Warm Starting of Mixed Integer Linear Optimization Problems via Parametric Disjunctive Cuts

Shannon Kelley [a], Ted Ralphs [a], Aleksandr M. Kazachkov [b]
[a] Department of Industrial and Systems Engineering, Lehigh University
[b] Department of Industrial and Systems Engineering, University of Florida

April 21, 2025

Abstract

Many applications require solving a sequence of mixed integer linear optimization problems within a parametric family that shares the same variables and constraints. Warm-starting—reusing information from previous solves—offers a powerful opportunity to reduce solution time in such cases. In this paper, we propose leveraging valid disjunctions generated from Branch-and-Bound trees of prior solves, paired with Farkas certificates that verify the validity of associated disjunctive cuts, to warm-start unsolved instances in the family. Together, these disjunctions and certificates form a parametric class of disjunctive cuts that can be efficiently applied to any member instance. We implemented this approach and conducted computational experiments to evaluate its potential.

1 Introduction

We examine warm-starting mixed-integer linear optimization problems (MILPs) using disjunctions and Farkas certificates from branch-and-bound trees of a family of perturbed prior instances by generating parametric disjunctive cuts. Resolving slight perturbations of previously solved MILPs frequently arises in mixed integer programming (MIP) literature, including Branch-and-Price [6, 13], Lagrangian Dual Decomposition [7, 9], Benders Decomposition [18], Mixed-Integer Nonlinear Programming (MINLP) [26, 10], Bilevel optimization [27], multi-objective optimization [29, 17], and online optimization [28, 22]. These contexts span diverse industrial applications such as energy management [28, 21], supply chain management [22, 15, 26], scheduling [6, 26], and revenue optimization [15]. Our approach aims to enhance decomposition methods and enable efficient real-time optimization, tackling theoretical and practical challenges central to modern MIP applications.

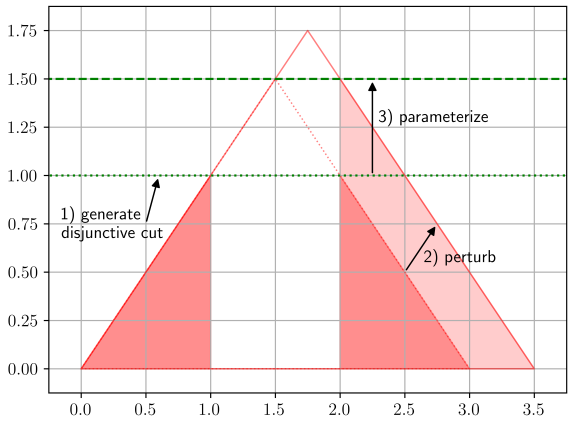


Figure 1: Given a disjunctive cut and a perturbed linear relaxation, we warm-start by parameterizing the cut to remain valid.

Solution methods for MILPs generate valuable information that is often discarded after solving, despite its potential for warm-starting similar instances [14, 23]. Examples include retaining variable branching priorities [21, 15] or predicting optimal solution properties [2, 20], both of which highlight the importance of heuristics in solver performance [19]. Beyond heuristics, proving optimality remains a critical step, and translating such proofs across related MILPs could significantly enhance efficiency. These *certificates*, often represented as disjunctions from terminal Branch-and-Bound trees, can be reused to warm-start the proof process for related instances. Practitioners may reuse these disjunctions

in various ways, such as adding constraints to enforce bounds on objective values [8] or initializing the node queue of a solution process with the disjunction's leaves [25, 16]. In this work, we propose retaining partial disjunctions from terminal Branch-and-Bound trees and Farkas (dual) multipliers for validating inequalities.

In more detail, the linear relaxation of a MILP can be tightened with additional inequalities, also known as *cutting planes* or *cuts*, that are *valid* for the MILP, in that they are satisfied by all feasible solutions. As it may not be straightforward to ensure that an inequality generated from another instance is valid directly for the MILP, one can identify a *disjunctive relaxation* [3] of the MILP using integrality restrictions to partition the domain into a set of *(disjunctive) terms* whose union contains all MILP-feasible points, with these terms comprising a (valid) *disjunction*. Every cut that is valid for a given disjunctive relaxation has a *Farkas certificate* of its validity, consisting of the disjunction and multipliers for each disjunctive term. We call such a Farkas certification a *parameterized (disjunctive) cut*.

The disjunctions we construct use only integrality information, arising as the leaf nodes of (partial) *Branch-and-Bound trees*. It follows that the Farkas certificate for a cut generated from such a disjunction and a base MILP will yield a valid inequality for *any* other MILP with the same number of constraints and the same integer variables. We focus on cuts derived from large disjunctions with potentially many terms, the computation of which may be expensive but offering opportunity for stronger cuts, for which the generation effort could be amortized across the sequence of instances. Our research objective is to explore whether retaining and exploiting disjunctive cutting plane information from solving a MILP is empirically effective to accelerate the solution of related instances.

Contributions. Our contributions are both theoretical and computational. In Theoretical Results, we establish key theoretical guarantees for warm-starting, including the validity and tightness conditions of parametric disjunctive cuts relative to the disjunctive hull, separation conditions for a parametric class of disjunctive cuts and basic solutions, and the NP-hardness of warm-starting MILPs in general. In Computational Results, we evaluate the performance of warm-starting with parametric disjunctive cuts, comparing it to the disjunctive cut generation method proposed by Balas and Kazachkov [4], a hybrid of the two approaches, and no disjunctive cuts at all. Our results show that warm-starting with disjunctive cuts consistently outperforms Balas and Kazachkov [4]'s method and is often advantageous compared to the default configuration of a leading solver.

2 Notation and Preliminaries

Our goal is to generate parametric valid inequalities to tighten the (natural) linear programming (LP) relaxation of a MILP instance from a family $\mathcal{K}$ of related instances. For ease of exposition, we assume that all instances share the same set of variables, though for correctness of our method, we only need that the set of integer variables is the same across all instances. Instance k takes the form

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^k x \\ & A^k x \geq b^k \\ & x_j \in \mathbb{Z} \qquad \text{for } j \in \mathcal{I}, \end{aligned} \tag{IP-k}$$

where $A^k \in \mathbb{Q}^{q \times n}$, $b^k \in \mathbb{Q}^q$, and $c^k \in \mathbb{Q}^{1 \times n}$. For simplicity, we assume that these constraints specify $x \geq 0$ and all additional variable bounds. For any positive integer n, let $[n] := \{1, \ldots, n\}$. The integer-restricted variables are denoted by $\mathcal{I} \subseteq [n]$. We define the LP relaxation of (IP-k) as

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^k x \\ & A^k x \geq b^k \end{aligned} \tag{LP-k}$$

where $\bar{x}^k \in \text{argLP-}k$ belongs to its optimal solution set. We refer to the feasible regions of (LP-k) and (IP-k), respectively, as $\mathcal{P}^k := \{x \in \mathbb{R}^n : A^k x \geq b^k\}$ and $\mathcal{S}^k := \{x \in \mathcal{P}^k : x_j \in \mathbb{Z} \text{ for } j \in \mathcal{I}\}$. We reference (LP-$k$) in standard form with $\tilde{\mathcal{P}}^k := \{\tilde{x} \in \mathbb{R}^{n+q} : \tilde{A}^k \tilde{x} = b^k\} = \{(x, s) \in \mathbb{R}^{n+q} : A^k x - s = b^k, s \geq 0\}$. For matrix A, we represent its ith row as "$A_{i\cdot}$" and its jth column as "$A_{\cdot j}$". $B \in \mathbb{N}^q$ and $N \in \mathbb{N}^n$ partition $[n+q]$ and index basic and nonbasic columns, respectively, of basic solution $\tilde{x}^{k,B}$ of (LP-k), which is defined $\tilde{x}^{k,B}_B := (\tilde{A}^k_{\cdot B})^{-1} b^k$ and $\tilde{x}^{k,B}_N := 0$ since $x \geq 0$. $\mathcal{N} \in \mathbb{N}^n$ indexes the corresponding active, linearly independent constraints, which define a translated cone with apex $x^{k,B} := \text{proj}_x(\tilde{x}^{k,B})$, called a *basis cone*, $\mathcal{C}^k_{\mathcal{N}} := \{x \in \mathbb{R}^n : A^k_{\mathcal{N}\cdot} x \geq b^k_{\mathcal{N}}\}$.

Valid disjunctions can be used to strengthen the formulation $\mathcal{P}^k$ (with respect to $\mathcal{S}^k$) by deriving *valid inequalities* for $\mathcal{S}^k$.

Definition 2.1. *A* valid inequality *for set $\mathcal{S} \subseteq \mathbb{R}^n$ is a pair $(\alpha, \beta) \in \mathbb{R}^n \times \mathbb{R}$ such that $\mathcal{S} \subseteq \{x \in \mathbb{R}^n : \alpha^\intercal x \geq \beta\}$.*

Definition 2.2. *A* valid disjunction *for set $\mathcal{S} \subseteq \mathbb{R}^n$ is a collection $\mathcal{X} := \{\mathcal{X}^t\}_{t \in T}$, where T is an index set, $\mathcal{X}^t \subset \mathbb{R}^n$ for $t \in T$, and $\mathcal{S} \subseteq \bigcup_{t=1}^{|T|} \mathcal{X}^t$.*

In what follows, we exploit disjunctions that are guaranteed to be valid for the feasible regions $\mathcal{S}^k$ of *all* instances $k \in \mathcal{K}$. Hence, we denote disjunctions independently of the index k. Let $\{\mathcal{X}^t\}_{t \in T}$ be such a disjunction, where *disjunctive term* $\mathcal{X}^t := \{x \in \mathbb{R}^n : D^t x \geq D^t_0\}$ is defined by q_t constraints with $D^t \in \mathbb{Q}^{q_t \times n}$ and $D^t_0 \in \mathbb{Q}^{q_t}$. Let

$$A^{kt} := \begin{bmatrix} A^k \\ D^t \end{bmatrix}, \quad b^{kt} := \begin{bmatrix} b^k \\ D^t_0 \end{bmatrix}, \quad \text{and} \quad \mathcal{Q}^{kt} := \mathcal{P}^k \cap \mathcal{X}^t = \{x \in \mathbb{R}^n : A^{kt} x \geq b^{kt}\}.$$

We formulate the LP relaxation of the application of disjunctive term $\mathcal{X}^t$ to IP-k as

$$\min_{x \in \mathcal{Q}^{kt}} \{c^k x\}. \tag{LP-kt}$$

where $\bar{x}^{kt} \in \text{argLP-}kt$ belongs to its optimal solution set. $\tilde{\mathcal{Q}}^{kt} \in \mathbb{R}^{n+q+q_t}$, $B^t \in \mathbb{N}^{q+q_t}$, $N^t \in \mathbb{N}^n$, and $\mathcal{N}^t \in \mathbb{N}^n$ are defined for (LP-kt) analogously to $\tilde{\mathcal{P}}^k$, B, N, and $\mathcal{N}$ for (LP-k). Similarly, basic solution $\tilde{x}^{kt,B^t}$ is defined $\tilde{x}^{kt,B^t}_{B^t} := (A^{kt}_{\cdot B^t})^{-1} b$ and $\tilde{x}^{kt,B^t}_{N^t} := 0$, yielding the basis cone $\mathcal{C}^{kt}_{\mathcal{N}^t} := \{x \in \mathbb{R}^n : A^{kt}_{\mathcal{N}^t \cdot} x \geq b^{kt}_{\mathcal{N}^t}\}$ with apex $x^{kt,B^t} := \text{proj}_x(\tilde{x}^{kt,B^t})$.

This work concerns generating a set of cuts $\{(\alpha^h, \beta^h)\}_{h \in H}$ valid for the *disjunctive hull* $\mathcal{P}^k_D := \text{cl conv}(\cup_{t \in T} \mathcal{Q}^{kt})$. The assumption that the disjunction is valid for all instances is important in what follows, is the basis for the warm-starting, and is the reason why we assume the set of variables is fixed across all instances. Note that $\{\mathcal{Q}^{kt}\}_{t \in T}$ is also a valid disjunction for instance k but may not be valid for other instances. Before proceeding to our methodology, we first summarize the fundamental concepts underlying our proposed cut generation paradigm.

2.1 Cut-Generating LP

The well-studied *cut-generating LP* (CGLP) is one way to generate inequalities valid for the disjunctive hull $\mathcal{P}^k_D$ for instance (IP-k), $k \in \mathcal{K}$, with respect to a valid disjunction $\{\mathcal{X}^t\}_{t \in T}$. The validity of the cuts from the CGLP follows from the theory of *disjunctive programming* [3, Section 4]. The key is that the CGLP is formulated in a higher-dimensional space that includes variables both for the cut and for the Farkas multipliers that certify the validity of that cut with respect to the given disjunction.

Concretely, an inequality $\alpha^\intercal x \geq \beta$ is valid for $\mathcal{P}^k_D$ if and only if the inequality is valid for each $\mathcal{Q}^{kt}$ with $t \in T$. Consequently, by Farkas's lemma [11], $(\alpha, \beta) \in \mathbb{R}^n \times \mathbb{R}$ is valid for $\mathcal{P}^k_D$ if and only if, for all $t \in T$, the following system is feasible in variables $(\alpha, \beta, \{v^t\}_{t \in T})$, where $\{v^t\}_{t \in T})$ is the *Farkas certificate* or *Farkas multipliers*:

$$\left.\begin{array}{l} \alpha^\intercal = v^t A^{kt} \\ \beta \leq v^t b^{kt} \\ v^t \in \mathbb{R}^{1 \times (q+q_t)}_{\geq 0} \end{array}\right\} \text{ for all } t \in T. \tag{FL}$$

To generate cuts with (FL), one typically maximizes the violation with respect to $\mathcal{S}^k$-infeasible points, after adding a normalization, which can be a crucial choice [12]. One straightforward normalization is to scale so that the right-hand of the inequality is fixed to $\bar{\beta} := 1$. For $\bar{x} \notin \mathcal{P}_D^k$, this amounts to solving CGLP-$k(\mathcal{X})$.

$$\begin{aligned}
\min_{(\alpha,\beta)\in\mathbb{R}^n\times\mathbb{R}} \quad & \alpha^\intercal \bar{x} - \beta \\
& \alpha^\intercal = v^t A^{kt} && \text{for } t \in T \\
& \beta \le v^t b^{kt} && \text{for } t \in T \\
& \beta = \bar{\beta} \\
& v^t \in \mathbb{R}^{1\times(q+q_t)}_{\ge 0} && \text{for } t \in T
\end{aligned} \tag{CGLP-$k(\mathcal{X})$}$$

2.2 Point-Ray LP

Solving the CGLP is often too expensive in practice when formulated beyond two-term disjunctions [24]. Since the VPC framework targets cuts from much larger disjunctions, Balas and Kazachkov [4] suggest two modifications to the cut-generation procedure: avoiding solving an LP with more than n variables, and reducing the number of constraints by only generating cuts for a relaxation of the disjunctive hull.

First, they consider the generation of cuts using the *$\mathcal{V}$-polyhedral* description of $\mathcal{P}_D^k$, through its extreme points and rays, as opposed to the inequality description involved in the CGLP. Let $\mathcal{E}^{kt}$ be the set of extreme points and $\mathcal{R}^{kt}$ be the set of extreme rays of $\mathcal{Q}^{kt}$. Further, define $\mathcal{E}^k := \cup_{t\in T}\mathcal{E}^{kt}$ and $\mathcal{R}^k := \cup_{t\in T}\mathcal{R}^{kt}$. Since $x \in \mathcal{P}_D^k$ if and only if $x \in \text{conv}(\mathcal{E}^k) + \text{cone}(\mathcal{R}^k)$, it holds that (α, β) is valid for $\mathcal{P}_D^k$ if and only if (α, β) satisfy the following system defining a *point-ray polyhedron* (PRP):

$$\begin{aligned}
& \alpha^\intercal p \ge \beta && \text{for all } p \in \mathcal{E}^k \\
& \alpha^\intercal r \ge 0 && \text{for all } r \in \mathcal{R}^k.
\end{aligned} \tag{PRP}$$

Cuts valid for (PRP) are referred to as VPCs [4]. The advantage of PRP over CGLP-$k(\mathcal{X})$ is the PRP involves only n variables, those of the cut coefficients.

