋮		⋮			⋮	
1000-4-40-75	8.92	42.90	41.38	150	53	5
1000-4-500-75	2.97	5.10	4.89	134	73	6
1000-4-50-50	13.01	17.12	15.26	113	60	4
mean	19.81	31.95	28.13	89.92	49.16	3.28

Table 1. Separation of wide split cuts on LBP instances

separated cuts. In the remaining 15 instances, strategy o is often highly competitive and requires a small number of cuts in total to close almost the same gap as strategies w/o and w.

3.3 Cut Strengthening

Ordinary split cuts can be seen to exploit the integrality requirements on the basic variables. In fact, since all non-basic variables are zero in the optimal simplex solution $\hat{x}$, the split violation is determined by the basic variables only. Intuitively, deriving a cutting plane by taking into account the integrality requirements of the non-basic variables as well should result in stronger cutting planes. This idea leads to the integer strengthening principle of intersection cuts from split disjunctions, outlined for example in [1], or recovered from the concept of monoidal strengthening introduced in [5]. Informally, this strengthening works by modifying the underlying split disjunctions in a certain way to another valid disjunction. An extension of this concept to wide split disjunctions is presented in [14, Sect. 4.2.3]. It turns out, however, that this strengthening is rather weak for wide split cuts, meaning that the improvement of the strengthened cutting plane is marginal. Intuitively, there is much less freedom in modifying a wide split disjunction to another valid disjunction than there is for ordinary split disjunctions. We experienced this weakness also computationally: substituting every wide split cut in the experiments in Tab. 1 by its strengthened version leads to a negligible improvement in almost all cases. Again in [14, Sect. 4.2.3] is developed some interesting and promising theory on the strengthening of wide split cuts when holes in domains are distributed regularly, which leads to more flexibility in the aforementioned modification of the underlying disjunction. However, the detection of regularly distributed holes is outside the scope of this paper.

The aforementioned strengthening principle of intersection cuts from ordinary split disjunctions can be shown to give precisely one of Gomory's Mixed Integer (GMI) cuts [10], see [1]. While the strengthening of wide split cuts is relatively weak, GMI cuts are generally considered to be able to give quite strong improvements. This can be seen as some kind of dilemma. Whenever we have a violated wide split disjunction, it is not clear whether the best strategy is to separate a wide split cut that can then be strengthened only weakly, or to weaken the disjunction to an ordinary split and separate an ordinary intersection cut, that can then be strengthened strongly to a GMI cut. To analyze this dilemma computationally, we introduce the new cut generation strategy w/o-g (with GMI cuts instead of wide splits): equal to w, except

	% gap closed	
instance	w	w/o-g
10000-4-100-75	12.53	9.21
10000-4-10-25	34.28	32.96
10000-4-10-75	42.21	34.31
⋮	⋮	
1000-4-40-75	42.90	8.55
1000-4-500-75	5.10	3.04
1000-4-50-50	17.12	13.01
mean	29.11	17.07

Table 2. GMI vs. wide split cuts on LBP instances

that for every basic variable that violates a wide split disjunction (in which case with strategy w we separate a wide split cut), we separate a GMI cut. Table 2 compares the gap closed by strategies w and w/o-g on the instances of Tab. 1. Strategy w beats w/o-g, meaning that GMI cuts are not able to close the gap that we can be closed by exploiting the wide split disjunction.

3.4 MIPLIB2010 Instances

In order to use wide split disjunctions on general MILPs, one needs to devise procedures to detect holes in the domains of the integer variables, possibly without exploiting the special structure of the problem like, instead, done for LBP. Although this is beyond the scope of the present paper, we performed a preliminary experiment on five MIPLIB2010 instances in which, by inspecting the cliques explicitly present in the problem, GUB-Link constraints with an integer variable containing holes in its domain can be defined. Table 3 reports the gap closed by adding five rounds of wide split cuts at the end of CPLEX 12.6.1's root node executed in either default (cpx_d) or aggressive cuts (cpx_a) mode. This admittedly limited set of experiments shows that there seems to be a value in adding wide split cuts on top of CPLEX 12.6.1's existing cut separation procedures, i.e., the information associated with the holes of the integer variables does not seem to be recovered by standard cuts.

	% gap closed			
instance	cpx_d	cpx_d+w	cpx_a	cpx_a+w
sp97ar	16.71	18.94	16.71	19.64
leo2	23.57	25.00	23.75	25.56
ns1830653	37.80	37.80	39.65	40.34
blp-ar98	77.86	78.40	80.98	81.27
n3div36	48.74	48.77	49.05	49.05
geo_mean	35.51	36.90	36.24	38.14

Table 3. Wide split cuts on top of CPLEX cuts

4 Branching on Wide Split Disjunctions in a Search Tree

In this final section we present some additional considerations and experimental algorithms on how wide split disjunctions can be useful not only for deriving cutting planes and thus strengthening the LP relaxation of a MILP, but also for solving the MILP to optimality. We assume to have a MILP as in (2)–(4) with the additional condition that some integer variables satisfy simple wide split disjunctions, which is imposed by means of the mixed integer constraints

$$x + Ex_a = g, \quad x_a \in \{0,1\}^k. \tag{22}$$

These constraints can be thought of as big-M or GUB-Link constraints including the auxiliary binary variables x_a. In order to solve the resulting MILP given by (2)–(4) and (22), we explore the following three strategies:

– B&C-F: Apply a MILP solver to the full model (2)–(4), (22).

