$\mathcal{V}$-Polyhedral Disjunctive Cuts

Egon Balas Aleksandr M. Kazachkov

February 18, 2024

Abstract

We introduce *$\mathcal{V}$-polyhedral disjunctive cuts* (VPCs) for generating valid inequalities from general disjunctions. Cuts are critical to integer programming solvers, but the benefit from many families is only realized when the cuts are applied recursively, causing numerical instability and "tailing off" of cut strength after several rounds. To mitigate these difficulties, the VPC framework offers a practical method for generating strong cuts without resorting to recursion. The framework starts with a disjunction whose terms partition the feasible region into smaller subproblems, then obtains a collection of points and rays from the disjunctive terms, from which we build a linear program whose feasible solutions correspond to valid disjunctive cuts. Though a naïve implementation would result in an exponentially-sized optimization problem, we show how to efficiently construct this linear program, such that it is much smaller than the one from the alternative higher-dimensional cut-generating linear program. This enables us to test strong multiterm disjunctions that arise from the leaf nodes of a partial branch-and-bound tree. In addition to proving useful theoretical properties of the cuts, we evaluate their performance computationally through an implementation in the open-source COIN-OR framework. In the results, VPCs from a strong disjunction significantly improve the gap closed compared to existing cuts in solvers, and they also decrease some instances' solving time when used with branch and bound.

1 Introduction

This paper presents a new framework for generating disjunctive *cutting planes*, or *cuts*, in which a large number of strong cuts can be generated efficiently and nonrecursively. We are motivated by a crucial drawback of many existing cut techniques, in their reliance on recursion to reach strong cuts, i.e., by computing cuts from previously-derived ones. This can result in numerical issues (e.g., due to compounding inaccuracies) and a "tailing off" of the strength of the cuts in later rounds [18, 30, 17, 59, 50]. Our primary computational innovation to circumvent recursion is an efficient use of the *$\mathcal{V}$-polyhedral* perspective, which has been avoided in the past because, used naïvely, it yields intractable representations of instances. The framework we introduce overcomes this hurdle and facilitates a cut generation scheme formulated in the original dimension of the problem, as opposed to a commonly-used higher-dimensional representation. This enables us to test large multiterm disjunctions arising from partial branch-and-bound trees, resulting in cuts that are strong (compared to cuts currently deployed in solvers) and have the potential to reduce solving time.

One way to derive a general-purpose cut is from a *disjunction*, which we define precisely later in this section. Balas [7] introduced the prevailing paradigm for disjunctive cuts, via a *cut-generating linear program* (CGLP) with additional variables beyond those defining the cut. The CGLP is too expensive to work with, despite being polynomially-sized (in the size of the original instance and disjunction). As a result, solvers only use this idea for the simplest—split—disjunctions [15, 16]. Successfully applying such *lift-and-project cuts* (L&PCs) for splits hinges on the ability to compute the cuts in the space of the original linear program, which is possible due to a correspondence of basic solutions of the extended formulation to those in the lower-dimensional space [14, 8, 21]. However, although such a correspondence exists for split disjunctions, it does not necessarily exist for more general ones [5, 11]. Hence, efficiently producing cuts from stronger disjunctions calls for a different perspective.

Our starting point for developing such a computationally-efficient procedure is a *polar* representation of polyhedra: rather than the inequality description—which is how an instance is usually provided and what underpins the CGLP—every polyhedron can also be equivalently represented by its $\mathcal{V}$-polyhedral description, i.e., through a collection of points and rays. Cuts can be generated by inputting these points and rays into a "point-ray" linear program with the same number of variables as that of the original problem. The drawback of the point-ray approach is that the number of rows of the linear program will typically be exponentially large in the size of an inequality formulation, and these rows are necessary in the sense that dropping them may result in invalid cuts. For this reason, prior work that has adopted the $\mathcal{V}$-polyhedral perspective resorts to row generation to guarantee validity [53, 49].

We show how to avoid the expensive row generation step via a properly chosen compact collection of points and rays that we prove suffices to produce valid cuts, albeit a subset of the entire pool of possible disjunctive cuts. We prove conditions under which the cuts obtained from our $\mathcal{V}$-polyhedral relaxation define facets for the disjunctive hull; for example, for a split disjunction, every cut we generate is facet-defining when there is no primal degeneracy.

Most existing cut-generation procedures are applied to shallow disjunctions, such as split disjunctions. Typically, strong cuts are attained by recursive applications, deriving cuts from other cuts, rather than only from the inequalities of the original system, and several shallow disjunctions are considered in parallel. In contrast, the efficiency of our technique enables testing larger disjunctions (up to 64 terms in our experiments). Our disjunctions are derived from generating a partial branch-and-bound tree for each instance. Theoretically, disjunctive cuts from such trees can solve the original integer programming problem without recursion [40, 24].

Another difference in our approach from usual cut generation paradigms is our choice of objective functions for the point-ray linear program. Instead of producing cuts that are maximally violated by an optimal solution to the linear programming relaxation, we aim for a diverse set of cuts with wider coverage of the disjunction: we explicitly aim for inequalities that are supporting on different points and rays. In the process, we constructively prove that cuts from our relaxed point-ray collection can provide the same objective value as from using the complete (exponential-sized) point-ray collection. Further, we develop and refine an objective selection scheme, which is a dynamic cut selection procedure, in that the next target direction depends on what cuts have been generated before. Our theory also aims to avoid infeasibility, unboundedness, and duplicate cuts while solving the point-ray

linear program, as these situations impose computational burden without generating new inequalities.

Supporting the theoretical results, our computational experiments indicate that applying our cuts, which we call *$\mathcal{V}$-polyhedral (disjunctive) cuts* (VPCs), significantly improves the percent integrality gap closed over the baseline of *Gomory mixed-integer cuts* (GMICs) [38] and the default cuts for the leading commercial solver Gurobi [39]. To complement our empirical evaluation of the strength of VPCs through the integrality gap they close, we also test the effect of VPCs on the solving time of Gurobi. The branch-and-bound results do not show an improvement in solver performance when using VPCs on average, but we observe a benefit for a number of instances. Our investigation leaves open the question of how to identify instances, hyperparameter settings, or solver modifications to take advantage of stronger disjunctive cuts in practice.

Paper organization. We describe properties of our cut generation scheme in Section 2, such as conditions under which we obtain facet-defining inequalities for the disjunctive hull and theoretical results that benefit our implementation. Section 3 specifies the disjunctions used in our experiments. Section 4 provides theory for the objective directions we consider. The computational results in Section 5 indicate that a vital and challenging question that remains outstanding is efficiently selecting multiterm disjunctions that yield good cuts.

Notation. Let P denote a polyhedron described by a set of m inequalities:

$$P := \{x \in \mathbb{R}^n : Ax \geq b\}. \tag{P}$$

Let P_I be the integer-feasible region:

$$P_I := \{x \in P : x_j \in \mathbb{Z} \text{ for all } j \in \mathcal{I}\}, \tag{P_I}$$

where $\mathcal{I} \subseteq [n] := \{1, \ldots, n\}$ is the index set of the integer-restricted variables. We assume that P is full dimensional and pointed, all data is rational, and all variable bounds are subsumed by $Ax \geq b$.[1] For a given $c \in \mathbb{R}^n$, our goal is to solve the *mixed-integer program*

$$\min_{x \in P_I} c^\intercal x. \tag{IP}$$

We start by solving the *linear programming relaxation* of (IP), obtained by removing the integrality restrictions on the variables:

$$\min_{x \in P} c^\intercal x. \tag{LP}$$

This yields an optimal solution, $\bar{x}$, which we assume does not belong to P_I. To proceed, integer programming solvers next tighten the relaxation P by the addition of inequalities that are valid for P_I but remove some of P. One way these cuts can be generated is via

[1]The full-dimensionality assumption is made for ease of exposition and the subsequent results generally apply, with minor modifications, when this assumption is relaxed.

a valid *disjunction*, which creates a partition such that P_I is contained in the union of the disjunctive terms. Concretely, a disjunction takes the form

$$\bigvee_{t\in\mathcal{T}} \{x \in \mathbb{R}^n : D^t x \geq D_0^t\}. \tag{1}$$

where $\mathcal{T}$ is a finite index set. We denote *disjunctive term* $t \in \mathcal{T}$ by

$$P^t := \{x \in P : D^t x \geq D_0^t\}. \tag{P^t}$$

Let $P_D := \operatorname{cl}\operatorname{conv}(\cup_{t\in\mathcal{T}} P^t)$ be the *disjunctive hull*, the *closed convex hull* of the elements of P satisfying the disjunction. We assume the disjunction satisfies $P_I \subseteq P_D$ and $\bar{x} \notin P_D$.

2 Point-ray linear program

Let $\mathcal{P}$ and $\mathcal{R}$ denote sets of points and rays in $\mathbb{R}^n$.[2] Define the *point-ray linear program* (PRLP), taking as an input the point-ray collection $(\mathcal{P}, \mathcal{R})$ and an objective direction $w \in \mathbb{R}^n$, as follows:

$$\begin{aligned} \min_{\alpha,\beta} \quad & \alpha^\intercal w & & \\ & \alpha^\intercal p \geq \beta & \text{for all } p \in \mathcal{P} & \\ & \alpha^\intercal r \geq 0 & \text{for all } r \in \mathcal{R}. & \end{aligned} \tag{PRLP}$$

Every feasible solution (α, β) to (PRLP) is an inequality $\alpha^\intercal x \geq \beta$; these are what we refer to as VPCs. Define the *point-ray hull* as $\operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$.

We immediately address the nature of the cuts obtainable from (PRLP). Theorem 1 shows that the extreme ray solutions to (PRLP) correspond to facet-defining inequalities for the point-ray hull (which, in accordance with convention, we simply refer to as *facets* of the point-ray hull).

Theorem 1. *The inequality $\alpha^\intercal x \geq \beta$ is valid for $\operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$ if and only if (α, β) is a feasible solution to* (PRLP)*. The inequality defines a facet of the point-ray hull if and only if the solution (α, β) is an extreme ray of* (PRLP)*.*

Proof. Every point $\hat{x} \in \operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$ can be expressed as a convex combination of the elements in $(\mathcal{P}, \mathcal{R})$. For any inequality $\alpha^\intercal x \geq \beta$ satisfied by all of these points and rays, it follows that $\alpha^\intercal \hat{x} \geq \beta$. Every extreme ray (α, β) to (PRLP) has the additional property that n affinely independent points and rays from the point-ray collection satisfy $\alpha^\intercal x = \beta$, which means that the inequality defines a facet of $\operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$. The reverse directions follow, respectively, from the definitions of valid and facet-defining inequality. □

There are two primary challenges encountered with the PRLP. First, the point-ray collection needs to be chosen judiciously, to balance the strength of the obtainable VPCs with the

[2] In this paper, *rays* refer to either (1) a direction of unboundedness, or (2) an "extended edge" of a polyhedron, i.e., an edge extended (to infinity) from one of its endpoints.

time required to generate them, while ensuring validity of the cuts. Second, it is critically important to intelligently select the objective directions w for (PRLP). The next sections are devoted to addressing these questions. We first detail properties of the feasible region of (PRLP). In particular, we describe how we normalize the PRLP, prove necessary and sufficient conditions for VPCs to be valid cuts for P_I, and discuss the conditions under which VPCs are facet-defining for the disjunctive hull.

2.1 Normalization of the PRLP

As presented, the linear program (PRLP) does not have a finite objective value for any nonzero objective direction w: if (α, β) is feasible, then so is $(\lambda\alpha, \lambda\beta)$ for any nonnegative $\lambda \in \mathbb{R}$. In theory, each ray (α, β) yields a valid cut; however, in practice, the ray returned by a solver might not be extreme, which may correspond to a weak cut. This is the same well-documented issue that arises with the CGLP. To resolve this, a *normalization* is applied, truncating the cone defining the feasible region of (PRLP). The choice of normalization can have a significant effect on the cuts that are ultimately generated [34, 47].

One solution is to constrain the magnitude of the cut coefficients, e.g., via (a linearization of) $\sum_{i=1}^{n} |\alpha_i| \leq 1$ [15]. This normalization has the undesirable characteristic that it may add extreme points to the feasible region of (PRLP) that do not correspond to facets of the disjunctive hull.

A second normalization proposed by Balas and Perregaard [13]—also used by Perregaard and Balas [53] and Louveaux, Poirrier, and Salvagnin [49]—is to add the constraint $\alpha^\intercal(p - \bar{x}) = 1$, for some $p \in P_D$. This idea has been studied by Serra [56] and Conforti and Wolsey [28], when the role of the objective and normalization is swapped. This guarantees that (PRLP) is always bounded and has other nice properties, but depends on a good choice of p.

We take a third approach for normalizing (PRLP) of fixing β to a constant value. As observed by Balas and Margot [12], it suffices to consider only three values for β: $\{-1, 0, +1\}$. This might indicate that one would need to solve three linear programs to generate cuts with such a normalization, as discussed by Louveaux, Poirrier, and Salvagnin [49]. We avoid this issue by formulating (PRLP) in the *nonbasic space* relative to $\bar{x}$. In this space, the (LP) optimal solution $\bar{x}$ is the origin. As a result, if we are looking for inequalities that are violated by $\bar{x}$, it suffices to fix $\beta = 1$. Another advantage of working in the nonbasic space is that (PRLP) may be much sparser than if it were formulated in the structural space of variables. This is because the number of nonzero components in every row roughly corresponds to the number of simplex pivots from $\bar{x}$ to the point or ray for that row. When we normalize with $\beta = 1$, every basic feasible solution to (PRLP) corresponds to an inequality violated by $\bar{x}$.

One limitation imposed by our normalization is that the cuts we generate are only those that remove $\bar{x}$. It is clearly necessary to remove this point in order to make progress beyond the linear programming relaxation towards an integer-feasible solution, but it is a "transient" point, in the sense that it is no longer truly important after the first cut that removes it (see, e.g., the discussion by Cadoux and Lemaréchal [23]). This limitation is easily avoided by generating cuts with $\beta = 0$ or -1, but effectively implementing this idea requires an independent in-depth investigation that we leave to future research.

2.2 Proper point-ray collections: ensuring valid inequalities

From Theorem 1, we have that feasible solutions to (PRLP) are valid for the point-ray hull. We further need to ensure that VPCs are valid for P_I. The point-ray collections with this guarantee will be called *proper*, adapting the definition from Kazachkov et al. [44] within the *generalized intersection cut* paradigm [12], which uses a linear program analogous to the PRLP but derives points and rays via a different procedure. We remove a dependency on $\bar{x}$ present in the prior definition, which enables our framework to produce arbitrary disjunctive inequalities that do not necessarily cut $\bar{x}$ and can be stronger than *any* intersection cut obtainable from the same disjunction [11]. This modification also simplifies the characterization of proper point-ray collections.

Definition 2. *The point-ray collection* $(\mathcal{P}, \mathcal{R})$ *is called* proper *if* $\alpha^\intercal x \geq \beta$ *is valid for* P_I *whenever* (α, β) *is feasible to* (PRLP)*.*

As a direct corollary to Theorem 1, we obtain a necessary and sufficient condition for a point-ray collection to be proper.

Corollary 3. *A point-ray collection* $(\mathcal{P}, \mathcal{R})$ *is proper if and only if* $P_I \subseteq \text{conv}(\mathcal{P}) + \text{cone}(\mathcal{R})$*.*

Proof. Sufficiency of the condition follows from Theorem 1: every feasible solution to (PRLP) is valid for $\text{conv}(\mathcal{P}) + \text{cone}(\mathcal{R})$ and hence for P_I. Necessity is similarly evident as, otherwise, there exists an extreme ray of (PRLP) that cuts a point in P_I. □

Of course, we do not work with the integer hull directly. The intermediary is the disjunctive hull. The next corollary is a key result for the development of a practical procedure working with points and rays. It states that as long as the point-ray hull forms a $\mathcal{V}$*-polyhedral relaxation* of P_D, then the point-ray collection is proper.

Corollary 4. *A point-ray collection* $(\mathcal{P}, \mathcal{R})$ *is proper if* $P^t \subseteq \text{conv}(\mathcal{P}) + \text{cone}(\mathcal{R})$ *for all* $t \in \mathcal{T}$*, or, equivalently, if* $P_D \subseteq \text{conv}(\mathcal{P}) + \text{cone}(\mathcal{R})$*.*

2.3 Simple VPCs from simple cone relaxations

To generate valid VPCs from the PRLP, we first have to compute a proper point-ray collection. A natural starting point is a $\mathcal{V}$-polyhedral description of the disjunctive hull. For $t \in \mathcal{T}$, let $\mathcal{P}^{t*}$ and $\mathcal{R}^{t*}$ be the complete set of extreme points and rays of P^t, and define $\mathcal{P}^* := \cup_{t \in \mathcal{T}} \mathcal{P}^{t*}$ and $\mathcal{R}^* := \cup_{t \in \mathcal{T}} \mathcal{R}^{t*}$. As a corollary of Theorem 1, we know not only that $(\mathcal{P}^*, \mathcal{R}^*)$ is proper but also that basic feasible solutions of the normalized (PRLP) from this point-ray collection correspond to facet-defining inequalities for the disjunctive hull, P_D.

Corollary 5. *The point-ray collection* $(\mathcal{P}^*, \mathcal{R}^*)$ *is proper. Every extreme ray solution* (α, β) *of the associated* (PRLP) *corresponds to a facet* $\alpha^\intercal x \geq \beta$ *of* P_D*. Conversely, for every facet* $\alpha^\intercal x \geq \beta$ *of* P_D*, the solution* (α, β) *to* (PRLP) *is feasible and extreme.*

The complete $\mathcal{V}$-polyhedral description of each disjunctive term is, however, impractical, as the number of points and rays can grow exponentially large in m and n. A reasonable

alternative would be to use some small subset of the point-ray collection $(\mathcal{P}^*, \mathcal{R}^*)$. It is not difficult to see that this could lead to invalid cuts, as we show by example in the extended manuscript [10]. It is for this reason that Perregaard and Balas [53] and Louveaux, Poirrier, and Salvagnin [49] employ constraint generation to obtain valid cuts.

We take an alternative approach, based on Corollary 4: instead of pursuing all facet-defining inequalities for the disjunctive hull, we use a relaxation of P_D with a compact $\mathcal{V}$-polyhedral description. One convenient relaxation for each disjunctive term is the *basis cone* C^t at an optimal solution p^t to $\min_x\{c^\intercal x : x \in P^t\}$, formed from p^t and a *cobasis* $\mathcal{N}(p^t)$ associated with p^t. This cone is defined as the intersection of the n inequalities corresponding to the nonbasic variables, which are indexed by $\mathcal{N}(p^t)$, and it has a compact $\mathcal{V}$-polyhedral description, with only one extreme point (p^t) and n extreme rays, which we denote $r^{t1}, \ldots, r^{tn}$. We refer to the union of these points and rays across all terms as the *simple point-ray collection* $(\mathcal{P}^0, \mathcal{R}^0)$, where $\mathcal{P}^0 := \cup_{t\in\mathcal{T}}\{p^t\}$ and $\mathcal{R}^0 := \cup_{t\in\mathcal{T}}\{r^{tj}\}_{j\in[n]}$, and we will use the shorthand $P_D^0 := \text{conv}(\mathcal{P}^0) + \text{cone}(\mathcal{R}^0)$ to denote the corresponding *simple point-ray hull*. The specific PRLP used in our experiments is given below as (PRLP^0). Recall that we formulate the problem in the nonbasic space. To make this explicit, for any $x \in \mathbb{R}^n$, we use $\tilde{x}$ for its representation in the nonbasic space.

$$\begin{aligned} \min_{\tilde{\alpha}} \quad & \tilde{\alpha}^\intercal \tilde{w} & & \\ & \tilde{\alpha}^\intercal \tilde{p}^t \geq 1 & \text{for all } t \in \mathcal{T} & \\ & \tilde{\alpha}^\intercal \tilde{r}^{tj} \geq 0 & \text{for all } t \in \mathcal{T},\ j \in [n] & \end{aligned} \tag{PRLP^0}$$

The cuts from (PRLP^0) will be called *simple* VPCs. We state their validity as Proposition 6.

Proposition 6. *The simple point-ray collection $(\mathcal{P}^0, \mathcal{R}^0)$ is proper.*

Proof. The result follows from Corollary 4 and the fact that $P_D \subseteq P_D^0$. □

It is useful at this point to make a theoretical comparison between the PRLP and extended formulation used for the CGLP. Any valid disjunctive cut can be theoretically found with a CGLP having size polynomial in the dimensions of the original problem and the number of disjunctive terms. In particular, the CGLP with a fixed right-hand side β has $(n+1)\cdot|\mathcal{T}|$ constraints and $n + (m + m_t)\cdot|\mathcal{T}|$ variables (where m_t denotes the number of rows of D^t). It is similarly possible to produce all valid disjunctive cuts with the $\mathcal{V}$-polyhedral framework, when $(\mathcal{P}, \mathcal{R}) = (\mathcal{P}^*, \mathcal{R}^*)$, as we proved in Corollary 5. The disadvantage is that the PRLP may now have exponentially many constraints, which is why we turned to the relaxation-based generator. This yields (PRLP^0), whose feasible region is defined by only $(n+1)\cdot|\mathcal{T}|$ constraints (the same as for the CGLP) and only n variables. Moreover, working in the original dimension of the problem confers significant computational efficiency over the CGLP framework, as we discuss in Section 3 via the experiments of Perregaard and Balas [53], but the tradeoff is being able to generate merely a subset of all valid disjunctive cuts. Nevertheless, the subsequent theoretical results of this section and the later computational results indicate that this subset already captures strong disjunctive cuts.

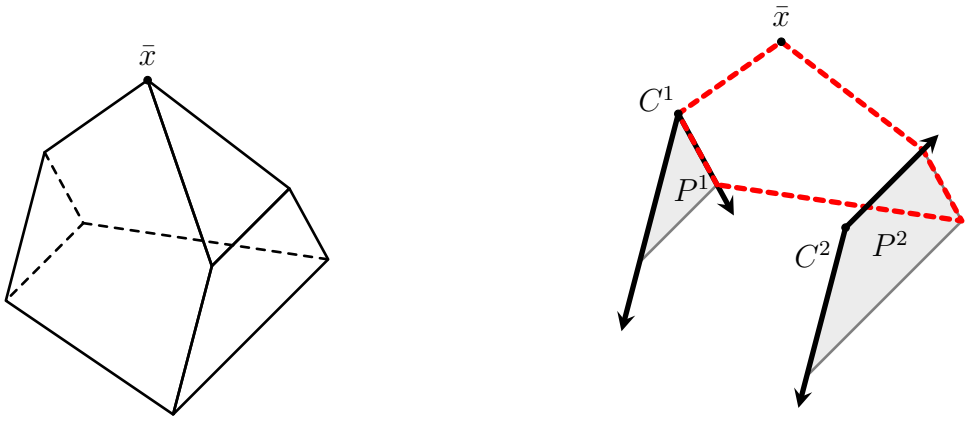


Figure 1: This example shows how the simple point-ray collection could limit the set of obtainable inequalities valid for the disjunctive hull. The right panel shows that the dashed inequality (that is valid for P_I) would violate a ray of the cone C^2. The cones are shown as two dimensional for ease of illustration.

2.4 Simple VPCs corresponding to facets of the disjunctive hull

In this section, we compare the disjunctive hull P_D with the relaxation P_D^0. Although P_D^0 is a drastic relaxation of P_D, in that it is defined by a small fraction of the inequalities defining P_D, we show that P_D^0 can tightly approximate P_D in the region of interest to us. We distinguish between two types of facets of P_D^0: those that are facets of P_D, and those that are not. Clearly not all facets of P_D are captured by P_D^0, which we illustrate in Figure 1. The results below concern precisely which facets of P_D exist as facets of P_D^0.

Consider a facet-defining inequality of P_D^0 that is violated by $\bar{x}$ and a corresponding basic feasible solution to (PRLP^0), after adding slack variables on the constraints. Since the cut coefficient variables $\tilde{\alpha}$ are unrestricted in sign, the nonbasic variables in this solution are all slack variables, and the corresponding tight constraints of (PRLP^0) identify n affinely independent points and rays of $(\mathcal{P}^0, \mathcal{R}^0)$ that lie on the inequality and certify that it defines a facet of P_D^0. We call these the "*nonbasic points*" and "*nonbasic rays*" with respect to the basic feasible solution.

By construction of the simple point-ray collection, each of the nonbasic points also belongs to P_D. Now assume that, for each of the nonbasic rays r, there is a point $p \in \mathcal{P}^0$ tight for the cut and a $\lambda > 0$ such that $p_r := p + \lambda r \in P_D$, which is thus also on the cut. Then the nonbasic points and the points p_r for the nonbasic rays certify that the inequality also defines a facet of P_D.

This assumption is, unfortunately, not generally satisfied: for some nonbasic rays, we might not find a point in P_D corresponding to that ray. One difficulty, illustrated in the example shown in Figure 2, is that although a ray originates from some particular point and term (when building the point-ray collection), it is ultimately added to *all* points in $\mathcal{P}^0$ to calculate P_D^0. In this example, a four-term disjunction is taken. The first panel shows P; the second panel shows P_D, as well as P^t and the cones C^t for each $t \in [4]$, with C^1 labeled;[3] and the third panel shows P_D^0 and the points and rays (as wavy arrows) that are tight for each of

[3]For simplicity in making the example, these C^t are not basis cones, due to degeneracy.

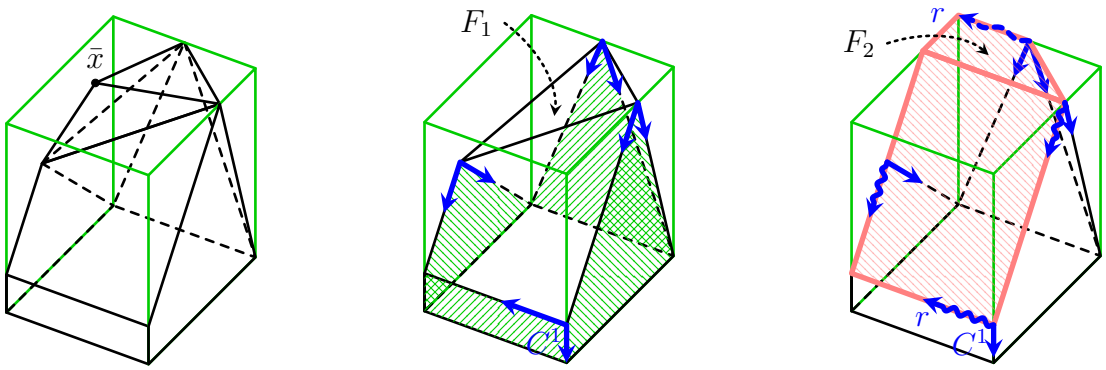


Figure 2: Rays in the point-ray collection impact the set of cuts that can be generated. The dashed wavy line in the third panel corresponds to a ray r that is added to $\mathcal{R}^0$ from one term of the disjunction, but affects the point-ray hull when it originates from a point from a different term.

the facets of this point-ray hull. Consider the ray r of C^1 labeled in two places in the third panel; it is added to the collection from one disjunctive term but impacts two facets. The effect of r is it causes F_1 (a facet-defining inequality for P_D) to be invalid for the point-ray hull, and it adds a facet (F_2) to the point-ray hull that is redundant for P.

Consider a basic feasible solution of (PRLP^0) and associated cut $\alpha^\intercal x \geq \beta$. If r^{tj} is a nonbasic ray in this solution, but there is no term $t' \in \mathcal{T}$ for which r^{tj} is an extreme ray of $C^{t'}$ and $\alpha^\intercal p^{t'} = \beta$, then we call this a *stray ray* (for the facet). Note that $t' \neq t$ is possible if a ray is extreme for multiple terms. With this, we say that a facet of P_D^0 is *standard* if there is a corresponding basis of (PRLP^0) with no stray rays, i.e., when $\tilde{\alpha}^\intercal \tilde{r}^{tj} = 0$ (with the row's slack variable nonbasic) implies that $\tilde{\alpha}^\intercal \tilde{p}^{t'} = 1$, for some $t' \in \mathcal{T}$ for which r^{tj} is an extreme ray of $C^{t'}$. Thus, facet F_2 in Figure 2 is not standard, due to stray ray r from C^1. We apply this concept in Theorem 7 to state a sufficient condition for a facet of the simple point-ray hull to be a facet of the disjunctive hull.

