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

Sean Kelley[1](✉), Aleksandr M. Kazachkov[2][0000−0002−4949−9565], and Ted Ralphs[1][0000−0002−4306−9089]

[1] Lehigh University, Bethlehem, PA, USA
sek519@lehigh.edu, ted@lehigh.edu
[2] University of Florida, Gainesville, FL, USA
akazachkov@ufl.edu

Abstract. Many applications require solving a series of closely related mixed integer linear optimization problems that all share the same variables and number of constraints. In such cases, warm-starting is an obvious avenue for improving overall solution time. In this paper, we explore the use of valid disjunctions generated from the branch-and-bound trees arising during previous solves, along with so-called Farkas certificates that prove the validity of associated disjunctive cuts, to warm start the solution of instances later in the sequence. The combination of a disjunction and a Farkas certificate yields a parameteric class of disjunctive cuts that can be inexpensively applied to any instance in the sequence. We implemented this idea and performed computation experiments to test its potential.

Keywords: mixed integer linear programming · disjunctive cuts · warm start · branch and bound

1 Introduction

We investigate the effect of reusing cutting planes generated from a mixed integer linear optimization problem (MILP) to warm start the solution of a sequence of related MILPs, obtained from the "base" instance by slightly perturbing the data of the objective, constraints, or variable bounds. This context is common in industrial applications, including energy [27] and transportation [22], in which minor modifications differentiate the optimization problem solved each day or it is desirable to explore alternative scenarios in view of uncertain input data.

Solution methods for MILPs generate a wide variety of information that is typically discarded once the solution process is completed, despite potential benefits when solving similar instances [13, 23], through retaining branching priorities of variables [21, 14] or predicting properties of an optimal solution [2, 20]. While the heuristic side of optimization solvers is crucial for performance [18], it would be ideal to also compute a "proof" of optimality for an instance that

translates to related MILPs. We propose to retain portions of *certificates* for the optimality of a MILP solution — via disjunctions arising from (partial) branch-and-bound trees — and for the validity of inequalities — with so-called Farkas (dual) multipliers associated to constraints.

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 MILP-feasible solutions. As it may not be straightforward to ensure that an inequality 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 cutting plane information from solving a MILP is empirically effective to accelerate the solution of a related MILP.

Related Work. Our work touches on a number of different areas of the literature on solution methods for MILP. We review here only the most closely related work. We primarily build on earlier work that develops a methodology for reusing branch-and-bound trees to solve a new instance by continuing in the same tree as a previous instance (after modifying the relaxations in the leaf nodes) [26, 15]. This approach is fully implemented in the SYMPHONY MILP solver [25] and is the basis, for example, of a generalized Benders algorithm for solving two-stage stochastic programs [16]. The main weakness of this method is that it is difficult to predict a priori when the previous disjunction will be effective for the new instance; when it turns out not to be the case, the size of the resulting tree can explode with very little recourse. In the present work, we hope that by starting the computation of the new instance with only valid inequalities from prior runs, which can ultimately be discarded if needed, we maintain more flexibility in the warm start and are less likely to see large regressions.

Research into effectively generating disjunctive cuts has a long history, with the first significant computational progress being lift-and-project cuts [6, 7] from a variable (two-term) split disjunction. Even for this simple special case, avoiding the higher-dimensional original formulation of these cuts is crucial for a practical implementation [5, 8], making it computationally prohibitive to realize otherwise theoretically-attractive methods for deriving disjunctive cuts from branch-and-bound trees [9, 10]. Perregaard and Balas [24] propose an alternative without

a lifted space, based on reverse polars, an idea the recurs through the literature [19]. The idea is recently further developed by Balas and Kazachkov [4], who avoid an expensive row generation step present in earlier papers and offer an efficient implementation of a procedure to calculate so-called *$\mathcal{V}$-polyhedral disjunctive cuts* (VPCs), which we use in this paper. Kazachkov and Balas [17] show how to compute the Farkas certificate of validity for VPCs.