However, the number of constraints of the PRP may be exponential in n and q, as opposed to the polynomial-sized CGLP. This leads to the second remedy used by Balas and Kazachkov [4]: relax the disjunctive terms by replacing the full set of constraints of $\mathcal{Q}^{kt}$ with just those associated with an optimal basis. Let $\mathcal{N}^t$ index those constraints and basis cone $\mathcal{C}^{kt}_{\mathcal{N}^t}$ represent the relaxation. Crucially, for all $t \in T$, $\mathcal{C}^{kt}_{\mathcal{N}^t}$ has a compact $\mathcal{V}$-polyhedral description consisting of a single point $\bar{x}^{kt}$ and n rays. Balas and Kazachkov [4] exploit this by replacing the point-ray collection $(\mathcal{E}^k, \mathcal{R}^k)$ used in (PRP) with the union of the points and rays defining the basis cones $\mathcal{C}^{kt}_{\mathcal{N}^t}$ for all $t \in T$, leading to a *point-ray LP* (PRLP) with at most $|T| \cdot (n + 1)$ constraints.

$$\begin{aligned}
\min_{(\alpha,\beta)\in\mathbb{R}^n\times\mathbb{R}} \quad & \alpha^\intercal w \\
& \alpha^\intercal p \ge \beta && \text{for all } p \in \mathcal{E}^k \\
& \alpha^\intercal r \ge 0 && \text{for all } r \in \mathcal{R}^k \\
& \beta = \bar{\beta}
\end{aligned} \tag{PRLP-$k(\mathcal{X}, w)$}$$

Similar to CGLP-$k(\mathcal{X})$, PRLP-$k(\mathcal{X}, w)$ requires normalizing, which can again be accomplished by constraining $\beta = \bar{\beta}$, appropriate values for which are discussed in Balas and Kazachkov [4]. Cuts obtained from PRLP-$k(\mathcal{X}, w)$ are referred to as *simple* VPCs, and are the ones used in our experiments. When the disjunction $\{\mathcal{X}^t\}_{t\in T}$ comes from the leaf nodes of a partial Branch-and-Bound tree, for each $t \in T$, the point and rays for $\mathcal{C}^{kt}_{\mathcal{N}^t}$ can be readily obtained from the optimal tableau of the LP relaxation at the corresponding leaf node. We demonstrate this process in Example A.1 of appendix A.

2.3 Deriving Farkas Certificates.

Given a valid disjunction $\{\mathcal{X}^t\}_{t\in T}$, let (α, β) be a simple VPC valid for $\mathcal{P}_D^k$, generated via (PRLP-$k(\mathcal{X}, w)$). As such, the Farkas certificate for this cut is not a byproduct of cut generation. Fortunately, Balas and Kazachkov [5, Section 3.1] show that, for the specific setting of simple VPCs, the Farkas certificate can be recovered efficiently.

Lemma 2.3. *If $\alpha^\intercal x \geq \beta$ is valid for $\mathcal{C}^{kt}_{\mathcal{N}^t}$, then the multiplier on constraint $i \in [q + q_t]$ has value $v_i^t = \bar{\alpha}^\intercal r^h$, where r^h is column h of $(A^{kt}_{\mathcal{N}^t\cdot})^{-1}$, if there exists $h \in [n]$ such that $\mathcal{N}_h^t = i$, else 0.*

Lemma 2.3 allows us to compute the Farkas certificate using the rays describing the basis cone, which can be read from the optimal tableau for disjunctive term t. Example A.2 in appendix A demonstrates this process. In principle, for simple VPCs derived from partial Branch-and-Bound trees, this tableau does not require additional computation due to our choice of p^{kt} as an optimal solution for term t, which is already calculated for each leaf node by the solver when constructing the partial tree.

3 Theoretical Results

Our work develops theory around two key ideas. First, parametric disjunctive cuts can reduce the feasible search space while preserving optimality. In Sections 3.1 - 3.3, we establish their validity against the generating disjunctive hull and provide conditions under which they both support and separate it from the LP relaxation. Second, while parameterization can aid warm-starting, it is not a universal solution. In Section 3.4, we prove that warm-starting in general is NP-hard, meaning parametric disjunctive cuts may not always simplify problem-solving. Section 4's computational results explore where practical outcomes fall between these extremes.

3.1 Parametric Disjunctive Cuts Are Valid for the Disjunctive Hull

Our disjunctive cut parameterization relies on a fundamental disjunctive programming theorem, which demonstrates how to reuse a disjunction and Farkas certificate from one instance to generate valid inequalities for another.

Theorem 3.1. *Let $k \in \mathcal{K}$ be the index of an arbitrary instance in a series. Then given a disjunction $\{\mathcal{X}^t\}_{t\in T}$ and a set $\{v^t\}_{t\in T}$ of nonnegative Farkas multipliers associated with each term of this disjunction, we have that $\alpha^\intercal x \geq \beta$ for all $x \in \mathcal{S}^k \cap \mathcal{P}_D^k$, where $\alpha_j := \max_{t\in T}\{v^t A^{kt}_{\cdot j}\}$ and $\beta := \min_{t\in T}\{v^t b^{kt}\}$ for all $j \in [n]$.*

Proof. Let $\gamma^t := v^t A^{kt}$ and $\gamma_0^t := v^t b^{kt}$ for all $t \in T$. We will show that $\alpha^\intercal x \geq \beta$ is valid for $\mathcal{Q}^{kt}$ for a fixed (arbitrary) index $t \in T$. We have that $\gamma^t x \geq \gamma_0^t$ is valid for $\mathcal{Q}^{kt}$. Then, if $x \in \mathcal{Q}^{kt}$, since we have nonnegativity on the variables ($\mathcal{Q}^{kt} \subseteq \mathbb{R}^n_{\geq 0}$), it follows that,

$$\alpha^\intercal x = \sum_{j\in[n]} \max_{t'\in T}\{\gamma_j^{t'}\}x_j \geq \sum_{j\in[n]} \gamma_j^t x_j \geq \gamma_0^t \geq \min_{t'\in T}\{\gamma_0^{t'}\} = \beta.$$

Hence, $\alpha^\intercal x \geq \beta$ is valid for $\bigcup_{t\in T} \mathcal{Q}^{kt} \supseteq \mathcal{S}^k \cap \mathcal{P}_D^k$. □

This theorem is significant because it yields a disjunctive cut that, under small perturbations, one could intuit to preserve much of its original strength. Furthermore, its derivation circumvents the need to solve complex optimization problems, such as LPs, by instead leveraging efficient vectorized operations. We illustrate the result of this theorem in Figure 1.

This result extends to all integer feasible points by analogously expanding the disjunction.

Corollary 3.2. *Let $k \in \mathcal{K}$ be the index of an arbitrary instance in a series. Then given a disjunction $\{\mathcal{X}^t\}_{t\in T}$ valid for $\mathbb{Z}^n$ and a set $\{v^t\}_{t\in T}$ of nonnegative Farkas multipliers associated with each term of this disjunction, we have that $\alpha^\intercal x \geq \beta$ for all $x \in \mathcal{S}^k$, where $\alpha_j := \max_{t\in T}\{v^t A^{kt}_{\cdot j}\}$ and $\beta := \min_{t\in T}\{v^t b^{kt}\}$ for all $j \in [n]$.*

Proof. Follow the steps of Theorem 3.1, but replace $\mathcal{S}^k$ for $\mathcal{S}^k \cap \mathcal{P}^k_D$. This replacement can be made because $\{\mathcal{X}^t\}_{t\in T}$ is valid for IP-k, which implies that $\mathcal{S}^k = \mathcal{S}^k \cap \mathcal{P}^k_D$. Thus, Theorem 3.1 allows us to conclude that for all $x \in \mathcal{S}^k$, we have $\alpha^\intercal x \geq \beta$. □

Expanding the disjunction ensures the resulting parametric disjunctive cuts remain valid for all integer feasible points. This is crucial in parameterization, where a disjunction valid for one problem may not hold for another if the feasible region expands beyond the original disjunctive hull. This approach preserves the efficiency gains from Theorem 3.1 while maintaining validity across all integer feasible points.

We illustrate the use of Theorem 3.1 to parameterize a disjunctive cut with Example A.3 in appendix A, and we implement Theorem 3.1 as generateValidParameterizedDisjunctiveCut in appendix B. With proof that Theorem 3.1 generates valid cuts for $\mathcal{P}^k_D$, we turn our attention to setting expectations.

3.2 Sufficient Conditions to Support the Disjunctive Hull

We establish that parametric disjunctive cuts can remain tight against the disjunctive hull when the coefficient matrix is unchanged by introducing the concept of *tight* inequalities, defined as follows.

Definition 3.3. *A* tight *inequality for set $\mathcal{P} \subseteq \mathbb{R}^n$ is a pair $(\alpha, \beta) \in \mathbb{R}^n \times \mathbb{R}$ such that $\mathcal{P} \subseteq \{x \in \mathbb{R}^n : \alpha^\intercal x \geq \beta\}$ and there exists $x^* \in \mathcal{P} : \alpha^\intercal x^* = \beta$.*

The proof follows from the observation that with constant coefficient matrices, Theorem 3.1 can generate inequalities that are tight for one disjunctive term and valid for all, resulting in tightness for the disjunctive hull.

Lemma 3.4. *Let $k, \ell \in \mathcal{K}$. Let $\{\mathcal{X}^t\}_{t\in T}$ be a disjunction, $\{v^t\}_{t\in T}$ be farkas multipliers, $\{\mathcal{N}^t\}_{t\in T}$ be a collection of constraint indices generating $\{v^t\}_{t\in T}$, and $\bar{\alpha} = v^1 A^{k1}$. If $A^k = A^\ell$ and if all $t \in T$ satisfy $v^t \geq 0$, $v^t A^{kt} = \bar{\alpha}$, $(A^{\ell t}_{\mathcal{N}^t \cdot})^{-1} b^{\ell t}_{\mathcal{N}^t} \in \mathcal{Q}^{\ell t}$, and $\{i \in q + q_t : v^t_i > 0\} \in \mathcal{N}^t$, then $(\alpha, \beta) =$ Theorem 3.1$(\ell, \{\mathcal{X}^t\}_{t\in T}, \{v^t\}_{t\in T})$ is tight for $\operatorname{cl}\operatorname{conv}(\cup_{t\in T}\mathcal{Q}^{\ell t})$ with $\alpha = \bar{\alpha}$.*

Proof. Let $t' = \arg\min_{t\in T}\{v^t b^{\ell t}\}$. Our hypothesis provides $v^t A^{\ell t} = v^t A^{kt} = \bar{\alpha}$ for all $t \in T$, which implies that $\max_{t\in T}\{v^t A^{\ell t}\} = v^{t'} A^{\ell t'}$. Applying Theorem 3.1 yields $(\alpha, \beta) = (\max_{t\in T}\{v^t A^{\ell t}\}, \min_{t\in T}\{v^t b^{\ell t}\}) = (v^{t'} A^{\ell t'}, v^{t'} b^{\ell t'})$, a valid inequality for $\operatorname{cl}\operatorname{conv}(\cup_{t\in T}\mathcal{Q}^{\ell t})$. Assumptions on $\mathcal{N}^{t'}$ imply that $\alpha = v^{t'}_{\mathcal{N}^{t'}} A^{\ell t'}_{\mathcal{N}^{t'}\cdot}$, $v^{t'}_{\mathcal{N}^{t'}} = \alpha (A^{\ell t'}_{\mathcal{N}^{t'}\cdot})^{-1}$, $v^{t'}_{\mathcal{N}^{t'}} b^{\ell t'}_{\mathcal{N}^{t'}} = \alpha (A^{\ell t'}_{\mathcal{N}^{t'}\cdot})^{-1} b^{\ell t'}_{\mathcal{N}^{t'}}$, and $(A^{\ell t'}_{\mathcal{N}^{t'}\cdot})^{-1} b^{\ell t'}_{\mathcal{N}^{t'}} \in \mathcal{Q}^{\ell t}$. Since $\beta = v^{t'}_{\mathcal{N}^{t'}} b^{\ell t'}_{\mathcal{N}^{t'}}$, (α, β) is tight for $\mathcal{Q}^{\ell t'}$ and, thus, for $\operatorname{cl}\operatorname{conv}(\cup_{t\in T}\mathcal{Q}^{\ell t})$. $\alpha = \bar{\alpha}$ follows from above conclusions that $\alpha = v^{t'} A^{\ell t'}$ and $v^{t'} A^{\ell t'} = v^{t'} A^{kt'} = \bar{\alpha}$. □

This result clarifies when generated cuts support the disjunctive hull, helping to interpret unexpected computational outcomes and rule out cut strength as a factor. Notably, this excludes cases where the coefficient matrix is perturbed: as shown in Figure 2, such perturbations can produce cuts that fail to support the disjunctive hull. While this complicates analysis, it remains practically useful, as small changes tend to yield similar, nearly tight cuts. We also exclude cases where a basis from the Farkas certificate becomes infeasible. In practice, this often results from an infeasible term, which can be dropped from the disjunction without compromising the tightness of the parametric cut. In rarer cases, the basis becomes infeasible while the overall problem remains feasible; like matrix perturbations, this typically results in a cut that is still nearly tight for small changes. We build on these insights to identify conditions under which parametric disjunctive cuts reliably separate the LP relaxation from the disjunctive hull.

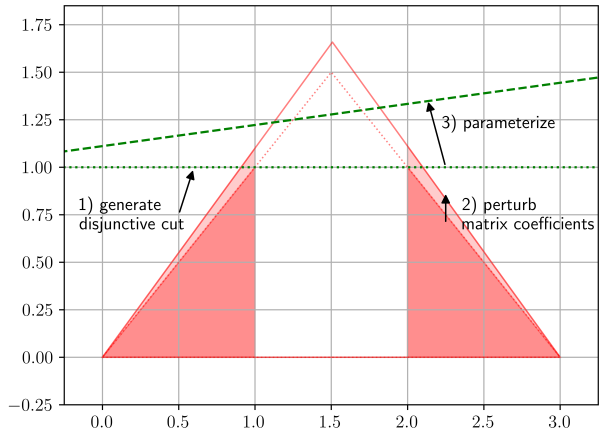


Figure 2: Applying Theorem 3.1 to coefficient matrix-perturbed instances may produce cuts that are not tight against the disjunctive hull.

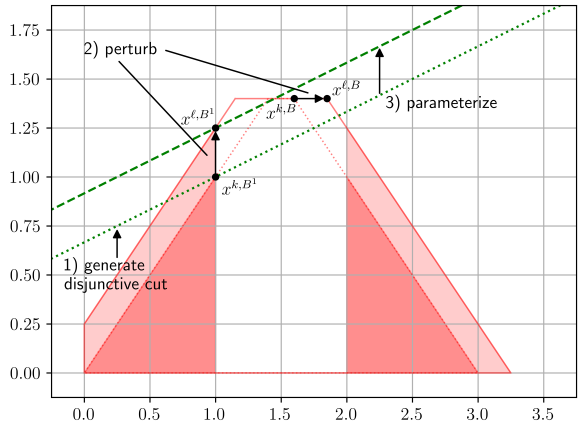


Figure 3: Tight parametric disjunctive cuts may not separate a basic solution ($x^{\ell,B}$) more than one pivot from the binding vertex ($x^{\ell t,B^t}$).

3.3 Sufficient Conditions to Separate LP-Relaxation Solutions

We demonstrate how to express the vector between neighboring basic solutions in terms of the solutions and coefficient matrix to establish conditions under which parametric disjunctive cuts separate one from the disjunctive hull.