Theorem 7. *Suppose the basis defining p^t is unique for each $t \in \mathcal{T}$. If a facet of P_D^0 is standard, then it is a facet of P_D that cuts $\bar{x}$.*

Proof. Given a standard facet of P_D^0 and a corresponding basic feasible solution of (PRLP^0) with no stray rays, we construct n affinely independent points from P_D that lie on the facet based on the nonbasic points and rays in this solution. For each nonbasic ray $r \in \mathcal{R}^0$, there is a term $t \in \mathcal{T}$ such that r is an extreme ray of C^t and the point p^t lies on the facet. Since the basis defining p^t is not primal degenerate, for small enough $\lambda > 0$, the point $p_r := p + \lambda r$ is in P^t (and thus in P_D), as otherwise the ray would be cut by a hyperplane of P^t that is tight at p^t but not defining C^t. The nonbasic points—all belonging to P_D—and the points p_r provide the desired n affinely independent points. □

The requirement that the facet of P_D has to cut $\bar{x}$ is due to the normalization of (PRLP^0) by $\tilde{\beta} = 1$.

The case of a split disjunction deserves special attention given its importance in prior work, especially in the context of L&PCs. Although a facet of P_D is not necessarily a simple

VPC (whereas it exists as an L&PC), we can conclude the converse using our much more compact formulation and Theorem 7, i.e., that all simple VPCs are facet-defining for the disjunctive hull.

Theorem 8. *Suppose that* (1) *is a split disjunction and the bases defining* p^1 *and* p^2 *are unique. Then every facet of* P_D^0 *that is tight on both* p^1 *and* p^2 *is a facet of* P_D*. Moreover, every facet of* P_D *that is tight on both* p^1 *and* p^2 *and cuts* $\bar{x}$ *exists as a facet of* P_D^0*.*

Proof. For the first statement, our assumptions imply that every facet of P_D^0 that is tight on both p^1 and p^2 is standard, because then all points of $\mathcal{P}$ have zero slack, so there can be no stray rays. The second statement follows from convexity, in that a facet of P_D that is tight on p^t will be valid for C^t, $t \in \{1, 2\}$, as otherwise that facet cuts a ray of C^t and the corresponding point from P_D in the ray's relative interior. □

Theorem 8 can be extended to more general disjunctions for facets of P_D^0 that are tight on all points p^t, $t \in \mathcal{T}$.

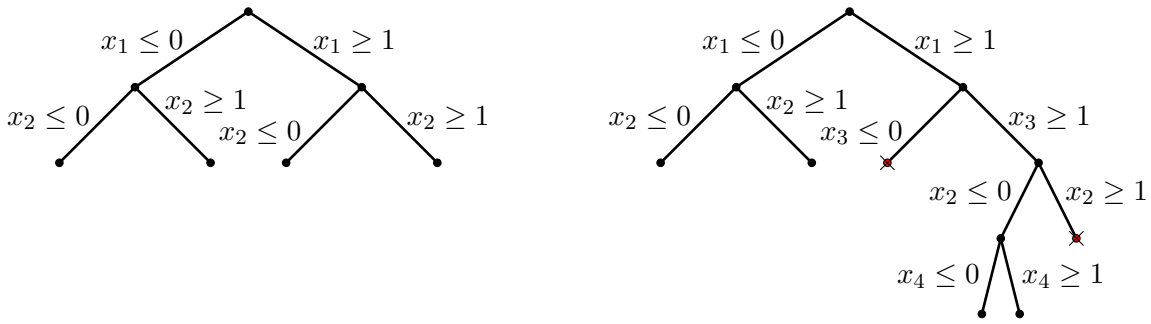
3 Choosing strong disjunctions

The previous section introduces our computationally-viable way to generate disjunctive cuts through the PRLP via a $\mathcal{V}$-polyhedral relaxation of the given disjunctive hull. To complete the setup of the constraints of (PRLP), it remains to specify which class of disjunctions we will use in our experiments.

Much of the focus in the recent literature on cutting planes has been on generating stronger cuts from shallow disjunctions, i.e., those that utilize relatively few (one or two) integer variables, based on indices of integer variables that are fractional in $\bar{x}$. This includes disjunctions based on the complements of triangles, quadrilaterals, and crosses. To compensate for the weakness of the disjunction, typically cuts are generated from several disjunctions from the same class in each round. These families of disjunctions do contain cuts that outperform Gomory cuts, but one needs to carefully select which disjunctions to test within each family, and the computational cost associated with finding the stronger cuts often outweighs their benefit (see, e.g., [32]).

We circumvent some of these difficulties by expending additional effort to generate one strong "deep" disjunction per instance for cut generation. Specifically, the disjunctions we define come from the set of leaf nodes of a partial branch-and-bound tree.[4] A partial branch-and-bound tree has the advantage of conveying additional information about (IP) (with respect to an alternative disjunction with the same number of terms obtained without branching). For instance, the partial tree may be asymmetric and include pruning by infeasibility, by integrality, and by bound. We demonstrate this by example in Figure 3, contrasting a cross disjunction generated from two integer variables x_1 and x_2 to a four-term disjunction that might be obtained using the branch-and-bound process.

[4]Besides intuitive appeal, our choice of disjunctions is further bolstered by the results in Appendix G of the extended manuscript [10], indicating that VPCs from a multitude of split or cross disjunctions are weaker.



(a) The disjunction is all possible assignments of x_1 and x_2.

(b) A different variable is branched on, resulting in some pruned nodes and two stronger disjunctive terms.

Figure 3: Two four-term disjunctions (the leaf nodes of the trees).

Disjunctive cuts coming from partial branch-and-bound trees have previously been proposed and tested in several contexts, indicating the potential impact of a more efficient method for obtaining such cuts. The majority of these previous experiments rely on variants of the higher-dimensional CGLP, e.g., in the context of the cutting plane tree algorithm [24, 25] and in stochastic mixed-integer programming applications [55, 52, 58, 51, 36, 54]. A famous example is the computational experience of solving the *seymour* problem with the aid of L&PCs, as documented by Ferris, Pataki, and Schmieta [33].

Methods resembling the VPC approach include the aforementioned cuts from the $\mathcal{V}$-polyhedral perspective based on row generation [53, 49]. One takeaway from the paper by Perregaard and Balas [53] is a comparison to generating disjunctive cuts via solving the higher-dimensional CGLP: the authors conclude that solving the CGLP becomes relatively much slower as the number of terms of the disjunction grows, a result that would be more pronounced if the expense of row generation were avoided. In other closely related work, Chvátal, Cook, and Espinoza [27] experiment with partial branch-and-bound trees as part of local or target cut algorithms [6, 22], which are dual to the separation schemes in this paper and that of [53, 49]. Most recently, Chen and Luedtke [26] generate objective cuts from multiterm disjunctions in the context of two-stage stochastic programs.

We refer the interested reader to the dissertation of Kazachkov [42] for a more in-depth treatment of related literature.

4 Choosing appropriate objectives

Having set up the constraints of the PRLP, we now analyze the theoretical strength of VPCs to drive our choices of objective functions $\tilde{w}$ for (PRLP^0). Choosing these carefully is critical to the success of any VPC algorithm, as the objectives directly determine the nature of the VPCs obtained. Aside from the type of cuts obtained, it is also important to make the cut-generating process efficient. We say that every time we solve (PRLP) and a new cut is

not generated, a *failure* occurs. In an early implementation of VPCs, failures were frequent (over 85% of the objectives tried).

One reason for failure is that (PRLP^0) may be infeasible for a particular point-ray collection. This occurs, for instance, if $\bar{x}$ belongs to P_D^0 (feasible solutions of (PRLP^0) are inequalities that separate $\bar{x}$, the origin in the nonbasic space). Figure 2 actually illustrates such a situation. The next proposition gives a sufficient condition for the feasibility of (PRLP^0) that we use in our implementation, though it can be extended to apply to (PRLP) more generally. Let p^* denote a point from $\mathcal{P}^0$ with minimum objective value, i.e.,

$$p^* \in \arg\min_t \{c^\intercal p^t : t \in \mathcal{T}\}. \tag{p^*}$$

Proposition 9. *If $\tilde{c}^\intercal \tilde{p^*} > 0$, then* (PRLP^0) *is feasible.*

Proof. It suffices to observe that $\tilde{\alpha} = \tilde{c}/\tilde{c}^\intercal \tilde{p^*}$ is a feasible solution to (PRLP^0), corresponding to the objective cut $\tilde{\alpha}^\intercal \tilde{x} \geq 1$, equivalently $c^\intercal x \geq c^\intercal p^*$. For $t \in \mathcal{T}$, $\tilde{\alpha}^\intercal \tilde{p}^t = \tilde{c}^\intercal \tilde{p}^t / \tilde{c}^\intercal \tilde{p^*} \geq 1$ by definition of p^*. For $r \in \mathcal{R}^0$, we need to show that $c^\intercal r \geq 0$. This follows from the fact that $p^t \in \arg\min_x \{c^\intercal x : x \in P^t\}$, which implies that p^t is also optimal when minimizing over C^t. For any ray $r \in C^t$, for all $\lambda > 0$, the point $p^t + \lambda r$ belongs to C^t. Hence, $c^\intercal p^t + \lambda c^\intercal r \geq c^\intercal p^t$.□

This condition is satisfied, for example, whenever each disjunctive term's LP relaxation has an optimal objective value worse than that of the original LP.

The two other primary reasons for failures we observed were that, for a given objective direction $\tilde{w}$: (1) (PRLP^0) was feasible but unbounded, or (2) (PRLP^0) had a finite optimal solution but the corresponding cut was a duplicate of a previously generated cut. We will not be able to completely eliminate failures, but in the remainder of this section, we work towards choosing objectives that help reduce the failure rate. At the same time, we will target generating strong VPCs, while mostly ignoring the potential effect of the cuts within branch and bound; this latter goal is poorly understood and hence difficult to target directly.

The first candidate for an objective direction is to maximize the violation by $\bar{x}$, as is done in the case of L&PCs. Unfortunately, in the nonbasic space and with $\beta = 1$, $\bar{x}$ is simply the origin and all cuts have violation equal to 1. As proxies, we use two other objectives. First, we try the all-ones objective, $\tilde{w} = e$. The interpretation is that we seek an inequality that puts equal weight on cutting each of the rays of the basis cone at $\bar{x}$. Second, we add to P a round of GMICs, separated from $\bar{x}$, and calculate a new optimal solution x'; we then use $\tilde{w} = \tilde{x}'$, which finds a cut maximizing the violation with respect to x'.

Finding cuts that maximize violation with respect to points not in P_D is a paradigm that may place too much emphasis on cutting away irrelevant parts of the relaxation. The alternative is to find inequalities that minimize the slack with respect to points that do belong to P_D. Within this latter perspective, we are able to utilize whatever structural information we possess about the disjunctive hull. We will now discuss precisely what kind of information can be inferred from our $\mathcal{V}$-polyhedral relaxations of the disjunctive hull.

The next result states that, despite the vastly relaxed simple point-ray collection, the optimal value over the disjunctive hull can be obtained by optimizing over the points in $\mathcal{P}^0$.

Proposition 10. *The point $p^* \in \arg\min_p \{c^\intercal p : p \in \mathcal{P}^0\}$ is also an optimal solution to $\min_x \{c^\intercal x : x \in P_D^0\}$ and to $\min_x \{c^\intercal x : x \in P_D\}$. Moreover, there are n facets of P_D^0 that can be added to P such that p^* is also an optimal solution to the resulting relaxation.*

Proof. This is a direct consequence of the fact that $p^t \in \arg\min_x\{c^\intercal x : x \in P^t\}$ for all $t \in \mathcal{T}$. The n inequalities are simply the ones determined by the cobasis of p^* from solving $\min_x\{c^\intercal x : x \in P_D^0\}$. □

We say that $c^\intercal p^*$ is the *disjunctive lower bound* and examine whether we can achieve it via VPCs. Note that we can always add the inequality $c^\intercal x \geq c^\intercal p^*$, but this is generally counterproductive, as such objective cuts tend to create multiple optimal solutions to the subsequent relaxation, which cause difficulties for solvers.

By Proposition 10, we know that we can attain the disjunctive lower bound via n facets of the point-ray hull that are tight at p^*. One way to generate a cut tight at p^* is to use the objective direction $\tilde{w} = \tilde{p^*}$. Absent numerical issues, the optimal solution will be some $\bar{\alpha}$ such that $\tilde{\bar{\alpha}}^\intercal \tilde{p^*} = 1$. Though this is only one cut, it can be used to find other objective directions. We will work with a modified (PRLP^0), which we refer to as *PRLP$^=$*, in which the constraint $\tilde{\alpha}^\intercal \tilde{p^*} \geq 1$ is changed to $\tilde{\alpha}^\intercal \tilde{p^*} = 1$. Let $\overline{\mathcal{R}}$ be the set of rays from $\mathcal{R}^0$ that are not tight for the cut $\tilde{\bar{\alpha}}^\intercal \tilde{x} \geq 1$, i.e., $\overline{\mathcal{R}} \coloneqq \{r \in \mathcal{R}^0 : \bar{\alpha}^\intercal r > 0\}$.

Proposition 11. *PRLP$^=$ with objective direction $\tilde{w} = \tilde{r}$, $r \in \overline{\mathcal{R}}$, has a finite optimal solution. The optimal value is strictly less than $\tilde{\bar{\alpha}}^\intercal \tilde{r}$ only if the resulting cut is distinct from $\tilde{\bar{\alpha}}^\intercal \tilde{x} \geq 1$. The optimal value is zero if and only if there exists a facet of P_D^0 that cuts $\bar{x}$ and is tight on r.*

Proof. The fact that PRLP$^=$ is finite and bounded is a direct result of the constraint $\tilde{\alpha}^\intercal \tilde{r} \geq 0$ and the feasibility of PRLP$^=$. The second statement is obvious. The last statement comes from the one-to-one correspondence between basic feasible solutions to PRLP$^=$ and facets of P_D^0 that cut away $\bar{x}$. □

Propostion 11 resolves issue (1) mentioned above, of having a feasible PRLP that is unbounded. Unfortunately, we may still get failures from issue (2), meaning the optimal solution to PRLP$^=$ corresponds to a cut we previously generated. For example, it may be the case that there exists $r \in \overline{R}$ such that, for all $\tilde{\alpha}$ feasible to PRLP$^=$, $\tilde{\alpha}^\intercal \tilde{r} \geq \tilde{\bar{\alpha}}^\intercal \tilde{r}$. Using such an r as the objective for PRLP$^=$ could reproduce the solution $\bar{\alpha}$. To prevent such phenomena from excessively slowing down cut generation, we add a failure rate parameter, which detects when there is an unacceptably small percent of the objectives successfully producing new cuts, triggering early termination of the procedure. We discuss this further in the extended manuscript [10].

Lastly, we address a practical question: will we always attain the disjunctive lower bound $c^\intercal p^*$ via simple VPCs? On one hand, Proposition 10 states that this bound is attainable via n facets of the point-ray hull tight at p^* (which may not be facets of P_D). However, the answer to this question can be no, given our algorithmic choices. To see why, we need to understand which inequalities can be generated from (PRLP^0) in our setup. The specific modifications we have made from the general case are that we fix $\beta > 0$ and work in the nonbasic space in which $\bar{x}$ is the origin. This implies that we will never generate any facet-defining inequalities for P_D^0 that are satisfied by $\bar{x}$. One might initially assume that these inequalities are not necessary in order to attain the bound $c^\intercal p^*$. We dispel that notion in Figure 4. The example demonstrates that ignoring inequalities that do not cut away $\bar{x}$ may lead to an optimal value (after adding cuts) that is strictly better than $c^\intercal p^*$. Note that in

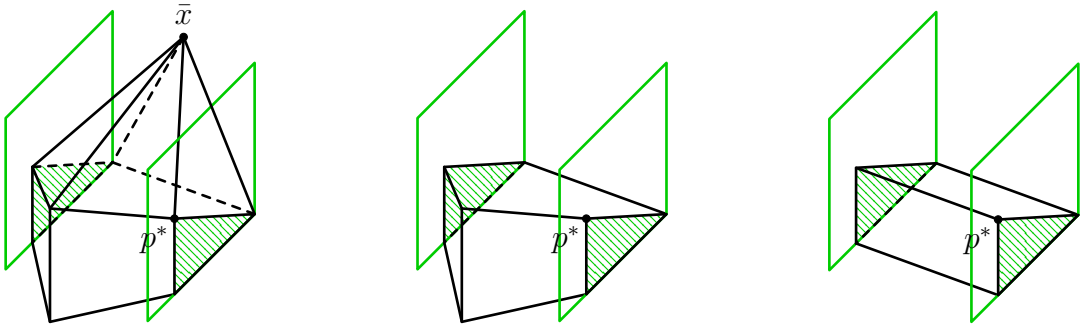


Figure 4: Example that shows an inequality tight at p^* that does not cut away $\bar{x}$ may be necessary for achieving the bound $c^\intercal p^*$. In this example, we assume we are maximizing along the vertical axis. The first panel shows the original polytope. The second panel is the polytope after adding the only split inequality that cuts away $\bar{x}$. The third panel shows the polytope after adding all the valid split cuts.

this example, the point-ray collection uses the complete $\mathcal{V}$-polyhedral description of each P^t, so, unlike the example in Figure 1, this situation is not a consequence of using a relaxation of P_D. This may partially explain why, in our experiments, despite our careful objective choices, we do not always obtain the bound $c^\intercal p^*$.

5 Computational results

Our computational experiments have two goals: (1) assess the strength of VPCs by the percent root gap closed by one round of the cuts, which we discuss in Section 5.2, and (2) evaluate the effectiveness of VPCs when added at the root and used as part of branch and bound, covered in Section 5.3. Before presenting our results, we review our algorithmic choices in Section 5.1.

5.1 Computational setup

The $\mathcal{V}$-polyhedral framework we have introduced is quite general, and there are many possibilities for implementing it. We experiment with only a small subset of the possible parameters, so a more thorough tuning may improve upon our reported results. Algorithm 1 summarizes our choices for generating VPCs; more details can be found in the extended manuscript [10]. All experiments are performed on computers equipped with i9-13900K CPUs. Our implementation is in C++ in the COIN-OR framework [48] using Clp [2] and Cbc [1] as the underlying linear programming and branch-and-bound solvers to generate VPCs. Gurobi 10.0.3 [39] is used to test the effectiveness of VPCs when embedded in branch and bound, averaged across 7 random seeds.

Instance selection. Our 332 test instances come from the union of the MIPLIB [19, 20, 4, 46, 37], CORAL [31], and NEOS sets, restricted to those with at most 5,000 rows and

Algorithm 1 Type 1 $\mathcal{V}$-Polyhedral Cuts

Input: Polyhedron P; objective direction c; disjunction $\vee_{t\in\mathcal{T}} P^t$.

for $t \in \mathcal{T}$ **do**
  $p^t \leftarrow$ optimal solution to $\min_x\{c^\intercal x : x \in P^t\}$.
  $C^t \leftarrow$ basis cone with respect to a cobasis of p^t.
  $\mathcal{P}^t \leftarrow \{p^t\}$ and $\mathcal{R}^t \leftarrow$ extreme rays of C^t.
$\mathcal{P} \leftarrow \cup_{t\in\mathcal{T}}\mathcal{P}^t$, $\mathcal{R} \leftarrow \cup_{t\in\mathcal{T}}\mathcal{R}^t$, $\mathcal{C} \leftarrow \emptyset$.
Solve (PRLP^0) with all-ones objective $\tilde{w} = e$; add resulting cut to $\mathcal{C}$.
Add GMICs to P; let x' be an optimal solution to the new relaxation.
Solve (PRLP^0) with $\tilde{w} = \tilde{x}'$ and add the resulting cut to $\mathcal{C}$.
$p^* \leftarrow$ point from $\arg\min_p\{c^\intercal p : p \in \mathcal{P}\}$.
Solve (PRLP^0) with $\tilde{w} = \tilde{p^*}$ and add the resulting cut, $\tilde{\bar{\alpha}}^\intercal \tilde{x} \geq 1$, to $\mathcal{C}$.
PRLP$^=$ $\leftarrow$ (PRLP^0) with added constraint $\tilde{\alpha}^\intercal \tilde{p^*} = 1$.
$\overline{\mathcal{P}} \leftarrow \{p \in \mathcal{P} : \tilde{\bar{\alpha}}^\intercal \tilde{p} > 1\}$, sorted in order of decreasing angle with c.
$\overline{\mathcal{R}} \leftarrow \{r \in \mathcal{R} : \tilde{\bar{\alpha}}^\intercal \tilde{r} > 0\}$, sorted in order of decreasing angle with c.
for all $w' \in \overline{\mathcal{P}} \cup \overline{\mathcal{R}}$ **do**
  Solve PRLP$^=$ with $\tilde{w} = \tilde{w}'$ and add the resulting cut $\tilde{\bar{\alpha}}^\intercal \tilde{x} \geq 1$ to $\mathcal{C}$.
  Remove from $\overline{\mathcal{P}} \cup \overline{\mathcal{R}}$ all points and rays that are tight for $\tilde{\bar{\alpha}}^\intercal \tilde{x} \geq 1$.
  If number of objectives tried is two times the cut limit, then break.
return Set $\mathcal{C}$ of generated cuts.

columns and by other criteria detailed in the extended manuscript [10]. Every instance is preprocessed once by Gurobi's presolve.

Generation of disjunction. The disjunctive terms provided as input to Algorithm 1 are the leaf nodes of a partial branch-and-bound tree terminated after reaching 2^ℓ, $\ell \in [6]$, leaf nodes of the partial tree, which form the disjunction that we input to Algorithm 1. The $\mathcal{V}$-polyhedral relaxation for each term is the simple point-ray collection defined in Section 2.3. The partial branch-and-bound tree is generated by the default node, variable, and branch selection rules for `Cbc`. Generating the partial tree can at times be expensive, and we make no claim that our enumeration technique is the best for cut generation; other, perhaps weaker but less costly, strategies also merit consideration [57].

Evaluation within branch and bound. The VPCs are given to Gurobi at the root as *user cuts*, which allows Gurobi to use its internal cut selection criteria. All of Gurobi's default parameters are used except the following:

- Set random seed to $i \cdot 628$, for $i \in \{1, \ldots, 7\}$.
- Set a time limit of 3600 seconds.
- Set maximum number of threads to 1.
- Disable presolve. (Each instance is already presolved during preprocessing.)

- Set `PreCrush` to 1. (Required when adding user cuts.)

Cut generation limits. We only use one round of cuts with a generation limit of one hour, all cuts are rank one with respect to P, and VPCs are unstrengthened (in the sense of coefficient modularization).[5] Cut generation is abandoned when (PRLP^0) is infeasible or fails to solve to optimality within a minute when using no objective, i.e., just the feasibility problem. We generate at most as many VPCs as the number of integer variables that are fractional at $\bar{x}$, i.e., the same as the limit on the number of Gomory cuts. This is in order to enhance comparability, but we have no evidence that this is a good choice.

In summary, we do not vary the $\mathcal{V}$-polyhedral relaxation of each term, and we do not impose limits on cut orthogonality or maximum density. For one of our experiments, we double the cut limit and use two cut rounds, but these parameters merit further exploration; prior work has repeatedly demonstrated that appropriately choosing such values can mean the difference between an algorithm that works in practice and one that seems to produce negative results (see, e.g., [16, 33] and the discussion in Karamanov [41, Chapter 3]).

5.2 Percent root gap closed

Table 1 provides a summary of the average percent gap closed by GMICs, VPCs, and VPCs used together with GMICs, as well as the percent gap closed by one round of cuts at the root by Gurobi and after the last round of cuts added by Gurobi at the root. The extended manuscript [10] contains the values for all the instances.

In the tables, "G" refers to GMICs, "V" refers to VPCs, "max(G,V)" refers to the best result (per instance) between GMICs and VPCs, "GurF" refers to Gurobi after one round of cuts at the root, "GurL" refers to Gurobi after the last round of cuts at the root, and "DB" refers to the value of the disjunctive lower bound $c^{\intercal}p^*$ from the partial branch-and-bound tree for each instance with 64 leaf nodes (an upper bound on the gap we can close using VPCs on their own). Unless otherwise stated, the result shown for VPCs is the best across all partial tree sizes tested for that instance. A finer level of analysis in this regard is given in Appendix A.

Column 1 indicates which instances are being considered in the corresponding row: the first pair of rows concerns the 332 instances for which the disjunctive lower bound is strictly greater than the LP optimal value, the second pair of rows pertains to the subset of 116 instances for which VPCs close at least 10% of the integrality gap with respect to GMICs, while the third pair of rows reports on the subset of 65 pure binary instances. The first row for each set gives the average for the percent gap closed across the instances. The second row for each set shows the number of "wins", where wins for columns "DB", "V", and "V+G" are relative to column "G"; wins for "V+GurF" are counted with respect to column "GurF"; and wins for "V+GurL" are with respect to column "GurL". An instance counts as a win when at least 10^{-3} percent more integrality gap is closed compared to the appropriate reference column.

[5] Efficiently strengthening general disjunctive cuts currently poses nontrivial barriers [43].

Table 1: Summary statistics for percent gap closed by VPCs. The wins row reports how many instances close at least ϵ more gap when comparing DB, V, V+G to G on its own, V+GurF to GurF, and V+GurL to GurL.

Set	# inst	Metric	G	DB	V	max(G,V)	V+G	GurF	V+GurF	GurL	V+GurL
All	332	Avg (%)	16.51	18.37	12.01	23.01	23.92	28.93	34.80	48.32	52.81
		Wins		178	132		226		250		234
≥10%	116	Avg (%)	16.80	38.10	29.80	34.22	35.90	26.41	38.59	45.27	55.40
		Wins		100	92		112		101		107
Binary	65	Avg (%)	13.64	21.80	17.31	25.48	26.39	19.23	31.26	32.94	42.80
		Wins		50	37		53		48		48

Column 2 gives the number of instances in each set. Next is given the percent gap closed by GMICs when they are added to the LP relaxation (column 4); the disjunctive lower bound from the partial tree with 64 leaf nodes (column 5; VPCs (column 6); the maximum per instance between GMICs and VPCs (column 7); GMICs and VPCs used together (column 8). Columns 9 and 10 show the percent gap closed by Gurobi cuts from one round at the root, first without and then with VPCs added as user cuts. Columns 11 and 12 show the same, but after the last round of cuts at the root. The Gurobi-related columns use the average percent gap closed by Gurobi across the 7 random seeds tested.

The results indicate that VPCs are strong compared to existing cuts. Namely, using VPCs and GMICs together leads the average percent gap closed at the root to increase from 16.5% to 23.9%. VPCs on their own close strictly more gap than GMICs for 132 instances. In comparison, for 178 instances, the disjunctive lower bound is greater than the optimal value after adding GMICs, so there are only 46 additional instances for which VPCs on their own could have gotten stronger results. For 24 of those 46 instances, we achieve the cut limit, implying that a higher percent gap might be achieved if we permit more cuts to be generated. VPCs used with GMICs together outperform GMICs for 226 of the 332 instances. Of the 106 instances in which VPCs and GMICs combined do not improve over GMICs alone, for 6 instances, GMICs already close ~100% of the gap; and for 33 instances, VPCs and GMICs together close 0% of the gap. The cut limit is achieved for 24 of the 106 nonimproving instances.

Perhaps even more indicative of the strength of VPCs is when VPCs are used as user cuts within Gurobi, which may employ a variety of cut classes, not only GMICs. For the first round of cuts at the root, the percent gap closed goes from 28.9% (without VPCs) to 34.8% (with them), with strictly better outcomes for 250 of the 332 instances. For the last round of cuts at the root, the percent gap closed increases from 48.3% to 52.8% when using VPCs.

On average, VPCs without GMICs close less of the integrality gap than GMICs do on their own; this occurs for 165 of the 332 instances. We offer two plausible explanations for this phenomenon. First, for 60 of the 165 instances, we generate very few VPCs compared to GMICs, which makes it difficult to compare the two families directly. Another 61 of these instances hit the cut limit, so there is further potential for VPCs to improve. Second, in

Table 2: Average percent gap closed broken down by the number of leaf nodes used to construct the partial branch-and-bound tree, for VPCs with and without GMICs, as well as at the root by Gurobi after the first and last round of cuts. "Best" refers to the maximum gap closed across all partial tree sizes.