Contributions. After introducing relevant notation and background in Section 2, we show in Section 3 how to obtain a Farkas certificate to parameterize VPCs. Section 4 contains details of our implementation and setup of computational experiments as well as a presentation of the results of parameterizing VPCs, showing that reusing this information improves the performance of solving a series of MILPs. Section 5 closes with a discussion of future research directions.

2 Notation and Preliminaries

Our goal is to generate parametric valid inequalities to tighten the (natural) linear programming (LP) relaxation of each MILP in a sequence of K 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

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

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 any variable bounds are specified among these constraints and that all variables have nonnegative lower bounds. For any positive integer n, let $[n] := \{1,\dots,n\}$. The integer-restricted variables are denoted by $I \subseteq [n]$. We also let $\mathcal{P}^k := \{x \in \mathbb{R}^n : A^k x \geq b^k\}$ be the feasible region of the LP relaxation of (IP-k) and $\mathcal{S}^k := \{x \in \mathcal{P}^k : x_j \in \mathbb{Z} \text{ for } j \in I\}$ be the feasible region of (IP-k). For matrix A, we represent its ith row as "$A_{i\cdot}$" and its jth column as "$A_{\cdot j}$".

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 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. *A* valid disjunction *for set $\mathcal{S} \subseteq \mathbb{R}^n$ is a collection $\{\mathcal{X}^t\}_{t\in[T]}$, 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 $k \in [K]$. Specifically, our disjunctions are valid for all MILP because they arise from partial branch-and-bound trees

constructed by branching on integer variables: each node is defined by modifying the lower or upper bound on an integer variable with respect to its parent. The LP relaxations of the leaf nodes from the tree constitute a disjunction that is valid even if the instance changes, if the integer variables remain the same; hence, we denote disjunctions independently of the index k.

Let $\{\mathcal{X}^t\}_{t\in[T]}$ be such a disjunction, where $\mathcal{X}^t := \{x \in \mathbb{R}^n : D^t x \geq D_0^t\}$ defined by q_t constraints with $D^t \in \mathbb{Q}^{q_t \times n}$ and $D_0^t \in \mathbb{Q}^{q_t}$. We also define

$$A^{kt} := \begin{bmatrix} A^k \\ D^t \end{bmatrix}, \quad b^{kt} := \begin{bmatrix} b^k \\ D_0^t \end{bmatrix}, \quad \text{and} \quad \mathcal{Q}^{kt} := \mathcal{P}^k \cap \mathcal{X}^t = \{x \in \mathbb{R}^n : A^{kt} x \geq b^{kt}\}.$$

The cuts we obtain will be valid for the *disjunctive hull* $\mathcal{P}_D^k := \operatorname{cl}\operatorname{conv}(\cup_{t\in[T]} \mathcal{Q}^{kt})$.

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.

Cut-Generating LP. The well-studied *cut-generating LP* (CGLP) is one way to generate inequalities valid for the disjunctive hull $\mathcal{P}_D^k$ for instance (IP-k), $k \in [K]$, 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 $\mathcal{P}_D^k$ 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 $\mathcal{P}_D^k$ 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 is a row vector of length $q + q_t$:

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

To generate cuts with (1), 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 a fixed $\bar{\beta} \in \mathbb{R}$ (often $\bar{\beta}$ is taken to be 1). As such, we can replace the region defined by (1) with

$$\left\{(\alpha, \{v^t\}_{t\in[T]}) : (\alpha, \bar{\beta}, \{v^t\}_{t\in[T]}) \text{ is feasible to (1)}\right\}. \tag{CGLP($\bar{\beta}$)}$$

Point-Ray LP. Formulating the CGLP is too expensive in practice even for two-term disjunctions based on a single variable split. The VPC framework targets cuts from much larger disjunctions, and hence Balas and Kazachkov [4] suggest two modifications to the cut-generation procedure: avoiding solving an LP

with more than n variables, and gaining efficiency 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 $(\alpha, \bar{\beta})$ is valid for $\mathcal{P}_D^k$ if and only if α is feasible to the following *point-ray LP* (PRLP):

$$\begin{aligned} \alpha^\intercal p &\geq \bar{\beta} \qquad \text{for all } p \in \mathcal{E}^k \\ \alpha^\intercal r &\geq 0 \qquad \text{for all } r \in \mathcal{R}^k. \end{aligned} \tag{PRLP($\bar{\beta}$)}$$

Cuts obtained via (PRLP($\bar{\beta}$)) are referred to as VPCs [4].