Lemma 3.5. *Let $x^{kt,B'}, x^{kt,B^t} \in \mathbb{R}^n$ be basic solutions of* (LP-kt) *such that $B' \setminus B^t = \{j^t\}$ for some $j^t \in [n+1, ..., q+q_t]$. Then $x^{kt,B'} - x^{kt,B^t} = \text{proj}_x(-(\tilde{A}^{kt}_{\cdot B^t})^{-1}\tilde{A}^{kt}_{\cdot j^t}\tilde{x}^{kt,B'}_{j^t})$.*

Proof. Let $\tilde{x}$ be a basic solution of (LP-kt). Then $\tilde{A}^{kt}\tilde{x} = b^{kt}$, leading to the following equivalences:

$$\begin{aligned}\tilde{A}^{kt}\tilde{x} = b^{kt} &\iff \tilde{A}^{kt}_{\cdot B^t}\tilde{x}_{B^t} + \tilde{A}^{kt}_{\cdot N^t}\tilde{x}_{N^t} = b^{kt} \iff \tilde{x}_{B^t} = (\tilde{A}^{kt}_{\cdot B^t})^{-1}(b^{kt} - \tilde{A}^{kt}_{\cdot N^t}\tilde{x}_{N^t}) \\ &\iff \tilde{x}_{B^t} = \tilde{x}^{kt,B^t}_{B^t} - (\tilde{A}^{kt}_{\cdot B^t})^{-1}\tilde{A}^{kt}_{\cdot N^t}\tilde{x}_{N^t}.\end{aligned}$$

Because $x^{kt,B'}$ is a basic solution of (LP-kt), $\tilde{x}^{kt,B'}_{B^t} = \tilde{x}^{kt,B^t}_{B^t} - (\tilde{A}^{kt}_{\cdot B^t})^{-1}\tilde{A}^{kt}_{\cdot N^t}\tilde{x}^{kt,B'}_{N^t}$. $\tilde{x}^{kt,B'}_{N'} = 0$ and $B' \setminus B^t = \{j^t\}$ then implies $\tilde{x}^{kt,B'}_{B^t} = \tilde{x}^{kt,B^t}_{B^t} - (\tilde{A}^{kt}_{\cdot B^t})^{-1}\tilde{A}^{kt}_{\cdot j^t}\tilde{x}^{kt,B'}_{j^t}$ or that $\tilde{x}^{kt,B'}_{B^t} - \tilde{x}^{kt,B^t}_{B^t} = -(\tilde{A}^{kt}_{\cdot B^t})^{-1}\tilde{A}^{kt}_{\cdot j^t}\tilde{x}^{kt,B'}_{j^t}$. Expanded, we have

$$\tilde{x}^{kt,B'}_{j} - \tilde{x}^{kt,B^t}_{j} = \begin{cases} -(\tilde{A}^{kt}_{\cdot B^t})^{-1}_{j\cdot}\tilde{A}^{kt}_{\cdot j^t}\tilde{x}^{kt,B'}_{j^t} & : j \in B^t \\ \tilde{x}^{kt,B'}_{j^t} & : j = j^t \\ 0 & : \text{else} \end{cases}$$

Since $j^t \in [n+1, ..., q+q_t]$, we have $\text{proj}_x(-(\tilde{A}^{kt}_{\cdot B^t})^{-1}\tilde{A}^{kt}_{\cdot j^t}\tilde{x}^{kt,B'}_{j^t}) = \text{proj}_x(\tilde{x}^{kt,B'} - \tilde{x}^{kt,B^t}) = x^{kt,B'} - x^{kt,B^t}$. □

if we have vector between neighboring basic solutions in terms of the coefficient matrix and we know that matrix doesn't change, then the above tells us that the vector between the neighboring basic solutions is just a multiple of itself from one perturbation to the next. Combining that fact with Lemma 3.4 telling us that the parametric disjunctive cut will be tight against the disjunctive hull, we can use vector algebra to conclude when one basic solution separates a neighbor from the disjunctive hull.

Theorem 3.6. *Let $k, \ell \in \mathcal{K}$, B be the basis for basic solutions $x^{k,B}$ and $x^{\ell,B}$, B^t be a basis and $\mathcal{N}^t$ be corresponding active constraints forming basis cones $\mathcal{C}^{kt}_{\mathcal{N}^t}$ for all $t \in T$, $\{\mathcal{X}^t\}_{t\in T}$ be a disjunction, $\{v^t\}_{t\in T}$ be Farkas multipliers, and $(\alpha^k, \beta^k) \in \mathbb{R}^{n+1}$ such that $A^k = A^\ell$, $(\alpha^k)^\intercal x^{k,B} < \beta^k$, and, for all*

$t \in T$, $B \setminus B^t = \{j^t\}$ for some $j^t \in [n+1, ..., n+q]$, $(A^{kt}_{\mathcal{N}^t \cdot}, b^{kt}_{\mathcal{N}^t}, \alpha^k, \beta^k, v^t)$ satisfies FL and $x^{\ell,B} \notin \mathcal{C}^{\ell t}_{\mathcal{N}^t}$. Then $\alpha^{\ell\mathsf{T}} x^{\ell,B} < \beta^\ell$ for $(\alpha^\ell, \beta^\ell)$ resulting from applying Theorem 3.1 to $(\ell, \{\mathcal{X}^t\}_{t\in T}, \{v^t\}_{t\in T})$.

Proof. Let $j \in [n]$. Theorem 3.1 yields $\alpha^\ell_j = \max_{t\in T}\{v^t A^{\ell t}_{\cdot j}\}$. Because it is assumed $A^\ell = A^k$ and $(A^k_{\mathcal{N}\cdot}, b^k_{\mathcal{N}}, \alpha^k, \beta^k, v^t)$ satisfies FL for all $t \in T$, it follows that $\max_{t\in T}\{v^t A^{\ell t}_{\cdot j}\} = \max_{t\in T}\{v^t A^{kt}_{\cdot j}\}$ and $\max_{t\in T}\{v^t A^{kt}_{\cdot j}\} = \max_{t\in T}\{\alpha^k_j\}$. Since $\max_{t\in T}\{\alpha^k_j\}$ reduces to α^k_j, we can conclude $\alpha^\ell_j = \alpha^k_j$. Thus, $\alpha^\ell = \alpha^k$.

Let $t \in T$ such that $\alpha^{\ell\mathsf{T}} x^{\ell t,B} = \beta^\ell$, which is known to exist by Theorem 3.1. FL implies that $\alpha^{k\mathsf{T}} x^{kt,B^t} \geq \beta^k$. Coupling this fact with our hypothesis yields $(\alpha^k)^\mathsf{T} x^{k,B} < \beta^k \leq \alpha^{k\mathsf{T}} x^{kt,B^t}$, which reduces to $\alpha^{k\mathsf{T}}(x^{k,B} - x^{kt,B^t}) < 0$ and allows us to deduce $x^{k,B} \notin \mathcal{C}^{kt}_{\mathcal{N}^t}$.

Let $B' = B \cup [n+q+1, ..., n+q+q_t]$, which implies $x^{k,B} = x^{kt,B'}$. From $B' \setminus B^t = \{j^t\}$, $j^t \in [n+1, ..., n+q]$, and Lemma 3.5, we can conclude that $x^{k,B} - x^{kt,B^t} = \text{proj}_x(-(\tilde{A}^{kt}_{\cdot B^t})^{-1}\tilde{A}^{kt}_{\cdot j^t}\tilde{x}^{kt,B'}_{j^t})$ and $\tilde{x}^{\ell,B} - \tilde{x}^{\ell t,B^t} = \text{proj}_x(-(\tilde{A}^{\ell t}_{\cdot B^t})^{-1}\tilde{A}^{\ell t}_{\cdot j^t}\tilde{x}^{\ell t,B'}_{j^t})$. Leveraging our hypothesis that $A^k = A^\ell$ and the above reductions, we make the following chain of substitutions:

$$A^k = A^\ell \implies \text{proj}_x(-(\tilde{A}^{kt}_{\cdot B^t})^{-1}\tilde{A}^{kt}_{\cdot j^t}) = \text{proj}_x(-(\tilde{A}^{\ell t}_{\cdot B^t})^{-1}\tilde{A}^{\ell t}_{\cdot j^t})$$
$$\implies \frac{\tilde{x}^{k,B} - \tilde{x}^{kt,B^t}}{\tilde{x}^{kt,B'}_{j^t}} = \frac{\tilde{x}^{\ell,B} - \tilde{x}^{\ell t,B^t}}{\tilde{x}^{\ell t,B'}_{j^t}} \implies (\tilde{x}^{k,B} - \tilde{x}^{kt,B^t}) = \frac{\tilde{x}^{kt,B'}_{j^t}}{\tilde{x}^{\ell t,B'}_{j^t}}(\tilde{x}^{\ell,B} - \tilde{x}^{\ell t,B^t}).$$

Since $x^{k,B} \notin \mathcal{C}^{kt}_{\mathcal{N}^t}$ and $x^{\ell,B} \notin \mathcal{C}^{\ell t}_{\mathcal{N}^t}$, $\frac{\tilde{x}^{kt,B'}_{j^t}}{\tilde{x}^{\ell t,B'}_{j^t}} > 0$, allowing further substitution resulting in

$$\alpha^{k\mathsf{T}}\frac{\tilde{x}^{kt,B'}_{j^t}}{\tilde{x}^{\ell t,B'}_{j^t}}(\tilde{x}^{\ell,B} - \tilde{x}^{\ell t,B^t}) < 0 \implies \alpha^{\ell\mathsf{T}}\tilde{x}^{\ell,B} < \alpha^{\ell\mathsf{T}}\tilde{x}^{\ell t,B}.$$

Thus, $\alpha^{\ell\mathsf{T}}\tilde{x}^{\ell,B} < \beta^\ell$. □

This ensures parametric disjunctive cuts not only support the disjunctive hull but also strengthen the LP relaxation, particularly near an optimal solution. Using the LP-relaxation solution as a proxy, we conclude that if a disjunctive cut sufficiently separates it from the disjunctive hull, it will do so consistently throughout the series, reinforcing the LP relaxation near optimal solutions.

While this result is useful, practical applications may extend beyond its guarantees by allowing multiple pivots from the LP-relaxation solution to each disjunctive term solution, as shown in Figure 3, or using larger disjunctions. This sacrifices a strict separation guarantee in favor of potentially deeper cuts that often still separate the LP-relaxation solution. Even when they do not, these cuts can still tighten the LP-relaxation and improve solve deeper in the branch-and-bound tree. For further discussion, see Balas and Kazachkov [4].

3.4 Warm-Starting MILPs Is NP-Hard

With sufficient conditions in Theorem 3.6 for parametric disjunctive cuts to separate the LP relaxation from the disjunctive hull, we aim to understand their impact on solve time. We first define a *certificate* as proof of optimality in a MILP and formally introduce the Mixed-Integer Linear Optimization Problem and its warm start.

Definition 3.7. *The pair $(x, \mathcal{X})$, consisting of a vector $x \in \arg$IP-k and disjunction $\mathcal{X} = \{\mathcal{X}^t\}_{t\in T}$ arising from the leaves of a Branch-and-Bound tree, is a* certificate *for IP-k when $\min_{t\in T} c^k \bar{x}^{kt} = c^k x$.*

Definition 3.8. *The* Mixed-Integer Linear Optimization Problem (MILP) *takes as an input the formulation IP-k and returns a certificate $(\bar{x}, \bar{\mathcal{X}})$ for IP-k if $\mathcal{S}^k \neq \emptyset$ and (NULL, NULL) otherwise.*

Definition 3.9. *The* Warm-Started Mixed-Integer Linear Optimization Problem (WS) *takes as inputs the formulations IP-k and IP-ℓ and a certificate $(x, \mathcal{X})$ for IP-k such that exactly one of the following statements is true:*

- $c^k \neq c^\ell$, *or*
- *there exists exactly one $i \in [q]$ such that $A^\ell_{i\cdot} \neq A^k_{i\cdot}$ or $b^\ell_i \neq b^k_i$.*

It returns a certificate $(\bar{x}, \bar{\mathcal{X}})$ for IP-ℓ if $\mathcal{S}^\ell \neq \emptyset$ and (NULL, NULL) otherwise.

Using these definitions, we construct a Cook Reduction to show that warm-starting MILPs is NP-Hard.

Theorem 3.10. *The Warm-Started Mixed-Integer Linear Optimization Problem is NP-Hard.*

Proof. We reduce MILP to WS via Cook Reduction. Let $h, \ell \in \mathcal{K}$ and $(\bar{x}, \bar{\mathcal{X}})$ be a certificate of IP-h. Then $A^h, A^\ell \in \mathbb{R}^{q \times n}$, $\bar{x} \in \mathbb{R}^n$ and $|\bar{\mathcal{X}}| < 2^n$, and thus the inputs to WS are asymptotically constant bound. Consider $(x, \mathcal{X})$ to be a certificate for IP-k created by solving IP-k as a MILP. Then $(x, \mathcal{X})$ is a feasible output for solving IP-k via our Cook Reduction as it is guaranteed to return a certificate for IP-k since it solves a series of restrictions ending with IP-k. Consider the converse, where $(x, \mathcal{X})$ is a certificate for IP-k created by solving IP-k via our Cook Reduction. Since $(x, \mathcal{X})$ is a certificate for IP-k, it is also a feasible output for solving IP-k as a MILP. Therefore, a bijection exists between the feasible instances of MILP and WS. Assuming we have a polynomial-time algorithm for WS, our Cook Reduction, constant bounds on the sizes of inputs to WS, and bijection between the feasible instances of MILP and WS show we can solve MILP in polynomial-time. However, MILP is known to be NP-Hard; thus, we conclude WS is NP-Hard, too.

Algorithm 1 Cook Reduction

Require: IP-k

1: $A^h = 0_{q \times n}$, $b^h = 0_q$, $c^h = 0_n$ ▷ Instantiate IP-h
2: $\bar{x} = 0_n$, $\bar{\mathcal{X}} = \{\mathbb{R}^n\}$ ▷ Initial certificate for IP-h is trivial
3: IP-ℓ = IP-h ▷ Instantiate IP-ℓ to a copy of IP-h
4: **for** $i \in [q]$ **do**
5: $\quad A^\ell_{i\cdot} = A^k_{i\cdot}$, $b^\ell_i = b^k_i$ ▷ Add IP-k's next constraint to IP-ℓ
6: $\quad \bar{x}, \bar{\mathcal{X}} =$ WS(IP-h, IP-ℓ, $\bar{x}, \bar{\mathcal{X}}$) ▷ Warm-start IP-$\ell$ with the previous certificate
7: $\quad$ **if** $\bar{x}$ is NULL **then** ▷ IP-ℓ infeasible implies IP-k infeasible
8: $\quad\quad$ **return** (NULL, NULL)
9: $\quad$ **end if**
10: $\quad$ IP-h = IP-ℓ
11: **end for**
12: $\bar{x}, \bar{\mathcal{X}} =$ WS(IP-h, IP-k, $\bar{x}, \bar{\mathcal{X}}$) ▷ Warm-start IP-$k$ with IP-h's certificate
13: **return** $\bar{x}, \bar{\mathcal{X}}$

□

Since cutting planes influence solve times indirectly, experiments may reveal cases where parametric disjunctive cuts provide no performance improvement. However, rather than attributing this entirely to their failure, we recognize above that warm-starting itself is inherently difficult.

4 Computational Results

The overarching goal of our experiments was to compare the effect generating VPCs in different ways has on various performance metrics when solving a series of related MILPs. This section is divided by the setup for the experiments (Section 4.1) and their results (Sections 4.2, 4.3, and 4.4).

4.1 Computational Setup

Before conducting our experiments, we generated series of randomly perturbed MILP instances from MIPLIB 2017 and implemented four disjunctive cut generators with varying degrees of parameterization. We then tested these generators on a server cluster using existing MILP solvers to produce the data analyzed in the following sections. This setup ensures a robust evaluation of disjunctive cuts' impact on solver performance.