	DB	V	max(G,V)	V+G	V+GurF	V+GurL
VPCs disabled	0.00	0.00	16.51	16.51	28.93	48.32
2 leaves	3.01	2.21	16.79	17.21	30.96	49.26
4 leaves	5.27	3.59	17.23	17.79	31.16	49.25
8 leaves	8.12	4.85	17.83	18.53	31.53	49.48
16 leaves	11.17	6.46	19.37	20.03	32.47	50.24
32 leaves	14.67	8.74	20.96	21.77	33.41	51.14
64 leaves	18.37	10.23	22.14	22.89	33.83	51.64
Best	18.37	12.01	23.01	23.92	34.80	52.81
Combined	17.80	12.49	23.21	24.12	34.46	52.07
Rounds	18.43	15.28	25.56	26.31	35.21	52.68

these results, no *a posteriori* strengthening techniques, such as modularization, are applied to VPCs, while GMICs do take advantage of modularization.

VPCs and GMICs seem to be strong for different types of instances. One indication is from column "max(G,V)": using the best result of only GMICs or only VPCs, per instance, the percent gap closed is 23%, only 1% less than combining both families together. Further evidence comes from the "$\geq$10%" set of instances. VPCs and GMICs together close over double the percent gap closed by GMICs alone, with improvements for 112 of the 116 instances in this set, and VPCs provide a 22% improvement in the gap closed after the last round of cuts at the root node of Gurobi (55.4% compared to 45.3%). VPCs also outperform GMICs on pure binary instances, offering 30% improvement in average root percent gap closed for Gurobi.

For the columns including VPCs, the result reported is the maximum percent gap closed across all partial tree sizes tested. One may initially assume that the strongest cuts would always come from the partial tree with 64 leaf nodes. This is indeed true for the disjunctive lower bound, but it does not always hold for VPCs for particular instances, though the *average* gap closed across instances steadily increases. One reason, meaningful in conjunction with the fact that we generate a fixed number of cuts, is that there are likely to be more facet-defining (i.e., essential) inequalities for the disjunctive hull from stronger disjunctions. As a result, achieving the disjunctive lower bound may become more difficult, in particular given our relatively conservative cut limit. Another reason, on an intuitive level, is that more of the facet-defining inequalities for the deeper disjunctions may not cut away $\bar{x}$, which are cuts we do not generate in these experiments. Finally, the rate of numerical issues goes up as the disjunctions get larger; we investigate this more in Appendix B.

Table 2 shows how the average percent gap closed increases with disjunction size. In this table, row "Best" corresponds to the same values as in Table 1, i.e., the best value per instance is used across all partial tree sizes tested. We show the same metrics as in Table 1.

In this table, VPCs close more gap as stronger disjunctions are used. However, we also see that there is much room for improvement in our implementation of the framework, as the gap closed by VPCs grows increasingly farther from the disjunctive lower bound with the use of stronger disjunctions.

We also report results from two additional experiments in Table 2. The penultimate row, "Combined", is obtained by applying all VPCs, across all disjunction sizes, though still constrained to an overall one-hour cut generation limit, which is why the value in column "DB" is smaller than the corresponding value from the preceding row "Best". The last row, "Rounds", reports the outcome of using "Combined" for two cut rounds, with doubling the cut limit per disjunction and per round. The "Rounds" setting only contains results for 322 instances, so it cannot be exactly compared to the 332 of the preceding rows. Modulo this caveat, it can be seen that the percent gap closed by VPCs alone increases from 12.5% to 15.3%, but the impact on Gurobi in column "V+GurL" appears to be relatively minor. For comparison, two rounds of GMICs correspondingly increases from 16.5% to 22.3%.

An important conclusion from Table 2 is that our procedure may help in avoiding the "tailing-off" effect from recursive applications of cuts: without requiring recursion, by simply using a (sufficiently) stronger disjunction, we make relatively steady progress toward the optimal value of (IP). However, this is only in terms of percent gap closed; as we discuss in the next section, the story when using the cuts within branch and bound is completely different, in which seemingly weaker cuts may lead to better performance.

5.3 Effect with branch and bound

We now turn to the second metric: the effect of our cuts on branch and bound in terms of time and number of nodes. We compare two solvers: "Gur" is the baseline of default Gurobi, and "V" denotes Gurobi with VPCs added as user cuts. Table 3 contains a summary of the statistics for several instance sets, called "Combined", "Rounds", and "ℓ leaves" for $\ell \in \{2, 4, 8, 16, 32, 64\}$, where

- "Combined" refers to all VPCs generated within an hour across all disjunction sizes considered;
- "Rounds" refers to two rounds of "Combined" VPC generation with a cut limit of twice the number of fractional integer variables in $\bar{x}$; and
- "ℓ leaves" refers to terminating partial branch-and-bound tree generation when there are ℓ open (leaf) nodes.

Each instance is solved with 7 random seeds per solver. For each seed that times out, for solving time, 7200 seconds (twice the time limit) is used, and for calculating nodes, the number of processed nodes is also multiplied by 2. Within each set, we create four bins, where bin [t,3600) contains the subset of the 332 instances used in Section 5.2 for which the average solution time (across 7 random seeds) is at least t seconds for both solvers and under 3600 seconds for at least one solver.

The first column of Table 3 indicates which set and bin is being considered. The second column is the number of instances in that subset. The next column indicates the two

summary statistics presented for each subset. The first statistic row, "Gmean", for each subset is a shifted geometric mean, with a shift of 1 for time and 1000 for nodes, modified from Achterberg [3]. The second row, "Wins", reports the number of instances for which each of the two solver options "Gur" and "V" perform better, where an instance counts as a win by time for a solver if the other solver has an average running time that is at least 10% slower, and an instance counts as a win by nodes for a solver when the number of nodes is strictly better. The remaining columns contain the corresponding statistics for each solver and metric, as well as column "Gen" that gives the geometric mean of cut generation time for instances within each set and bin.

From Table 3, we conclude that adding VPCs tends to slightly degrade the running time of Gurobi, even though the number of processed nodes frequently decreases. For both the "Combined" and "Rounds" sets, the default "Gur" solver runs faster, especially when considering the running time reported in column "Gen", though there is a reduction in nodes for the $[1000, 3600)$ bucket. In the remaining sets, there are isolated cases of promise. For example, under "4 leaves", we see that the geometric mean of the number of processed nodes decreases for the first three buckets when using VPCs, though the corresponding geometric mean running times are about the same for both solvers. There is a small improvement in running time for the "harder" instances, in the $[1000, 3600)$ bucket, for several parameter settings: "2 leaves", "8 leaves", and "16 leaves" (even when considering cut generation time). We caution that, despite our best efforts, we might not have completely removed confounding factors such as machine variability, which may cause inconsistent results when repeating the experiments. Nevertheless, these results indicate that there is a significant portion of instances from the dataset that may benefit from VPCs. At the same time, it is not clear how to identify such instances, effectively select VPC hyperparameters, or incorporate VPCs within Gurobi in general.

Table 3: Summary statistics when solving instances with branch and bound.

			Time (s)			Nodes (#)	
Set	# inst	Metric	Gur	V	Gen	Gur	V
Combined [0,3600)	288	Gmean	11.26	11.84	27.62	7,682	7,794
		Wins	117	62		128	131
Combined [10,3600)	119	Gmean	157.26	168.45	55.30	75,194	76,956
		Wins	44	32		54	65
Combined [100,3600)	69	Gmean	505.12	550.56	43.70	244,802	255,028
		Wins	32	15		38	31
Combined [1000,3600)	24	Gmean	1,972.95	2,073.90	35.86	1,008,340	996,885
		Wins	10	5		11	13
Rounds [0,3600)	279	Gmean	11.62	12.47	128.88	7,846	7,920
		Wins	144	61		129	123
Rounds [10,3600)	117	Gmean	155.23	165.44	216.40	74,002	74,706
		Wins	45	34		58	59
Rounds [100,3600)	64	Gmean	526.97	558.17	193.58	235,024	241,115
		Wins	22	16		36	28

Set	# inst	Metric	Time (s)			Nodes (#)	
			Gur	V	Gen	Gur	V
Rounds [1000,3600)	23	Gmean	1,868.61	1,982.47	166.96	1,041,128	1,032,614
		Wins	8	5		13	10
2 leaves [0,3600)	261	Gmean	9.26	9.51	0.56	6,310	6,453
		Wins	64	52		102	135
2 leaves [10,3600)	102	Gmean	129.68	139.35	1.19	59,631	64,032
		Wins	33	22		44	58
2 leaves [100,3600)	50	Gmean	503.50	534.57	1.00	186,172	199,676
		Wins	13	10		18	32
2 leaves [1000,3600)	15	Gmean	1,974.56	1,894.64	0.30	799,032	729,535
		Wins	4	4		4	11
4 leaves [0,3600)	266	Gmean	8.81	8.80	1.19	6,017	5,963
		Wins	70	55		112	127
4 leaves [10,3600)	104	Gmean	117.38	115.83	2.38	55,206	53,956
		Wins	26	29		49	55
4 leaves [100,3600)	48	Gmean	466.63	458.66	2.56	166,328	162,475
		Wins	12	12		19	29
4 leaves [1000,3600)	13	Gmean	1,802.02	1,875.01	0.96	790,760	792,726
		Wins	4	4		7	6
8 leaves [0,3600)	259	Gmean	9.19	9.68	2.21	6,718	7,035
		Wins	89	46		127	107
8 leaves [10,3600)	99	Gmean	139.39	153.05	4.72	74,523	82,416
		Wins	42	24		58	41
8 leaves [100,3600)	52	Gmean	510.32	557.75	5.16	213,765	241,450
		Wins	24	13		29	23
8 leaves [1000,3600)	16	Gmean	2,046.17	1,965.60	2.64	750,097	709,701
		Wins	7	6		7	9
16 leaves [0,3600)	261	Gmean	9.22	9.46	3.91	6,965	7,020
		Wins	78	63		104	133
16 leaves [10,3600)	100	Gmean	139.44	147.97	6.66	80,058	83,603
		Wins	30	28		45	55
16 leaves [100,3600)	50	Gmean	535.23	562.07	7.83	226,992	236,018
		Wins	15	14		21	29
16 leaves [1000,3600)	18	Gmean	1,891.26	1,824.11	4.82	995,350	889,380
		Wins	5	7		4	14
32 leaves [0,3600)	246	Gmean	8.35	8.66	7.33	6,474	6,411
		Wins	85	53		104	117
32 leaves [10,3600)	91	Gmean	128.97	136.26	13.09	78,583	77,146
		Wins	30	26		35	56
32 leaves [100,3600)	43	Gmean	519.65	567.70	12.13	259,502	261,929

Set	# inst	Metric	Time (s)			Nodes (#)	
			Gur	V	Gen	Gur	V
32 leaves [1000,3600)	14	Wins	14	10		15	28
		Gmean	1,876.72	1,940.80	10.71	613,962	547,304
		Wins	6	5		3	11
64 leaves [0,3600)	229	Gmean	7.88	8.35	10.39	6,674	6,817
		Wins	90	47		107	99
64 leaves [10,3600)	82	Gmean	135.62	150.60	18.13	92,630	96,976
		Wins	35	18		42	40
64 leaves [100,3600)	41	Gmean	483.71	558.89	22.10	240,939	257,322
		Wins	14	10		20	21
64 leaves [1000,3600)	12	Gmean	1,804.24	2,254.48	20.51	546,739	549,839
		Wins	6	2		6	6

6 Conclusion & open problems

This paper presents a step toward merging cut-generation and branching in integer programming solvers by providing a computationally tractable method for generating cuts from partial branch-and-bound trees. The framework we introduce is to (1) select a disjunction, (2) choose a (compact) $\mathcal{V}$-polyhedral relaxation for each disjunctive term, and (3) selectively generate cuts by judiciously choosing objective directions to optimize over (PRLP) formed from the point-ray collection.

Our investigation touches on each of these aspects. The first is the disjunction choice, which in our experiments is the set of leaf nodes of a partial branch-and-bound tree. We only experiment with the size of the disjunction, but we do not extensively test alternative disjunctions or claim to prescribe the best (or even good) disjunctions for the purposes of cut generation, which is left as an open problem for future work. As discussed in Appendix G of the extended manuscript [10], just one sufficiently strong disjunction chosen this way can lead to cuts that are stronger than those from all possible split and cross disjunctions. Although using stronger disjunctions does lead to better gap closed, our results indicate that this monotonicity does not hold when embedding the cuts within branch and bound in a solver. A better understanding of the interaction between the branch-and-bound process and cutting planes remains an open problem meriting future research [29, 45].

The relaxation for each disjunctive term that we use is quite simple, but therein lies its advantage. Nevertheless, we do show examples highlighting the weakness of our simple relaxations—that only a subset of all the valid disjunctive cuts can be produced—and the computational results do, at times, reflect this weakness. Thus, there is an opportunity to improve the quality of the generated VPCs by considering tighter relaxations generated from structural information about each instance.

For the objective directions, we provide theoretical support for objective directions to (PRLP) that yield new and strong VPCs more frequently than previous approaches (reducing the percent of objectives failing to produce a cut from 80% in early experiments, to around

30% in the current implementation).

Overall, the cuts we generate are strong, as evidenced by the percent integrality gap they close in our experiments (compared to both GMICs and the default cut setting of Gurobi). Moreover, the integrality gap closed by VPCs increases steadily with the use of stronger disjunctions, which may help to avoid the common tailing off of strength experienced by other cut families that require recursive applications to reach strong cuts. In addition, for some instances, our results show that the extra computational effort pays off in reduced branch-and-bound time, but on average across the instances tested, employing VPCs causes slower performance. We conclude that, though VPCs may not yet improve most solvers, the VPC framework has theoretical advantages with the potential for practical impact on cut generation.

Acknowledgments. This work was supported in part by NSF grant CMMI1560828 and ONR contract N00014-15-12082. We also greatly appreciate the reviewers' valuable feedback.

Conflicts of interest. The authors have no conflicts of interest to declare.

References

[1] COIN-OR Branch and Cut. `https://github.com/coin-or/Cbc`.

[2] COIN-OR Linear Programming. `https://github.com/coin-or/Clp`.

[3] Tobias Achterberg. Conflict analysis in mixed integer programming. *Discrete Optim.*, 4(1):4–20, 2007. URL `https://doi.org/10.1016/j.disopt.2006.10.006`.

[4] Tobias Achterberg, Thorsten Koch, and Alexander Martin. MIPLIB 2003. *Oper. Res. Lett.*, 34(4):361–372, 2006. URL `http://dx.doi.org/10.1016/j.orl.2005.07.009`.

[5] Kent Andersen, Gérard Cornuéjols, and Yanjun Li. Split closure and intersection cuts. *Math. Program.*, 102(3, Ser. A):457–493, 2005. URL `http://dx.doi.org/10.1007/s10107-004-0558-z`.

[6] David Applegate, Robert Bixby, Vašek Chvátal, and William Cook. Finding cuts in the TSP (a preliminary report). Technical report, Center for Discrete Mathematics & Theoretical Computer Science, 1995.

[7] Egon Balas. Disjunctive programming. *Ann. Discrete Math.*, 5:3–51, 1979. URL `https://doi.org/10.1016/S0167-5060(08)70342-X`.

[8] Egon Balas and Pierre Bonami. Generating lift-and-project cuts from the LP simplex tableau: Open source implementation and testing of new variants. *Math. Program. Comput.*, 1(2-3):165–199, 2009. URL `http://dx.doi.org/10.1007/s12532-009-0006-4`.

[9] Egon Balas and Robert G. Jeroslow. Strengthening cuts for mixed integer programs. *European J. Oper. Res.*, 4(4):224–234, 1980. URL `https://doi.org/10.1016/0377-2217(80)90106-X`.

[10] Egon Balas and Aleksandr M. Kazachkov. $\mathcal{V}$-polyhedral disjunctive cuts, 2022. URL https://arxiv.org/abs/2207.13619. Working paper.

[11] Egon Balas and Tamás Kis. On the relationship between standard intersection cuts, lift-and-project cuts and generalized intersection cuts. *Math. Program.*, pages 1–30, 2016. URL https://doi.org/10.1007/s10107-015-0975-1.

[12] Egon Balas and François Margot. Generalized intersection cuts and a new cut generating paradigm. *Math. Program.*, 137(1-2, Ser. A):19–35, 2013. URL http://dx.doi.org/10.1007/s10107-011-0483-x.

[13] Egon Balas and Michael Perregaard. Lift-and-project for mixed 0-1 programming: Recent progress. *Discrete Appl. Math.*, 123(1-3):129–154, 2002. URL http://dx.doi.org/10.1016/S0166-218X(01)00340-7. Workshop on Discrete Optimization, DO'99 (Piscataway, NJ).

[14] Egon Balas and Michael Perregaard. A precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer Gomory cuts for 0-1 programming. *Math. Program.*, 94(2-3, Ser. B):221–245, 2003. URL http://dx.doi.org/10.1007/s10107-002-0317-y. The Aussois 2000 Workshop in Combinatorial Optimization.

[15] Egon Balas, Sebastián Ceria, and Gérard Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Math. Program.*, 58(3, Ser. A):295–324, 1993. URL http://dx.doi.org/10.1007/BF01581273.

[16] Egon Balas, Sebastián Ceria, and Gérard Cornuéjols. Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. *Man. Sci.*, 42(9):1229–1246, 1996. URL http://mansci.journal.informs.org/content/42/9/1229.abstract.

[17] Egon Balas, Matteo Fischetti, and Arrigo Zanette. On the enumerative nature of Gomory's dual cutting plane method. *Math. Program.*, 125(2, Ser. B):325–351, 2010. URL http://dx.doi.org/10.1007/s10107-010-0392-4.

[18] David E. Bell and Marshall L. Fisher. Improved integer programming bounds using intersections of corner polyhedra. *Math. Program.*, 8:345–368, 1975. URL https://doi.org/10.1007/BF01580451.

[19] R. E. Bixby, E. A. Boyd, and R. R. Indovina. MIPLIB: A test set of mixed integer programming problems. *SIAM News*, 25:16, 1992.

[20] R. E. Bixby, S. Ceria, C. M. McZeal, and M. W. P Savelsbergh. An updated mixed integer programming library: MIPLIB 3.0. *Optima*, 58:12–15, 6 1998.

[21] Pierre Bonami. On optimizing over lift-and-project closures. *Math. Program. Comput.*, 4(2):151–179, 2012. URL http://dx.doi.org/10.1007/s12532-012-0037-0.

[22] Christoph Buchheim, Frauke Liers, and Marcus Oswald. Local cuts revisited. *Oper. Res. Lett.*, 36(4):430–433, 2008. URL http://dx.doi.org/10.1016/j.orl.2008.01.004.

[23] Florent Cadoux and Claude Lemaréchal. Reflections on generating (disjunctive) cuts. *EURO Journal on Computational Optimization*, 1(1-2):51–69, 2013. URL http://dx.doi.org/10.1007/s13675-012-0006-4.

[24] Binyuan Chen, Simge Küçükyavuz, and Suvrajeet Sen. Finite disjunctive programming characterizations for general mixed-integer linear programs. *Oper. Res.*, 59(1):202–210, 2011. URL https://doi.org/10.1287/opre.1100.0882.

[25] Binyuan Chen, Simge Küçükyavuz, and Suvrajeet Sen. A computational study of the cutting plane tree algorithm for general mixed-integer linear programs. *Oper. Res. Lett.*, 40(1):15–19, 2012. URL https://doi.org/10.1016/j.orl.2011.10.009.

[26] Rui Chen and James Luedtke. Sparse multi-term disjunctive cuts for the epigraph of a function of binary variables. *Math. Program.*, 2023. URL https://doi.org/10.1007/s10107-023-02019-2.

[27] Vašek Chvátal, William Cook, and Daniel Espinoza. Local cuts for mixed-integer programming. *Math. Program. Comput.*, 5(2):171–200, 2013. URL http://dx.doi.org/10.1007/s12532-013-0052-9.

[28] Michele Conforti and Laurence A. Wolsey. "Facet" separation with one linear program. *Math. Program.*, 178(1):361–380, 2019. URL https://doi.org/10.1007/s10107-018-1299-8.

[29] Claudio Contardo, Andrea Lodi, and Andrea Tramontani. Cutting planes from the branch-and-bound tree: Challenges and opportunities. *INFORMS J. Comput.*, 35(1):2–4, 2022. URL https://doi.org/10.1287/ijoc.2022.1248.

[30] William Cook, Sanjeeb Dash, Ricardo Fukasawa, and Marcos Goycoolea. Numerically safe Gomory mixed-integer cuts. *INFORMS J. Comput.*, 21(4):641–649, 2009. URL https://doi.org/10.1287/ijoc.1090.0324.

[31] CORAL. Computational Optimization Research at Lehigh. MIP instances. http://coral.ise.lehigh.edu/data-sets/mixed-integer-instances/, 2020. Accessed September 2020.

[32] Sanjeeb Dash, Oktay Günlük, and Juan Pablo Vielma. Computational experiments with cross and crooked cross cuts. *INFORMS J. Comput.*, 26(4):780–797, 2014. URL http://dx.doi.org/10.1287/ijoc.2014.0598.

[33] Michael C. Ferris, Gábor Pataki, and Stefan Schmieta. Solving the *seymour* problem. *Optima*, 66:2–6, 2001.

[34] Matteo Fischetti, Andrea Lodi, and Andrea Tramontani. On the separation of disjunctive cuts. *Math. Program.*, 128(1-2, Ser. A):205–230, 2011. URL http://dx.doi.org/10.1007/s10107-009-0300-y.

[35] D. R. Fulkerson, G. L. Nemhauser, and L. E. Trotter. Two computationally difficult set covering problems that arise in computing the 1-width of incidence matrices of Steiner triple systems. *Math. Program. Stud.*, 2:72–81, 1974. URL http://dx.doi.org/10.1007/BFb0120689.

[36] Dinakar Gade, Simge Küçükyavuz, and Suvrajeet Sen. Decomposition algorithms with parametric Gomory cuts for two-stage stochastic integer programs. *Math. Program.*, 144(1-2, Ser. A):39–64, 2014. URL https://doi.org/10.1007/s10107-012-0615-y.

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

[38] Ralph E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bull. Amer. Math. Soc.*, 64:275–278, 1958.

[39] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023. URL https://www.gurobi.com. Version 10.0.3.

[40] Markus Jörg. *k-disjunctive cuts and cutting plane algorithms for general mixed integer linear programs*. PhD thesis, Technischen Universität München, 8 2008.

[41] Miroslav Karamanov. *Branch and cut: an empirical study*. PhD thesis, Carnegie Mellon University, 9 2006.

[42] Aleksandr M. Kazachkov. *Non-Recursive Cut Generation*. PhD thesis, Carnegie Mellon University, 2018. URL https://kilthub.cmu.edu/articles/thesis/Non-Recursive_Cut_Generation/6720881.

[43] Aleksandr M. Kazachkov and Egon Balas. Monoidal strengthening of simple $\mathcal{V}$-polyhedral disjunctive cuts. In Alberto Del Pia and Volker Kaibel, editors, *Integer Programming and Combinatorial Optimization*, Lecture Notes in Comput. Sci., pages 275–290, Cham, 2023. Springer. URL https://doi.org/10.1007/978-3-031-32726-1_20.

[44] Aleksandr M. Kazachkov, Selvaprabu Nadarajah, Egon Balas, and François Margot. Partial hyperplane activation for generalized intersection cuts. *Math. Program. Comp.*, 12(1):69–107, 3 2020. URL https://doi.org/10.1007/s12532-019-00166-2.

[45] Aleksandr M. Kazachkov, Pierre Le Bodic, and Sriram Sankaranarayanan. An abstract model for branch and cut. *Math. Program.*, 2023. URL https://doi.org/10.1007/s10107-023-01991-z.

[46] Thorsten Koch, Tobias Achterberg, Erling Andersen, Oliver Bastert, Timo Berthold, Robert E. Bixby, Emilie Danna, Gerald Gamrath, Ambros M. Gleixner, Stefan Heinz, Andrea Lodi, Hans Mittelmann, Ted Ralphs, Domenico Salvagnin, Daniel E. Steffy,

and Kati Wolter. MIPLIB 2010: mixed integer programming library version 5. *Math. Program. Comput.*, 3(2):103–163, 2011. URL http://dx.doi.org/10.1007/s12532-011-0025-9.

[47] Andrea Lodi, Mathieu Tanneau, and Juan-Pablo Vielma. Disjunctive cuts in mixed-integer conic optimization. *Math. Program.*, 199(1-2):671–719, 2023. URL https://doi.org/10.1007/s10107-022-01844-1.

[48] Robin Lougee-Heimer. The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development*, 47, 2003.

[49] Quentin Louveaux, Laurent Poirrier, and Domenico Salvagnin. The strength of multi-row models. *Math. Program. Comput.*, 7(2):113–148, 2015. URL http://dx.doi.org/10.1007/s12532-014-0076-9.

[50] Matthias Miltenberger, Ted Ralphs, and Daniel E. Steffy. Exploring the numerics of branch-and-cut for mixed integer linear optimization. In Natalia Kliewer, Jan Fabian Ehmke, and Ralf Borndörfer, editors, *Operations Research Proceedings 2017*, pages 151–157, Cham, 2018. Springer International Publishing. URL https://doi.org/10.1007/978-3-319-89920-6_21.

[51] Lewis Ntaimo. Disjunctive decomposition for two-stage stochastic mixed-binary programs with random recourse. *Oper. Res.*, 58(1):229–243, 2010. URL https://doi.org/10.1287/opre.1090.0693.

[52] Lewis Ntaimo and Matthew W. Tanner. Computations with disjunctive cuts for two-stage stochastic mixed 0-1 integer programs. *J. Global Optim.*, 41(3):365–384, 2008. URL https://doi.org/10.1007/s10898-007-9245-y.

[53] Michael Perregaard and Egon Balas. Generating cuts from multiple-term disjunctions. In *Integer Programming and Combinatorial Optimization*, volume 2081 of *Lecture Notes in Comput. Sci.*, pages 348–360. Springer, Berlin, 2001. URL http://dx.doi.org/10.1007/3-540-45535-3_27.

[54] Yunwei Qi and Suvrajeet Sen. The ancestral Benders' cutting plane algorithm with multi-term disjunctions for mixed-integer recourse decisions in stochastic programming. *Math. Program.*, 161(1-2, Ser. A):193–235, 2017. URL https://doi.org/10.1007/s10107-016-1006-6.

[55] S. Sen and H. D. Sherali. Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Math. Program.*, 106(2):203–223, 2006. URL https://doi.org/10.1007/s10107-005-0592-5.

[56] Thiago Serra. Reformulating the disjunctive cut generating linear program. *Ann. Oper. Res.*, 295(1):363–384, 2020. URL https://doi.org/10.1007/s10479-020-03709-2.

[57] Hanif D. Sherali and J. Cole Smith. Higher-level RLT or disjunctive cuts based on a partial enumeration strategy for 0-1 mixed-integer programs. *Optim. Lett.*, 6(1):127–139, 2012. URL https://doi.org/10.1007/s11590-010-0255-1.

[58] Yang Yuan and Suvrajeet Sen. Enhanced cut generation methods for decomposition-based branch and cut for two-stage stochastic mixed-integer programs. *INFORMS J. Comput.*, 21(3):480–487, 2009. URL https://doi.org/10.1287/ijoc.1080.0300.

[59] Arrigo Zanette, Matteo Fischetti, and Egon Balas. Lexicography and degeneracy: can a pure cutting plane algorithm work? *Math. Program.*, 130(1, Ser. A):153–176, 2011. URL http://dx.doi.org/10.1007/s10107-009-0335-0.

A Analysis of effect of disjunction size

One aspect that is hidden in the results of Section 5 is the number of leaf nodes used for the partial branch-and-bound tree to obtain the reported gap closed and branch-and-bound times. Table 1 reports the best result across all the tested sizes of the partial tree, i.e., with number of leaf nodes $\ell \in \{2, 4, 8, 16, 32, 64\}$. We next disaggregate the analysis to see how the different size partial trees perform alone. For data including VPCs, a partial tree size may be specified (either as "V (ℓ)" or "ℓ leaves"), indicating that these results concern only the runs for partial trees with ℓ leaf nodes.

Table 4 gives the number of instances for which the best result over the appropriate baseline is achieved when restricted to a particular number of leaves. We use the same tolerances to count wins as in Section 5. The first column is the tree size, including the option of no VPCs, the second through fifth columns refer to gap closed for the 332 instances used in the strength experiments, while the last columns give solving time and nodes for the 282 instances solved within 3600 seconds on average for either the default Gurobi solver or for Gurobi with VPCs for at least one setting for ℓ. We see, as in Table 2, that the tree with the most leaf nodes produces the best percent gap closed quite often, but not always; there are even instances for which $\ell = 2$, i.e., a single split disjunction, suffices to achieve the best result. In contrast, there is no clear winner in terms of branch-and-bound metrics.