	B&C-F		B&C-R		B&C-R+cuts	
instance	time	(nodes)	time	(nodes)	time	(nodes)
1-1	540.78	(66302)	∞	(∞)	287.02	(39925)
1-2	930.40	(108000)	∞	(∞)	134.33	(19451)
1-3	384.16	(39605)	159.64	(25244)	158.93	(25244)
⋮	⋮		⋮		⋮	
5-3	86.06	(15599)	∞	(∞)	33.30	(7803)
5-4	4339.04	(137497)	∞	(∞)	17.70	(3788)
5-5	37.43	(9077)	∞	(∞)	6.61	(1326)
mean	621.67	(51667.00)	-	-	107.55	(16357.04)

Table 4. Comparison of B&C-F, B&C-R and B&C-R+cuts on neos-555424 instances

- B&C-R: Apply a MILP solver to the relaxed model (2)–(4), but whenever an incumbent is found, check for satisfaction the wide split disjunctions. If some disjunction $(\mathbf{e}_i^T, \pi_l, \pi_u)$ for x_i is violated, branch with π_l and π_u as new upper and lower bound, respectively: $x_i \leq \pi_l$ OR $x_i \geq \pi_u$.
- B&C-R+cuts: the same as B&C-R with the separation of r rounds of wide split cuts at the root node.

Of course, all three strategies are ways of solving model (2)–(4), (22) exactly. The idea behind B&C-R is that the model size is reduced with respect to B&C-F, depending on the number of holes, more or less significantly. An additional advantage of B&C-R+cuts is that information used for strengthening the LP relaxation contained in (22) is potentially preserved by wide split cuts.

All three procedures have been implemented in C/C++ with CPLEX 12.6.1 as a MILP-solver. B&C-R and B&C-R+cuts can be implemented by using CPLEX's incumbent- and branch-callbacks. We further used randomly generated test instances. In particular, we took all MIP or IP instances from MIPLIB2010 (with problem status *easy* as per January 2016), and randomly generated holes, distributed in the domains of the involved integer variables. Here, we discuss an excerpt of the outcome for 25 such random instances based on the MIPLIB2010 one neos-555424. Table 4 shows solution times in seconds and number of branch-and-bound nodes explored for all three strategies in that case. A time limit of two hours was imposed, the number of rounds r was set to 10 and (22) were encoded by big-M constraints. We immediately note that B&C-R hits the time limit in almost all cases. Essentially, the algorithm keeps finding incumbent solutions that are then rejected. Remarkably, the separation of wide split cuts in B&C-R+cuts consistently avoids this phenomenon. In the second instance for example, a total of two separated cuts is enough to avoid hitting the time limit. More importantly, B&C-R+cuts clearly beats B&C-F, showing how the exploitation of wide split disjunctions in branching and through cuts can lead to significant computational advantages.

The picture of Tab. 4 does not persist throughout all variations of all the original MIPLIB2010 instances we tested. There are cases in which already B&C-

R wins, and cases in which B&C-F is clearly the winning strategy. In general, there seem to be problems that suffer the step from B&C-F to B&C-R, like in the case of neos-555424, and others that do not. A problem from the literature that belongs to the latter class is the TSPMTW, where B&C-R led to an average reduction in computing times of around 20% with respect to B&C-F on our testbed. However, B&C-R+cuts does not lead to an additional improvement. This is probably due to the fact that wide split cuts act on the scheduling component of the TSPMTW, leaving the LP relaxation at the root node still very weak due to the TSP component. A future direction of the considerations in this section is therefore the experimentation of the separation of wide split cuts at nodes inside the search tree. Also, there is clearly a more efficient way of replicating strategy B&C-R, that is, by incorporating the branching on wide split disjunctions directly in the branching process of the MILP-solver instead of applying it only when incumbent solutions are found. However, both aspects are outside the scope of this paper.

References

1. K. Andersen, G. Cornuéjols, Y. Li. Reduce-and-split cuts: Improving the performance of mixed-integer Gomory cuts. *Management Science*, 51:1720–1732, 2005.
2. K. Andersen, G. Cornuéjols, Y. Li. Split closure and intersection cuts. *Mathematical programming*, 102:457–493, 2005.
3. E. Balas. Disjunctive programming: Properties of the convex hull of feasible points. *Discrete Applied Mathematics*, 89:3–44, 1998.
4. E. Balas. Intersection cuts–a new type of cutting planes for integer programming. *Operations Research*, 19:19–39, 1971.
5. E. Balas, R.G. Jeroslow. Strengthening cuts for mixed integer programs. *European Journal of Operational Research*, 4:224–234, 1980.
6. S. Belhaiza, P. Hansen, G. Laporte. A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. *Computers & Operations Research*, 52:269–281, 2014.
7. W. Cook, R. Kannan, A. Schrijver. Chvátal closures for mixed integer programming problems. *Mathematical Programming*, 47:155–174, 1990.
8. E. Coughlan, M. Lübbecke, J. Schulz. A branch-and-price algorithm for multi-mode resource leveling. In Paola Festa, editor, *Experimental Algorithms*, volume 6049 of *Lecture Notes in Computer Science*, pages 226–238. Springer, 2010.
9. F. Furini, I. Ljubić, M. Sinnl. ILP and CP Formulations for the Lazy Bureaucrat Problem. In *Integration of AI and OR Techniques in Constraint Programming*, pages 255–270. Springer, 2015.
10. R.E. Gomory. An algorithm for integer solutions to linear programs. In Robert L. Graves, Philip Wolfe, editors, *Recent Advances in Mathematical Programming*, pages 269–302. McGraw-Hill, New York, 1963.
11. S. Martello, D. Pisinger, P. Toth. Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science*, 45:414–424, 1999.
12. S. Martello, P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. Wiley, Chichester, UK, 1990.
13. D. Pisinger. `http://www.diku.dk/~pisinger/codes.html`. Accessed: 2016-02-19.
14. S. Wiese. *On the interplay of Mixed Integer Linear, Mixed Integer Nonlinear and Constraint Programming*. PhD thesis, University of Bologna, 2016.