4.1.1 MIPLIB 2017 Generates Series of Randomly Perturbed MILP Instances

The MILP instances used in our experiments consist of two groups: a set of *base instances*, indexed by $\mathcal{B}$, and a collection of *test instances*, indexed by $\mathcal{T}$. Base instances are drawn from the MIPLIB 2017 dataset and include those with at most 5,000 variables and 5,000 constraints after presolving, provided they solve to optimality within 1 hour. Test instances, subindexed by $(k, u, \theta) \in \mathcal{B} \times \{A, b, c\} \times \{.5, 2\}$, are derived from a base instance k using Algorithm 3(u^k, θ). These instances are created by perturbing c^k if $u = c$, b^k if $u = b$, or A^k if $u = A$, up to a specified *degree* θ. A test instance $\ell \in \mathcal{T}_{ku\theta}$ is considered to be within θ degree(s) of perturbation of the base instance k if Algorithm 2(u^k, u^ℓ) returns a value no greater than θ. For each $(k, u, \theta) \in \mathcal{T}$, Algorithm 3$(u^k, \theta)$ is executed iteratively, stopping after one of the following conditions is met: 1000 attempts, 4 hours, or the successful generation of three test instances (unless specified otherwise). Consequently, the number of test instances generated for each base instance varies.

Algorithm 3 employs two black-box subroutines: `toVector` and `toMatrix`. The `toVector` subroutine converts a matrix to a vector, while `toMatrix` restores the resulting vector to the shape of the original matrix.

Algorithm 2 Find Degree

Require: u^k, u_{new}

angle_difference $\leftarrow \cos^{-1}\left(\frac{u^k \cdot u_{\text{new}}}{\|u^k\| \|u_{\text{new}}\|}\right)$ ▷ Angle between vectors

norm_difference $\leftarrow \left|\frac{\|u^k\| - \|u_{\text{new}}\|}{\|u^k\|}\right|$ ▷ Relative change in vector norms

return max{angle_difference, norm_difference}

Algorithm 3 Find Perturbation

Require: u^k, θ ▷ starting vector, desired perturbation

$u^k \leftarrow$ `toVector`(u^k) if $u^k \in \mathbb{R}^{q \times n}$ else u^k ▷ Flatten to vector if needed

$u_{\text{final}}, \epsilon \leftarrow$ NULL, 1

while $\epsilon \geq 10^{-6}$ and $u_{\text{final}} ==$ NULL **do** ▷ until tolerance met or perturbed vector found

  $u_{\text{new}}, u_{\text{prev}} \leftarrow u^k$, NULL

  while Find Degree$(u^k, u_{\text{new}}) < \theta$ **do** ▷ Until desired perturbation reached

    $u_{\text{prev}} \leftarrow u_{\text{new}}$

    $i \leftarrow$ `random`$([|u|])$ ▷ Select random element

    $u_{\text{new}}[i] \leftarrow u_{\text{new}}[i] +$ `random`$([-\epsilon, \epsilon])$ ▷ Randomly perturb by $\pm\epsilon$

  end while

  if $u_{\text{prev}} \neq u^k$ **then** ▷ If an iteration produced a vector within the desired perturbation

    $u_{\text{final}} \leftarrow u_{\text{prev}}$

  end if

  $\epsilon \leftarrow \epsilon/2$

end while

return `toMatrix`(u_{final}) if $u_{\text{final}} \in \mathbb{R}^{qn}$ else u_{final} ▷ Return as matrix if needed

4.1.2 Four Disjunctive Cut Generators Varying in Parameterization Are Implemented

Our experiments explore variations in how disjunctive cuts are generated. We use d to denote the number of disjunctive terms in the Branch-and-Bound queue at the point when a disjunction is generated from the leaves of the solve's Branch-and-Bound tree. For each combination of $d \in \{4, 16, 64\}$ and $(k, u, \theta) \in \mathcal{T}$, we evaluate four approaches to generating disjunctive cuts. Algorithm 4 (Default) tests solving the series $\mathcal{T}_{ku\theta}$ using default solver settings without disjunctive cuts. Algorithm 5 (Calculated Disjunction, Calculated Cuts) follows the disjunctive cut generation method prescribed in Balas and Kazachkov [4, Algorithm 1]. Algorithm 6 (Parametric Disjunction, Calculated Cuts) and Algorithm 7 (Parametric Disjunction, Parametric Cuts) apply disjunctive cuts with varying levels of parameterization to solve the series. The difference between Param Disj, Calc Cuts and Param Disj, Param Cuts is that the latter generates cuts as described in Theorem 3.1 while the former recylces a previously found disjunction when solving PRLPs.

Algorithm 4 Default

Require: $k, u, \theta, \mathcal{T}$

for all $\ell \in \mathcal{T}_{ku\theta}$ **do**
  `Branch-and-Cut` (IP-ℓ) ▷ Solve each test instance under default settings
end for

Algorithm 5 Calculated Disjunction, Calculated Cuts

Require: $d, k, u, \theta, \mathcal{T}$

for all $\ell \in \mathcal{T}_{ku\theta}$ **do**. ▷ For each test instance
  $\{\mathcal{X}^t\}_{t \in T} \leftarrow$ `Branch-and-Bound` (IP-ℓ, d) ▷ Generate a new disjunction
  $T^\ell_{\text{feas}} \leftarrow \{t \in T : \mathcal{Q}^{\ell t} \neq \emptyset\}$ ▷ Using feasible terms only
  $\Pi^\ell \leftarrow$ Balas and Kazachkov [4, Algorithm 1](IP-ℓ, $\{\mathcal{X}^t\}_{t \in T^\ell_{\text{feas}}}$) ▷ Get cuts from PRLPs
  `Branch-and-Cut` (IP-ℓ, Π^ℓ) ▷ Solve with disjunctive cuts
end for

Algorithm 6 Parametric Disjunction, Calculated Cuts

Require: $d, k, u, \theta, \mathcal{T}$

$\{\mathcal{X}^t\}_{t \in T} \leftarrow$ `Branch-and-Bound` (IP-k, d) ▷ Generate initial disjunction to parameterize
for all $\ell \in \mathcal{T}_{ku\theta}$ **do**. ▷ For each test instance
  $T^\ell_{\text{feas}} \leftarrow \{t \in T : \mathcal{Q}^{\ell t} \neq \emptyset\}$ ▷ Using feasible terms only
  $\Pi^\ell \leftarrow$ Balas and Kazachkov [4, Algorithm 1](IP-ℓ, $\{\mathcal{X}^t\}_{t \in T^\ell_{\text{feas}}}$) ▷ Get cuts from PRLPs
  `Branch-and-Cut` (IP-ℓ, Π^ℓ) ▷ Solve with disjunctive cuts
end for

Algorithms 4 - 7 utilize the subroutines `Branch-and-Cut` and `Branch-and-Bound`. `Branch-and-Cut` solves a MILP using a standard solver (e.g., CBC or Gurobi) with default settings unless specified otherwise. Its arguments include: IP-k, the instance to solve; optionally, $d \in \mathbb{N}$, the number of queued nodes for termination; $\{\mathcal{X}^t\}_{t \in T} \subseteq \mathbb{R}^n$, a starting disjunction such that $\bigcap_{t \in T} \mathcal{X}^t = \emptyset$; and $\Pi \subseteq \mathbb{R}^{n+1}$, an initial pool of cutting planes. `Branch-and-Cut` outputs $\{\bar{\mathcal{X}^t}\}_{t \in T} \subseteq \mathbb{R}^n$, the terminating disjunction. `Branch-and-Bound` is similar to `Branch-and-Cut` but disables cut generation, omitting the initial pool of cutting planes as an argument. Its configuration follows Balas and Kazachkov [4, Appendix C].

Algorithms 4 - 7 require the selection of several parameters. For each $d \in \{4, 16, 64\}$, a disjunction $\{\mathcal{X}^t\}_{t \in T}$ generated from partially solving IP-$k$includes the d queued terms, the infeasible terms, and terms implicitly added due to tightening during strong branching. While Balas and Kazachkov [4] excludes pruned terms in their experiments, we must compute Farkas certificates for all terms, as problem perturbations can invalidate a term's pruning. Consequently, we include all feasible terms in

Algorithm 7 Parametric Disjunction, Parametric Cuts

Require: $d, k, u, \theta, \mathcal{T}$

$\{\mathcal{X}^t\}_{t\in T} \leftarrow$ `Branch-and-Bound` (IP-k, d) ▷ Generate initial disjunction to parameterize
$T^k_{\text{feas}} \leftarrow \{t \in T : \mathcal{Q}^{kt} \neq \emptyset\}$ ▷ Using feasible terms only
$\Pi^\ell \leftarrow$ Balas and Kazachkov [4, Algorithm 1](IP-ℓ, $\{\mathcal{X}^t\}_{t\in T^k_{\text{feas}}}$) ▷ Get initial cuts from PRLPs
$\mathcal{V} \leftarrow \emptyset$
for all $(\alpha^k, \beta^k) \in \Pi^k$ **do**. ▷ For each cut
  $v \leftarrow \emptyset$
  for all $t \in T$ **do**. ▷ For each term
    $\mathcal{N}^t \leftarrow \mathcal{N}$ such that $A^{kt}_{\mathcal{N}\cdot}\bar{x}^{kt} = b^{kt}$ **if** $\mathcal{Q}^{kt} \neq \emptyset$ **else** NULL ▷ Get term's optimal basis
    $v^t \leftarrow$ Lemma 2.3(IP-k, $\mathcal{C}^{kt}_{\mathcal{N}^t}, \alpha, \beta$) ▷ Calculate its certificate
  end for
  $\mathcal{V} \leftarrow \mathcal{V} \cup \{v\}$
end for
for all $\ell \in \mathcal{T}_{ku\theta}$ **do** ▷ For each test instance
  $\Pi^\ell \leftarrow \emptyset$
  $T^\ell_{\text{feas}} \leftarrow \{t \in T : \mathcal{Q}^{\ell t} \neq \emptyset\}$ ▷ Using feasible terms only
  for all $v \in \mathcal{V}$ **do** ▷ For each certificate
    $(\alpha^\ell, \beta^\ell) \leftarrow$ Theorem 3.1(IP-k, $\{\mathcal{X}^t\}_{t\in T^\ell_{\text{feas}}}, \{v^t\}_{t\in T^\ell_{\text{feas}}}$) ▷ Parameterize the corresponding cut
    $\Pi^\ell \leftarrow \Pi^\ell \cup \{(\alpha^\ell, \beta^\ell)\}$
  end for
  `Branch-and-Cut` (IP-ℓ, Π^ℓ) ▷ Solve with disjunctive cuts
end for

the PRLP and set the Farkas certificate for infeasible terms to 0, since Lemma 2.3 requires IP-kt to be feasible for $t \in T$. When applying Theorem 3.1 to ℓ, $\mathcal{X}$, and $\{v^t\}_{t\in T}$, we disregard $t \in T$ where $\mathcal{Q}^{\ell t} = \emptyset$. The number of VPCs generated from PRLPs does not exceed the number of fractional integer variables in the root LP relaxation. Each PRLP is solved using a sequence of carefully chosen objective functions to produce a single round of rank-one cuts. These cuts are valid for the same disjunction and are obtained nonrecursively. For further details, see [4].

4.1.3 Experiments Assess Disjunctive Cuts' Impact on Gurobi

Our experiments relied on the following computing resources. We used the C++ library VPC [1] to generate disjunctions and VPCs from solving PRLPs, creating a fork so that disjunctions would be lattice free. We relied on Gurobi 10 as our `Branch-and-Cut` solver, writing a callback to add our cuts at the root node, and CBC 2.10 for `Branch-and-Bound`. The experiments are conducted on a server cluster with 16 AMD Opteron Processor 6128's; 2 CPUs and up to 15GB of RAM were dedicated to each experiment. We allowed 3600 seconds each for generating disjunctive cuts as well as for solving each MILP instance.

We use these resources to evaluate disjunctive cuts' impact on Gurobi, starting with a high-performing subset (Section 4.2) before expanding our analysis to assess broader performance trends across related MILPs and varying input parameters. For each test instance, we evaluate performance using two approaches, leading to several metrics categorized as follows: (1) *Root node performance*: Measures the strength of VPCs and the time required to generate them. (2) *Branch-and-Cut performance*: Reports the time and number of nodes processed to reach optimality. Root node and Branch-and-Cut performance are analyzed in detail in subsections Section 4.3 and Section 4.4, respectively.

Table 1: Solve time (in seconds) for a subset of series with significant solve time improvement when enabling disjunctive cut generation.

Degree	Terms	Type	Base Instance	Default	Param Disj Param Cuts	Param Disj Calc Cuts	Calc Disj Calc Cuts	Series Length
0.5	4	matrix	neos-631517	1282	1785	1081	855	11
	16	objective	arki001	1408	508	388	337	11
	16	rhs	tr12-30	2471	1797	1793	1652	10
2.0	64	matrix	tr12-30	431	455	411	397	11
	16	objective	neos-631517	3880	6813	4102	2758	9
	64	rhs	neos-631517	383	500	265	119	5

Table 2: For each combination of perturbation degree and disjunctive terms, average solve time (in seconds) for each of 372 bm23 test instances excluding the time to generate disjunctive cuts.

Degree	Terms	Default	Param Disj Param Cuts	Param Disj Calc Cuts	Calc Disj Calc Cuts
0.5	4	0.202	0.184	0.181	0.170
	16	0.187	0.174	0.167	0.166
	64	0.188	0.140	0.140	0.159
2.0	4	0.179	0.178	0.174	0.175
	16	0.172	0.166	0.160	0.162
	64	0.177	0.137	0.132	0.142

4.2 High Performing Subset of Experiments

Our experiments in Sections 4.3 and 4.4 are motivated by two key findings on selected MILP series: parametric disjunctive cuts can significantly reduce total solve time, with greater improvement as parameterization increases (Table 1), and test instance solve time reductions tend to grow with the number of disjunctive terms used for cut generation (Table 2).

4.2.1 Parametric Disjunctive Cuts Can Significantly Reduce Series' Solve Time

Table 1 is structured to highlight this first point. Column 1 represents the perturbation "Degree" ($\theta \in \{.5, 2\}$), while Column 2 indicates the number of "Terms" ($d \in \{4, 16, 64\}$) queued in a partial Branch-and-Bound solve when generating a disjunction. Column 3 specifies the "Type" of perturbation differentiating test instances from their base instance, and Column 4 lists the base instance name from MIPLIB 2017. Columns 5 through 8 tell us for given degree θ, number of disjunctive terms d, type of perturbation u, and base instance k, the total amount of time (in seconds) to solve base instance k plus all test instances in $\mathcal{T}_{ku\theta}$ for each of our four disjunctive cut generation algorithms. Column 9 tells us the number of instances in the series, including the base instance.

The rows of Table 1 were selected to illustrate key observations. First, despite the complexity of generating disjunctive cuts by solving PRLPs for the base instance in Calc Disj, Calc Cuts, Param Disj, Calc Cuts, and Param Disj, Param Cuts, total series solve time can be significantly reduced compared to Default, where no disjunctive cuts are used. Greater improvements are achieved when parameterizing disjunctive cuts (Param Disj, Calc Cuts and Param Disj, Param Cuts). Additionally, reusing both Farkas multipliers and the disjunction that generated previous cuts (Param Disj, Param Cuts) yields even better results compared to reusing only the disjunction (Param Disj, Calc Cuts).

This subset was chosen to show that these improvements are not limited to a specific "right"

Table 3: The number of base and test instances where VPC generation succeeds for all combinations of possible degree of perturbation and terms.

Type	Base Instances	Test Instances
matrix	42	103
objective	101	361
rhs	29	75

combination of parameters. Table 1 demonstrates that substantial reductions in total solve time occur across a variety of perturbation degrees, perturbation types, disjunctive term counts, and base instances. These results align with our expectations: disjunctive cuts, particularly the VPCs used in this study, are exceptionally strong. If their generation cost can be amortized across a series of related MILPs, total solve time can be significantly reduced. Since this impact can be substantial, improvements are achievable across a wide range of parameter settings.