This phenomenon of stronger cuts not being directly correlated to better branch and bound performance is not easy to remedy, due to the hard-to-predict effect of cuts on the branch-and-bound process. Nevertheless, there is one aspect of Table 3 that suggests a possible explanation. Observe that the number of wins with VPCs in terms of number of nodes is typically higher than in terms of time. For each instance in which the number of nodes decreases but the time increases, solving the relaxation at each node of the branch-and-bound tree might be slower with additional cuts added. It is commonly known that making the coefficient matrix denser will slow down the solution of a linear program. We next look at how the density of VPCs changes as deeper disjunctions are used.

Table 5 gives the average cut density statistics across the different partial tree sizes for the set "All". The density of a cut is defined as the number of nonzero cut coefficients divided by the total number of coefficients. The first row is the number of instances having VPCs for each tree size. The second row is the number of instances where "V" wins on time compared

Table 4: Number of leaf nodes yielding the best result for each experiment per instance.

	Gap				Time	Nodes
	V	V+G	V+GurF	V+GurL	All	All
No improvement	200	106	82	98	178	79
2 leaves	0	10	53	39	27	38
4 leaves	2	8	65	42	37	37
8 leaves	5	20	60	37	23	35
16 leaves	4	18	79	44	25	41
32 leaves	16	37	72	44	32	40
64 leaves	105	143	122	96	27	38

Table 5: Statistics about the density of generated cuts broken down by partial tree size.

	V (2)	V (4)	V (8)	V (16)	V (32)	V (64)
# inst w/VPCs and time < 3600s	261	266	259	261	246	229
# wins by time	52	55	46	63	53	47
Avg min cut density	0.217	0.236	0.274	0.301	0.349	0.392
Avg max cut density	0.344	0.383	0.405	0.427	0.484	0.488
Avg avg cut density	0.280	0.311	0.343	0.373	0.427	0.453
Avg avg cut density (win by time)	0.290	0.298	0.213	0.277	0.361	0.399
Avg avg cut density (lose by time)	0.279	0.311	0.347	0.398	0.487	0.501

to "Gur". The next three rows give the average of the minimum, maximum, and average densities of VPCs for each instance. The last two rows give the average of the average cut densities for (1) the instances counted in the second row, i.e., those where VPCs improve time with respect to "Gur", and (2) the instances for which "Gur" wins over "V".

The first observation is that the number of instances with VPCs decreases from 261 for the setting with 2 leaf nodes to 229 for the 64 leaf node setting. The average cut density goes from 0.280 to 0.453 on average, which could help explain the fact that the stronger cuts from stronger disjunctions can yield worse branch and bound times. In addition, the last two rows suggest that density is correlated to whether VPCs help for an instance. Namely, for each column, the average cut density is always lower for those instances that win by time. Future experiments may benefit from doing cut filtering by density, or possibly reducing the density of generated cuts *a posteriori* while sacrificing strength.

B Analysis of objective function choices

Next, via Table 6, we discuss our objective function choices, including statistics on the frequency of failures as a function of disjunction size. A caveat is that our analysis of objectives functions is somewhat limited by our relatively conservative strategy in selecting objectives to use, in order to limit time spent on cut generation and the types of failures discussed in Section 4. The objectives we choose utilize the results of Kazachkov et al. [44], suggesting that a successful class of objectives is the set of points and rays of the point-ray collection, but, motivated by Proposition 10, we only do this for cuts tight at p^* (the loop starting at step 14 of Algorithm 1). It is natural to consider cuts that only lie on deeper points, but this, as well as cuts that are satisfied by $\bar{x}$, remains a topic for future research.

Table 6 summarizes objective failures using all 332 instances from the gap closed experiments. The columns of the table are the same as Table 5. The rows are divided into several blocks. The first block gives statistics on the number of instances for each column for which: (1) objectives were tried, (2) VPCs were generated, (3) objectives were not tried, (4) none of the objectives yielded VPCs, and (5) all of the objectives led to distinct VPCs. The next block gives the average percent of the objectives that were failures. The subsequent block of rows looks at the cause of these failures, which fall into one of four categories:

- "Dup": the optimal solution to the PRLP is an exact duplicate of an existing cut

- "Unbdd": (PRLP) does not have a finite solution for that objective
- "Tilim": the time limit for (PRLP) is attained for that objective
- "Dyn": the dynamism of the new cut is too high

The following block of rows looks at the percent of failures for each class of objectives: "all ones" ($w = e$), "post-GMIC" (step 8 of Algorithm 1), and "DB" (step 14 of Algorithm 1). The last block of rows looks at the average number of objectives required to generate each distinct VPC, as well as the average number of seconds taken per objective and per cut.

This table shows that failures become more frequent when using disjunctions with more terms, with average failure rate increasing from 29% to 34%. The primary reason for this is that more objectives lead to cuts that were previously generated, leading to, on average, up to 57% of the failures. The cause is that there are more unsuccessful objectives being tried for the "DB" class of objectives. The last set of rows of the table also show that cut generation can be extremely costly for stronger disjunctions, which could be mitigated by reducing the failure rate. Table 8 provides objective statistics just for the best run per instance (across all partial tree sizes).

Lastly, we look at which classes of objective functions are more likely to lead to active cuts (after the addition of all cuts), as a different measure of the effect of our cuts. Table 7 gives averages for which cuts are active at the optimal solution after adding both GMICs and VPCs to P, for GMICs and VPCs, as well as individually based on the objective producing each VPC. The first row is the percent of GMICs that are active. The second row is the percent of VPCs that are active, averaged across those instances per each column for which VPCs were generated. The next rows come in pairs and give the average percent of cuts that come from the four subclasses of VPCs within our procedure, as well as the average percent of these cuts that are active (across instances for which there exist cuts from that class). The first class concerns a set of (rarely encountered) cuts that is added in our procedure, which we call "one-sided". In the process of generating VPCs, while selecting variables for strong branching, we occasionally detect that one of the two possible branches is infeasible. In this case, for a variable x_k, $k \in \mathcal{I}$, we generate the "one-sided cut" $x_k \leq \lfloor \bar{x}_k \rfloor$ or $x_k \geq \lceil \bar{x}_k \rceil$. These cuts are generated for 5 of the 332 instances, with a total of only 6 cuts.

Table 7 shows that the proportion of active VPCs somewhat increases while the percent of active GMICs decreases as VPCs from stronger disjunctions are used. Aside from the one-sided cuts, which are always active in our results, the objectives "all ones" and "post-GMIC opt" lead to cuts that are frequently active, though these objectives yield at most two cuts in total per instance. Though the "DB" class of objectives leads to a smaller percentage of active cuts, it is the source for the majority of the cuts that we generate.

Table 6: Statistics about objectives leading to failures, by partial tree size.

	V (2)	V (4)	V (8)	V (16)	V (32)	V (64)
# inst w/obj	311	318	304	306	289	271
# inst w/succ obj	308	315	300	302	285	263
# inst no obj	21	14	28	26	43	61
# inst all obj fail	3	3	4	4	4	8
# inst all obj succ	30	41	31	34	27	24
% obj fails	28.57	25.28	26.50	30.77	31.83	34.08
% fails dup	50.67	39.34	41.11	48.78	51.02	56.85
% fails unbdd	38.74	47.59	45.41	35.74	32.21	27.50
% fails tilim	1.38	3.67	3.61	5.13	6.27	5.91
% fails dyn	9.08	8.74	9.61	9.42	9.31	9.38
% fails all ones	26.75	30.73	25.85	20.69	17.44	14.17
% fails post-GMIC obj	14.59	20.45	22.88	21.25	20.51	17.38
% fails DB	58.66	48.82	51.27	58.07	62.06	68.45
# obj / cut	2.57	2.31	2.38	2.76	2.97	3.44
(s) / obj	0.29	4.42	0.77	0.92	1.93	3.02
(s) / cut	0.36	1.68	6.32	4.98	11.67	22.38

Table 7: Statistics about when generated cuts are active, broken down by partial tree size.

	V+G (2)	V+G (4)	V+G (8)	V+G (16)	V+G (32)	V+G (64)
% active GMIC	44.09	43.52	42.17	41.60	40.88	40.68
% active VPC	30.20	30.59	31.90	35.50	34.48	33.40
% cuts one-sided	0.77	0.72	0.74	1.11	0.82	0.79
% active one-sided	100.00	100.00	100.00	100.00	100.00	100.00
% cuts all ones	11.82	6.72	7.81	9.10	8.40	9.00
% active all ones	91.45	84.78	81.82	77.02	79.63	79.27
% cuts post-GMIC opt	2.62	3.40	2.42	2.19	1.70	2.74
% active post-GMIC opt	85.90	71.43	67.92	62.75	63.83	61.90
% cuts DB	84.79	89.16	89.03	87.61	89.08	87.47
% active DB	63.75	58.32	56.36	51.85	49.31	43.00

Table 8: Information about objectives and time to generate cuts corresponding to the parameters leading to the best gap closed per instance, as reported in Table 10.

	Objectives				Time (s)		
Instance	Obj	Succ	Fails	% fails	Total	(s)/obj	(s)/cut
10teams	214	16	198	92.5	3,604.6	16.8	225.3
23588	76	75	1	1.3	28.6	0.4	0.4
30n20b8	193	190	3	1.6	40.0	0.2	0.2
50v-10	30	29	1	3.3	4.5	0.1	0.2
a1c1s1	3	3	0	0.0	0.5	0.2	0.2
a2c1s1	20	18	2	10.0	0.2	0.0	0.0
aflow30a	28	25	3	10.7	6.4	0.2	0.3
aflow40b	38	36	2	5.3	82.0	2.2	2.3
aligninq	164	163	1	0.6	3,616.4	22.1	22.2
app3	38	15	23	60.5	18.9	0.5	1.3
arki001	8	5	3	37.5	0.3	0.0	0.1
assign1-5-8	229	114	115	50.2	11.6	0.1	0.1
b1c1s1	9	9	0	0.0	3.3	0.4	0.4
b2c1s1	4	4	0	0.0	25.4	6.3	6.3
bc1	5	5	0	0.0	45.2	9.0	9.0
bc	17	16	1	5.9	51.7	3.0	3.2
beasleyC1	17	17	0	0.0	2.2	0.1	0.1
beasleyC2	34	32	2	5.9	4.6	0.1	0.1
beasleyC3	130	124	6	4.6	3.7	0.0	0.0
beavma	6	4	2	33.3	0.0	0.0	0.0
bell3a	5	3	2	40.0	0.0	0.0	0.0
bell3b	50	22	29	58.0	0.1	0.0	0.0
bell4	23	7	16	69.6	0.0	0.0	0.0
bell5	12	10	2	16.7	0.0	0.0	0.0
berlin_5_8_0	43	9	34	79.1	0.5	0.0	0.1
bg512142	66	63	3	4.5	0.9	0.0	0.0
bienst1	53	13	40	75.5	126.5	2.4	9.7
bienst2	67	7	60	89.6	62.1	0.9	8.9
binkar10_1	40	38	2	5.0	4.5	0.1	0.1
blend2	8	6	2	25.0	0.0	0.0	0.0
blp-ir98	46	45	1	2.2	6.3	0.1	0.1
bm23	7	6	1	14.3	0.1	0.0	0.0
bnatt400	5	5	0	0.0	2.9	0.6	0.6
bppc8-02	36	13	23	63.9	0.1	0.0	0.0
bppc8-09	36	30	6	16.7	0.2	0.0	0.0
breastcancer-regularized	204	200	4	2.0	21.9	0.1	0.1
cap6000	3	2	1	33.3	22.7	7.6	11.4
cod105	392	391	1	0.3	173.5	0.4	0.4
control30-3-2-3	25	24	1	4.0	0.1	0.0	0.0

	Objectives				Time (s)		
Instance	Obj	Succ	Fails	% fails	Total	(s)/obj	(s)/cut
cost266-UUE	68	56	12	17.6	10.0	0.1	0.2
cov1075	21	1	20	95.2	7.5	0.4	7.5
csched007	137	136	1	0.7	442.4	3.2	3.3
csched010	125	124	1	0.8	615.7	4.9	5.0
cvs08r139-94	387	339	48	12.4	3,605.9	9.3	10.6
cvs16r106-72	914	838	76	8.3	523.6	0.6	0.6
cvs16r128-89	770	734	36	4.7	1,620.3	2.1	2.2
cvs16r70-62	788	773	15	1.9	1,031.0	1.3	1.3
cvs16r89-60	227	218	9	4.0	3,649.8	16.1	16.7
d10200	155	154	1	0.6	966.0	6.2	6.3
danoint	67	52	15	22.4	1.7	0.0	0.0
dcmulti	14	13	1	7.1	0.5	0.0	0.0
dfn-gwin-UUM	49	45	4	8.2	5.0	0.1	0.1
dg012142	262	27	235	89.7	53.1	0.2	2.0
eilB101	90	89	1	1.1	1,284.9	14.3	14.4
eild76	65	63	2	3.1	538.7	8.3	8.6
exp-1-500-5-5	5	3	2	40.0	0.8	0.2	0.3
f2gap201600	18	16	2	11.1	6.8	0.4	0.4
f2gap401600	32	29	3	9.4	68.0	2.1	2.3
f2gap801600	6	5	1	16.7	0.7	0.1	0.1
fiber	47	38	9	19.1	1.1	0.0	0.0
fixnet4	16	15	1	6.3	0.1	0.0	0.0
fixnet6	13	12	1	7.7	0.5	0.0	0.0
g200x740	31	25	6	19.4	1.9	0.1	0.1
g200x740i	33	30	3	9.1	2.5	0.1	0.1
gen-ip002	18	18	0	0.0	0.2	0.0	0.0
gen-ip016	14	13	1	7.1	0.1	0.0	0.0
gen-ip021	15	15	0	0.0	0.2	0.0	0.0
gen-ip036	18	17	1	5.6	0.2	0.0	0.0
ger50-17-ptp-pop-6t	8	8	0	0.0	4.1	0.5	0.5
gesa2-o	8	8	0	0.0	0.2	0.0	0.0
gesa2	8	8	0	0.0	0.2	0.0	0.0
gesa3_o	6	5	1	16.7	0.8	0.1	0.2
gesa3	60	59	1	1.7	6.4	0.1	0.1
glass4	7	6	1	14.3	0.1	0.0	0.0
gmu-35-40	24	9	15	62.5	3.2	0.1	0.4
gmu-35-50	24	17	7	29.2	3.3	0.1	0.2
go19	367	357	10	2.7	741.9	2.0	2.1
graph20-20-1rand	34	4	30	88.2	2.8	0.1	0.7
graphdraw-domain	3	2	1	33.3	0.1	0.0	0.0
graphdraw-gemcutter	26	26	0	0.0	0.0	0.0	0.0
gsvm2rl3	32	31	1	3.1	0.6	0.0	0.0

	Objectives				Time (s)		
Instance	Obj	Succ	Fails	% fails	Total	(s)/obj	(s)/cut
gsvm2rl5	46	45	1	2.2	11.9	0.3	0.3
gt2	30	3	27	90.0	0.2	0.0	0.1
gus-sch	99	17	82	82.8	59.9	0.6	3.5
h50x2450	9	10	1	11.1	4.1	0.5	0.4
haprp	510	170	340	66.7	6.8	0.0	0.0
harp2	28	27	1	3.6	3.5	0.1	0.1
hgms-det	8	6	2	25.0	1.4	0.2	0.2
ic97_potential	4	2	2	50.0	0.2	0.1	0.1
ic97_tension	9	4	5	55.6	0.0	0.0	0.0
icir97_tension	3	3	0	0.0	1.8	0.6	0.6
iis-100-0-cov	103	100	3	2.9	48.5	0.5	0.5
iis-bupa-cov	153	153	0	0.0	533.5	3.5	3.5
janos-us-DDM	5	4	1	20.0	0.7	0.1	0.2
k16x240	15	14	1	6.7	0.5	0.0	0.0
k16x240b	16	15	1	6.3	0.5	0.0	0.0
khb05250	21	19	2	9.5	0.2	0.0	0.0
l152lav	54	52	2	3.7	221.6	4.1	4.3
lectsched-4-obj	4	2	2	50.0	0.1	0.0	0.1
lotsize	3	2	1	33.3	0.8	0.3	0.4
lrn	36	6	30	83.3	2.2	0.1	0.4
lseu	9	8	1	11.1	0.1	0.0	0.0
macrophage	921	46	875	95.0	16.9	0.0	0.4
mas074	13	12	1	7.7	0.6	0.0	0.0
mas076	13	11	2	15.4	0.6	0.0	0.1
mas284	22	20	2	9.1	3.4	0.2	0.2
maxgasflow	6	6	0	0.0	5.7	1.0	1.0
mc11	246	227	19	7.7	3.0	0.0	0.0
mc7	346	341	5	1.4	13.3	0.0	0.0
mc8	375	363	12	3.2	18.8	0.1	0.1
mcsched	1,259	1,224	35	2.8	3,577.9	2.8	2.9
mik-250-1-100-1	125	100	25	20.0	1.0	0.0	0.0
mik-250-20-75-1	97	75	22	22.7	2.8	0.0	0.0
mik-250-20-75-2	119	75	44	37.0	9.8	0.1	0.1
mik-250-20-75-3	61	20	41	67.2	0.1	0.0	0.0
mik-250-20-75-4	127	75	52	40.9	0.8	0.0	0.0
mik-250-20-75-5	136	75	61	44.9	2.3	0.0	0.0
milo-v12-6-r2-40-1	101	5	96	95.0	312.1	3.1	62.4
milo-v13-4-3d-3-0	25	23	2	8.0	12.8	0.5	0.6
mine-90-10	355	346	9	2.5	303.7	0.9	0.9
misc03	20	18	2	10.0	0.9	0.0	0.0
misc07	18	16	2	11.1	3.1	0.2	0.2
mkc1	136	54	82	60.3	15.4	0.1	0.3

	Objectives				Time (s)		
Instance	Obj	Succ	Fails	% fails	Total	(s)/obj	(s)/cut
mkc	23	21	2	8.7	11.6	0.5	0.6
mod008	13	6	7	53.8	0.2	0.0	0.0
mod013	6	5	1	16.7	0.1	0.0	0.0
modglob	38	30	8	21.1	0.7	0.0	0.0
mtest4ma	123	111	12	9.8	5.0	0.0	0.0
n13-3	4	4	0	0.0	0.3	0.1	0.1
n2seq36f	38	24	14	36.8	0.8	0.0	0.0
n4-3	38	36	2	5.3	58.7	1.5	1.6
n5-3	44	35	9	20.5	33.8	0.8	1.0
n6-3	41	38	3	7.3	490.4	12.0	12.9
n7-3	46	30	16	34.8	243.3	5.3	8.1
neos-1058477	29	28	1	3.4	0.1	0.0	0.0
neos-1215259	163	150	13	8.0	198.3	1.2	1.3
neos-1225589	52	20	32	61.5	12.2	0.2	0.6
neos-1281048	264	31	233	88.3	435.8	1.7	14.1
neos-1330346	32	32	0	0.0	4.7	0.1	0.1
neos-1396125	119	69	50	42.0	1,408.4	11.8	20.4
neos-1413153	354	269	85	24.0	99.7	0.3	0.4
neos-1415183	81	4	77	95.1	43.5	0.5	10.9
neos-1420205	61	3	58	95.1	1.8	0.0	0.6
neos-1480121	11	7	4	36.4	0.0	0.0	0.0
neos-1489999	439	438	1	0.2	423.7	1.0	1.0
neos-1582420	298	295	3	1.0	3,132.7	10.5	10.6
neos-1595230	101	5	96	95.0	141.6	1.4	28.3
neos-1599274	21	1	20	95.2	198.4	9.4	198.4
neos-1601936	32	25	7	21.9	479.9	15.0	19.2
neos-1605061	133	70	63	47.4	458.9	3.5	6.6
neos-1605075	468	434	34	7.3	2,205.3	4.7	5.1
neos-1616732	353	200	153	43.3	264.5	0.7	1.3
neos-1620807	21	1	20	95.2	9.4	0.4	9.4
neos-2328163-agri	362	84	278	76.8	2,642.9	7.3	31.5
neos-3024952-loue	11	5	6	54.5	0.5	0.0	0.1
neos-3046601-motu	13	11	2	15.4	0.2	0.0	0.0
neos-3046615-murg	12	10	2	16.7	0.1	0.0	0.0
neos-3072252-nete	5	1	4	80.0	0.0	0.0	0.0
neos-3083819-nubu	29	29	0	0.0	22.3	0.8	0.8
neos-3118745-obra	50	6	44	88.0	0.1	0.0	0.0
neos-3216931-puriri	26	22	4	15.4	1,660.7	63.9	75.5
neos-3373491-avoca	4	2	3	75.0	1.6	0.4	0.8
neos-3381206-awhea	3	3	0	0.0	20.2	6.7	6.7
neos-3421095-cinca	36	35	1	2.8	0.0	0.0	0.0
neos-3592146-hawea	12	7	5	41.7	30.4	2.5	4.3

	Objectives				Time (s)		
Instance	Obj	Succ	Fails	% fails	Total	(s)/obj	(s)/cut
neos-3610040-iskar	41	2	39	95.1	1.2	0.0	0.6
neos-3610051-istra	78	7	71	91.0	9.0	0.1	1.3
neos-3610173-itata	5	3	2	40.0	0.0	0.0	0.0
neos-3611447-jijia	20	4	16	80.0	0.6	0.0	0.1
neos-3611689-kaihu	58	11	47	81.0	2.2	0.0	0.2
neos-3627168-kasai	185	152	33	17.8	80.5	0.4	0.5
neos-3660371-kurow	241	160	81	33.6	424.3	1.8	2.7
neos-3665875-lesum	3	2	1	33.3	0.6	0.2	0.3
neos-3754480-nidda	43	41	2	4.7	6.0	0.1	0.1
neos-3762025-ognon	0	1	0	-	0.5	-	0.5
neos-4333464-siret	19	15	4	21.1	7.5	0.4	0.5
neos-4333596-skien	62	43	19	30.6	5.9	0.1	0.1
neos-4387871-tavua	68	33	35	51.5	42.1	0.6	1.3
neos-4393408-tinui	66	33	33	50.0	15.3	0.2	0.5
neos-4650160-yukon	8	7	1	12.5	2.4	0.3	0.3
neos-480878	25	22	3	12.0	4.8	0.2	0.2
neos-4954672-berkel	7	5	2	28.6	0.5	0.1	0.1
neos-501453	1	1	0	0.0	0.0	0.0	0.0
neos-504674	193	153	40	20.7	105.1	0.5	0.7
neos-504815	157	115	42	26.8	34.0	0.2	0.3
neos-5051588-culgoa	159	22	137	86.2	35.7	0.2	1.6
neos-5075914-elvire	231	125	106	45.9	4.4	0.0	0.0
neos-5078479-escaut	105	7	98	93.3	5.2	0.0	0.7
neos-512201	287	49	238	82.9	62.5	0.2	1.3
neos-5140963-mincio	28	24	4	14.3	0.7	0.0	0.0
neos-5182409-nasivi	170	143	27	15.9	6.8	0.0	0.0
neos-522351	213	29	184	86.4	40.7	0.2	1.4
neos-5261882-treska	51	48	3	5.9	95.7	1.9	2.0
neos-538867	88	47	41	46.6	19.4	0.2	0.4
neos-538916	63	53	10	15.9	12.4	0.2	0.2
neos-547911	88	87	1	1.1	16.8	0.2	0.2
neos-555884	254	62	192	75.6	1.2	0.0	0.0
neos-565815	158	28	130	82.3	1,311.6	8.3	46.8
neos-570431	303	215	88	29.0	553.1	1.8	2.6
neos-574665	90	30	60	66.7	2.3	0.0	0.1
neos-584851	239	231	8	3.3	120.7	0.5	0.5
neos-585192	12	12	0	0.0	1.1	0.1	0.1
neos-585467	17	12	5	29.4	5.7	0.3	0.5
neos-593853	13	10	3	23.1	1.8	0.1	0.2
neos-595904	52	47	5	9.6	48.3	0.9	1.0
neos-598183	19	19	0	0.0	9.3	0.5	0.5
neos-603073	22	20	2	9.1	1.9	0.1	0.1

	Objectives				Time (s)		
Instance	Obj	Succ	Fails	% fails	Total	(s)/obj	(s)/cut
neos-631517	5	5	0	0.0	8.7	1.7	1.7
neos-686190	76	74	2	2.6	3,601.8	47.4	48.7
neos-691058	166	151	15	9.0	371.7	2.2	2.5
neos-717614	15	14	1	6.7	0.2	0.0	0.0
neos-775946	64	8	56	87.5	377.2	5.9	47.1
neos-796608	3	2	1	33.3	0.0	0.0	0.0
neos-801834	457	450	7	1.5	3,602.7	7.9	8.0
neos-803219	20	20	0	0.0	3.5	0.2	0.2
neos-803220	20	20	0	0.0	1.8	0.1	0.1
neos-806323	153	120	33	21.6	1.4	0.0	0.0
neos-807639	89	80	9	10.1	5.1	0.1	0.1
neos-807705	107	91	16	15.0	4.6	0.0	0.1
neos-810326	199	198	1	0.5	1,454.7	7.3	7.3
neos-831188	156	79	77	49.4	3,615.0	23.2	45.8
neos-839859	290	132	158	54.5	351.4	1.2	2.7
neos-862348	27	15	12	44.4	0.2	0.0	0.0
neos-880324	4	2	2	50.0	0.8	0.2	0.4
neos-886822	21	1	20	95.2	295.4	14.1	295.4
neos-892255	221	197	24	10.9	6.4	0.0	0.0
neos-906865	32	29	3	9.4	62.2	1.9	2.1
neos-911880	72	48	24	33.3	0.3	0.0	0.0
neos-911970	87	7	80	92.0	5.3	0.1	0.8
neos-916792	89	89	0	0.0	271.0	3.0	3.0
neos-942830	4	3	1	25.0	131.0	32.8	43.7
neos14	9	8	1	11.1	0.1	0.0	0.0
neos15	20	13	7	35.0	0.4	0.0	0.0
neos16	5	5	0	0.0	0.0	0.0	0.0
neos17	173	171	2	1.2	21.8	0.1	0.1
neos18	5	2	3	60.0	1.7	0.3	0.9
neos22	102	66	36	35.3	14.2	0.1	0.2
neos2	21	20	1	4.8	1.4	0.1	0.1
neos3	58	18	40	69.0	10.9	0.2	0.6
neos5	21	1	20	95.2	0.3	0.0	0.3
neos7	21	1	20	95.2	94.3	4.5	94.3
newdano	101	5	96	95.0	94.1	0.9	18.8
nexp-150-20-1-5	94	13	81	86.2	0.8	0.0	0.1
nexp-50-20-1-1	68	8	60	88.2	0.7	0.0	0.1
nexp-50-20-4-2	5	5	0	0.0	0.1	0.0	0.0
nh97_potential	7	5	2	28.6	2.1	0.3	0.4
nobel-eu-DBE	49	47	2	4.1	1.3	0.0	0.0
ns1208400	3	1	2	66.7	71.3	23.8	71.3
ns1606230	117	90	27	23.1	326.2	2.8	3.6

	Objectives				Time (s)		
Instance	Obj	Succ	Fails	% fails	Total	(s)/obj	(s)/cut
ns1688347	16	4	12	75.0	42.2	2.6	10.6
ns1830653	457	124	333	72.9	370.8	0.8	3.0
ns2081729	8	7	1	12.5	0.1	0.0	0.0
ns894788	601	253	348	57.9	3,600.2	6.0	14.2
nsa	14	13	1	7.1	2.4	0.2	0.2
nsrand-ipx	26	26	0	0.0	1.7	0.1	0.1
nu120-pr12	34	8	26	76.5	1.5	0.0	0.2
nu25-pr12	12	11	1	8.3	2.3	0.2	0.2
p0282	26	24	2	7.7	0.3	0.0	0.0
p0548	4	2	2	50.0	0.0	0.0	0.0
p100x588b	11	11	0	0.0	1.3	0.1	0.1
p200x1188c	6	5	1	16.7	6.9	1.1	1.4
p2756	6	4	2	33.3	0.5	0.1	0.1
p6000	3	2	1	33.3	22.7	7.6	11.3
p6b	375	375	0	0.0	2,797.9	7.5	7.5
p80x400b	4	3	1	25.0	0.3	0.1	0.1
pg5_34	4	4	0	0.0	1.7	0.4	0.4
pg	3	2	1	33.3	1.7	0.6	0.8
pigeon-10	129	72	57	44.2	0.2	0.0	0.0
piperout-03	5	4	1	20.0	19.7	3.9	4.9
piperout-d20	254	144	110	43.3	6.4	0.0	0.0
piperout-d27	60	58	2	3.3	3.3	0.1	0.1
pipex	8	6	2	25.0	0.0	0.0	0.0
pp08aCUTS	70	46	24	34.3	1.9	0.0	0.0
pp08a	5	3	2	40.0	0.0	0.0	0.0
probportfolio	20	20	0	0.0	0.0	0.0	0.0
prod1	42	40	2	4.8	3.3	0.1	0.1
prod2	46	44	2	4.3	3.3	0.1	0.1
protfold	85	46	39	45.9	280.0	3.3	6.1
qiu	39	36	3	7.7	154.1	4.0	4.3
qnet1_o	10	10	0	0.0	23.3	2.3	2.3
qnet1	51	47	4	7.8	64.8	1.3	1.4
queens-30	851	756	95	11.2	3,602.7	4.2	4.8
r50x360	44	43	1	2.3	1.6	0.0	0.0
r80x800	9	9	0	0.0	1.5	0.2	0.2
railway_8_1_0	6	5	1	16.7	0.7	0.1	0.1
ran12x21	19	17	2	10.5	0.7	0.0	0.0
ran13x13	18	18	0	0.0	0.4	0.0	0.0
ran14x18-disj-8	88	86	2	2.3	102.2	1.2	1.2
ran14x18	19	18	1	5.3	0.8	0.0	0.0
ran16x16	21	20	1	4.8	0.7	0.0	0.0
reblock115	427	396	31	7.3	3,602.1	8.4	9.1