Relaxation-Based Generation. The advantage of (PRLP($\bar{\beta}$)) over (CGLP($\bar{\beta}$)) is the PRLP involves only n variables, those of the cut coefficients. However, the number of constraints of the PRLP 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 p^{kt} be a vertex of $\mathcal{Q}^{kt}$, for $t \in [T]$. Then there exists a set of n linearly independent constraints of $\mathcal{Q}^{kt}$ tight at p^{kt}. The region satisfying these n constraints defines a (translated) polyhedral cone $\mathcal{C}^{kt} \supseteq \mathcal{Q}^{kt}$ with apex at p^{kt}, referred to as a *basis cone*.

Crucially, each basis cone $\mathcal{C}^{kt}$ has a compact $\mathcal{V}$-polyhedral description consisting of a single point p^{kt} and n rays. Balas and Kazachkov [4] exploit this by replacing the point-ray collection $(\mathcal{E}^k, \mathcal{R}^k)$ used for the constraints of (PRLP($\bar{\beta}$)) with the union of the points and rays defining the basis cones $\mathcal{C}^{kt}$ for all $t \in [T]$, leading to a *simple* PRLP with at most $T \cdot (n + 1)$ constraints. Cuts obtained from that PRLP 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, the points and rays for the simple PRLP can be readily obtained from the optimal tableau of the LP relaxation at each leaf node.

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($\bar{\beta}$)). As such, the Farkas certificate for this cut is not a byproduct of cut generation. Fortunately, Kazachkov and Balas [17, Section 3.1] show that, for the specific setting of simple VPCs, the Farkas certificate can be recovered efficiently.

Lemma 3. *Suppose that the first n inequalities in the description of $\mathcal{Q}^{kt}$ are $C^{kt}x \geq C_0^{kt}$ and are linearly independent, forming a basis cone $\mathcal{C}^{kt}$. If $\bar{\alpha}^\intercal x \geq \bar{\beta}$ is valid for $\mathcal{C}^{kt}$, then the multiplier on constraint $i \in [n]$ has value $v_i^t = \bar{\alpha}^\intercal r^i$, where r^i is column i of $-(C^{kt})^{-1}$.*

Lemma 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. 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 Generating Disjunctive Cuts

Our procedure relies on the following lemma, based on fundamental disjunctive programming concepts, which shows how to reuse a Farkas certificate from one instance in the sequence to produce valid inequalities for another instance.

Lemma 4. *Let $k \in [K]$ be the index of an arbitrary instance in a series. Then given a valid 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 $x \in \mathcal{S}^k$, 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 $\gamma^t_0 := v^t b^{kt}$ 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 $\gamma^t x \geq \gamma^t_0$ 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,

$$\alpha^\intercal x = \sum_{j\in[n]} \max_{t'\in[T]}\{\gamma^{t'}_j\}x_j \geq \sum_{j\in[n]} \gamma^t_j x_j \geq \gamma^t_0 \geq \min_{t'\in[T]}\{\gamma^{t'}_0\} = \beta.$$

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

Disjunctive Cuts via PRLP. When we talk below about generating disjunctive cuts via PRLP, we mean VPCs generated via the method described by Balas and Kazachkov [4, Algorithm 1], given a disjunction $\{(D^t, D^t_0)\}_{t\in[T]}$ constructed when solving another instance or as part of solving the same instance.

Disjunctive Cuts via Parameterization. Disjunctive cuts generated by parameterization are generated using Lemma 4 from stored certificates of validity $\{(v^t, D^t, D^t_0)\}_{t\in[T]}$ associated with a cut generated from another instance.