4.2.2 Solve Time Can Inversely Correlate with the Number of Disjunctive Terms

We structure Table 2 to illustrate the second point. This table is indexed by its first two columns, which have the same meaning as in Table 1. Columns 3 through 6 report the average solve time (in seconds) for each test instance in $\mathcal{T}_{ku\theta}$, given degree θ, disjunctive terms d, and base instance bm23. These values are averaged across all perturbation types and *exclude disjunctive cut generation time* for each of the four disjunctive cut generation algorithms.

This table highlights the relationship between test instance solve time and the number of disjunctive terms used to generate cuts, showing that solve time tends to decrease as more terms are added—at least when excluding disjunctive cut generation time. This trend aligns with expectations, as additional disjunctive terms produce stronger cuts that can accelerate solving. Since this effect is independent of perturbation degree, we anticipate the pattern observed in Table 2, where solve time (excluding disjunctive cut generation) improves consistently with more disjunctive terms, regardless of perturbation degree. Additionally, we expect that as test instances deviate further from the base instance (i.e., with higher perturbation degrees), parameterized cuts become less effective at refining the LP relaxation near the optimal solution. This diminished effectiveness is generally reflected in Table 2.

4.3 Root Node Performance

The next subsections evaluate the strength and generation time of disjunctive cuts across our four generators, showing that less parameterized cuts tend to be stronger, while more parameterized ones generate faster. These trends become more pronounced as the number of disjunctive terms increases.

For consistency, only test instances for which VPCs are generated for all combinations of possible degree of perturbation and terms are considered when evaluating root node performance, counts of which are detailed in Table 3. Given a time limit on VPC generation and that many pruned, feasible terms can be added to disjunctions when solving a PRLP, some test instances hit their time limit for Calc Disj, Calc Cuts and Param Disj, Calc Cuts resulting in their exclusion. Further, fewer base instances were tested for matrix and constraint bound (rhs) perturbations than base instances as our method for creating test instances often resulted in infeasibility when perturbing the feasible region. Table 3 breaks down the total number of base and test instances considered for each type of perturbation. For reference, 104 MIPLIB 2017 base instances meet our requirements of at most 5000 presolved rows and columns.

Table 4: Average percent integrality gap closed by disjunctive cuts and implied by disjunctions, grouped by degree of perturbation, size of disjunction, and type of perturbation.

Degree	Terms	Type	Calc Disj	Param Disj	Calc Disj Calc Cuts	Param Disj Calc Cuts	Param Disj Param Cuts
0.5	4	matrix	6.09	4.99	4.07	3.48	3.42
		objective	5.49	4.92	3.90	3.72	3.42
		rhs	5.29	3.94	3.10	3.02	2.61
	16	matrix	11.09	9.41	6.11	6.06	4.99
		objective	9.52	8.64	5.62	5.37	4.70
		rhs	9.37	7.40	5.89	5.73	4.65
	64	matrix	16.83	13.94	8.59	8.32	6.30
		objective	14.39	12.65	8.46	7.59	6.39
		rhs	13.72	10.52	9.06	7.63	6.25
2.0	4	matrix	9.25	4.79	4.85	3.04	2.07
		objective	6.13	4.69	4.32	3.54	3.11
		rhs	12.84	3.51	3.33	2.77	1.51
	16	matrix	16.13	7.99	7.43	5.10	1.83
		objective	10.71	7.81	6.05	5.17	3.93
		rhs	16.75	6.64	5.85	4.81	2.64
	64	matrix	22.50	10.66	10.46	6.41	1.48
		objective	15.78	11.50	8.98	7.15	5.41
		rhs	21.42	9.31	8.47	6.52	3.34

4.3.1 Parametric Disjunctive Cuts Improve Dual Bound at the Root Node

To measure the strength of disjunctive cuts, we calculate the amount of integrality gap they close (defined in appendix C) at the root node. We look at this from two perspectives. The first way is to look at the gap closed by disjunctive cuts and their generating disjunctions alone. The results of this can be seen in Table 4. The second way is to look at the gap closed by disjunctive cuts when combined with the default cuts CBC generates at the root node. The results are in Table 5.

In Table 4, column 1 represents the "Degree" ($\theta \in \{.5, 2\}$) of perturbation, while column 2 indicates the number of "Terms" ($d \in \{4, 16, 64\}$) queued in a partial Branch-and-Bound solve when a disjunction is generated. Columns 3 and 4 show the average integrality gap closed by the disjunctions used to generate our disjunctive cuts, where Calc Disj corresponds to Calc Disj, Calc Cuts, and Param Disj corresponds to Param Disj, Calc Cuts and Param Disj, Param Cuts. The final three columns present the average integrality gap closed at the root node, considering only the VPCs added according to the specified disjunctive cut generator.

The trends amongst VPCs and their generating disjunctions regarding integrality gap closed at the root node in Table 4 are to be expected. First, as the size of the disjunctions used grow, the integrality gap closed by disjunctive cuts and their generating disjunctions improves. This is unsurprising as the disjunctions are based off of partial Branch-and-Bound trees, and the dual bound improves monotonically as Branch-and-Bound progresses. Second, the disjunctions (Calc Disj, Param Disj) close at least as much gap as the cuts they generate (Calc Disj, Calc Cuts, Param Disj, Calc Cuts, Param Disj, Param Cuts), which follows from a property of disjunctive cut generation. Third, disjunctions generated in Calc Disj close more integrality gap than disjunctions generated in Param Disj, and the same relationship exists between the cuts they generate (Calc Disj, Calc Cuts and Param Disj, Calc Cuts, respectively). For a well-tuned solver like Gurobi, this is again a predictable result as changes to a MILP instance often result in a solver applying different branching decisions in an attempt to more

efficiently close the integrality gap. Fourth, differences in integrality gap closed between Calc Disj and Param Disj as well as between Calc Disj, Calc Cuts and Param Disj, Calc Cuts both increase as degree of perturbation increased. This is to be expected as greater differences in problem structure lead to greater differences in disjunctions employed while solving them, causing a weakening of old disjunctions and farkas coefficients. Finally, we see VPCs generated from Param Disj, Param Cuts tend to not close as much integrality gap as VPCs generated from Param Disj, Calc Cuts. Param Disj, Param Cuts recycles Farkas certificates generating VPCs whereas Param Disj, Calc Cuts tailors its VPCs' Farkas certificates by solving PRLPs, which often results in stronger cuts. Further, this difference tends to be most pronounced for matrix perturbations when broken down by perturbation type. Although Lemma 3.4 shows parametric disjunctive cuts support the disjunctive hull when the coefficient matrix does not change, the result does not hold for general coefficient matrix perturbations, as shown by Figure 2.

Table 5: Average percent root integrality gap closed from Gurobi's default cut generators plus different choice of VPCs added based on experiment run. VPCs parameterize best for large disjunctions and small perturbations.

Degree	Terms	Type	Default	Param Disj Param Cuts	Calc Disj Calc Cuts
0.5	4	matrix	70.12	70.11	70.32
		objective	59.96	60.55	60.51
		rhs	59.76	59.96	60.27
	16	matrix	70.12	70.17	70.62
		objective	59.96	61.10	61.06
		rhs	59.76	61.00	61.58
	64	matrix	70.12	70.71	70.55
		objective	59.96	61.56	61.62
		rhs	59.76	61.87	63.27
2.0	4	matrix	75.50	75.31	75.80
		objective	60.39	60.42	60.39
		rhs	61.70	61.15	61.24
	16	matrix	75.50	75.81	75.81
		objective	60.39	60.46	60.58
		rhs	61.70	61.42	62.52
	64	matrix	75.50	75.54	75.90
		objective	60.39	60.96	61.64
		rhs	61.70	62.17	64.27

The columns in Table 5 can be understood as follows. "Degree" and "Terms" take the same meaning as in Table 4. The columns Default, Param Disj, Param Cuts, and Calc Disj, Calc Cuts refer to the integrality gap Gurobi closed at the root node with its default cut generators plus VPCs generated by the respective disjunctive cut generator.

With respect to Table 4, a couple of similar results are apparent in Table 5. Even with the addition of default cuts generated at the root node, VPCs from Calc Disj, Calc Cuts still close at least as much integrality gap than VPCs from Param Disj, Param Cuts, and, when holding perturbation constant, VPCs close more integrality gap when they are generated against more disjunctive terms. Again, there appears to be an inverse relationship between the degree of perturbation and difference of integrality gap closed between Calc Disj, Calc Cuts to Param Disj, Param Cuts.

Table 5 also yields a couple of new trends. Instances with VPCs (Param Disj, Param Cuts,

Calc Disj, Calc Cuts) often see an improvement in root integrality gap closed compared to the same instances without VPCs (Default). This aligns with the results recorded by Balas and Kazachkov [4] in similar experiments and could be due to the strength of VPCs or that VPCs separate a MILP's relaxation from more extrema than its optimal solution. The difference of gap closed between VPCs in Param Disj, Param Cuts and Calc Disj, Calc Cuts is much smaller when applied along with root cuts as compared to Table 4, which suggests that default root cuts recover some of the refined relaxation lost from using VPCs from Param Disj, Param Cuts instead of from Calc Disj, Calc Cuts. Additionally, the inverse relationship between perturbation and differences of integrality gap closed extends to Default when compared to each of Calc Disj, Calc Cuts and Param Disj, Param Cuts. Lastly, the effectiveness of default cuts for matrix perturbed instances obscures if there is a performance degradation due to these cuts not supporting the disjunctive hull, which could be due to our perturbation method only generating test instances for a subset of the base instances.

4.3.2 Parameterization Reduces Disjunctive Cut Generation Time

Table 6: Average root node processing time (in seconds) broken out by different choice of VPCs added and type of perturbation based on experiment run.

Degree	Terms	Type	Default	Param Disj Param Cuts	Param Disj Calc Cuts	Calc Disj Calc Cuts
0.5	4	matrix	0.96	0.98	3.15	33.36
		objective	1.01	1.09	8.12	5.85
		rhs	0.49	0.58	1.20	1.34
	16	matrix	0.89	1.37	8.61	56.71
		objective	1.04	1.51	20.35	26.80
		rhs	0.49	0.90	3.24	4.99
	64	matrix	0.90	2.07	47.80	34.16
		objective	1.02	2.42	67.67	70.95
		rhs	0.48	1.20	13.47	18.46
2.0	4	matrix	0.71	0.78	5.17	5.62
		objective	1.03	1.07	15.38	4.30
		rhs	0.48	0.46	0.96	2.43
	16	matrix	0.70	1.35	9.20	11.02
		objective	1.01	1.56	20.63	22.25
		rhs	0.41	0.67	2.56	4.11
	64	matrix	0.71	2.63	31.44	20.93
		objective	0.99	2.44	81.49	64.11
		rhs	0.42	1.11	10.63	13.19

The columns in Table 6 are defined as follows. "Degree," "Terms," and "Type" retain their previously established meanings. The columns Default, Param Disj, Param Cuts, Param Disj, Calc Cuts, and Calc Disj, Calc Cuts represent the time (in seconds) that Gurobi requires to process the root node under each respective disjunctive cut generator.

The trends observed in Table 6 align with expectations. Processing time is shortest for Default, as it involves only a subset of the tasks performed by the other three options. Param Disj, Calc Cuts and Calc Disj, Calc Cuts take significantly longer than Param Disj, Param Cuts since the first two solve a series of PRLPs, which the third does not. Among the two, Param Disj, Calc Cuts tends to be faster than Calc Disj, Calc Cuts because it reuses an existing disjunction rather than generating a new one through a partial Branch-and-Bound solve. Processing times for Param Disj, Param Cuts,

Param Disj, Calc Cuts, and Calc Disj, Calc Cuts increase with the number of terms, as Param Disj, Param Cuts has more terms to parameterize and the PRLPs solved by Param Disj, Calc Cuts and Calc Disj, Calc Cuts grow larger. By contrast, Default remains relatively unaffected by changes in degree, terms, or type, with any minor variations possibly attributable to differences in the underlying test sets and problem structures.

A couple trends were expected to be absent in Table 6, and this was indeed the case. For instance, while generating a disjunction from a partial Branch-and-Bound tree, some perturbations add more pruned nodes than others, especially as the number of terms queued before generating a disjunction increases. This might explain the variance observed for Param Disj, Param Cuts, Param Disj, Calc Cuts, and Calc Disj, Calc Cuts. Furthermore, there are no consistent trends distinguishing the type of perturbation, as it does not affect the complexity of any disjunctive cut generator.

4.3.3 Parameterization Is (Nearly) Pareto Optimal in Generation Time and Strength

Table 7: Root node summary statistics by disjunctive cut generator with average optimality gap closed (in %) and processing time (in seconds).

Disjunctive Cut Generator	Gap Closed (%)	Time
Default	62.67	0.90
Param Disj, Param Cuts	63.21	1.53
Param Disj, Calc Cuts	63.26	27.95
Calc Disj, Calc Cuts	63.47	27.87

Table 7 summarizes the relationship between root node processing time and optimality gap closed for the four disjunctive cut generation methods. Column 1 identifies the cut generation method, while Columns 2 and 3 report the average integrality gap closed (percentage) and the time (seconds) required for root node processing, respectively.

As shown in Table 7, the relationship of optimality gap closed and processing time at the root node is nearly Pareto optimal. This is expected, as increased complexity in the disjunctive cut generators allows for more tailored cuts to the instance, but also demands more processing time. The slight exception arises from a negligible decrease in time from Param Disj, Calc Cuts to Calc Disj, Calc Cuts, which is reasonable since the complexity lies almost entirely in solving the same series of PRLPs, whose variance may obscure the minor time increase associated with generating a new disjunction. Moreover, for small perturbations, parametric disjunctive cuts capture the majority of the bound improvement achieved by generating disjunctive cuts from PRLPs, all while requiring only a fraction of the time.

4.4 Branch-and-Cut Performance

In the following paragraphs, we compare test instances based on the time required to reach optimality for Branch-and-Cut and, where relevant, the number of nodes processed. This analysis is conducted from two perspectives: the distribution of runtime changes relative to Default for the other three disjunctive cut generators, and the proportion of test instances where Param Disj, Param Cuts or Default *win*, defined as one significantly outperforming the other.

To ensure consistency, we evaluate performance using test instances solved to optimality within a time frame of 10 seconds to 1 hour, as detailed in Table 9. The lower limit minimizes the effects of machine variability, while the upper limit reflects our computational resource constraints. Disjunctive cut generation time is excluded from the 1-hour limit to focus on their impact on the broader Branch-and-Cut process, though it is included for holistic comparisons.

As shown in Table 9, test set counts generally decrease with higher degrees of perturbation and more terms, as increased perturbation often leads to infeasible instances (which are discarded), and disjunctive cut generation is more likely to exceed its time limit. Notably, the set of test instances

examined in this section differs from the previous one. Many instances that are successfully perturbed and generate root cuts fail to reach optimality within 1 hour, while some that complete Branch-and-Cut within the time limit for one degree of perturbation or term count fail under different conditions. By analyzing each set independently rather than restricting to their intersection, we increase the sample size for each scenario, yielding more robust results. Each combination of degree and terms includes at least 300 test instances solved within 1 hour.

4.4.1 Parameterization Expands Test Cases with Improved Solve Time

One way to measure the effectiveness of disjunctive cuts is by recording the change in time to reach optimality for each test instance when run with Param Disj, Param Cuts, Param Disj, Calc Cuts, and Calc Disj, Calc Cuts relative to Default. This analysis is presented using cumulative distributions, which show the proportion of test instances achieving a specific amount of improvement or degradation. For example, a y-value at -0.25 indicates the proportion of instances solved at least 25% faster with a given disjunctive cut generator compared to Default, while a y-value at +0.25 indicates those solved no more than 25% slower. We compare these cumulative distributions across disjunctive cut generators, the number of terms ($d \in \{4, 16, 64\}$) queued at partial Branch-and-Bound solves where disjunctions are generated, and the degree of perturbation ($\theta \in \{.5, 2\}$).