	Objectives				Time (s)		
Instance	Obj	Succ	Fails	% fails	Total	(s)/obj	(s)/cut
reblock67	469	442	27	5.8	1,712.1	3.7	3.9
rgn	25	19	6	24.0	0.1	0.0	0.0
rlp1	95	6	89	93.7	5.2	0.1	0.9
rococoB10-011000	321	302	19	5.9	934.4	2.9	3.1
rococoC10-001000	308	157	151	49.0	5.7	0.0	0.0
roll3000	4	3	1	25.0	7.3	1.8	2.4
rout	85	34	51	60.0	55.5	0.7	1.6
roy	17	13	4	23.5	0.2	0.0	0.0
sentoy	9	8	1	11.1	0.2	0.0	0.0
set1al	4	3	1	25.0	0.4	0.1	0.1
set1ch	98	71	27	27.6	1.1	0.0	0.0
set1cl	4	3	1	25.0	0.4	0.1	0.1
set3-10	4	3	1	25.0	7.9	2.0	2.6
set3-15	5	4	1	20.0	0.8	0.2	0.2
set3-20	3	2	1	33.3	0.2	0.1	0.1
seymour-disj-10	899	621	279	31.0	787.5	0.9	1.3
seymour1	96	74	22	22.9	1,106.1	11.5	14.9
seymour	504	481	23	4.6	2,091.4	4.1	4.3
sorrell8	1,189	1,075	114	9.6	2,315.1	1.9	2.2
sp150x300d	8	6	2	25.0	0.2	0.0	0.0
sp98ir	112	111	1	0.9	1,259.2	11.2	11.3
square23	92	90	2	2.2	181.1	2.0	2.0
stein27_nocard	41	2	39	95.1	0.2	0.0	0.1
stein45_nocard	21	1	20	95.2	0.6	0.0	0.6
supportcase17	10	7	3	30.0	0.1	0.0	0.0
supportcase20	7	7	0	0.0	0.7	0.1	0.1
supportcase25	120	115	5	4.2	43.6	0.4	0.4
supportcase26	4	2	2	50.0	0.0	0.0	0.0
ta1-UUM	96	10	86	89.6	2.6	0.0	0.3
timtab1CUTS	146	122	24	16.4	9.5	0.1	0.1
timtab1	7	5	2	28.6	0.0	0.0	0.0
timtab2	4	3	1	25.0	0.0	0.0	0.0
toll-like	1,086	179	907	83.5	14.7	0.0	0.1
tr12-30	3	2	1	33.3	0.8	0.3	0.4
traininstance6	41	2	39	95.1	0.0	0.0	0.0
uct-subprob	65	23	42	64.6	133.2	2.0	5.8
umts	279	276	3	1.1	2,379.5	8.5	8.6
usAbbrv-8-25_70	4	4	0	0.0	0.6	0.1	0.1
vpm1	34	10	24	70.6	0.1	0.0	0.0
vpm2	36	25	11	30.6	0.2	0.0	0.0
zib54-UUE	62	56	6	9.7	63.3	1.0	1.1
Average				28.8	267.7	2.0	5.6

C Example of invalid cuts from a point-ray collection

This example shows that the using as the point-ray collection the optimal points p^t on each term $t \in \mathcal{T}$ along with the neighbors of p^t may lead to the generation of invalid cuts from the associated (PRLP).

$$
\begin{aligned}
\max_{x_1,x_2,x_3} \quad & x_3 \\
& -x_3 \leq -1/2 \\
& -(7/4)x_1 + 5x_2 - 2x_3 \leq 1 \\
& -x_1 - 5x_2 + 2x_3 \leq -1 \\
& -x_1 - (20/3)x_2 + (7/3)x_3 \leq -3/2 \\
& x_1 - x_2 + (3/2)x_3 \leq 3/2 \\
& 2x_1 - x_2 + 3x_3 \leq 7/2 \\
& -x_1 + 4x_2 + 2x_3 \leq 7/2 \\
& -x_1 + 4x_2 \leq 2 \\
& x_1, x_2, x_3 \in [0,1] \\
& x_1 \quad \text{integer}
\end{aligned}
$$

Let P denote the feasible region of the linear relaxation of the above integer program. Figure 5 shows the feasible region of P. Point q^1 denotes the optimal solution to the linear programming relaxation. The vertices of P are:

$$
\begin{aligned}
q^1 &= \{1/2, 1/2, 1\} \\
q^2 &= \{1, 3/4, 3/4\} \\
q^3 &= \{1, 3/4, 1/2\} \\
q^4 &= \{1, 1/2, 2/3\} \\
q^5 &= \{1, 1/4, 1/2\} \\
q^6 &= \{0, 1/2, 3/4\} \\
q^7 &= \{0, 2/5, 1/2\}.
\end{aligned}
$$

We use as the valid disjunction the elementary split on x_1. If we solve $\max\{x_3 : x \in P,\ x_1 = 0\}$, the maximum is achieved by q^6. Solving $\max\{x_3 : x \in P,\ x_1 = 1\}$, the maximum is achieved by q^2. Consider using q^2 and q^6 and their neighbors as the collection of points given to (PRLP). Thus $\mathcal{P} = \{q^2, q^3, q^4, q^6, q^7\}$ and $\mathcal{R} = \emptyset$.

One cut that can be obtained from this set of points is $\bar{\alpha}^\intercal x \geq 1$ where $\bar{\alpha} = (-1/6, 5, -2)$, which goes through q^6, q^7, and q^4, leaving q^2 and q^3 on the feasible side, and cutting off not only q^1, but also q^5. Indeed, $\bar{\alpha}^\intercal q^1 = -1/12 + 5/2 - 2 = 5/12 < 1$, while $\bar{\alpha}^\intercal q^4 = \bar{\alpha}^\intercal q^7 = \bar{\alpha}^\intercal q^6 = 1$, and $\bar{\alpha}^\intercal q^2 = 25/12 > 1$ and $\bar{\alpha}^\intercal q^3 = 31/12 > 1$. However, as seen in Figure 5, $\bar{\alpha}^\intercal q^5 = 1/12 < 1$, so that point of $P \cap \{x : x_1 = 1\}$ is cut off, making the cut invalid.

There is a relatively simple resolution for the above counterexample. If we require the generated cuts to be tight at the optimal solutions on each facet, q^6 and q^2, then all generated cuts will be valid for $\text{conv}(P_D)$. We state this in Theorem 12.

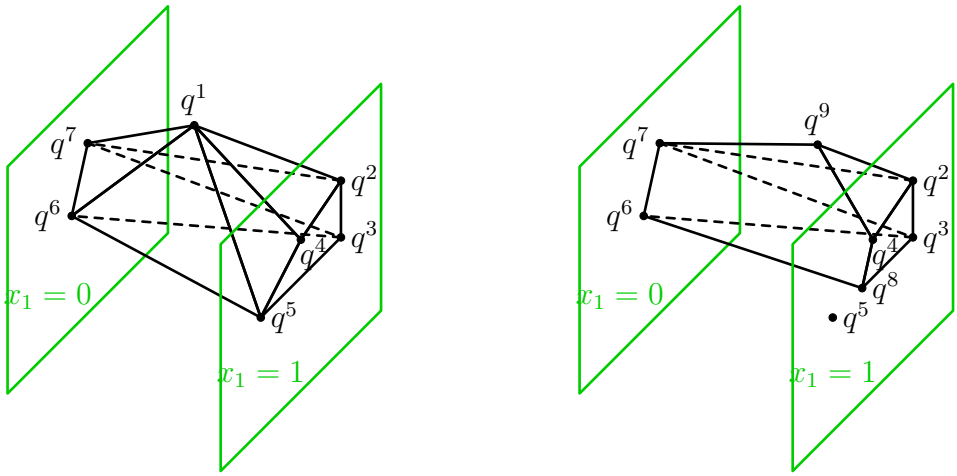


Figure 5: The LP polytope P for the counter-example showing an optimal point on each disjunctive term and its neighbors as the point-ray collection may lead to invalid cuts.

Theorem 12. *For each $t \in \mathcal{T}$, set $\mathcal{P}^t$ to be an optimal solution p^t to* $\arg\min_x\{c^\top x : x \in P^t\}$ *as well as one point on each of the edges emanating from p^t within P^t. Let $\mathcal{P} := \cup_{t\in\mathcal{T}}\mathcal{P}^t$ and $\mathcal{R} := \emptyset$. Any feasible solution to* (PRLP) *formulated from these points and amended with the condition $\alpha^\top p^t = \beta$ for all $t \in \mathcal{T}$ yields a valid cut for P_I.*

Proof. Let (α, β) be a feasible solution to (PRLP) such that $\alpha^\top p^t = \beta$ for all $t \in \mathcal{T}$. Since $\alpha^\top p \geq \beta$ for all $p \in \mathcal{P}^t$, it holds that $\alpha^\top(p - p^t) \geq 0$. This means that, for each $t \in \mathcal{T}$, the generated cut is valid for the cone with apex at p^t and rays going through each of the points $p \in \mathcal{P}^t$. By convexity, this cone is a relaxation of P^t. It follows, by Corollary 4, that the cut is valid for P_I. □

D Sample partial branch-and-bound tree

The computational experiments in the paper use a partial branch-and-bound tree as the source of the disjunction for cut generation. The partial trees are produced from the branching strategy described in Section 5.1. In particular, there may exist nodes of the tree that are pruned, and the tree may be very asymmetric. We illustrate this with one sample tree, shown in Figure 6, constructed from the instance `bm23` and terminated after finding 64 leaf nodes. This tree includes two pruned nodes (the leftmost leaf node, from branching on x_{15}, and an adjacent leaf on the same level, from branching on x_8).

E Computational setup: additional details

This section expands on Section 5.1 with additional details on the tolerances and settings used in our VPC implementation.

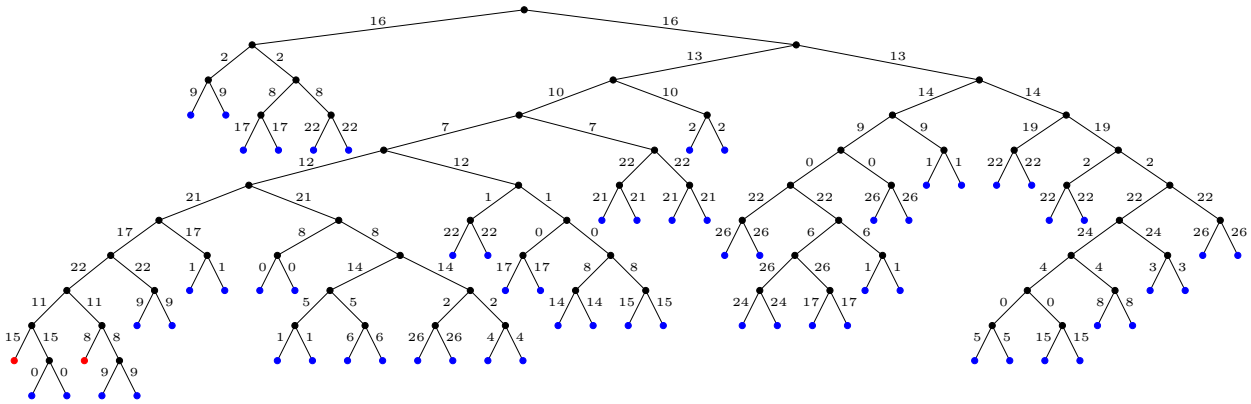


Figure 6: Partial branch-and-bound tree with 64 leaf nodes for instance `bm23`.

E.1 Evaluation

We evaluate cuts from two different perspectives: strength and effect on branch and bound. The strength of the cuts is assessed by the *percent integrality (root) gap closed* by one round of VPCs. Let x_I denote an optimal solution to (IP), and let x' be an optimal solution to (LP) after a set of cuts have been added. We measure the quantity

$$\% \text{ integrality gap closed} := 100 \times \frac{c^\intercal x' - c^\intercal \bar{x}}{c^\intercal x_I - c^\intercal \bar{x}}.$$

As a baseline, we also report the percent gap closed by adding one round of GMICs, as well as the percent gap closed by using both VPCs and GMICs together. In addition, we report, both with and without the use of VPCs, the root gap closed by Gurobi after one cut pass and after the last round of cuts is added at the root. The effect on branch and bound is measured by the time Gurobi takes to solve the problem with VPCs added as user cuts; this is compared to the time taken without VPCs.

Note that there are two sources of cut strength, one from the disjunction from which the cut is produced, and one from *modularization* applied to variables other than those involved in the disjunction [9]. In contrast to GMICs, VPCs do not use this second approach, as applying the technique to VPCs is more complex and requires the use of additional information about the cut when it is derived from a disjunction that is not simple.

E.2 Generating a partial branch-and-bound tree

The partial branch-and-bound tree is generated by the node, variable, and branch selection rules that follow, which are the defaults for `Cbc`. Node selection is roughly by the *best-first* or *best-bound* rule, in which the next node to explore will be the one with the minimum objective value (though the number of fractional variables at each node is also considered, it is to a much lesser extent). Variable selection utilizes the outcome of strong branching on up to five fractional variables at each node. Between the two possible children of the node when branching on the variable that is selected, the direction is chosen by a similar rule to the node selection criteria, i.e., typically in the direction of the child with a lower optimal value.

E.3 Instance preprocessing

Every instance is first preprocessed by Gurobi's presolve. This procedure is used in order to improve the fairness of the testing environment. It allows VPCs to be generated from the same version of the instance that would be used internally by the branch-and-bound solver. This is also a reason for turning presolve off during the subsequent branch and bound tests, as one round of presolve has already been applied. At the same time, for some instances, preprocessing closes a significant portion of the integrality gap, which could make the process of finding strong cuts more difficult. For reproducibility, one must be aware that not only might Gurobi's presolve algorithms change with a new version (release of the software), but also they depend on the random seed given to the solver. We do not experiment with this latter variability (we presolve with the random seed 628 only).

E.4 Setting up (PRLP)

When constructing the associated (PRLP) (in the nonbasic space defined by the cobasis at $\bar{x}$), we remove all duplicate rows. In addition, any rows that are actually bounds on the α variables are removed as explicit constraints and kept as bounds instead. Henceforth, we assume that $(\mathcal{P}, \mathcal{R})$ has no duplicates. We proceed with generating cuts from a given (PRLP) if it is feasible and solves to optimality within a minute when using no objective, i.e., just the feasibility problem.

E.5 Normalization

As mentioned, we formulate the PRLP in the nonbasic space with respect to $\bar{x}$ (in which $\bar{x}$ is represented as the origin), and we normalize (PRLP) to have $\beta = 1$, which restricts the set of obtainable inequalities to those that cut $\bar{x}$. In practice, we will actually use some scaled positive constant determined by the input, as $\beta = 1$ could lead to numerical issues, due to how it causes the cut coefficients scale. To illustrate this, suppose $c^\intercal p^t \geq 10^8$ for all $t \in \mathcal{T}$. Then, if $\beta = 1$, $\alpha = c/10^8$ is a feasible solution to (PRLP). However, coefficients less than 10^{-7} are often regarded as zero by solvers, so we may end up generating cuts incorrectly with improper scaling.

E.6 Cut processing and objective failures

Not every objective function we try for (PRLP) leads to a new cut. As we discussed in Section 4, (PRLP) can be unbounded or lead to a duplicate cut. Other failures are imposed by conditions that we set. If the time to solve (PRLP) for an objective is greater than 5 seconds, we abandon the objective. If we do obtain a solution, since (PRLP) is formulated in the nonbasic space, we first convert the cut to the structural space, yielding a cut $\gamma^\intercal x \geq \gamma_0$. Next, we remove small coefficients: for $j \in [n]$, if $|\gamma_j| < \epsilon$, we ignore the coefficient, and if $\epsilon \leq |\gamma_j| < \epsilon_{\text{coeff}}$, we replace x_j by either its lower or upper bound and adjust γ_0. In our experiments, $\epsilon = 10^{-7}$, and $\epsilon_{\text{coeff}} = 10^{-5}$. Assume that $\gamma^\intercal x \geq \gamma_0$ has been processed in this way. The cut is rejected if it is a duplicate of or dominated by a previously generated VPC. It will also be rejected if its dynamism ($\max_{j\in[n]}|\gamma_j| / \min_{j\in[n]}\{|\gamma_j| : \gamma_j \neq 0\}$) is higher

than 10^8. In addition, if there exists some previously generated VPC $\alpha^\intercal x \geq \beta$ that is nearly parallel to $\gamma^\intercal x \geq \gamma_0$, i.e., if $\alpha \cdot \gamma / (\|\alpha\| \cdot \|\gamma\|) < \epsilon_{\text{orth}}$ ($\epsilon_{\text{orth}} = 10^{-3}$ in our setup), then we keep only one of these two cuts (the one that separates $\bar{x}$ by a greater Euclidean distance, or if these are equal, the sparser cut).

We solve (PRLP) until we exhaust all objectives or reach one of these stopping criteria: (1) Numerical difficulties are encountered while solving (PRLP). (2) The time limit for cut generation is reached. (3) The cut limit is reached. (4) The failure limit is reached.

An example of numerical difficulties we have encountered is when (PRLP) solves to optimality for one objective but is deemed primal infeasible for another. The time limit for cut generation is 900 seconds (the time to set up the partial tree and build (PRLP) is not counted against this). The cut limit is equal to the number k of fractional variables at the LP optimal solution.

The failure limit we use comes from some nontrivial experimentation. It varies based on several parameters: there are different maximum failure rates depending on whether "few" or "many" cuts have been generated, and whether "many" objective functions have been attempted. Let $\phi_{\text{few\_cuts}} := 0.95$, $\phi_{\text{many\_cuts}} := 0.90$, $\phi_{\text{many\_obj}} := 0.80$. We define "few" cuts as $n_{\text{few\_cuts}} := 1$, "many" cuts as $n_{\text{many\_cuts}} := \lceil k/4 \rceil$, and "many" objectives as

$$n_{\text{many\_obj}} := \max\{\lceil n_{\text{few\_cuts}}/(1-\phi_{\text{few\_cuts}}) \rceil, \lceil n_{\text{many\_cuts}}/(1-\phi_{\text{many\_cuts}}) \rceil\}.$$

Hence, the default for $n_{\text{many\_obj}}$ is $\max\{20, 10\lceil k/4 \rceil\}$. After each cut, we test whether the current failure ratio (number of unsuccessful objectives as a proportion of the total number of objectives attempted) is greater than the appropriate threshold ($\phi_{\text{few\_cuts}}$, $\phi_{\text{many\_cuts}}$, or $\phi_{\text{many\_obj}}$); if it is, then we return that we have reached the failure limit. We also say we reached the failure limit if the first $\lceil n_{\text{few\_cuts}}/(1-\phi_{\text{few\_cuts}}) \rceil$ objectives all lead to failures; this is often an indicator of numerical issues with the instance. As more cuts are generated and more objectives are tried, the acceptable failure rate decreases in this setup, as there is likely to be diminishing marginal benefit for additional cuts and we wish to avoid spending excessive amounts of time attempting to generate cuts unsuccessfully.

F Instance selection

Instances were selected by the following criteria:

Criterion 1. The IP optimal value is known and is not equal to the LP optimal value.

Criterion 2. There are at most 5,000 rows and 5,000 columns in the presolved instance.

Criterion 3. No partial branch-and-bound tree used finds an IP optimal solution.

Criterion 4. For at least one partial branch-and-bound tree, (a) either the disjunctive lower bound, $c^\intercal p^*$, is strictly less than the maximum objective value of any leaf node, or it is strictly greater than $c^\intercal \bar{x}$, (b) the corresponding (PRLP) is primal feasible, (c) it is proved feasible within a one minute time limit, and (d) at least one VPC is added to cut pool after processing according to Section E.6.

Criterion 4 is put in place because infeasible instantiations of (PRLP) typically occur when $\max_{p \in \mathcal{P}} c^\intercal p = c^\intercal p^* = c^\intercal \bar{x}$. When we report gap closed, we will also filter by another condition, that $c^\intercal p^* > c^\intercal \bar{x}$ from the 64-leaf partial tree, as instances in which the disjunctive lower bound and optimal value of the LP relaxation coincide are not good candidates for evaluation by gap closed. The branch and bound results only include those instances that are solved in under an hour with Gurobi (either with or without VPCs).

We modify the instances `stein27` and `stein45` from the versions in MIPLIB; in particular, we remove a cardinality constraint enforcing a lower bound for the objective value, which is not present in the original formulation of the problem [35].

In total, there are 1,458 instances across the MIPLIB, CORAL, and NEOS sets. Many of these are ultimately not considered. Initially, we eliminate 42 instances that have indicator constraints from MIPLIB 2017, and 81 nonindicator instances that are infeasible or unbounded. Then, from the 1,335 instances that are left, we eliminate 349 instances with unknown IP optimal value. We then remove 45 instances that have no objective (feasibility instance). From the 941 instances remaining, 295 instances have either more than 20,000 rows or more than 20,000 columns, so these are never preprocessed and removed from consideration, as we deemed it unlikely that the resulting presolved instance would satisfy Criterion 2. From the remaining 646 instances, 53 are eliminated because of less than 0.001 integrality gap after presolve, and another 153 are eliminated due to having too many rows or columns after presolve, Further, 4 more instances (bley_xs2, control20-5-10-5, ej, gen-ip016) encountered numerical issues. For example, for bley_xs2, Gurobi presolve declares the instance unbounded, and for control20-5-10-5, Gurobi will declare the instance unbounded or infeasible depending on seed. The above exclusions count neos-3661949-lesse, which appears to have no integrality gap after presolve, though the issue may be numerical as the optimal value found after presolve is 689,000,000, compared to the listed value 688,995,225. This leaves 436 instances.

Of these 436 instances, Table 9 lists the 104 that were removed from consideration and the reason for removal. As the table shows, one of the most common reasons for discarding an instance was that no cuts were generated for that instance due either to $\max_{p \in \mathcal{P}} c^\intercal p = c^\intercal \bar{x}$ (the optimal value over the best leaf node equals the optimal value of the LP) or (PRLP) being primal infeasible. The situations are treated together because the former anyway typically results in (PRLP) being infeasible, though it also implies that our metric of gap closed is not reasonable for that instance. There is potential for these instances to either generate inequalities that do not cut away $\bar{x}$ or to try to find a different partial branch-and-bound tree, but that has been beyond the scope of our investigation. There are an additional 50 instances for which branch and bound results are not shown, because these instances were unable to be solved within an hour by Gurobi (neither with or without cuts, averaged across all random seeds). Finally, we comment that results were not obtained with instance neos-3734794-moppy because of a slight mismatch between the objective values computed for the disjunctive terms within Cbc versus outside of Cbc' this issue could be resolved with loosening the associated tolerance, but we refrained from this manual adjustment for these experiments.

Table 9: Instances not considered along with violated selection criteria.

		Number of leaves					
Instance	Set	2	4	8	16	32	64
22433	miplib2017			3	3	3	3
air01	miplib2	3	3	3	3	3	3
app1-1	miplib2017					3	3
b-ball	miplib2017	4(a)	4(a)	4(a)	4(b)	4(b)	4(b)
bppc4-08	miplib2017	4(a)	4(a)	4(b)	4(b)	4(b)	4(c)
bppc6-02	miplib2017	4(a)	4(a)	4(b)	4(c)	4(c)	4(c)
chromaticindex32-8	miplib2017	4(a)	4(a)	4(a)	4(b)	4(b)	4(c)
csched008	miplib2017	4(a)	4(a)	4(a)	4(a)	4(a)	4(b)
egout	miplib3					3	3
eil33-2	miplib2017					3	3
f2gap40400	miplib2017		3	3	3	3	3
fastxgemm-n2r6s0t2	miplib2017	4(a)	4(a)	4(a)	4(b)	4(b)	4(b)
fastxgemm-n2r7s4t1	miplib2017	4(a)	4(a)	4(a)	4(a)	4(b)	4(b)
fixnet3	miplib2						3
gen	miplib2017	3	3	3	3	3	3
gr4x6	miplib2017					3	3
icir97_potential	miplib2017	4(a)	4(a)	4(a)	4(a)	4(a)	4(a)
lp4l	miplib2					3	3
mad	miplib2017	4(a)	4(a)	4(a)	4(b)	4(b)	4(b)
markshare1	miplib2017	4(a)	4(a)	4(a)	4(a)	4(a)	4(a)
markshare2	miplib2017	4(a)	4(a)	4(a)	4(a)	4(a)	4(a)
markshare_4_0	miplib2017	4(a)	4(a)	4(a)	4(a)	4(a)	4(a)
markshare_5_0	miplib2017	4(a)	4(a)	4(a)	4(a)	4(a)	4(b)
misc01	miplib2					3	3
misc02	miplib2			3	3	3	3
misc04	miplib2				3	3	3
misc05	miplib2						3
misc06	miplib3						3
mod010	miplib2017				3	3	3
neos-1112782	miplib2017	4(d)	4(d)	4(d)	4(d)	4(d)	4(d)
neos-1112787	miplib2017	4(d)	4(d)	4(d)	4(d)	4(d)	4(d)
neos-1200887	coral	4(a)	4(b)	4(b)	4(b)	4(b)	4(b)
neos-1211578	coral	4(a)	4(a)	4(a)	4(a)	4(b)	4(b)
neos-1228986	coral	4(a)	4(a)	4(a)	4(b)	4(b)	4(b)
neos-1337489	coral	4(a)	4(a)	4(a)	4(a)	4(b)	4(b)
neos-1425699	miplib2017	3	3	3	3	3	3
neos-1426635	miplib2010	4(a)	4(a)	4(a)	4(a)	4(a)	4(a)
neos-1426662	miplib2010	4(a)	4(a)	4(a)	4(a)	4(a)	4(a)
neos-1430701	miplib2017	4(a)	4(a)	4(a)	4(a)	4(b)	4(b)
neos-1436709	miplib2010	4(a)	4(a)	4(a)	4(a)	4(a)	4(a)
neos-1437164	coral	4(a)	4(a)	4(a)	4(b)	4(b)	4(b)
neos-1440447	coral	4(a)	4(a)	4(a)	4(a)	4(b)	4(b)
neos-1440460	miplib2010	4(a)	4(a)	4(a)	4(a)	4(a)	4(b)

		Number of leaves					
Instance	Set	2	4	8	16	32	64
neos-1441553	coral	4(a)	4(a)	4(a)	4(a)	4(b)	4(b)
neos-1442119	miplib2017	4(a)	4(a)	4(a)	4(a)	4(a)	4(a)
neos-1442657	miplib2010	4(a)	4(a)	4(a)	4(a)	4(a)	4(a)
neos-1445532	miplib2017				3	3	3
neos-1516309	miplib2017	3	3	3	3	3	3
neos-1620770	miplib2010	4(a)	4(a)	4(a)	4(a)	4(c)	4(c)
neos-2624317-amur	miplib2017	4(a)	4(a)	4(a)	4(b)	4(b)	4(b)
neos-2652786-brda	miplib2017	4(a)	4(a)	4(a)	4(a)	4(a)	4(a)
neos-2657525-crna	miplib2017	4(a)	4(a)	4(a)	4(a)	4(a)	4(a)
neos-3214367-sovi	miplib2017	4(c)	4(c)	4(c)	?	?	?
neos-3530903-gauja	miplib2017	4(a)	4(a)	4(a)	4(a)	4(b)	4(b)
neos-3530905-gaula	miplib2017	4(a)	4(a)	4(a)	4(a)	4(a)	4(a)
neos-3734794-moppy	miplib2017	?	?	?	?	?	?
neos-430149	coral	4(a)	4(a)	4(a)	4(a)	4(b)	4(b)
neos-4338804-snowy	miplib2017	4(a)	4(a)	4(a)	4(a)	4(a)	4(a)
neos-530627	coral					?	?
neos-555001	miplib2017	4(a)	4(a)	4(a)	4(a)	4(a)	4(a)
neos-555694	coral	4(a)	4(a)	4(a)	4(a)	4(b)	4(c)
neos-555771	coral	4(a)	4(a)	4(b)	4(b)	4(b)	4(b)
neos-555927	coral	4(a)	4(a)	4(a)	4(a)	4(a)	4(a)
neos-631694	coral	4(a)	4(a)	4(b)	4(b)	4(c)	4(c)
neos-825075	coral	4(a)	4(b)	4(b)	4(b)	3	3
neos-847302	miplib2010	4(a)	4(a)	4(b)	4(b)	4(b)	4(b)
neos-850681	miplib2017	4(a)	4(a)	4(a)	4(a)	4(c)	4(c)
neos-860300	miplib2017						3
neos-933562	miplib2017	4(a)	4(c)	4(c)	4(c)	4(c)	4(c)
neos-955215	coral	4(a)	4(a)	4(a)	4(b)	4(b)	4(b)
neos11	coral	4(a)	4(a)	4(a)	4(c)	4(c)	4(c)
neos20	coral	4(a)	4(b)	4(b)	4(b)	4(b)	4(b)
noswot	miplib2017	4(a)	4(a)	4(a)	4(a)	4(b)	4(b)
ns2071214	miplib2017	4(a)	4(a)	4(a)	4(a)	4(c)	4(c)
ns4-pr6	miplib2017				3	3	3
opt1217	miplib2017	4(a)	4(a)	4(a)	4(a)	4(a)	4(b)
p0033	miplib3					3	3
p0201	miplib2017						3
p0291	miplib2						3
pigeon-08	miplib2017	4(a)	4(a)	4(a)	4(a)	4(b)	4(b)
pigeon-11	miplib2010	4(a)	4(a)	4(a)	4(b)	4(b)	4(b)
pigeon-12	miplib2010	4(a)	4(a)	4(a)	4(a)	4(a)	4(b)
pigeon-13	miplib2017	4(a)	4(a)	4(a)	4(b)	4(b)	4(b)
pigeon-16	miplib2017	4(a)	4(a)	4(a)	4(a)	4(a)	4(a)
pigeon-19	miplib2010	4(a)	4(a)	4(a)	4(a)	4(a)	4(a)
pigeon-20	miplib2017	4(a)	4(a)	4(a)	4(b)	4(b)	4(b)
pk1	miplib2017	4(a)	4(a)	4(b)	4(b)	4(b)	4(b)
pw-myciel4	miplib2017	4(a)	4(a)	4(a)	4(b)	4(b)	4(c)

		Number of leaves					
Instance	Set	2	4	8	16	32	64
qap10	miplib2017			3	3	3	3
rentacar	miplib2017			4(d)		3	3
rlp2	neos		3	3	3	3	3
rocI-3-11	miplib2017	4(a)	4(a)	4(a)	4(b)	4(b)	4(b)
rocI-4-11	miplib2017	4(a)	4(a)	4(a)	4(a)	4(b)	4(b)
sample2	miplib2		3	3	3	3	3
sct2	miplib2017	4(a)	4(a)	4(a)	4(b)	4(b)	4(c)
shiftreg1-4	miplib2017	4(a)	4(a)	4(b)	4(c)	4(c)	4(c)
stein09_nocard	miplib2			3	3	3	3
stein15_nocard	miplib2					3	3
supportcase14	miplib2017			3	3	3	3
supportcase16	miplib2017		3	3	3	3	3
tanglegram2	miplib2010		3	3	3	3	3
tanglegram6	miplib2017			3	3	3	3
traininstance2	miplib2017	4(a)	4(a)	4(a)	4(a)	4(a)	4(a)
wachplan	miplib2017	4(a)	4(a)	4(a)	4(a)	4(a)	4(a)

G Additional results for partial branch-and-bound tree experiments

This section contains additional details for the experimental results, left out of the main text due to length.

Table 10 shows the number of rows and columns for each of the 332 instances used in the gap closed experiments, the number of cuts produced to yield the "best" VPC objective value (across partial tree sizes), and the percent gap closed for each instance. Columns 2 and 3 give, for each instance, the number of constraints and variables after preprocessing. The next two columns show the number of cuts generated. Column 6 is the percent gap closed by GMICs when they are added to the LP relaxation. Column 7 is the percent gap closed as implied by the disjunctive lower bound from the partial tree with 64 leaf nodes. Column 8 is the percent gap closed by VPCs. Column 9 is the best percent gap closed for each instance between the value with GMICs alone in column "G" and the value with VPCs alone in column "V". Column 10 is the percent gap closed when GMICs and VPCs are used together. Columns 11 and 12 show the percent gap closed by Gurobi cuts from one round at the root, first without and then with VPCs added as user cuts. Columns 13 and 14 show the same, but after the last round of cuts at the root. The values in columns 11 and 13 are the maximum percent gap closed across seven random seeds. The last two rows give the average and number of wins, reproducing the summary data in Table 1.

Table 11 provides the time and number of nodes taken by each instance for the partial tree size per instance that led to the best outcome for Gurobi with VPCs across the six partial tree sizes tested. Columns 2 and 3 show the number of terms and number of cuts that led to the time for "V". The table is in increasing order by column 5, "V".

Table 10: Percent gap closed by instance for GMICs (G), VPCs (V), both VPCs and GMICs used together, and the bound implied by the partial branch-and-bound tree with 64 leaf nodes (DB). Also shown are the sizes of the instances, the number of cuts added, and the percent gap closed by Gurobi at the root (after one round (GurF) and after the last round (GurL)).

			# cuts		% gap closed								
Instance	Rows	Cols	G	V	G	DB	V	max(G,V)	V+G	GurF	V+GurF	GurL	V+GurL
10teams	210	1,600	153	16	100.00	0.00	0.00	100.00	100.00	93.88	93.88	100.00	100.00
23588	137	237	75	75	5.77	72.18	71.61	71.61	71.62	15.59	68.94	24.38	71.78
30n20b8	387	4,191	184	190	11.10	1.56	0.03	11.10	11.10	0.93	1.33	16.91	16.30
50v-10	233	2,013	29	29	45.75	18.01	11.18	45.75	45.82	49.86	50.15	73.17	74.44
a1c1s1	1,876	2,489	154	3	25.11	4.90	1.09	25.11	25.38	45.58	46.25	88.81	88.73
a2c1s1	1,856	2,459	157	18	24.37	3.34	0.46	24.37	24.37	42.80	44.25	91.53	92.02
aflow30a	449	812	25	25	16.54	17.17	16.83	16.83	19.29	29.60	29.63	64.90	65.25
aflow40b	1,401	2,687	34	36	12.00	14.59	13.34	13.34	15.42	30.75	32.15	56.33	58.70
aligninq	337	1,831	182	163	11.59	64.30	61.10	61.10	61.19	32.48	60.67	50.80	62.44
app3	370	1,392	18	15	20.41	55.67	15.52	20.41	26.67	28.43	34.82	83.90	84.43
arki001	693	957	64	5	35.06	13.18	4.08	35.06	35.10	20.25	28.01	48.87	48.87
assign1-5-8	161	156	114	114	6.62	10.22	9.10	9.10	9.66	5.98	9.12	8.52	9.82
b1c1s1	2,520	2,651	240	9	23.01	3.74	0.05	23.01	23.01	22.48	27.76	76.32	76.69
b2c1s1	2,546	2,677	238	4	19.81	2.01	0.09	19.81	19.81	23.07	23.40	72.08	72.32
bc1	1,338	1,044	5	5	0.59	12.14	8.78	8.78	8.78	30.32	30.70	44.93	45.03
bc	1,877	1,275	15	16	2.35	13.70	13.10	13.10	13.27	15.13	17.71	30.26	30.65
beasleyC1	700	1,099	17	17	22.73	38.64	25.66	25.66	33.38	51.07	56.22	93.71	95.59
beasleyC2	680	1,072	32	32	4.10	13.91	9.74	9.74	11.21	49.44	49.92	97.28	97.50
beasleyC3	790	1,220	124	124	1.30	2.81	1.24	1.30	1.80	57.74	63.23	97.72	97.87
beavma	247	272	39	4	48.24	25.05	8.83	48.24	52.33	68.31	71.31	98.18	99.86
bell3a	63	82	7	3	36.98	53.71	39.92	39.92	45.96	38.39	45.54	45.55	48.03
bell3b	73	91	24	22	31.17	85.44	83.84	83.84	84.61	37.15	86.21	52.14	95.62
bell4	73	88	27	7	25.57	21.72	14.22	25.57	26.15	25.26	26.10	30.72	31.24
bell5	34	56	10	10	13.84	84.75	73.82	73.82	75.11	13.68	25.22	36.66	76.81
berlin_5_8_0	1,328	979	235	9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
bg512142	897	757	204	63	0.81	2.00	0.81	0.81	1.39	3.51	3.81	10.77	10.99
bienst1	520	449	26	13	11.95	43.61	43.61	43.61	43.61	7.86	43.61	14.66	43.61
bienst2	520	449	33	7	9.77	35.63	24.18	24.18	24.18	7.17	26.62	12.28	35.63
binkar10_1	813	1,399	38	38	10.94	9.96	7.44	10.94	12.59	23.15	38.11	74.04	74.59
blend2	154	302	12	6	5.46	29.37	12.03	12.03	12.03	5.46	10.87	20.58	21.18
blp-ir98	473	4,675	43	45	40.79	19.07	4.18	40.79	40.79	29.67	32.64	86.39	87.34
bm23	20	27	6	6	16.81	71.47	69.64	69.64	69.64	21.16	66.20	39.06	71.15
bnatt400	3,808	1,901	545	5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
bppc8-02	57	204	17	13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
bppc8-09	75	364	30	30	3.08	3.30	0.59	3.08	3.08	1.19	1.59	3.61	3.17
breastcancer-regularized	485	477	9	200	0.01	22.65	8.87	8.87	8.88	0.20	7.68	24.70	27.25
cap6000	1,725	4,596	2	2	41.65	67.98	63.48	63.48	63.48	35.70	35.70	36.02	38.82
cod105	1,024	1,024	598	391	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
control30-3-2-3	324	247	10	24	0.00	0.00	0.00	0.00	0.00	0.02	0.02	0.02	0.02
cost266-UUE	1,302	3,882	55	56	23.92	19.13	4.96	23.92	24.67	20.10	22.27	37.29	38.68
cov1075	637	120	120	1	1.55	10.66	10.66	10.66	10.66	2.80	26.18	5.37	26.54
csched007	274	1,680	130	136	5.10	7.21	6.16	6.16	8.31	20.61	21.35	38.91	38.27
csched010	272	1,678	121	124	4.17	6.05	5.51	5.51	6.02	14.19	16.07	33.74	34.09
cvs08r139-94	2,398	1,864	1,582	339	1.09	10.47	1.24	1.24	1.63	1.58	1.96	2.51	2.78
cvs16r106-72	3,608	2,848	2,440	838	0.75	1.98	0.00	0.75	0.83	9.27	9.60	13.27	13.43

			# cuts		% gap closed								
Instance	Rows	Cols	G	V	G	DB	V	max(G,V)	V+G	GurF	V+GurF	GurL	V+GurL
cvs16r128-89	4,633	3,472	3,200	734	0.86	3.02	0.14	0.86	0.95	5.13	5.13	6.69	6.53
cvs16r70-62	3,278	2,112	1,846	773	0.52	4.08	0.00	0.52	0.64	26.25	26.25	30.92	30.98
cvs16r89-60	3,068	2,384	1,935	218	0.53	3.37	0.56	0.56	0.91	12.97	14.28	21.03	20.96
d10200	906	697	144	154	0.51	14.33	14.15	14.15	14.15	0.00	0.00	0.00	0.00
danoint	656	513	32	52	0.26	3.43	1.59	1.59	1.59	1.17	1.48	3.27	3.29
dcmulti	271	529	46	13	38.35	27.04	11.56	38.35	45.57	54.28	59.63	87.77	88.63
dfn-gwin-UUM	156	936	45	45	45.85	33.45	24.23	45.85	47.82	46.38	47.00	65.06	65.32
dg012142	1,987	1,899	397	27	0.48	0.10	0.03	0.48	0.48	0.57	0.61	1.10	1.06
eilB101	100	2,815	71	89	2.47	12.02	1.12	2.47	3.13	12.03	12.03	43.85	43.55
eild76	75	1,893	55	63	1.09	9.28	1.01	1.09	1.36	10.51	10.53	51.97	54.35
exp-1-500-5-5	550	990	131	3	28.15	9.37	1.71	28.15	28.51	33.48	36.52	79.01	80.37
f2gap201600	20	1,600	16	16	60.27	8.85	7.22	60.27	60.27	18.18	74.07	18.18	74.07
f2gap401600	40	1,600	29	29	62.97	4.72	2.52	62.97	63.31	24.07	93.90	24.07	93.90
f2gap801600	80	1,600	66	5	78.59	6.09	1.11	78.59	78.61	49.79	95.41	49.79	95.41
fiber	267	998	20	38	69.33	9.09	5.36	69.33	69.58	66.16	72.72	93.64	93.23
fixnet4	477	877	14	15	43.19	84.68	25.21	43.19	51.09	46.73	57.83	99.34	99.72
fixnet6	477	877	11	12	22.11	67.96	34.96	34.96	40.48	40.71	52.42	80.00	87.32
g200x740	934	1,472	144	25	30.19	11.09	2.57	30.19	31.05	76.54	77.62	94.59	94.23
g200x740i	940	1,480	113	30	1.93	2.78	2.30	2.30	3.34	71.52	71.62	88.64	89.08
gen-ip002	24	41	18	18	2.15	12.81	12.39	12.39	12.39	1.32	1.32	3.65	12.29
gen-ip016	24	28	8	13	0.31	6.11	5.96	5.96	5.96	0.31	5.57	0.52	5.88
gen-ip021	28	35	15	15	1.01	15.75	15.20	15.20	15.20	0.65	14.65	1.44	15.18
gen-ip036	46	29	17	17	2.44	17.79	17.73	17.73	17.73	2.10	16.82	2.84	17.63
ger50-17-ptp-pop-6t	544	4,860	265	8	48.43	0.91	0.78	48.43	48.43	48.63	48.77	89.44	89.51
gesa2-o	1,200	1,176	71	8	26.74	15.71	6.67	26.74	30.11	34.95	36.13	98.48	98.76
gesa2	1,344	1,176	43	8	23.86	35.40	6.70	23.86	28.45	66.12	70.07	99.09	99.31
gesa3_o	1,152	1,080	98	5	42.20	39.42	7.77	42.20	42.21	61.56	61.81	86.86	87.30
gesa3	1,296	1,080	59	59	7.28	48.28	33.52	33.52	35.12	40.51	47.22	73.69	82.09
glass4	390	316	64	6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
gmu-35-40	357	652	11	9	0.07	0.00	0.00	0.07	0.07	4.60	4.60	9.41	9.65
gmu-35-50	358	953	17	17	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.01
go19	361	361	357	357	2.00	15.41	15.18	15.18	15.18	2.23	14.91	5.26	15.27
graph20-20-1rand	4,810	1,924	416	4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
graphdraw-domain	865	254	100	2	0.00	0.00	0.00	0.00	0.00	0.72	0.72	4.90	4.84
graphdraw-gemcutter	474	166	66	26	0.00	0.00	0.00	0.00	0.00	1.95	1.95	6.84	6.84
gsvm2rl3	180	241	31	31	6.25	18.74	5.77	6.25	11.27	10.45	11.68	30.37	33.77
gsvm2rl5	300	401	45	45	0.42	11.55	7.87	7.87	8.05	1.73	4.58	4.04	9.65
gt2	28	173	14	3	87.06	2.04	1.66	87.06	87.06	49.97	50.04	62.54	69.54
gus-sch	834	1,378	49	17	45.20	18.96	7.64	45.20	45.20	51.12	54.15	84.89	84.36
h50x2450	2,451	4,763	57	10	12.19	19.01	6.52	12.19	16.97	48.18	50.27	93.99	93.83
haprp	694	756	254	170	50.50	0.35	0.17	50.50	50.50	100.00	100.00	100.00	100.00
harp2	92	1,013	27	27	17.28	15.51	11.31	17.28	17.64	16.99	17.26	39.22	40.15
hgms-det	4,599	950	150	6	0.41	0.01	0.00	0.41	0.41	0.88	0.89	2.62	2.11
ic97_potential	998	726	290	2	4.63	0.00	0.00	4.63	4.63	17.31	18.92	53.84	51.77
ic97_tension	215	523	119	4	5.86	0.00	0.00	5.86	5.86	33.27	33.27	85.93	87.26
icir97_tension	818	1,816	668	3	3.45	0.00	0.00	3.45	3.45	42.90	42.98	74.27	74.26
iis-100-0-cov	3,831	100	100	100	1.15	35.42	35.40	35.40	35.40	4.20	35.25	8.80	35.39
iis-bupa-cov	4,796	337	151	153	0.85	30.82	30.76	30.76	30.76	4.36	30.47	6.10	31.07
janos-us-DDM	755	2,179	81	4	11.98	3.75	2.22	11.98	11.98	23.84	23.84	29.01	29.01
k16x240	256	480	13	14	7.62	23.34	18.27	18.27	19.74	67.24	68.47	79.76	80.21
k16x240b	256	480	14	15	6.04	20.95	17.49	17.49	18.87	61.18	62.97	82.23	82.51

			# cuts		% gap closed								
Instance	Rows	Cols	G	V	G	DB	V	max(G,V)	V+G	GurF	V+GurF	GurL	V+GurL
khb05250	100	1,299	19	19	69.79	57.82	21.92	69.79	70.57	78.60	79.34	98.33	99.88
l152lav	97	1,987	13	52	10.84	62.58	54.33	54.33	54.49	1.00	52.26	4.08	54.26
lectsched-4-obj	3,221	1,875	418	2	0.00	0.00	0.00	0.00	0.00	57.14	57.14	100.00	100.00
lotsize	1,920	2,985	477	2	13.60	2.14	0.24	13.60	13.63	20.04	22.89	83.40	84.91
lrn	4,880	4,332	243	6	56.93	57.78	0.00	56.93	56.93	58.74	60.16	84.57	84.37
lseu	28	78	8	8	32.91	23.47	18.41	32.91	38.36	34.19	42.26	66.18	69.07
macrophage	2,308	1,673	386	46	9.35	3.76	3.76	9.35	12.46	88.82	88.82	99.41	99.44
mas074	13	148	12	12	6.67	13.51	13.31	13.31	13.33	1.83	2.12	8.31	13.36
mas076	12	148	11	11	6.42	13.30	12.15	12.15	12.48	2.33	3.66	14.26	14.71
mas284	68	148	20	20	0.87	34.87	34.10	34.10	34.10	0.51	33.80	10.07	33.92
maxgasflow	3,935	4,125	641	6	22.48	0.02	0.01	22.48	22.48	22.50	22.52	28.77	29.34
mc11	1,917	3,035	206	227	0.48	0.61	0.19	0.48	0.56	78.28	78.41	97.24	97.43
mc7	1,914	3,031	341	341	0.50	0.21	0.18	0.50	0.54	85.82	85.90	97.41	97.40
mc8	1,914	3,031	363	363	0.91	0.85	0.39	0.91	1.11	68.50	68.84	96.54	96.81
mcsched	1,793	1,450	1,224	1,224	0.03	3.65	2.69	2.69	2.69	0.06	0.21	0.64	2.76
mik-250-1-100-1	100	251	100	100	53.52	12.03	8.54	53.52	53.52	67.71	67.71	75.08	75.09
mik-250-20-75-1	76	269	75	75	65.25	15.47	12.23	65.25	65.56	65.25	65.25	82.91	82.91
mik-250-20-75-2	76	271	74	75	61.43	16.67	6.55	61.43	62.24	62.32	62.90	85.15	85.15
mik-250-20-75-3	76	268	75	20	61.59	20.55	4.69	61.59	62.25	61.59	62.25	76.70	76.68
mik-250-20-75-4	76	267	75	75	53.83	17.58	12.04	53.83	54.20	53.83	53.89	76.84	77.69
mik-250-20-75-5	76	266	75	75	63.44	18.04	5.98	63.44	64.03	63.44	64.01	79.34	82.10
milo-v12-6-r2-40-1	4,157	1,897	224	5	21.22	9.43	1.22	21.22	21.24	64.18	66.11	88.29	88.53
milo-v13-4-3d-3-0	655	295	23	23	1.03	9.19	8.53	8.53	8.56	0.59	8.39	0.60	8.54
mine-90-10	4,118	778	345	346	6.25	34.42	31.29	31.29	31.37	29.51	38.99	38.02	41.61
misc03	95	138	12	18	0.00	57.07	37.74	37.74	37.74	0.00	32.58	25.69	45.23
misc07	211	232	11	16	0.72	6.99	3.23	3.23	3.24	0.72	3.23	7.43	7.86
mkc1	2,738	4,751	67	54	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.33	0.85
mkc	961	2,933	116	21	6.08	5.47	3.46	6.08	6.44	6.05	9.40	46.26	46.58
mod008	6	319	6	6	24.37	26.26	21.61	24.37	27.04	7.80	16.47	89.94	91.36
mod013	62	96	5	5	4.41	51.54	43.58	43.58	46.12	31.44	41.05	78.20	82.47
modglob	286	354	27	30	17.40	25.31	5.71	17.40	20.22	65.46	69.37	96.37	97.22
mtest4ma	1,097	1,873	118	111	59.60	12.00	1.75	59.60	60.36	43.67	45.71	99.83	99.87
n13-3	224	506	16	4	43.24	47.39	26.52	43.24	45.12	65.84	66.73	91.36	91.84
n2seq36f	251	3,231	24	24	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
n4-3	884	2,950	36	36	10.72	23.10	14.95	14.95	17.70	35.73	35.94	81.12	81.78
n5-3	744	1,974	35	35	12.94	31.46	18.49	18.49	24.58	44.75	48.81	86.93	88.99
n6-3	1,693	4,988	38	38	21.28	18.47	9.40	21.28	23.94	49.49	51.34	94.75	95.00
n7-3	960	2,772	30	30	17.08	31.00	24.10	24.10	28.46	53.80	55.60	87.45	87.91
neos-1058477	966	769	28	28	0.76	76.16	0.00	0.76	0.76	10.93	10.93	99.20	88.66
neos-1215259	1,197	1,512	123	150	3.47	29.13	18.05	18.05	18.05	17.74	23.87	33.90	41.75
neos-1225589	650	1,250	25	20	3.94	12.09	4.82	4.82	6.12	85.07	85.44	98.98	99.09
neos-1281048	459	685	121	31	0.00	33.11	20.51	20.51	20.51	14.24	21.06	94.70	99.34
neos-1330346	1,620	2,628	565	32	0.00	14.29	0.00	0.00	0.00	2.04	2.04	8.16	6.12
neos-1396125	1,434	1,161	69	69	5.40	0.44	0.16	5.40	5.40	52.68	52.68	54.71	54.57
neos-1413153	2,500	2,451	257	269	14.37	30.65	3.92	14.37	14.79	16.13	16.31	47.02	47.57
neos-1415183	2,809	2,757	237	4	11.13	27.67	15.96	15.96	16.15	21.99	21.99	26.07	58.86
neos-1420205	341	231	44	3	0.00	20.00	20.00	20.00	20.00	20.00	20.00	35.14	35.14
neos-1480121	183	151	7	7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
neos-1489999	870	438	438	438	0.47	27.75	18.36	18.36	18.39	0.61	15.27	2.15	18.60
neos-1582420	2,487	2,407	295	295	3.19	30.81	28.92	28.92	28.92	11.54	11.55	50.90	53.46
neos-1595230	677	490	114	5	4.01	7.14	7.14	7.14	7.14	0.34	4.08	1.54	7.43

			# cuts		% gap closed								
Instance	Rows	Cols	G	V	G	DB	V	max(G,V)	V+G	GurF	V+GurF	GurL	V+GurL
neos-1599274	1,169	4,200	60	1	34.65	0.00	0.00	34.65	34.65	51.66	51.66	51.66	51.66
neos-1601936	3,088	4,022	589	25	100.00	100.00	50.00	100.00	100.00	100.00	100.00	100.00	100.00
neos-1605061	3,447	4,023	729	70	36.21	17.10	0.00	36.21	36.21	86.21	86.21	100.00	100.00
neos-1605075	3,440	4,085	966	434	2.90	0.00	0.00	2.90	2.90	73.67	73.67	89.74	91.68
neos-1616732	1,026	200	200	200	1.47	24.78	24.74	24.74	24.74	1.90	24.72	5.86	24.72
neos-1620807	405	231	39	1	0.00	16.67	16.67	16.67	16.67	0.00	16.67	0.00	16.67
neos-2328163-agri	1,924	2,236	178	84	0.59	0.81	0.48	0.59	0.76	9.96	9.98	86.74	87.77
neos-3024952-loue	3,633	3,218	391	5	2.00	0.00	0.00	2.00	2.00	24.90	24.96	57.06	57.00
neos-3046601-motu	282	163	11	11	0.00	0.08	0.08	0.08	0.08	72.81	92.88	99.75	99.65
neos-3046615-murg	249	145	14	10	0.00	0.00	0.00	0.00	0.00	90.69	90.69	96.84	96.95
neos-3072252-nete	429	570	93	1	19.69	0.93	0.00	19.69	19.69	19.47	19.47	73.16	73.93
neos-3083819-nubu	340	2,230	12	29	42.07	33.87	12.43	42.07	46.60	64.14	64.14	86.83	91.88
neos-3118745-obra	137	1,130	24	6	18.80	0.00	0.00	18.80	18.80	15.31	15.31	58.78	53.06
neos-3216931-puriri	3,427	2,993	409	22	0.93	21.97	0.00	0.93	0.93	1.64	1.71	5.32	5.32
neos-3373491-avoca	1,267	2,152	66	2	0.01	0.00	0.00	0.01	0.01	0.01	0.01	0.03	0.03
neos-3381206-awhea	479	2,375	478	3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
neos-3421095-cinca	597	292	56	35	0.00	0.00	0.00	0.00	0.00	3.60	3.68	83.16	83.16
neos-3592146-hawea	765	4,205	240	7	16.02	4.84	2.20	16.02	16.07	15.85	18.09	28.59	28.61
neos-3610040-iskar	215	275	18	2	7.12	22.23	14.86	14.86	14.86	9.86	15.41	31.03	41.32
neos-3610051-istra	389	444	27	7	5.56	26.41	26.41	26.41	26.41	16.78	25.66	24.53	27.74
neos-3610173-itata	435	491	26	3	7.81	43.61	0.51	7.81	7.81	19.17	19.27	55.09	57.36
neos-3611447-jijia	262	324	23	4	19.97	20.29	5.51	19.97	19.97	46.09	46.09	66.47	66.53
neos-3611689-kaihu	256	317	27	11	17.83	21.49	20.48	20.48	22.66	50.89	53.86	61.28	60.70
neos-3627168-kasai	1,190	1,400	152	152	17.11	0.73	0.59	17.11	17.11	6.06	6.55	31.66	31.62
neos-3660371-kurow	1,524	1,488	144	160	0.22	0.81	0.61	0.61	0.66	0.23	0.59	1.50	1.46
neos-3665875-lesum	3,184	3,128	179	2	1.27	0.00	0.00	1.27	1.27	6.04	6.04	92.35	92.63
neos-3754480-nidda	402	253	41	41	19.64	22.88	22.28	22.28	27.34	48.50	48.71	50.47	50.99
neos-3762025-ognon	2,636	4,507	339	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
neos-4333464-siret	2,278	2,186	15	15	5.65	4.88	4.37	5.65	8.31	21.68	21.93	68.13	68.83
neos-4333596-skien	438	390	21	43	20.84	10.05	7.08	20.84	20.84	38.54	39.87	67.11	67.82
neos-4387871-tavua	3,918	3,638	33	33	2.04	6.48	4.08	4.08	5.00	15.14	16.68	61.26	62.55
neos-4393408-tinui	3,918	3,640	33	33	9.03	28.73	22.87	22.87	28.53	4.20	27.13	18.26	39.45
neos-4650160-yukon	1,551	980	83	7	85.84	2.07	0.61	85.84	85.85	90.49	90.60	93.19	93.19
neos-480878	1,265	746	21	22	1.84	18.72	7.22	7.22	8.12	1.80	6.06	7.92	11.58
neos-4954672-berkel	315	630	60	5	8.44	5.05	2.28	8.44	8.90	12.75	16.94	53.47	53.57
neos-501453	13	52	1	1	70.00	0.08	0.08	70.00	70.00	70.00	70.00	70.00	70.00
neos-504674	1,099	617	153	153	9.86	11.99	8.59	9.86	12.46	14.16	15.80	22.96	22.63
neos-504815	857	483	115	115	16.04	15.51	7.60	16.04	16.41	22.14	22.58	29.76	29.85
neos-5051588-culgoa	4,076	3,902	112	22	0.00	4.74	4.35	4.35	4.35	30.92	31.61	46.74	49.38
neos-5075914-elvire	1,975	1,951	216	125	32.92	8.17	0.00	32.92	32.92	1.47	1.68	4.04	4.04
neos-5078479-escaut	2,201	2,025	205	7	5.62	0.00	0.00	5.62	5.62	0.45	0.94	1.89	1.89
neos-512201	1,082	611	141	49	7.20	23.15	15.10	15.10	16.80	16.74	23.54	25.46	29.86
neos-5140963-mincio	184	196	11	24	16.99	23.74	22.77	22.77	23.16	18.97	22.91	27.56	29.33
neos-5182409-nasivi	404	1,904	142	143	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	2.97
neos-522351	1,705	1,524	104	29	58.21	29.93	29.92	58.21	58.21	79.77	79.77	99.83	99.81
neos-5261882-treska	2,469	2,031	20	48	3.44	4.95	3.08	3.44	4.41	3.44	3.44	34.45	34.73
neos-538867	1,042	666	47	47	0.00	5.25	1.99	1.99	1.99	0.00	1.43	0.00	7.30
neos-538916	1,154	704	53	53	0.00	3.48	2.51	2.51	2.51	0.00	1.36	0.00	6.62
neos-547911	357	2,352	84	87	0.91	6.43	0.00	0.91	0.91	1.26	1.53	10.16	10.08
neos-555884	1,952	2,478	108	62	14.47	17.11	0.00	14.47	14.47	87.94	87.94	93.42	93.42
neos-565815	1,746	1,248	238	28	12.16	100.00	91.80	91.80	91.80	9.77	83.16	28.40	93.14