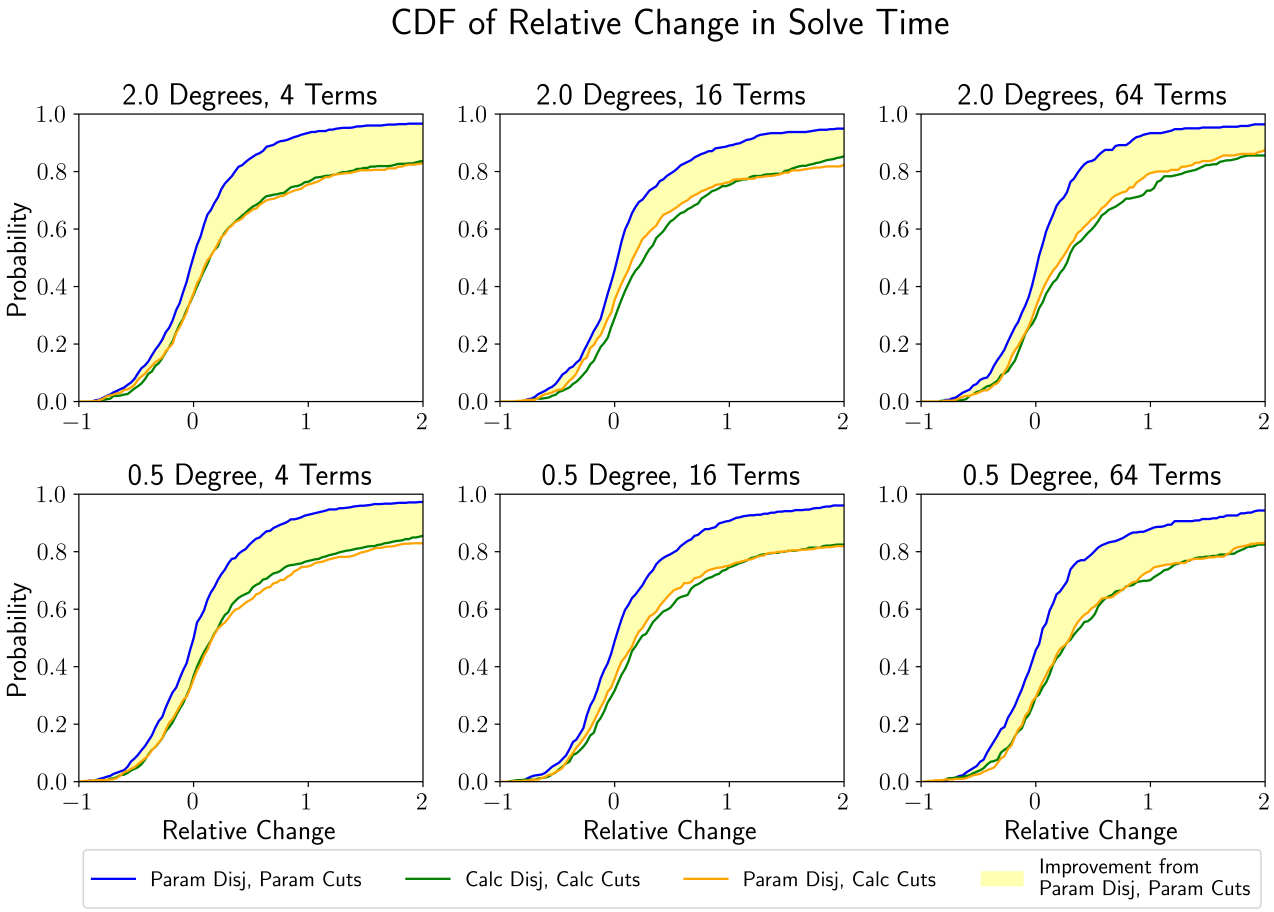


Figure 4: Cumulative distribution of change in run time relative to Default for remaining disjunctive cut generators. Param Disj, Param Cuts advantage over Param Disj, Calc Cuts and Calc Disj, Calc Cuts highlighted in yellow.

In Figure 4, columns and rows correspond to the number of disjunctive terms and the degree of perturbation, respectively. Each plot presents the cumulative distribution of runtime changes for Param Disj, Param Cuts, Param Disj, Calc Cuts, and Calc Disj, Calc Cuts relative to Default.

The yellow-highlighted region shows the additional proportion of test instances where Param Disj, Param Cuts achieves a given improvement or better compared to both Param Disj, Calc Cuts and Calc Disj, Calc Cuts, each relative to Default.

The trends in Figure 4 align with expectations. Param Disj, Param Cuts achieves a comparable improvement in the optimality gap closed at the root node to Param Disj, Calc Cuts and Calc Disj, Calc Cuts but in significantly less time, leading to a higher proportion of instances reaching a given level of improvement relative to Default. A similar pattern emerges between Param Disj, Calc Cuts and Calc Disj, Calc Cuts as the number of disjunctive terms increases: Param Disj, Calc Cuts becomes more efficient while retaining most of the optimality gap closed, as shown in Table 8. Consequently, Param Disj, Calc Cuts begins to outperform Calc Disj, Calc Cuts with more disjunctive terms.

Table 8: Root node summary statistics with average optimality gap closed (in %) and root node processing time (in seconds) for disjunctive cut generators Default (D), Calc Disj, Calc Cuts (CD,CC), Param Disj, Calc Cuts (PD,CC), and Param Disj, Param Cuts (PD,PC) against the test set where Branch-and-Cut reached optimality between 10 seconds and 1 hour.

Degree	Terms	Gap Closed (%)				Time (s)			
		D	CD,CC	PD,CC	PD,PC	D	CD,CC	PD,CC	PD,PC
0.5	4	17.31	20.36	20.47	20.06	1.181	6.706	5.843	1.331
	16	20.37	21.61	22.55	21.41	0.968	10.761	8.350	1.477
	64	20.89	24.26	23.81	22.69	0.587	13.447	9.786	1.505
2.0	4	19.75	21.02	20.51	20.24	1.367	8.000	7.113	1.512
	16	18.71	20.41	19.01	18.94	1.061	13.321	10.514	1.722
	64	19.34	22.95	20.62	19.95	0.681	14.999	10.636	1.756

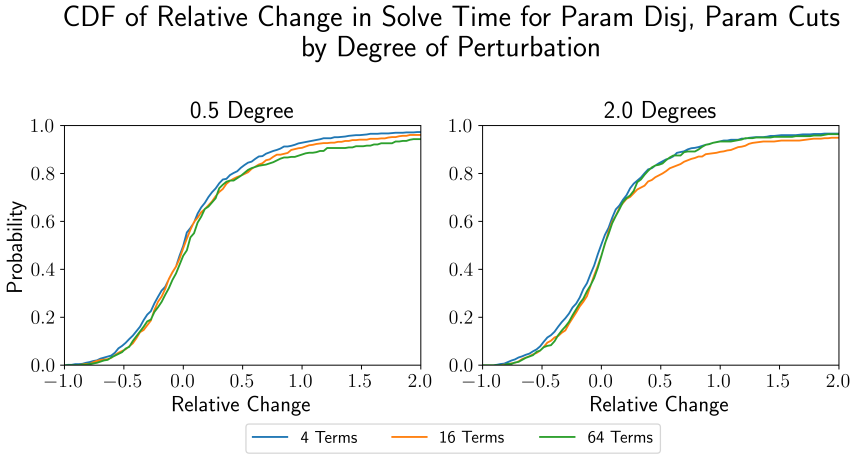


Figure 5: Cumulative distribution of change in run time relative to Default for Param Disj, Param Cuts broken out by degree of perturbation.

In Figure 5, the columns represent the degree of perturbation, and each plot shows the cumulative distribution of runtime changes for Param Disj, Param Cuts relative to Default, broken out by the number of disjunctive terms. As highlighted in the previous section, reducing generation time while retaining improvements in the root optimality gap closed is critical for improving time to reach optimality. Accordingly, we would expect fewer terms to yield better performance, which holds true for a perturbation degree of 0.5. However, this pattern does not persist for a perturbation degree of 2, where the outlier is 16 disjunctive terms. As shown in Table 8, the processing time and optimality gap closed at the root node for 16 terms are worse than for 64 terms, explaining the deviation.

In Figure 6, the columns represent the number of disjunctive terms, and each plot shows the

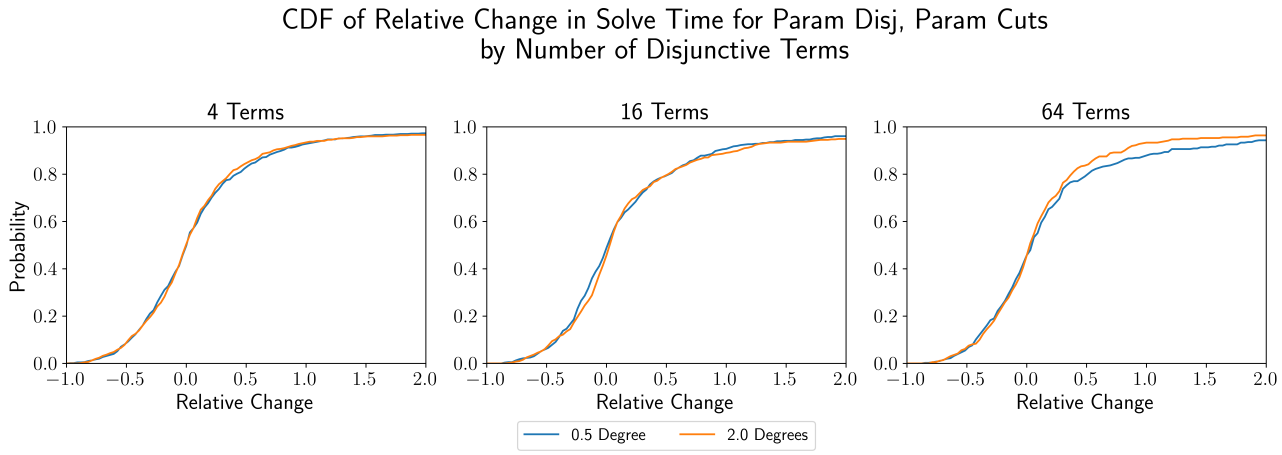


Figure 6: Cumulative distribution of change in run time relative to Default for Param Disj, Param Cuts broken out by terms.

cumulative distribution of runtime changes for Param Disj, Param Cuts relative to Default, broken out by the degree of perturbation. For a given number of terms, we would expect less perturbation to perform better due to the advantage of stronger cuts, especially since the increase in root node processing time for Param Disj, Param Cuts relative to Default is expected to remain consistent across degrees of perturbation. However, as discussed earlier, solve time is highly sensitive to root node processing time. This explains why, for 4 and 16 terms, the better-performing degree of perturbation aligns with whichever achieves faster root node processing, as shown in Table 8.

The case of 64 terms, however, is an exception. Here, the cuts are weaker, and root node processing time is longer, but yet solve time performance is better, breaking both expected trends. While the exact cause remains unclear, this behavior highlights an open opportunity to further explore the complex relationship between cut strength, root node processing time, and overall runtime performance at higher disjunctive term counts.

4.4.2 Parametric Disjunctive Cuts Often Reduce Solve Time and Nodes Processed

Ultimately, we aim to determine when disjunctive cuts should be used and which generator is most effective. Building on the previous section, which identified Param Disj, Param Cuts as the most performant cut generator, we now focus on understanding when it should be used versus leaving disjunctive cuts disabled, as in Default. To measure this, we record the proportion of the test set where a disjunctive cut generator *wins* for a given metric compared to Default. A win for Param Disj, Param Cuts is defined as achieving at least a 10% reduction in the given metric for a test instance relative to Default, and vice versa.

In Table 9, column 1 lists the Degree" of perturbation, $\theta \in \{.5, 2\}$, and column 2 shows the number of Terms", $d \in \{4, 16, 64\}$, queued in a partial Branch-and-Bound solve where a disjunction is generated. Columns 3 and 4 report the percentage of the test set where Param Disj, Param Cuts and Default, respectively, win based on the number of nodes processed. Columns 5 and 6 show the corresponding percentages for solve time, while columns 7 and 8 present results for solve time excluding the time required to generate disjunctive cuts. Finally, columns 9 and 10 provide the counts of base and test instances used in each experiment. All proportions are expressed as percentages.

The trends in Table 9 largely align with expectations. As demonstrated by Balas and Kazachkov [4], Calc Disj, Calc Cuts regularly outperforms Default when considering its best performance across multiple solve repetitions. Since Param Disj, Param Cuts retains much of Calc Disj, Calc Cuts's strength at a fraction of the generation cost, as shown in Table 8, we similarly expect Param Disj, Param Cuts to win consistently, even with a single repetition. Columns 5 and 6 confirm this expectation for solve time, and similar trends appear for both nodes processed and solve time excluding

Table 9: Winning ratios of test instances for Param Disj, Param Cuts (PD,PC) and Default by performance metric and counts of base and test instances. Wins are defined by one disjunctive cut generator recording a value 10% better than the corresponding value for the other disjunctive cut generator and a given metric, which includes nodes processed, solve time, and solve time excluding VPC generation.

Degree	Terms	Nodes Processed		Time		Time w/o VPC Gen.		Base	Test
		PD,PC	Default	PD,PC	Default	PD,PC	Default		
0.5	4	36.30	34.15	37.73	38.31	38.16	37.88	100	697
	16	35.60	34.88	38.46	39.00	40.25	38.10	96	559
	64	37.28	32.59	35.06	41.98	37.78	38.27	74	405
2.0	4	34.47	36.98	37.52	35.19	38.06	34.47	100	557
	16	34.43	35.31	31.80	37.72	35.31	35.31	90	456
	64	35.33	35.03	31.74	39.82	33.83	34.43	74	334

disjunctive cut generation, as these metrics are closely related to overall solve time.

Another clear trend is that, for a fixed metric and number of disjunctive terms, lower perturbation increases the likelihood of Param Disj, Param Cuts winning. This is expected, as less perturbation allows cuts to retain more of their strength while maintaining comparable generation times. Similarly, when holding the degree of perturbation constant and increasing the number of terms, Param Disj, Param Cuts achieves more wins for nodes processed. This aligns with the idea that stronger root cuts can significantly reduce the size of the branch-and-cut tree. Combining these observations, we would also expect that increasing the number of terms while decreasing perturbation would further improve Param Disj, Param Cuts 's performance relative to Default. This pattern is indeed evident for nodes processed.

However, for solve time, the expected increase in wins with more disjunctive terms is less pronounced. While disjunctive cuts strengthen as terms increase, the performance of Param Disj, Param Cuts relative to Default remains highly sensitive to root node cut generation time, which also increases with more terms. This sensitivity helps explain why solve time wins decline as terms increase. When disjunctive cut generation time is excluded, we observe win percentages shifting back in favor of smaller disjunctions for Param Disj, Param Cuts, as expected.

Despite this, the patterns for solve time differ from those for nodes processed, where more terms and less perturbation consistently improve performance. This discrepancy can be explained by the properties of disjunctive cuts: their density often slows down pivot algorithms, increasing processing time per node and, consequently, overall solve time, even when fewer nodes are processed. For a more detailed discussion of this phenomenon, we refer the reader to Balas and Kazachkov [4].

4.4.3 Toggling Parameterization May Unlock Consistent Gains

While Param Disj, Param Cuts often provides significant advantages over Default, the reverse is also true, prompting a desire to combine their strengths. In practice, both approaches could be run in parallel, terminating when the first satisfies a termination condition—similar to how Gurobi simultaneously runs primal simplex, dual simplex, and barrier methods when solving LPs. Another possibility is to predict when parametric disjunctive cuts would be advantageous over no disjunctive cuts and selectively apply Param Disj, Param Cuts in such cases.