			# cuts		% gap closed								
Instance	Rows	Cols	G	V	G	DB	V	max(G,V)	V+G	GurF	V+GurF	GurL	V+GurL
neos-570431	925	495	215	215	1.36	39.78	38.92	38.92	38.92	14.23	38.54	38.85	43.32
neos-574665	3,020	326	44	30	3.39	4.28	4.16	4.16	4.19	18.53	21.51	49.77	52.68
neos-584851	465	389	231	231	0.94	8.70	8.68	8.68	9.38	3.41	11.31	14.15	16.03
neos-585192	1,149	1,314	16	12	0.15	19.01	5.59	5.59	5.67	6.45	8.96	8.35	10.63
neos-585467	948	1,067	12	12	0.15	37.57	20.65	20.65	20.70	9.16	18.34	12.19	24.97
neos-593853	242	1,230	10	10	22.04	49.15	41.27	41.27	42.24	24.20	41.91	56.50	54.15
neos-595904	1,432	2,835	47	47	25.57	8.44	5.62	25.57	25.57	55.76	57.68	98.10	97.84
neos-598183	452	916	19	19	16.89	24.89	15.83	16.89	22.52	4.34	18.45	95.73	97.58
neos-603073	452	974	20	20	4.34	4.57	3.26	4.34	5.41	2.28	3.88	29.75	31.30
neos-631517	343	1,037	202	5	21.88	0.18	0.14	21.88	21.88	15.76	40.91	99.91	99.90
neos-686190	3,658	3,660	55	74	4.43	61.71	54.77	54.77	54.77	5.18	5.18	11.37	53.55
neos-691058	1,953	2,871	354	151	54.80	0.00	0.00	54.80	54.80	72.78	72.78	97.30	99.60
neos-717614	811	3,049	138	14	66.48	3.98	2.07	66.48	67.26	87.61	88.98	98.39	98.51
neos-775946	1,381	2,876	160	8	36.93	43.49	31.88	36.93	36.93	46.09	46.09	97.42	98.75
neos-796608	64	104	15	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
neos-801834	3,300	3,260	481	450	2.95	62.64	34.64	34.64	34.64	2.67	17.74	9.60	35.48
neos-803219	621	360	20	20	13.00	69.68	61.45	61.45	62.32	18.57	49.12	18.57	62.15
neos-803220	611	350	20	20	14.88	55.11	41.84	41.84	44.29	21.92	27.84	22.79	43.56
neos-806323	1,025	650	120	120	25.78	21.77	21.73	25.78	32.39	22.58	23.37	37.18	37.86
neos-807639	953	580	80	80	19.99	77.18	75.12	75.12	75.32	19.99	70.61	37.05	75.43
neos-807705	1,024	604	77	91	10.26	34.36	33.92	33.92	34.33	17.57	33.10	21.42	35.22
neos-810326	1,730	1,701	159	198	3.80	60.83	41.11	41.11	41.11	13.94	33.63	27.47	43.94
neos-831188	2,153	4,580	892	79	0.47	16.94	7.01	7.01	7.06	0.77	1.01	1.24	6.76
neos-839859	2,601	1,950	141	132	11.17	26.97	4.27	11.17	12.49	4.07	6.92	17.68	18.78
neos-862348	916	1,810	117	15	54.07	0.00	0.00	54.07	54.07	57.56	57.56	74.62	79.23
neos-880324	173	135	48	2	0.00	14.54	14.54	14.54	14.54	12.06	14.54	25.13	25.80
neos-886822	1,057	1,025	376	1	3.18	3.05	0.43	3.18	3.51	2.04	2.33	2.84	2.52
neos-892255	1,675	1,521	189	197	0.00	33.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00
neos-906865	1,562	1,160	19	29	13.90	37.52	36.36	36.36	36.82	13.76	33.91	18.48	36.79
neos-911880	83	888	48	48	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
neos-911970	107	888	43	7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	12.33	12.54
neos-916792	1,413	1,465	88	89	4.06	15.72	7.87	7.87	8.91	0.12	7.97	3.24	8.83
neos-942830	589	831	250	3	6.25	0.00	0.00	6.25	6.25	18.45	18.45	30.36	35.04
neos14	438	677	130	8	55.47	7.14	2.63	55.47	56.08	62.56	63.30	80.22	81.08
neos15	461	700	153	13	48.80	6.30	4.15	48.80	49.33	60.48	61.14	77.79	79.19
neos16	346	208	90	5	2.78	0.00	0.00	2.78	2.78	50.86	58.80	100.00	100.00
neos17	486	511	7	171	0.00	1.16	0.50	0.50	0.50	84.00	84.29	92.10	93.43
neos18	3,052	758	567	2	11.90	13.04	13.04	13.04	24.94	60.87	60.87	63.31	63.59
neos22	2,594	2,066	66	66	5.94	5.38	5.25	5.94	11.19	48.49	56.98	83.66	62.80
neos2	793	1,487	20	20	3.22	0.00	0.00	3.22	3.22	4.91	4.91	20.86	22.43
neos3	1,146	2,160	28	18	2.49	2.61	0.74	2.49	2.65	5.89	6.78	20.70	23.53
neos5	63	63	35	1	11.11	37.50	37.50	37.50	37.50	4.17	37.50	17.58	37.50
neos7	1,807	1,501	19	1	23.17	30.06	28.83	28.83	29.12	35.06	35.19	46.56	46.91
newdano	520	449	52	5	7.76	28.32	28.32	28.32	28.32	7.47	28.32	10.66	28.32
nexp-150-20-1-5	2,033	3,931	46	13	0.00	5.17	0.00	0.00	0.00	17.24	17.24	29.02	44.21
nexp-50-20-1-1	267	443	33	8	4.48	8.39	8.39	8.39	9.11	100.00	100.00	100.00	100.00
nexp-50-20-4-2	493	1,101	41	5	25.76	0.11	0.11	25.76	25.76	53.95	58.70	94.36	94.48
nh97_potential	1,836	1,180	545	5	0.00	0.00	0.00	0.00	0.00	38.10	45.24	69.05	73.81
nobel-eu-DBE	726	3,460	46	47	10.84	0.00	0.00	10.84	10.84	13.09	22.91	49.86	52.10
ns1208400	2,054	2,682	438	1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	91.07	71.43
ns1606230	3,459	3,261	905	90	4.16	1.70	0.41	4.16	4.16	73.36	73.36	100.00	100.00

			# cuts		% gap closed								
Instance	Rows	Cols	G	V	G	DB	V	max(G,V)	V+G	GurF	V+GurF	GurL	V+GurL
ns1688347	2,336	1,235	249	4	10.38	29.56	28.30	28.30	28.30	39.62	39.62	39.62	39.62
ns1830653	1,310	586	214	124	10.57	28.70	18.07	18.07	18.71	15.00	19.57	27.80	30.53
ns2081729	1,130	601	14	7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ns894788	923	2,012	322	253	100.00	9.76	4.11	100.00	100.00	100.00	100.00	100.00	100.00
nsa	600	176	13	13	2.97	22.39	19.52	19.52	19.52	3.91	19.59	8.54	19.80
nsrand-ipx	510	3,739	67	26	17.29	12.43	4.01	17.29	19.53	13.59	17.77	60.88	60.46
nu120-pr12	534	1,402	14	8	27.71	74.23	47.43	47.43	55.28	31.41	56.38	57.39	61.44
nu25-pr12	534	1,402	11	11	28.21	51.43	21.21	28.21	35.29	33.03	38.59	45.20	48.08
p0282	157	199	24	24	3.18	13.48	7.63	7.63	8.28	75.97	75.99	99.78	99.90
p0548	119	371	36	2	38.09	56.39	44.40	44.40	70.14	88.23	89.51	99.89	99.91
p100x588b	688	1,176	70	11	8.63	5.43	2.87	8.63	9.98	78.81	79.98	92.08	92.25
p200x1188c	1,388	2,376	3	5	0.40	0.89	0.78	0.78	0.83	52.05	52.18	93.04	94.92
p2756	590	2,143	97	4	93.51	11.61	2.46	93.51	93.51	90.00	91.63	98.73	98.64
p6000	1,725	4,596	2	2	41.65	67.98	60.72	60.72	63.20	35.70	35.70	36.02	38.78
p6b	502	451	368	375	0.91	13.54	13.38	13.38	13.38	0.67	12.90	2.14	13.34
p80x400b	456	768	51	3	11.19	7.70	2.86	11.19	12.16	74.75	76.56	90.66	90.77
pg5_34	225	2,600	88	4	23.45	6.70	1.01	23.45	23.91	93.75	95.34	98.80	98.79
pg	125	2,700	91	2	24.29	9.18	0.54	24.29	24.64	81.74	81.74	84.40	84.40
pigeon-10	525	210	72	72	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
piperout-03	3,529	4,414	766	4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.10	20.38
piperout-d20	3,557	4,499	142	144	100.00	100.00	35.33	100.00	100.00	100.00	100.00	100.00	100.00
piperout-d27	3,867	4,890	164	58	100.00	0.00	0.00	100.00	100.00	100.00	100.00	100.00	100.00
pipex	25	48	6	6	35.56	37.81	32.89	35.56	39.52	16.98	42.88	93.52	98.39
pp08aCUTS	228	235	46	46	31.53	21.04	15.88	31.53	32.43	53.08	58.88	90.89	90.93
pp08a	133	234	52	3	51.25	22.40	5.50	51.25	51.25	62.60	63.63	95.61	95.20
probportfolio	302	320	125	20	25.14	20.41	0.00	25.14	25.41	12.80	15.59	32.85	33.54
prod1	75	117	40	40	4.66	42.45	40.11	40.11	40.34	10.68	38.85	30.58	41.44
prod2	92	182	44	44	2.31	39.64	27.63	27.63	27.65	5.92	25.35	29.86	33.04
protfold	2,110	1,835	490	46	4.95	9.82	3.30	4.95	6.71	4.46	6.93	10.79	11.71
qiu	1,192	840	36	36	0.86	41.96	32.56	32.56	32.63	0.98	30.94	5.15	32.78
qnet1_o	237	1,314	10	10	29.71	63.67	57.94	57.94	61.28	46.49	62.88	87.96	89.90
qnet1	360	1,417	40	47	13.25	25.88	6.32	13.25	16.14	23.77	25.18	80.54	84.05
queens-30	900	900	900	756	2.29	2.98	0.17	2.29	2.29	0.27	0.28	2.24	2.21
r50x360	407	716	43	43	15.74	13.46	10.45	15.74	18.08	77.49	81.48	94.79	95.38
r80x800	880	1,600	57	9	31.60	20.13	6.72	31.60	33.67	70.35	73.32	89.92	89.75
railway_8_1_0	2,060	1,557	317	5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ran12x21	285	504	17	17	3.23	24.92	24.13	24.13	24.19	34.69	42.37	61.85	61.96
ran13x13	195	338	18	18	11.39	27.85	24.81	24.81	25.90	39.21	46.96	62.31	63.15
ran14x18-disj-8	449	506	80	86	0.14	9.70	5.92	5.92	5.92	0.93	5.84	8.07	9.27
ran14x18	284	504	18	18	5.40	10.47	8.34	8.34	8.89	20.71	25.23	49.35	55.77
ran16x16	288	512	20	20	7.49	20.19	17.09	17.09	17.99	34.93	39.49	62.05	62.15
reblock115	3,953	1,062	814	396	11.61	46.66	24.27	24.27	27.78	19.45	31.57	31.91	38.82
reblock67	1,928	585	442	442	10.68	31.20	27.93	27.93	28.64	25.45	30.99	40.37	41.67
rgn	24	180	19	19	9.66	64.07	9.64	9.66	15.26	41.17	44.63	66.84	84.86
rlp1	52	316	47	6	11.18	0.00	0.00	11.18	11.18	15.76	15.82	32.62	32.45
rococoB10-011000	765	3,555	274	302	12.59	1.27	0.91	12.59	12.59	22.29	22.31	49.85	49.91
rococoC10-001000	617	2,483	157	157	32.89	0.03	0.01	32.89	32.89	35.13	35.13	81.24	81.36
roll3000	981	892	157	3	4.25	1.43	0.27	4.25	4.39	33.14	51.70	88.07	89.55
rout	290	555	40	34	1.51	17.48	17.17	17.17	17.17	0.71	16.55	3.98	17.17
roy	147	139	13	13	11.98	39.13	26.70	26.70	28.19	30.05	35.14	89.54	91.27
sentoy	30	60	8	8	19.31	61.36	57.65	57.65	58.09	13.73	14.46	39.60	59.34

			# cuts		% gap closed								
Instance	Rows	Cols	G	V	G	DB	V	max(G,V)	V+G	GurF	V+GurF	GurL	V+GurL
set1al	432	652	196	3	98.83	3.66	1.11	98.83	98.83	99.15	99.15	99.96	99.96
set1ch	423	643	129	71	28.20	2.19	2.07	28.20	28.20	67.24	67.26	99.91	99.91
set1cl	431	651	200	3	100.00	3.59	1.09	100.00	100.00	100.00	100.00	100.00	100.00
set3-10	2,481	2,677	176	3	17.62	3.77	0.00	17.62	18.81	8.77	8.80	39.66	39.72
set3-15	2,537	2,677	174	4	7.36	1.59	0.16	7.36	7.36	7.40	7.65	29.61	29.67
set3-20	2,537	2,677	176	2	10.55	1.09	0.00	10.55	11.04	8.45	9.22	33.73	33.71
seymour-disj-10	4,777	1,031	614	621	0.21	5.99	0.89	0.89	0.89	0.17	0.79	0.37	1.13
seymour1	4,452	897	58	74	17.43	44.27	37.05	37.05	37.05	37.38	41.52	46.59	47.78
seymour	4,369	893	474	481	6.08	19.22	15.26	15.26	15.45	23.48	23.72	43.18	43.86
sorrell8	1,554	1,504	1,063	1,075	1.74	6.70	5.97	5.97	6.38	10.24	11.85	16.32	16.42
sp150x300d	269	419	31	6	1.67	6.72	4.88	4.88	6.12	78.94	79.49	89.46	89.12
sp98ir	1,400	1,586	110	111	3.69	29.11	27.12	27.12	27.18	13.59	14.53	49.39	51.60
square23	560	4,345	86	90	9.18	42.26	2.48	9.18	9.18	3.07	5.49	14.60	14.60
stein27_nocard	117	27	27	2	8.30	55.56	55.56	55.56	55.56	14.78	55.56	29.08	55.56
stein45_nocard	330	45	45	1	4.69	53.96	53.96	53.96	53.96	12.69	59.09	26.34	59.09
supportcase17	1,421	736	247	7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.43	0.40
supportcase20	598	896	270	7	35.68	2.83	1.49	35.68	36.28	41.02	42.00	80.74	81.89
supportcase25	3,832	3,664	110	115	5.41	7.57	4.20	5.41	7.99	31.03	32.73	44.47	44.59
supportcase26	830	436	18	2	0.00	0.00	0.00	0.00	0.00	5.16	5.16	23.99	24.25
ta1-UUM	424	1,875	40	10	4.09	3.95	0.00	4.09	4.09	0.00	0.00	0.00	0.00
timtab1CUTS	349	365	121	122	0.73	9.26	7.88	7.88	7.90	3.71	8.08	11.35	11.87
timtab1	165	365	128	5	24.08	11.49	4.44	24.08	24.08	32.68	33.03	59.87	59.95
timtab2	285	625	214	3	13.66	9.25	2.36	13.66	13.66	24.68	24.92	47.85	48.90
toll-like	3,208	2,146	435	179	4.38	2.87	2.73	4.38	6.48	64.76	67.67	91.43	91.44
tr12-30	710	1,028	321	2	58.36	1.74	0.48	58.36	58.36	60.24	60.44	98.55	98.69
traininstance6	497	265	21	2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
uct-subprob	1,595	1,932	594	23	2.61	9.67	6.76	6.76	7.33	19.13	19.64	48.46	49.10
umts	1,749	1,648	275	276	0.97	0.21	0.11	0.97	0.97	1.41	1.46	4.79	5.02
usAbbrv-8-25_70	2,813	2,073	722	4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
vpm1	128	188	16	10	16.93	7.79	4.67	16.93	16.93	45.86	56.94	68.16	72.13
vpm2	127	187	25	25	17.85	14.29	7.83	17.85	19.71	43.30	50.63	73.26	73.53
zib54-UUE	1,114	3,726	56	56	10.70	17.64	7.60	10.70	15.60	43.11	54.10	68.33	68.23
Average					16.51	18.37	12.01	23.01	23.92	28.93	34.80	48.32	52.81
Wins						178	132		226		250		234

Table 11: Time (in seconds) and number nodes taken to solve each instance, for the disjunction size with best solving time with VPCs per instance. The table is sorted by column 5 ("V" under "Time (s)").

		# cuts	Time (s)			Nodes (#)	
Instance	# terms	V	Gur	V	Gen	Gur	V
neos-796608	16	2	0.00	0.00	0.02	1	1
neos-501453	64	1	0.00	0.00	0.02	1	1
gt2	4	3	0.00	0.00	0.01	1	1
vpm1	8	5	0.00	0.00	0.02	1	1
set1cl	64	3	0.00	0.00	0.35	1	1
pipex	4	6	0.02	0.01	0.00	7	1
nexp-50-20-1-1	16	2	0.01	0.01	0.08	1	1
p0548	64	3	0.01	0.01	0.20	1	1
sp150x300d	64	6	0.01	0.01	0.23	1	1
set1al	32	2	0.01	0.01	0.18	2	2
haprp	2	3	0.01	0.01	0.06	1	1
f2gap201600	32	16	0.02	0.01	6.78	1	1
roy	4	7	0.02	0.02	0.01	7	8
f2gap401600	2	29	0.02	0.02	0.26	1	1
mod008	64	6	0.02	0.02	0.40	3	1
f2gap801600	8	5	0.03	0.02	0.74	1	1
mod013	64	5	0.03	0.03	0.06	56	41
khb05250	2	3	0.02	0.03	0.05	1	1
beavma	32	3	0.03	0.03	0.07	1	1
p0282	2	14	0.03	0.03	0.01	1	1
neos22	4	66	0.03	0.03	0.50	1	1
bm23	32	6	0.04	0.04	0.03	149	168
neos-3046601-motu	16	5	0.04	0.04	0.03	20	29
neos-1599274	16	2	0.04	0.04	97.62	1	1
bell5	4	2	0.04	0.04	0.00	711	687
fixnet4	32	15	0.05	0.04	0.52	1	1
lseu	2	8	0.05	0.05	0.00	226	148
sentoy	64	8	0.08	0.05	0.15	73	67
set1ch	32	8	0.05	0.05	0.26	37	41
neos-1058477	2	28	0.06	0.05	0.12	2	1
modglob	32	30	0.06	0.06	0.35	37	27
control30-3-2-3	16	24	0.07	0.07	0.07	92	81
fiber	16	18	0.08	0.07	0.34	59	58
cap6000	8	2	0.08	0.08	2.50	1	1
p6000	8	2	0.08	0.08	2.51	1	1
bell4	2	3	0.09	0.08	0.00	861	892
gesa2	2	7	0.10	0.08	0.05	8	3
neos-3610173-itata	16	3	0.08	0.09	0.44	71	65
p2756	8	4	0.11	0.09	0.47	38	23
neos-3611447-jijia	2	3	0.10	0.09	0.02	325	284
dcmulti	8	11	0.10	0.09	0.09	34	31
neos-3610040-iskar	8	25	0.12	0.10	0.10	550	372
lectsched-4-obj	64	3	0.09	0.10	4.86	1	1
bppc8-02	4	13	0.11	0.10	0.05	600	523
neos-1225589	4	25	0.10	0.11	0.12	18	17
neos-522351	2	3	0.10	0.11	0.09	4	3

		# cuts	Time (s)			Nodes (#)	
Instance	# terms	V	Gur	V	Gen	Gur	V
nu120-pr12	2	16	0.10	0.11	0.06	32	25
n13-3	64	4	0.12	0.11	0.34	159	152
fixnet6	4	12	0.14	0.11	0.06	21	14
gesa2-o	16	7	0.14	0.12	0.36	32	18
gesa3	4	24	0.11	0.12	0.13	27	17
neos-880324	2	6	0.14	0.12	0.02	437	354
neos-775946	2	7	0.13	0.13	0.55	1	1
p200x1188c	4	5	0.18	0.13	0.33	1	1
vpm2	8	8	0.13	0.13	0.02	439	442
nu25-pr12	16	11	0.14	0.13	0.47	90	88
neos-3610051-istra	32	19	0.17	0.13	3.70	329	227
gesa3_o	64	5	0.13	0.14	1.85	39	50
blend2	8	6	0.14	0.14	0.03	501	424
rgn	32	19	0.14	0.14	0.13	1,210	1,068
stein27_nocard	16	2	0.16	0.14	0.06	2,719	2,780
beasleyC1	4	6	0.16	0.15	0.07	11	4
mtest4ma	8	16	0.16	0.16	0.38	1	1
bell3b	32	9	0.13	0.16	0.03	1,570	1,924
neos-717614	4	14	0.17	0.16	0.40	27	39
neos-1480121	16	2	0.23	0.16	0.12	1,297	1,009
nexp-50-20-4-2	2	3	0.18	0.16	0.04	37	11
qnet1_o	8	10	0.17	0.17	0.34	30	24
bell3a	16	1	0.18	0.18	0.01	2,928	2,928
neos-1489999	4	85	0.17	0.19	0.30	91	101
neos-598183	16	19	0.23	0.19	1.43	36	31
neos-593853	8	10	0.19	0.20	0.31	354	469
misc03	64	18	0.33	0.21	0.85	1,374	579
mik-250-20-75-2	32	39	0.23	0.22	0.26	1,362	1,092
pp08aCUTS	64	46	0.27	0.24	1.94	579	510
gus-sch	2	26	0.29	0.25	0.10	114	67
beasleyC2	8	32	0.26	0.25	0.30	19	14
neos-3118745-obra	16	4	0.20	0.26	1.83	207	280
qnet1	2	47	0.27	0.26	0.18	25	31
neos-3083819-nubu	64	29	0.29	0.28	64.62	253	207
neos-555884	64	21	0.26	0.29	72.94	11	11
n2seq36f	16	3	0.23	0.29	7.73	209	251
app3	8	13	0.29	0.29	1.88	77	80
neos-595904	16	2	0.36	0.29	4.83	48	29
pp08a	2	4	0.29	0.32	0.01	779	843
piperout-03	16	3	0.32	0.34	4.21	20	1
neos-862348	4	9	0.38	0.35	0.43	90	55
piperout-d20	2	26	0.35	0.39	0.56	1	1
neos-3611689-kaihu	2	14	0.51	0.39	0.03	1,850	1,403
macrophage	32	14	0.43	0.39	2.05	67	40
l152lav	2	52	0.37	0.42	0.79	394	357
23588	4	75	0.54	0.47	0.36	882	951
neos-1420205	8	4	0.71	0.48	0.19	3,521	2,168
mik-250-20-75-5	4	2	0.58	0.55	0.02	1,299	1,497
mik-250-20-75-1	64	75	0.62	0.57	2.82	1,469	1,672
mik-250-20-75-3	2	1	0.58	0.61	0.01	1,756	1,728
neos-585467	4	12	0.69	0.61	0.13	119	103

		# cuts	Time (s)			Nodes (#)	
Instance	# terms	V	Gur	V	Gen	Gur	V
neos2	4	20	0.63	0.64	0.35	778	757
neos17	64	171	1.01	0.65	21.76	1,785	1,859
neos-565815	16	37	8.30	0.70	318.52	932	1
nexp-150-20-1-5	4	13	1.01	0.71	0.75	369	296
railway_8_1_0	32	3	0.94	0.72	1.35	1,329	963
piperout-d27	16	17	0.68	0.75	12.99	9	18
neos-807639	2	59	0.76	0.86	0.08	952	1,011
neos16	4	8	1.11	0.91	0.01	1,313	1,082
neos-1415183	16	4	1.78	0.91	43.46	851	193
neos-1281048	32	37	1.15	0.97	144.23	235	122
neos-691058	16	64	1.05	1.02	3,601.91	1	6
blp-ir98	16	45	1.13	1.04	61.18	202	111
neos7	2	3	1.17	1.09	0.09	1,750	1,498
neos-631517	16	2	8.72	1.24	0.83	8,295	634
neos-3381206-awhea	64	3	1.22	1.25	66.12	1,094	1,310
mik-250-20-75-4	16	50	1.85	1.31	0.32	7,435	5,509
neos18	8	2	1.80	1.48	0.96	2,301	1,866
neos-803219	8	5	1.22	1.48	0.27	2,950	3,508
neos-585192	4	16	1.55	1.51	0.21	881	840
eild76	4	63	1.51	1.53	7.75	176	179
neos-603073	2	15	1.70	1.59	0.06	780	882
neos-807705	32	91	1.29	1.59	2.15	1,338	1,702
neos-1413153	2	25	1.44	1.67	0.55	716	930
aligninq	64	163	2.58	1.67	3,616.38	629	576
exp-1-500-5-5	64	3	1.80	1.69	0.79	616	600
neos-570431	4	204	1.81	1.72	7.14	676	627
10teams	2	77	1.82	1.75	5.01	280	219
nsa	64	13	3.52	1.78	2.38	6,241	3,713
beasleyC3	8	61	1.86	1.79	0.56	678	542
nh97_potential	64	5	1.88	1.87	2.13	729	870
neos3	8	4	2.09	1.93	0.51	2,394	2,318
neos-806323	16	20	2.22	2.13	0.21	1,288	1,245
neos-584851	4	12	2.26	2.17	0.20	1,444	1,487
pg	32	2	2.39	2.39	3.23	921	921
mkc1	4	54	1.97	2.42	15.41	3,451	4,405
h50x2450	4	63	2.53	2.47	1.22	1,311	865
n7-3	16	23	3.05	2.53	10.93	575	476
neos-803220	8	20	2.73	2.60	0.18	7,785	7,204
mas284	2	20	2.27	2.67	0.03	14,635	14,512
aflow30a	2	25	2.64	2.75	0.06	2,665	2,576
neos-801834	16	482	2.84	3.03	720.73	402	370
n6-3	2	38	3.93	3.10	0.55	1,083	814
neos-504815	4	14	3.46	3.17	0.13	3,147	2,917
n5-3	2	27	4.78	3.34	0.14	1,120	847
binkar10_1	4	38	4.16	3.50	0.31	5,264	4,290
ic97_tension	8	3	2.44	3.58	0.04	3,725	4,803
neos-892255	8	116	3.21	3.65	136.85	524	540
traininstance6	8	3	4.05	3.81	0.12	13,733	11,945
stein45_nocard	16	1	3.92	3.88	0.22	44,351	47,875
ran13x13	64	18	3.91	3.89	0.41	6,494	6,451
neos-1582420	2	221	4.23	3.93	7.31	825	755

		# cuts	Time (s)			Nodes (#)	
Instance	# terms	V	Gur	V	Gen	Gur	V
mc11	16	227	4.05	4.05	3.00	569	558
neos-512201	16	143	4.22	4.30	5.90	2,455	2,526
roll3000	64	5	6.04	4.78	25.74	1,141	1,007
r50x360	2	19	4.41	5.26	0.03	1,475	1,695
neos-906865	32	29	6.87	5.29	68.48	6,066	4,296
arki001	2	39	7.52	5.48	0.14	6,152	5,037
ran12x21	16	17	7.53	5.91	0.14	9,972	8,749
neos-839859	8	87	6.40	5.94	128.13	2,247	1,899
ns1688347	32	4	5.09	6.29	42.24	822	1,047
neos-480878	2	22	8.60	6.32	0.08	8,721	5,086
gsvm2rl3	64	31	6.76	6.53	1.63	11,447	10,296
neos-538867	64	47	9.54	6.92	19.40	13,503	7,650
bc1	4	5	8.62	7.19	2.21	1,743	1,269
sp98ir	4	111	8.64	7.35	8.12	1,570	1,422
g200x740	2	5	8.21	8.04	0.06	3,898	3,547
neos-4393408-tinui	32	23	16.72	8.61	6.86	10,577	2,504
neos-1215259	16	150	9.01	9.11	198.27	1,200	1,059
neos-504674	2	23	9.86	9.20	0.09	6,039	5,295
ns1208400	32	1	6.42	9.31	365.97	272	412
misc07	4	16	12.29	9.54	0.06	30,047	27,125
rout	8	42	9.34	9.55	1.28	18,180	13,302
30n20b8	4	190	15.51	9.72	151.13	809	566
neos-3046615-murg	16	7	11.04	9.75	0.03	23,277	21,494
harp2	16	27	9.70	10.12	0.63	14,622	14,364
neos-686190	16	96	11.69	10.66	293.59	4,265	3,748
prod1	32	40	14.26	10.89	0.99	39,637	32,829
mc8	8	21	14.44	11.05	0.87	1,159	930
neos-810326	4	143	10.74	11.20	13.31	1,477	1,387
ns1606230	2	90	14.50	11.56	326.21	173	58
bienst1	4	26	14.29	12.18	1.49	11,326	9,389
mine-90-10	4	4	15.06	12.67	0.49	13,319	11,051
neos-5051588-culgoa	4	4	126.94	12.74	3.53	18,926	1,143
mc7	4	25	19.06	14.34	0.44	1,175	859
square23	4	90	14.94	15.47	181.14	417	439
neos-538916	64	53	28.22	16.02	12.43	12,252	6,240
qiu	32	36	16.30	16.03	154.10	10,141	9,267
mik-250-1-100-1	8	28	13.52	16.97	0.04	33,184	41,848
seymour1	8	8	22.31	17.59	6.59	1,913	1,479
nsrand-ipx	16	3	17.04	17.91	10.97	4,988	5,573
lrn	4	2	14.89	17.93	43.49	1,033	1,365
neos-1396125	16	69	22.25	19.74	151.63	2,750	2,508
graphdraw-gemcutter	16	43	21.79	20.34	0.18	32,814	31,496
ran16x16	4	11	19.89	21.41	0.03	23,264	25,512
neos-3072252-nete	32	5	28.12	21.76	0.24	37,762	25,057
mas076	8	8	24.44	22.67	0.05	196,202	203,272
supportcase25	4	15	21.73	24.48	0.68	6,743	7,792
pg5_34	64	3	27.43	25.34	6.67	14,486	13,358
supportcase17	32	23	25.69	26.40	0.89	17,134	17,558
timtab1CUTS	4	45	30.71	28.41	0.14	24,852	22,213
neos-886822	2	167	25.06	28.43	8.81	11,190	12,526
neos-3665875-lesum	32	2	35.40	28.79	23.53	3,393	2,643

		# cuts	Time (s)			Nodes (#)	
Instance	# terms	V	Gur	V	Gen	Gur	V
eilB101	8	89	36.35	30.67	59.00	4,167	3,203
neos-5182409-nasivi	32	1	33.62	33.96	133.40	25,964	20,842
neos-1601936	4	25	75.25	35.99	479.86	686	435
glass4	8	4	51.86	37.49	0.03	180,559	116,890
mcsched	2	89	42.14	41.31	0.76	16,778	15,931
timtab1	32	2	45.01	41.46	0.07	34,735	35,424
ns1830653	4	4	38.63	42.27	1.10	6,549	6,841
bc	2	16	50.74	43.04	0.80	7,093	5,362
n4-3	4	36	41.74	44.02	0.84	3,135	3,308
neos14	16	8	64.61	49.81	0.10	82,144	61,013
neos-4650160-yukon	32	8	43.54	54.13	1.17	46,881	59,524
neos-1605075	2	434	63.20	56.04	2,205.31	659	497
gmu-35-40	32	3	170.52	61.00	0.71	482,215	161,203
reblock67	32	442	63.41	61.53	667.20	36,497	32,724
ns2081729	32	6	276.82	62.02	0.32	1,009,684	233,820
neos-5075914-elvire	4	19	212.91	63.53	30.26	26,062	7,421
neos-3421095-cinca	4	36	122.66	64.98	0.05	221,154	122,193
prod2	64	44	98.57	65.43	3.30	133,462	98,808
neos-4333596-skien	4	4	83.01	67.56	0.04	203,135	166,544
gen-ip036	2	15	67.59	68.67	0.01	1,263,100	1,265,201
k16x240b	4	14	89.76	70.61	0.03	66,445	44,605
umts	8	276	69.24	71.11	78.76	107,399	98,107
bienst2	4	33	86.24	74.03	1.97	87,484	87,024
tr12-30	32	5	81.41	77.12	0.42	94,948	89,717
breastcancer-regularized	64	200	86.25	83.13	21.91	70,461	73,506
gen-ip021	2	15	91.33	87.84	0.01	1,834,786	1,750,552
neos-916792	8	89	106.78	89.12	5.16	110,878	99,137
icir97_tension	64	3	133.01	97.29	1.83	92,396	58,179
k16x240	16	14	102.65	98.19	0.12	114,169	111,043
neos-831188	2	47	101.16	101.22	3.14	6,634	6,363
neos-3216931-puriri	2	22	105.82	103.67	1,660.66	1,247	1,076
graphdraw-domain	16	2	102.85	115.31	0.09	79,886	88,283
aflow40b	2	36	117.88	117.70	0.37	30,696	31,618
neos-1605061	4	176	201.37	118.20	3,609.85	3,296	2,142
rococoC10-001000	8	21	162.18	127.78	1.09	32,448	28,031
dfn-gwin-UUM	4	17	112.40	129.70	0.07	99,727	119,263
zib54-UUE	64	56	146.07	139.07	63.32	9,681	9,957
milo-v12-6-r2-40-1	64	5	174.93	149.73	312.06	57,487	48,876
neos-1620807	4	10	219.78	156.39	0.46	354,665	211,312
supportcase26	64	2	193.92	170.62	0.47	181,915	154,879
neos5	64	1	201.01	173.43	0.60	991,446	481,597
csched007	32	136	174.73	175.95	129.31	27,096	25,884
neos-547911	2	87	138.53	183.11	4.16	16,942	58,100
neos-5078479-escaut	4	1	88.23	184.08	3.09	14,657	24,747
neos-3660371-kurow	2	26	194.65	197.49	0.23	121,101	111,705
bnatt400	16	2	188.68	197.88	29.12	5,012	5,455
gmu-35-50	32	17	82.54	239.46	1.29	182,483	562,625
neos-1595230	2	114	270.73	245.27	0.52	101,859	100,370
neos15	4	8	267.40	261.97	0.03	173,755	170,372
a2c1s1	2	18	291.47	290.17	0.16	25,431	25,164
gen-ip002	4	18	303.86	296.12	0.01	4,262,718	4,165,621

		# cuts	Time (s)			Nodes (#)	
Instance	# terms	V	Gur	V	Gen	Gur	V
mas074	4	12	302.34	306.38	0.02	2,946,495	2,971,458
csched010	4	124	347.70	315.05	4.04	45,064	44,640
p80x400b	64	4	435.81	331.45	0.58	246,583	189,296
iis-100-0-cov	2	33	343.00	373.03	0.78	129,699	138,641
neos-3592146-hawea	2	11	531.73	416.99	9.58	27,991	23,822
ran14x18	4	18	551.03	424.91	0.04	502,202	364,247
ran14x18-disj-8	16	86	504.12	459.40	29.31	443,531	284,982
a1c1s1	8	3	523.14	499.18	0.54	46,066	45,913
reblock115	2	115	544.67	509.08	1.21	374,093	353,691
neos-5140963-mincio	32	24	512.37	523.90	0.27	4,229,762	4,177,225
ns894788	4	26	938.63	657.61	48.41	93,106	83,978
neos-2328163-agri	32	141	1,028.61	683.71	867.92	68,637	44,304
neos-4333464-siret	8	15	931.40	718.73	0.71	35,773	23,412
neos-1330346	4	12	1,298.32	740.23	293.93	1,115,343	438,141
neos-1616732	64	200	1,092.27	900.70	264.46	1,271,371	959,082
danoint	16	11	1,191.43	1,126.42	53.88	503,027	466,329
uct-subprob	32	16	1,176.32	1,145.54	36.16	119,812	117,491
iis-bupa-cov	4	153	1,197.89	1,155.29	6.84	241,282	228,666
toll-like	64	179	1,114.16	1,157.96	14.65	102,922	114,050
neos-911970	8	15	5,179.34	1,170.48	1.86	16,443,286	4,761,581
neos-911880	4	48	2,351.45	1,230.27	0.29	6,956,611	3,265,946
cost266-UUE	8	56	1,595.05	1,320.01	1.45	115,358	93,401
nobel-eu-DBE	32	47	1,706.44	1,320.64	8.50	635,334	511,472
pigeon-10	16	72	1,437.67	1,434.01	0.22	9,000,483	8,975,846
ic97_potential	32	2	1,528.57	1,441.99	0.48	770,495	677,844
cov1075	4	120	1,768.86	1,866.61	0.80	1,904,781	1,822,317
b1c1s1	16	9	2,138.32	1,937.98	3.28	97,151	90,411
newdano	16	16	1,878.95	2,006.42	35.16	1,560,936	1,074,754
neos-574665	2	11	4,117.02	2,074.51	0.05	11,845,245	5,263,881
50v-10	2	10	4,424.19	2,236.30	0.10	2,299,227	1,209,503
maxgasflow	8	6	1,781.33	2,323.15	1.74	225,353	311,268
neos-3754480-nidda	32	41	2,585.34	2,859.40	1.52	4,550,435	5,206,727
Gmean			8.78	7.81	1.88	5,761	5,031
Wins			31	106	48	69	187

H Alternative cut-generating sets and point-ray collections

In this section, we briefly mention experiments with other choices for cut-generating sets (instead of those derived from partial branch-and-bound trees) and with potential refinements of the simple point-ray collection used for all of the previous results. **Importantly, we report results from *preliminary* experiments, with an old version of the code. Thus, the numbers in these tables will not correspond to the other experiments in the paper, even though many of the instances are the same.**

Table 12: Summary on small instance set of average percent gap closed, average percent of GMICs and VPCs active at the optimum after adding both sets of cuts to P, average "cut ratio" between number of VPCs and number of GMICs, and the shifted geometric mean (by 60) for generating VPCs for each partial tree size as well as on runs with multiple split and cross disjunctions.

	G	V+G (2)	V+G (4)	V+G (8)	V+G (16)	V+G (32)	V+G (46)	V+G (splits)	V+G (crosses)
% gap closed	16.33	18.48	19.55	21.73	25.52	30.74	35.26	21.79	26.92
% active GMIC		50.40	49.68	45.01	40.11	35.37	36.90	45.65	35.82
% active VPC		22.09	22.59	30.57	29.81	29.26	27.90	14.19	10.45
Cut ratio		0.63	0.64	0.66	0.69	0.71	0.72	14.27	108.23
Time (gmean)		0.16	0.54	2.34	7.24	13.16	14.43	7.38	53.40

H.1 Gap closed using "multiple" split and cross disjunctions

Instead of using one large, multiterm disjunction, as we do above, the typical approach in the literature is to generate cuts from the union of several shallower disjunctions. We report results on preliminary computational experiments to assess the strength of VPCs obtained from multiple split disjunctions or multiple 2-branch (cross) disjunctions. An alternative that we do not test but merits exploration in the future is that of several partial branch-and-bound trees produced from different branching strategies.

Let $\sigma := \{j \in \mathcal{I} : \bar{x}_j \notin I\}$ be the set of indices of integer variables that take fractional values in $\bar{x}$. For each $k \in \sigma$, there is a corresponding elementary split disjunction $(x_k \leq \lfloor \bar{x}_k \rfloor) \vee (x_k \geq \lceil \bar{x}_k \rceil)$. We generate VPCs from each of the elementary split disjunctions applied to P. We call these "multiple" split cuts. We also report on the strength of VPCs from the nonconvex P_I-free set corresponding to a union of two split disjunctions from pairs of indices in σ.

We experiment on a smaller set of instances (37 in total) in order to conserve computational resources, and we report only the percent gap closed (without testing the cuts' effect on branch and bound time). The reason for both of these restrictions is that these early experiments strongly support the use of partial trees as the cut-generating set over multiple split or cross disjunctions. The size restriction for the experiments in this section is at most 500 rows and 500 columns, and instances from MIPLIB 2017 were not considered. The other instance selection criteria remain unchanged. The limit on the number of cuts per split or cross disjunction is set as $|\sigma|$ (which is the same as the limit from each partial branch-and-bound tree).

Table 12 has columns for "G" (GMICs), "V+G (ℓ)" for $\ell \in \{2, 4, 8, 16, 32, 64\}$ (VPCs used together with GMICs from a partial branch-and-bound tree with ℓ leaf nodes), and "V+G (splits)" and "V+G (crosses)" (values corresponding to using splits and crosses, respectively). The rows give the average percent gap closed, the average percent of active GMICs and VPCs at the post-cut optimum, the ratio between the number of VPCs and number of GMICs, and the geometric mean (with a shift of 60) of the time needed to generate cuts (including the time to generate the partial trees and set up the point-ray collections).

As Table 12 shows, the gap closed by VPCs from multiple split disjunctions is comparable to that from using a partial tree with 8 leaves, while multiple cross disjunctions yield a gap closed similar to that from partial trees with 16 leaves. However, when using splits and crosses, the number of VPCs is considerably larger than the number of GMICs, and cut generation time is also on average much greater. This data supports our conclusion that using partial branch-and-bound trees to generate disjunctions for our procedure is preferable to using multiple split or cross disjunctions.

H.2 Tightening the $\mathcal{V}$-polyhedral relaxation

We have seen in Figures 1 and 2 that using the relaxations C^t for each term $t \in \mathcal{T}$ can limit the set of cuts that can be generated. A natural question to consider is whether a different relaxation would lead to stronger cuts.

One approach, which we have not tested computationally, involves refining the relaxations of each disjunctive term to some $\tilde{C}^t \subseteq C^t$ for each $t \in \mathcal{T}$ such that the following condition is satisfied for all $t, t' \in \mathcal{T}$, $t \neq t'$:

$$\tilde{C}^t \cap \{x \in \mathbb{R}^n : D^{t'} x \geq D_0^{t'}\} = \emptyset.$$

This would avoid the type of problem shown in Figure 2. The essential idea would involve activating hyperplanes, but the process is generally made simpler by the fact that the disjunctive inequalities in practice all take on a simple form (each is a bound on a variable).

A different idea is to keep a simple cone as the relaxation for each term of the disjunction, but to use a different cobasis as the origin. Specifically, we can apply the VPC procedure based on Proposition 6, not on p^t, but on a neighbor of p^t obtained by pivoting along any edge of P^t. We tested this procedure on VPCs generated from the set of elementary split disjunctions. The results were negative, in the sense that only a small additional percent gap was closed, whereas the extra computational expense involved was significant. Our interpretation of this outcome is that the VPCs from simple cones contain the vast majority of the cuts that affect the objective function value and are obtainable from each elementary split. This was in fact a primary motivation for pursuing more complicated disjunctions for cut generation.

I Complete tables for experiments with other cut-generating sets

The tables in this section show that, overall, using partial branch-and-bound trees leads to comparable or better percent gap closed than generating cuts from multiple split or cross disjunctions (Table 13), in less time (Table 14) and with fewer cuts (Table 15). A summary of these tables appeared in Table 12.

The columns of Table 13 give the following information for each instance. Columns 2 and 3 give the dimensions of the instance. Column 4 gives the number of GMICs generated (one for each elementary split on an integer variable fractional at $\bar{x}$). Column 5 contains the number of VPCs generated, while the next column (6) specifies the number of cuts that are

active, i.e., tight, at the optimum of the LP after adding the cuts. Finally, columns 7 and 8 give the percentage of the integrality gap closed by the GMICs and the VPCs. The last column (9) gives the time used for generating the cuts.

Table 13: Comparison of percent gap closed by VPCs from partial branch-and-bound trees to using multiple split or cross disjunctions.

Instance	G	V+G (2)	V+G (4)	V+G (8)	V+G (16)	V+G (32)	V+G (46)	V+G (best)	V+G (splits)	V+G (crosses)
23588	5.80	17.30	28.70	35.70	47.80	60.90	71.80	71.80	28.70	39.80
bell3a	37.00	37.00	37.00	37.00	43.60	43.60	43.60	43.60	37.00	43.90
bell3b	31.20	84.10	84.10	84.10	84.50	84.50	84.70	84.70	51.40	84.30
bell4	25.60	25.60	25.80	26.10	26.00	26.10	26.40	26.40	25.60	29.90
bell5	13.80	13.80	13.80	25.70	25.60	77.80	62.90	77.80	14.80	22.70
blend2	5.50	5.50	7.50	12.10	13.10	15.50	23.00	23.00	7.10	8.90
bm23	16.80	18.00	19.80	19.80	39.50	56.20	70.90	70.90	18.00	20.20
glass4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
go19	2.00	2.50	3.70	5.60	9.80	13.10	2.00	13.10	8.50	5.80
gt2	87.10	87.10	87.10	87.10	87.10	87.10	87.10	87.10	87.10	87.10
k16x240	11.40	11.40	12.90	12.80	16.90	17.70	22.50	22.50	13.50	16.40
lseu	5.80	6.10	6.10	6.10	10.10	11.40	19.20	19.20	6.20	9.10
mas074	6.70	6.70	7.20	7.90	9.00	11.10	13.40	13.40	6.90	7.20
mas076	6.40	6.40	6.50	6.40	6.80	8.90	13.00	13.00	6.40	6.50
mas284	0.90	1.80	8.70	12.40	15.60	25.30	33.80	33.80	3.30	10.60
mik-250-1-100-1	53.50	53.50	53.50	53.50	53.50	53.50	53.50	53.50	53.50	53.50
misc03	0.00	0.00	0.00	5.40	10.50	17.40	44.30	44.30	0.00	4.10
misc07	0.70	0.70	0.70	0.70	0.90	1.90	5.50	5.50	0.70	0.70
mod008	24.40	24.40	24.40	25.80	25.50	26.70	27.80	27.80	24.40	24.40
mod013	5.90	9.00	9.90	20.60	21.10	36.90	47.40	47.40	9.00	11.60
modglob	18.10	18.50	18.80	19.60	18.90	18.40	18.60	19.60	30.30	44.80
neos-1420205	0.00	0.00	0.00	0.00	0.00	11.40	14.70	14.70	0.00	0.40
neos5	11.10	11.10	11.10	18.80	21.90	29.20	37.50	37.50	11.20	15.70
neos-880324	0.00	0.00	0.00	0.00	0.00	0.00	36.30	36.30	8.50	16.90
p0282	3.20	3.20	3.20	3.40	6.40	7.30	10.60	10.60	65.20	77.90
pipex	35.60	35.60	35.70	35.70	35.60	35.60	36.70	36.70	35.60	35.60
pp08aCUTS	33.80	33.80	34.90	34.90	35.70	34.60	33.80	35.70	39.90	51.90
pp08a	54.50	54.50	54.50	54.50	54.50	54.50	54.50	54.50	56.10	70.30
probportfolio	4.60	4.80	5.40	5.70	7.00	8.40	11.30	11.30	14.70	13.50
prod1	4.70	5.10	8.30	12.20	18.60	29.60	36.30	36.30	9.70	22.40
rgn	8.00	8.00	8.00	8.00	21.60	33.80	45.40	45.40	8.00	8.10
roy	4.50	7.10	10.40	11.30	15.80	24.30	30.30	30.30	7.40	9.30
sentoy	19.30	19.30	20.00	27.50	45.90	55.00	59.80	59.80	21.30	22.90
stein27_nocard	8.30	13.10	17.90	29.50	44.40	50.00	59.30	59.30	28.50	36.40
timtab1	24.10	24.10	24.10	24.10	24.10	24.10	24.10	24.10	27.20	34.10
vpm1	15.70	15.70	15.70	15.70	27.60	26.20	22.80	27.60	16.50	17.30
vpm2	18.20	18.60	18.20	18.60	19.00	19.30	20.10	20.10	23.80	31.80
Average	16.33	18.48	19.55	21.73	25.52	30.74	35.26	36.17	21.79	26.92

Table 14: Time to generate VPCs for each of the different partial branch-and-bound tree sizes and for multiple split and cross disjunctions.

Instance	V (2)	V (4)	V (8)	V (16)	V (32)	V (46)	V (splits)	V (crosses)
23588	0.50	2.30	8.80	23.90	77.50	224.80	36.20	912.90
bell3a	0.00	0.00	0.00	0.00	0.10	0.20	0.00	0.10
bell3b	0.00	0.00	0.00	0.10	0.10	0.20	0.10	2.40
bell4	0.00	0.00	0.00	0.10	0.10	0.20	0.10	2.60
bell5	0.00	0.00	0.00	0.00	0.10	0.10	0.10	0.30
blend2	0.10	0.20	0.20	0.50	1.60	4.30	0.60	6.90
bm23	0.00	0.00	0.00	0.10	0.10	0.30	0.10	0.20
glass4	0.00	0.10	0.10	0.30	0.50	1.60	1.50	100.80
go19	1.60	7.50	52.10	314.40	906.30	71.70	619.50	926.50
gt2	0.00	0.00	0.00	0.20	0.40	0.90	0.20	2.90
k16x240	0.10	0.10	0.30	0.70	1.60	3.50	0.70	8.80
lseu	0.00	0.00	0.00	0.10	0.20	0.40	0.10	1.00
mas074	0.00	0.10	0.20	0.60	1.40	4.40	0.40	4.80
mas076	0.00	0.10	0.30	0.80	1.90	4.20	0.40	4.80
mas284	0.10	0.30	0.80	4.10	13.50	52.20	1.70	48.80
mik-250-1-100-1	0.10	0.10	0.20	0.40	2.10	9.30	3.80	327.50
misc03	0.10	0.10	0.50	1.10	3.20	8.70	0.90	20.10
misc07	0.10	0.30	0.90	3.00	13.00	33.30	1.30	29.50
mod008	0.10	0.10	0.20	0.40	1.20	2.90	0.20	1.20
mod013	0.00	0.00	0.00	0.10	0.20	0.40	0.10	0.20
modglob	0.10	0.20	0.40	1.20	1.90	3.40	0.80	18.60
neos-1420205	0.20	0.30	0.30	0.90	4.10	7.10	2.40	170.80
neos5	0.10	0.20	0.40	0.70	1.80	4.90	1.40	114.60
neos-880324	0.10	0.60	0.40	1.30	15.30	20.10	2.20	267.10
p0282	0.00	0.10	0.10	0.40	0.70	1.60	0.50	10.90
pipex	0.00	0.00	0.00	0.10	0.10	0.20	0.00	0.20
pp08aCUTS	0.10	0.10	0.20	0.80	2.90	17.60	3.70	184.60
pp08a	0.00	0.00	0.10	0.20	0.40	1.10	0.90	39.10
probportfolio	2.30	7.40	41.20	264.70	682.70	903.30	88.50	907.20
prod1	0.10	0.30	1.20	2.70	6.10	11.80	2.00	218.30
rgn	0.00	0.10	0.20	0.30	1.40	2.50	0.40	7.40
roy	0.00	0.00	0.10	0.20	0.30	0.60	0.20	1.40
sentoy	0.00	0.00	0.10	0.20	0.40	0.90	0.10	0.70
stein27_nocard	0.00	0.10	0.10	0.20	0.40	0.50	0.40	19.20
timtab1	0.00	0.00	0.10	0.20	0.30	0.70	2.30	339.60
vpm1	0.00	0.00	0.10	0.20	0.30	0.70	0.10	1.10
vpm2	0.00	0.00	0.10	0.20	0.40	0.80	0.50	8.90
Gmean	0.16	0.54	2.34	7.24	13.16	14.43	7.38	53.40

Table 15: Number of rows, columns, GMICs, and VPCs for small instances used to test multiple split and cross disjunctions. The last row gives the average ratio of number of VPCs as a fraction of the number of GMICs.

			# cuts								
Instance	Rows	Cols	G	V (2)	V (4)	V (8)	V (16)	V (32)	V (46)	V (splits)	V (crosses)
23588	137	237	75	36	75	75	75	75	75	4,890	32,920
bell3a	63	82	7	1	5	7	4	5	7	4	19
bell3b	73	91	24	3	3	7	21	25	25	63	328
bell4	73	88	27	3	4	8	5	7	9	22	178
bell5	34	56	10	3	2	5	4	10	10	19	82
blend2	154	302	13	13	13	13	13	13	13	117	840
bm23	20	27	6	6	6	6	6	6	6	36	90
glass4	392	317	72	2	4	10	14	0	72	142	906
go19	361	361	357	97	228	357	357	175	0	35,760	22,557
gt2	28	173	14	3	4	3	7	2	1	21	242
k16x240	256	480	14	14	9	14	14	14	14	166	977
lseu	28	79	9	9	9	9	9	9	9	81	280
mas074	13	148	12	12	12	12	12	12	12	144	779
mas076	12	148	11	11	8	4	10	11	11	112	497
mas284	68	148	20	20	20	20	6	9	4	387	3,607
mik-250-1-100-1	100	251	100	1	2	8	13	74	100	1	100
misc03	95	138	18	18	18	18	18	18	18	310	2,616
misc07	211	232	16	16	16	16	16	16	16	256	1,901
mod008	6	319	6	6	6	3	6	6	6	30	88
mod013	62	96	5	5	5	5	5	5	5	23	50
modglob	286	354	29	21	25	29	29	29	29	209	1,049
neos-1420205	341	231	44	40	44	0	0	2	1	613	9,756
neos5	63	63	35	25	35	2	1	1	1	719	8,989
neos-880324	182	135	45	10	7	0	0	0	11	543	17,738
p0282	160	200	24	13	10	24	24	24	24	181	1,697
pipex	25	48	6	6	6	6	6	6	6	19	42
pp08aCUTS	239	235	46	6	7	3	5	11	46	924	12,766
pp08a	133	234	53	4	5	4	4	2	2	246	1,145
probportfolio	302	320	105	105	105	105	105	105	48	6,157	9,432
prod1	75	117	40	18	40	40	40	40	40	1,184	30,508
rgn	24	180	18	16	18	18	18	18	3	222	1,606
roy	147	139	11	7	6	11	11	11	11	104	368
sentoy	30	60	8	8	8	8	8	8	8	64	224
stein27_nocard	117	27	27	7	11	27	1	2	1	163	6,102
timtab1	165	365	128	5	2	3	4	2	2	557	13,566
vpm1	128	188	11	11	0	5	11	9	11	56	104
vpm2	127	187	24	24	7	13	24	24	24	247	1,209
Avg (cut ratio)				1	1	1	1	1	1	14	108