To explore this, we further analyzed Table 9 by breaking down results based on runtime length (Table 10) and perturbation type (Table 12) to identify potential patterns. Additionally, we performed a linear regression analysis (Table 14, Table 15) using data generated in this work, augmented with tags from the MIPLIB 2017 metadata file, to investigate whether any linear relationships exist regarding the relative advantage of Param Disj, Param Cuts. Although these analyses revealed no clear trends,

we include them in the appendix for completeness.

5 Conclusion

We contributed both theoretical insights and computational results to advance the understanding of warm-starting MILPs with disjunctive cuts. Theoretically, we established key guarantees, including the validity and tightness of parametric disjunctive cuts relative to the disjunctive hull, separation conditions for a parametric class of disjunctive cuts and basic solutions, and the NP-hardness of warm-starting MILPs in general. Computationally, we demonstrated that warm-starting with parametric disjunctive cuts (Param Disj, Param Cuts) consistently outperforms alternative approaches, including the disjunctive cut generation method proposed by Balas and Kazachkov [4] (Calc Disj, Calc Cuts) and its hybrid warm-starting extension (Param Disj, Calc Cuts). In many cases, Param Disj, Param Cuts also surpassed Gurobi under its default settings (Default), highlighting its practical advantages. These results underscore the significant potential of parametric disjunctive cuts to enhance MILP solvers, offering a promising direction for future research and solver development.

While parametric disjunctive cuts show significant promise, several opportunities exist to further improve their performance. For matrix perturbations, developing a tightening scheme could enhance the effectiveness of generated cuts, as indicated in Figure 10 and Table 4. Further, decisions such as placing all feasible disjunctive terms in the PRLP, assigning Farkas coefficients of 0 to infeasible terms, and disabling cut generation during disjunction creation merit deeper investigation.

For example, removing pruned yet feasible terms from the PRLP could reduce computation time by shrinking the problem size, but this may weaken the resulting cuts. Similarly, using 0 as a placeholder Farkas coefficient for infeasible terms—while valid—may not be optimal, particularly as perturbations increase the likelihood of those terms becoming feasible. Exploring alternative methods for generating Farkas certificates in such cases is a promising direction for further study.

Additionally, while parametric disjunctive cuts recover a significant portion of the dual bound refined by their generating disjunction (Table 4), their impact on additional root node gap closure remains muted (Table 5). This may occur because Balas and Kazachkov [4] disable cut generation in Calc Disj, causing disjunctive cuts and Gurobi's default cuts to refine similar regions of the LP relaxation. Enabling cut generation during disjunction creation might push these cuts to target different parts of the LP relaxation, increasing their combined effectiveness.

Another key question is determining when parametric disjunctive cuts should be applied. Although our linear regression analysis revealed no clear relationships between problem features and the relative advantage of parametric disjunctive cuts, nonlinear patterns may exist. Training nonlinear models, such as neural networks, could uncover these relationships.

Further computational experiments are also needed to compare parametric disjunctive cuts against the warm-starting techniques proposed by Ralphs and Güzelsoy [25]. Finally, given the importance of restarts in satisfiability problems, it would be worth exploring how parametric disjunctive cuts could support restart strategies in the context of MILP solvers.

Overall, this work establishes a strong foundation for warm-starting MILPs with parametric disjunctive cuts, both theoretically and computationally. By demonstrating their effectiveness across a range of problem settings, we highlight their potential to become a key tool in MILP solvers. However, realizing their full impact requires further refinement, particularly in improving cut generation strategies, identifying when they offer the greatest advantage, and integrating them with existing solver techniques. Future research that addresses these challenges could unlock even greater efficiency gains, ultimately advancing the state of MILP solving and expanding the practical applications of warm-starting methods.

References

[1] V-polyhedral Disjunctive Cuts. https://github.com/spkelle2/vpc.

[2] Kyri Baker. Learning warm-start points for ac optimal power flow. In *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2019. doi: 10.1109/MLSP.2019.8918690.

[3] Egon Balas. Disjunctive programming. *Ann. Discrete Math.*, 5:3–51, 1979. URL https://www.sciencedirect.com/science/article/pii/S016750600870342X.

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

[5] Egon Balas and Aleksandr M. Kazachkov. Monoidal strengthening of $\mathcal{V}$-polyhedral disjunctive cuts, 2023. URL https://optimization-online.org/2023/02/monoidal-strengthening-of-simple-v-polyhedral-disjunctive-cuts/.

[6] Cynthia Barnhart, Ellis L. Johnson, George L. Nemhauser, Martin W. P. Savelsbergh, and Pamela H. Vance. Branch-and-price: column generation for solving huge integer programs. *Oper. Res.*, 46(3):316–329, 1998. ISSN 0030-364X,1526-5463. doi: 10.1287/opre.46.3.316. URL https://doi.org/10.1287/opre.46.3.316.

[7] John E. Beasley. *Lagrangian relaxation*, page 243–303. John Wiley & Sons, Inc., USA, 1993. ISBN 0470220791.

[8] DE Bell. A cutting plane algorithm for integer programs with an easy proof of convergence. 1973.

[9] Claus C. Carø e and Rüdiger Schultz. Dual decomposition in stochastic integer programming. *Oper. Res. Lett.*, 24(1-2):37–45, 1999. ISSN 0167-6377,1872-7468. doi: 10.1016/S0167-6377(98)00050-9. URL https://doi.org/10.1016/S0167-6377(98)00050-9.

[10] Marco A. Duran and Ignacio E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Math. Programming*, 36(3):307–339, 1986. ISSN 0025-5610,1436-4646. doi: 10.1007/BF02592064. URL https://doi.org/10.1007/BF02592064.

[11] Julius Farkas. Theorie der einfachen Ungleichungen. *J. Reine Angew. Math.*, 124:1–27, 1902. URL https://doi.org/10.1515/crll.1902.124.1.

[12] 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.

[13] M.V. Galati and Ted K. Ralphs. Decomposition in integer programming. Technical Report 04T-019, Lehigh University Industrial and Systems Engineering Department, 2005.

[14] Gerald Gamrath, Benjamin Hiller, and Jakob Witzig. Reoptimization techniques for MIP solvers. In Evripidis Bampis, editor, *Experimental Algorithms - 14th International Symposium, SEA 2015, Paris, France, June 29 - July 1, 2015, Proceedings*, volume 9125 of *Lecture Notes in Computer Science*, pages 181–192. Springer, 2015. doi: 10.1007/978-3-319-20086-6_14. URL https://doi.org/10.1007/978-3-319-20086-6_14.

[15] Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 15554–15566, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/d14c2267d848abeb81fd590f371d39bd-Abstract.html.

[16] M. Güzelsoy. *Dual Methods in Mixed Integer Linear Programming*. PhD, Lehigh University, 2009. URL http://coral.ie.lehigh.edu/~ted/files/papers/MenalGuzelsoyDissertation09.pdf.

[17] Yacov Y. Haimes, Leon S. Lasdon, and David A. Wismer. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Trans. Systems Man Cybernet.*, SMC-1:296–297, 1971. ISSN 0018-9472.

[18] A. Hassanzadeh and T.K. Ralphs. A Generalization of Benders' Algorithm for Two-Stage Stochastic Optimization Problems with Mixed Integer Recourse. Technical Report 14T-005, COR@L Laboratory, Lehigh University, 2014. URL http://coral.ie.lehigh.edu/~ted/files/papers/SMILPGenBenders14.pdf.

[19] Andrea Lodi. *The Heuristic (Dark) Side of MIP Solvers*, pages 273–284. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-30671-6. doi: 10.1007/978-3-642-30671-6_10. URL https://doi.org/10.1007/978-3-642-30671-6_10.

[20] Sidhant Misra, Line Roald, and Yeesian Ng. Learning for constrained optimization: Identifying optimal active constraint sets. *INFORMS J. Comput.*, 34(1):463–480, 2022. doi: 10.1287/ijoc.2020.1037. URL https://doi.org/10.1287/ijoc.2020.1037.

[21] Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, Ravichandra Addanki, Tharindi Hapuarachchi, Thomas Keck, James Keeling, Pushmeet Kohli, Ira Ktena, Yujia Li, Oriol Vinyals, and Yori Zwols. Solving mixed integer programs using neural networks. *CoRR*, abs/2012.13349, 2020. URL https://arxiv.org/abs/2012.13349.

[22] MohammadReza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takac. Reinforcement learning for solving the vehicle routing problem. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/9fb4651c05b2ed70fba5afe0b039a550-Paper.pdf.

[23] Krunal Patel. Progressively strengthening and tuning mip solvers for reoptimization, 2023. URL https://arxiv.org/pdf/2308.08986.pdf.

[24] Michael Perregaard and Egon Balas. Generating cuts from multiple-term disjunctions. In *Integer programming and combinatorial optimization (Utrecht, 2001)*, volume 2081 of *Lecture Notes in Comput. Sci.*, pages 348–360. Springer, Berlin, 2001. ISBN 3-540-42225-0. doi: 10.1007/3-540-45535-3_27. URL https://doi.org/10.1007/3-540-45535-3_27.

[25] T.K. Ralphs and M. Güzelsoy. Duality and Warm Starting in Integer Programming. In *The Proceedings of the 2006 NSF Design, Service, and Manufacturing Grantees and Research Conference*, 2006. URL http://coral.ie.lehigh.edu/~ted/files/papers/DMII06.pdf.

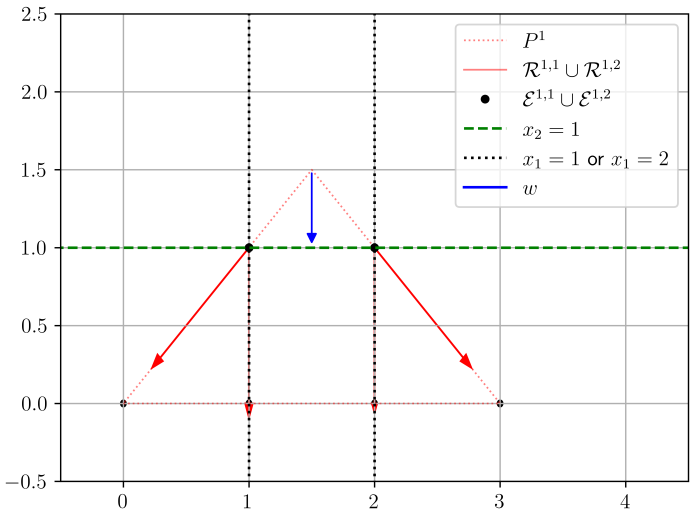
[26] R. Raman and I.E. Grossmann. Modelling and computational techniques for logic based integer programming. *Computers & Chemical Engineering*, 18(7):563–578, 1994. ISSN 0098-1354. doi: https://doi.org/10.1016/0098-1354(93)E0010-7. URL https://www.sciencedirect.com/science/article/pii/0098135493E00107. An International Journal of Computer Applications in Chemical Engineering.

[27] Sahar Tahernejad, Ted K. Ralphs, and Scott T. DeNegre. A branch-and-cut algorithm for mixed integer bilevel linear optimization problems and its implementation. *Math. Program. Comput.*, 12(4):529–568, 2020. ISSN 1867-2949,1867-2957. doi: 10.1007/s12532-020-00183-6. URL https://doi.org/10.1007/s12532-020-00183-6.

[28] Álinson S. Xavier, Feng Qiu, and Shabbir Ahmed. Learning to solve large-scale security-constrained unit commitment problems. *INFORMS J. Comput.*, 33(2):739–756, 2021. ISSN 1091-9856,1526-5528. doi: 10.1287/ijoc.2020.0976. URL https://doi.org/10.1287/ijoc.2020.0976.

[29] L. Zadeh. Optimality and non-scalar-valued performance criteria. *IEEE Transactions on Automatic Control*, 8(1):59–60, 1963. doi: 10.1109/TAC.1963.1105511.

A Examples



$$\begin{aligned} \min_{x \in \mathbb{R}^2} \quad & -x_2 \\ x_1 \quad & -x_2 \geq 0 \\ -x_1 \quad & -x_2 \geq -3 \\ x_1 \quad & \geq 0 \\ & x_2 \geq 0 \\ x_1, \quad & x_2 \in \mathbb{Z} \end{aligned} \tag{IP-1}$$

Figure 7: Solving PRLP-1$(\{\mathcal{X}^1, \mathcal{X}^2\}, w)$ with $w = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$, $\mathcal{X}^1 = \{x \in \mathbb{R}^2 : -x_1 \geq -1\}$, and $\mathcal{X}^2 = \{x \in \mathbb{R}^2 : x_1 \geq 2\}\}$ yields the VPC $x_2 \leq 1$, a valid inequality for IP-1.

Example A.1. *This example illustrates how to generate a VPC by solving a PRLP given IP-1, the disjunction* $\{\mathcal{X}^1, \mathcal{X}^2\}$*, and objective direction* w *as defined in Figure 7. Let* $B^1 = [1, 2, 4, 5, 6]$ *and* $N^1 = [3, 7]$ *be the optimal basic and nonbasic columns of LP-*$1, 1$ *, respectively, and let* $B^2 = [1, 2, 3, 5, 6]$ *and* $N^2 = [4, 7]$ *be the optimal basic and nonbasic columns of LP-*$1, 2$ *, respectively. Thus, we have*

$$\left(\tilde{A}^{1,1}_{\cdot B^1}\right)^{-1} = \begin{bmatrix} 0 & 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 & -1 \\ 1 & -1 & 0 & 0 & 2 \\ 0 & 0 & -1 & 0 & -1 \\ -1 & 0 & 0 & -1 & -1 \end{bmatrix}, \; \tilde{A}^{1,1}_{\cdot N^1} = \begin{bmatrix} -1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & -1 \end{bmatrix}, \; b^{1,1} = \begin{bmatrix} 0 \\ -3 \\ 0 \\ 0 \\ -1 \end{bmatrix}$$

$$\left(\tilde{A}^{1,2}_{\cdot B^2}\right)^{-1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & -1 \\ -1 & 1 & 0 & 0 & 2 \\ 0 & 0 & -1 & 0 & 1 \\ 0 & -1 & 0 & -1 & -1 \end{bmatrix}, \; \tilde{A}^{1,2}_{\cdot N^2} = \begin{bmatrix} 0 & 0 \\ -1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & -1 \end{bmatrix}, \text{ and } b^{1,2} = \begin{bmatrix} 0 \\ -3 \\ 0 \\ 0 \\ 2 \end{bmatrix}.$$

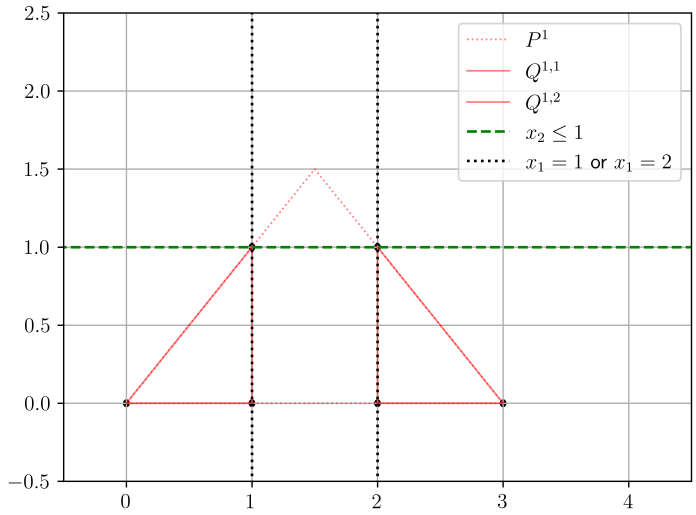
For $t \in T$*, basis cone* $\mathcal{C}^{kt}_{\mathcal{N}^t} = \bar{x}^{kt} + \text{cone}\left\{r^{k,t,h} : r^{k,t,h} = \text{proj}_x\left(\left(-\tilde{A}^{kt}_{\cdot B^t}\right)^{-1} \tilde{A}^{kt}_{\cdot j}\right), j = N^t_h\}\right\}$*, where*

$$\bar{x}^{11} = \text{proj}_x\left(\left(\tilde{A}^{1,1}_{\cdot B^1}\right)^{-1} b^{1,1}\right) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \; \bar{x}^{12} = \text{proj}_x\left(\left(\tilde{A}^{1,2}_{\cdot B^2}\right)^{-1} b^{1,2}\right) = \begin{bmatrix} 2 \\ 1 \end{bmatrix},$$

$$r^{1,1,1} = -\text{proj}_x\left(\left(\tilde{A}^{1,1}_{\cdot B^1}\right)^{-1} \tilde{A}^{1,1}_{\cdot N^1_1}\right) = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \; r^{1,1,2} = -\text{proj}_x\left(\left(\tilde{A}^{1,1}_{\cdot B^1}\right)^{-1} \tilde{A}^{1,1}_{\cdot N^1_2}\right) = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

$$r^{1,2,1} = -\text{proj}_x\left(\left(\tilde{A}^{1,2}_{\cdot B^2}\right)^{-1} \tilde{A}^{1,2}_{\cdot N^2_1}\right) = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \text{ and } r^{1,2,2} = -\text{proj}_x\left(\left(\tilde{A}^{1,2}_{\cdot B^2}\right)^{-1} \tilde{A}^{1,2}_{\cdot N^2_2}\right) = \begin{bmatrix} 1 \\ -1 \end{bmatrix},$$

yielding the translated cones pictured in Figure 7. Therefore, PRLP-$1(\mathcal{X}, w)$ *is formulated with* $\mathcal{E}^k = \left\{\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}\right\}$ *and* $\mathcal{R}^k = \left\{\begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}\right\}$*, which has optimal solution* $(\alpha, \beta) = ((0, -1), -1)$*, implying* $-x_2 \geq -1$*, or* $x_2 \leq 1$ *as written above.*



$$\begin{array}{lrl} \min\limits_{x\in\mathbb{R}^2} & -x_2 & \\ & x_1 \quad -x_2 & \geq 0 \\ & -x_1 \quad -x_2 & \geq -3 \\ & x_1 \qquad & \geq 0 \\ & \qquad x_2 & \geq 0 \\ & x_1, \quad x_2 & \in \mathbb{Z} \end{array} \tag{IP-1}$$

Figure 8: The LP relaxation of IP-1, $\mathcal{P}^1$, overlaid with the simple VPC $-x_2 \geq -1$ and LP relaxations $\mathcal{Q}^{1,1}$ and $\mathcal{Q}^{1,2}$, which arise from the application of disjunction $\{\mathcal{X}^1, \mathcal{X}^2\}$ where $\mathcal{X}^1 := \{x \in \mathbb{R}^2 : -x_1 \geq -1\}$ and $\mathcal{X}^2 := \{x \in \mathbb{R}^2 : x_1 \geq 2\}$.

Example A.2. *This example illustrates how to calculate a Farkas certificate $\{v^1, v^2\}$ given IP-1, the disjunction $\{\mathcal{X}^1, \mathcal{X}^2\}$, and the simple vpc $-x_2 \geq -1$ as defined in Figure 8. Let $N^{1,1} := [1, 5]$ and $N^{1,2} := [2, 5]$ be the optimal nonbasis elements for LP-1, 1 and LP-1, 2 , respectively. Let*

$$C^{1,1} := A^{1,1}_{N^{1,1}\cdot} = \begin{bmatrix} 1 & -1 \\ -1 & 0 \end{bmatrix}, \qquad C^{1,1}_0 := b^{1,1}_{N^{1,1}} = \begin{bmatrix} 0 \\ -1 \end{bmatrix},$$
$$C^{1,2} := A^{1,2}_{N^{1,2}\cdot} = \begin{bmatrix} -1 & -1 \\ 1 & 0 \end{bmatrix}, \text{ and } C^{1,2}_0 := b^{1,2}_{N^{1,2}} = \begin{bmatrix} -3 \\ 2 \end{bmatrix}.$$

Then by Lemma 2.3, we have that

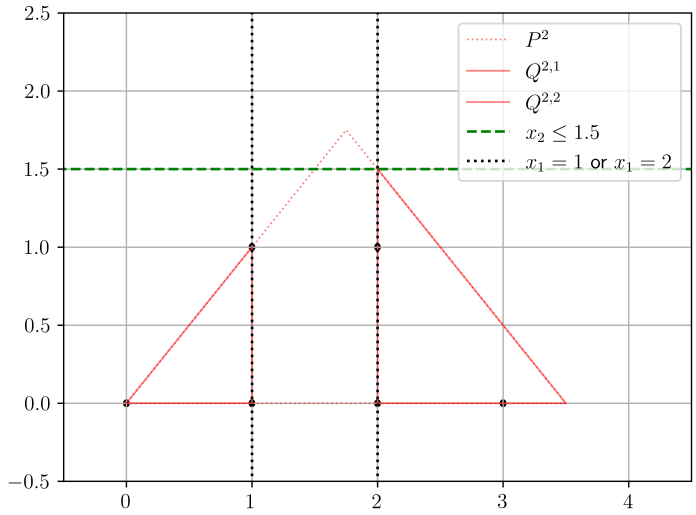
$$v^1_{N^{1,1}} = \alpha^\intercal (C^{1,1})^{-1} = \begin{bmatrix} 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & -1 \\ -1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \end{bmatrix} \text{ and}$$
$$v^2_{N^{1,2}} = \alpha^\intercal (C^{1,2})^{-1} = \begin{bmatrix} 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \end{bmatrix},$$

which means $v^1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \end{bmatrix}$ and $v^2 = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \end{bmatrix}$. FL holds when inputting the certificate $\{v^1, v^2\}$ and the vpc $-x_2 \geq -1$

$$v^1 A^{1,1} x \geq v^1 b^{1,1} \implies \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & -1 \\ 1 & 0 \\ 0 & 1 \\ -1 & 0 \end{bmatrix} x \geq \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ -3 \\ 0 \\ 0 \\ -1 \end{bmatrix} \implies \begin{bmatrix} 0 & -1 \end{bmatrix} x \geq -1$$

$$v^2 A^{1,2} x \geq v^2 b^{1,2} \implies \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & -1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} x \geq \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ -3 \\ 0 \\ 0 \\ 2 \end{bmatrix} \implies \begin{bmatrix} 0 & -1 \end{bmatrix} x \geq -1$$

Thus, the certificate $\{v^1, v^2\}$ is correct.



$$\begin{array}{lrl} \min\limits_{x \in \mathbb{R}^2} & -x_2 & \\ & x_1 \quad -x_2 & \geq 0 \\ & -x_1 \quad -x_2 & \geq -3.5 \\ & x_1 \qquad & \geq 0 \\ & \qquad x_2 & \geq 0 \\ & x_1, \quad x_2 & \in \mathbb{Z} \end{array} \tag{IP-2}$$

Figure 9: The LP relaxation of IP-2, $\mathcal{P}^2$, overlaid with the LP relaxations $\mathcal{Q}^{2,1}$ and $\mathcal{Q}^{2,2}$ arising from the application of disjunction $\{\mathcal{X}^1, \mathcal{X}^2\}$. Also, plotted is $-x_2 \geq -1.5$, the parameterization of the simple VPC $-x_2 \geq -1$ from Example A.2.

Example A.3. *Applying the disjunction $\{\mathcal{X}^1, \mathcal{X}^2\}$ and the Farkas certificate $\{v^1, v^2\}$ from Example A.2 to Theorem 3.1 and IP-2 yields a valid parameterized disjunctive cut for IP-2. This cut is calculated as follows:*

$$v^1 A^{2,1} x \geq v^1 b^{2,1} \implies \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & -1 \\ 1 & 0 \\ 0 & 1 \\ -1 & 0 \end{bmatrix} x \geq \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ -3.5 \\ 0 \\ 0 \\ -1 \end{bmatrix} \implies \begin{bmatrix} 0 & -1 \end{bmatrix} x \geq -1$$

$$v^2 A^{2,2} x \geq v^2 b^{2,2} \implies \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & -1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} x \geq \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ -3.5 \\ 0 \\ 0 \\ 2 \end{bmatrix} \implies \begin{bmatrix} 0 & -1 \end{bmatrix} x \geq -1.5$$

$$\alpha = \begin{bmatrix} \max\{v^1 A^{2,1}_{\cdot 1}, v^2 A^{2,2}_{\cdot 1}\} \\ \max\{v^1 A^{2,1}_{\cdot 2}, v^2 A^{2,2}_{\cdot 2}\} \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \text{ and } \beta = \min\{v^1 b^{2,1}, v^2 b^{2,2}\} = \min\{-1, -1.5\} = -1.5$$

This yields the cut $-x_2 \geq -1.5$, which is plotted in Figure 9.

B Algorithms

C Evaluation

As mentioned in **??**, cuts are evaluated in two different ways, their their effect on the root node and their effect on branch and bound. At the root node, we measure the strength of the cuts by the percent integrality (root) gap closed by one round of VPCs or their generating disjunction. Let x_I denote an optimal solution to a MILP, let $\bar{x}$ denote an optimal solution to a MILP's LP relaxation, and let x_0 be an optimal solution to the MILP's LP relaxation after a set of cuts have been added or disjunction

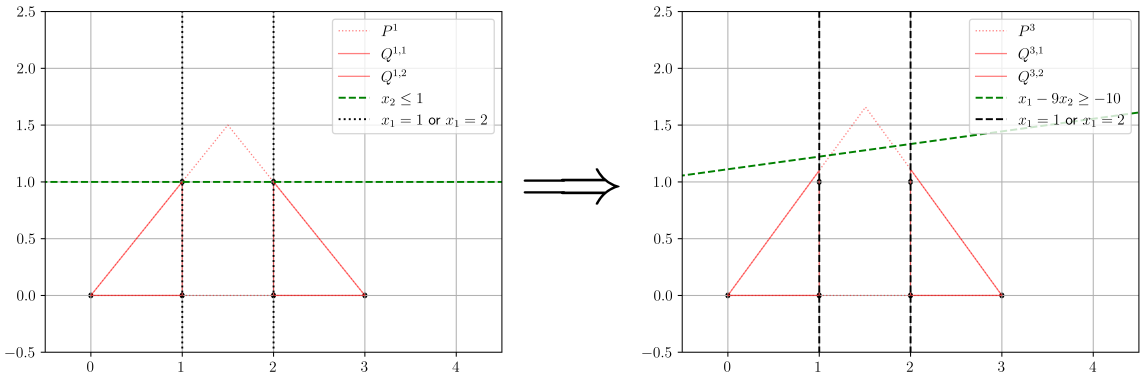


Figure 10: Consider the VPC $x_2 \leq 1$ from the left instance. Using Theorem 3.1 with its Farkas certificate on the right instance (a matrix perturbation of the left) produces a cut that is not tight for the right's disjunctive hull.

Algorithm 8 generateValidParameterizedDisjunctiveCut

Require: IP-k,$\mathcal{X}, v, T$
Ensure: $\{\mathcal{X}^t\}_{t\in T}$ is a disjunction and $\{v^t\}_{t\in T} \geq 0$
1: Let $\alpha \in \mathbb{R}^n$, $\beta \in \mathbb{R}$, and $\gamma^t \in \mathbb{R}^n$ for all $t \in T$
2: **for all** $t \in T$ **do**
3: $\quad \gamma^t \leftarrow v^t A^{kt}$
4: **end for**
5: **for** $j \leftarrow 1$ to n **do**
6: $\quad \alpha_j \leftarrow \max_{t\in T}\{\gamma_j^t\}$
7: **end for**
8: $\beta \leftarrow \min_{t\in T}\{v^t b^{kt}\}$
9: **return** (α, β)

applied. We define measure the quantity

$$\% \text{ integrality gap closed} := 100 \times \frac{c^T x_0 - c^T \bar{x}}{c^T x_I - c^T \bar{x}}$$

D Additional Tables and Figures

Table 10: Winning ratios of test instances for Param Disj, Param Cuts (PD,PC) and Default by solve time categories. Wins are defined by one disjunctive cut generator solving a test instance, excluding VPC generation, to optimality in 10% less time than the other disjunctive cut generator. Short includes solves less than 60 seconds, Medium from 60 up to 600 seconds, and Long from 600 to 3600 seconds.

Degree	Terms	Short		Medium		Long	
		PD,PC	Default	PD,PC	Default	PD,PC	Default
0.5	4	38.25	37.54	37.36	39.19	39.57	35.97
	16	36.62	37.09	45.06	38.63	37.17	38.94
	64	38.41	37.09	39.39	35.76	33.71	44.94
2.0	4	36.41	39.32	39.23	32.31	38.46	29.67
	16	34.57	42.59	36.19	31.90	34.52	29.76
	64	36.21	36.21	32.70	37.11	32.20	23.73

Table 11: Counts of base and test instances by solve time categories. Short includes solves of Default less than 60 seconds, Medium from 60 up to 600 seconds, and Long from 600 to 3600 seconds.

Degree	Terms	Short		Medium		Long	
		Base	Test	Base	Test	Base	Test
0.5	4	63	285	64	273	35	139
	16	61	213	55	233	29	113
	64	48	151	41	165	25	89
2.0	4	64	206	60	260	28	91
	16	57	162	52	210	24	84
	64	45	116	41	159	19	59

Table 12: Winning ratios of test instances for Param Disj, Param Cuts (PD,PC) and Default by type of perturbation. Wins are defined by one disjunctive cut generator solving a test instance, excluding VPC generation, to optimality in 10% less time than the other disjunctive cut generator.

Degree	Terms	Matrix		Objective		RHS	
		PD,PC	Default	PD,PC	Default	PD,PC	Default
0.5	4	41.45	35.23	36.76	39.95	37.50	34.38
	16	38.46	36.36	39.76	39.45	44.94	35.96
	64	40.91	38.18	35.48	38.25	39.74	38.46
2.0	4	31.93	34.45	38.74	35.71	44.59	28.38
	16	34.41	32.26	36.51	35.53	30.51	38.98
	64	42.86	25.40	33.48	36.61	23.40	36.17

Table 13: Counts of base and test instances by perturbation.

Degree	Terms	Matrix		Objective		RHS	
		Base	Test	Base	Test	Base	Test
0.5	4	62	193	86	408	38	96
	16	53	143	78	327	38	89
	64	44	110	64	217	32	78
2.0	4	44	119	88	364	32	74
	16	37	93	79	304	26	59
	64	29	63	63	224	24	47

Table 14: Linear Regression Model Metadata

Metric	**Value**
Mean Absolute Error (MAE)	0.8929
Mean Squared Error (MSE)	13.1607
R^2 Score	0.0504
Intercept	0.4831

Table 15: Linear Regression Coefficients (Sorted by Absolute Value)

Feature	**Coefficient**
Root Optimality Gap Improvement	4.2107
indicator	2.1385
numerics	-1.8152
equation_knapsack	1.7023
integer_knapsack	1.4748
bound	-1.4389
mixed_binary	-1.2352
set_packing	-1.1955
cardinality	0.7873
binary	0.6683
objective	0.5729
decomposition	-0.5401
general_linear	-0.5371
matrix	0.5106
binpacking	0.4801
density	-0.4759
aggregations	0.4171
precedence	-0.4094
infeasibleTermsRatio Farkas	0.4019
knapsack	-0.3744
rhs	0.3554
set_covering	-0.3137
variable_bound	0.2485
hard	0.1539
easy	-0.1539
benchmark	-0.0871
benchmark_suitable	-0.0871
degree	-0.0424
set_partitioning	0.0402
invariant_knapsack	-0.0253
actualTerms Farkas	-0.0110
terms	0.0097
numCuts Farkas	0.0028
integers	-0.0001
variables	-0.0001
continuous	0.0000
binaries	-0.0000
constraints	-0.0000
nonz.	0.0000
feasibility	0.0000
no_solution	-0.0000
infeasible	-0.0000
open	-0.0000
infeasibleToFeasibleTermsRatio Farkas	0.0000
zeroInfeasibleToFeasibleTerms Farkas	0.0000
feasibleToInfeasibleTermsRatio Farkas	0.0000