4 Computational Experiments

Computational Setup. 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.

Our test instances are drawn from among the MIPLIB instances with at most 1,000 variables (of which at most 150 are integer) and 1,000 constraints. From each base instance, we created a series by perturbing different elements of the input (objective direction, constraint matrix coefficients, constraint / variable bounds) to different degrees. Starting with a "base" instance, we perturbed

one selected input vector/matrix by increasing/decreasing random elements by random integer amounts, measuring the degree of perturbation by the angle between the original vector and the perturbed vector. Up to ten perturbed instances were created for each of three different degrees of perturbation (0.5, 1, 2) and for each of the types of elements perturbed: objective vector coefficients, constraint coefficient matrix, constraint / variable bounds. For variable bounds, we only performed perturbations that tightened bounds. We excluded perturbations that led to an infeasible instance. We also excluded some instances that were too small — perturbing the base instance data even just by one caused a too large (in angle) deviation. As a result, the number of perturbed instances in each series varies.

After the above process, we have one series for each combination of degree of perturbation and perturbation type (which vector is perturbed). For each combination of these series and three different sizes of disjunction (4, 16, and 64), we ran four separate experiments, the details of which are bulleted below. For the three sizes of disjunction, the number of terms excludes those added implicitly from tightening that occurs during strong branching. For these experiments, we used the latest "master" branch commit of CBC [1] with its default settings when called from the command line, including all default MILP cuts, with the exception that preprocessing is turned off. When an experiment calls for VPCs, they are generated only in the root node using a PRLP arising from a single disjunction, and they are added to a subclass of CglStored, a custom cut generator to CBC. We generate at most as many VPCs as we have fractional integer variables in the root LP relaxation. Each PRLP is solved with a sequence of (carefully selected) objective functions to obtain one round of "rank-one" cuts, all valid for the same disjunction and obtained nonrecursively. For details, see [4].

The experiments are conducted on a server cluster with 16 AMD Opteron Processor 6128's; 4 CPUs and 8GB of RAM were dedicated to each experiment, and each MILP instance solved is allotted 1,200 seconds of run time.

Experiments. For each instance, we can evaluate performance by several metrics: (1) *percent integrality gap closed* at the root, measuring improvement in the bound from the initial LP relaxation optimal objective value toward the (known) integer optimal value; (2) *branch-and-bound performance*: reporting time (in seconds), number of nodes processed, or total number of LP iterations to solve an instance to optimality.

Assuming we obtained a valid disjunction and parameterized cuts from the base instance in each series, there are 4 options to solve each perturbed instance:

Default:	Solve the instance with default settings, without any VPCs.
Calc Disj, Calc Cuts:	Generate a disjunction from a partial branch-and-bound tree and obtain VPCs from resulting PRLP. Solve the instance with these VPCs added to the root relaxation.
Param Disj, Calc Cuts:	Apply parameterized disjunction from base instance, with each term recalculated based on current instance data; ob-

tain VPCs from resulting PRLP. Solve the instance with these VPCs added to the root relaxation.

Param Disj, Param Cuts: Apply parameterized VPCs using Farkas certificates from base instance cuts, without a new disjunction or PRLP.

This leads to four sets of experiments:

Experiment 1. Use the default setting; this is the baseline for a cold solve.

Experiment 2. Test whether VPCs are effective in a best case scenario: generating a disjunction from an instance, generating VPCs from that disjunction, and applying these VPCs to the same instance (equivalently, applying parameterized cuts for perturbation of degree zero).

Experiment 3. Test whether the disjunction from one instance is effective when applied to another instance in the same series.

Experiment 4. Test whether the combination of a disjunction and a set of Farkas certificates generated when solving one instance is effective when applied to another instance.

Strength of Parameterized Cuts. To measure the strength of disjunctive cuts, we calculate the amount of integrality gap they close (defined in appendix A). 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 1. 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 2.

Table 1: Average percent integrality gap closed by disjunctive cuts and implied by disjunctions, grouped by degree of perturbation and size of disjunction.

Degree	Terms	Calc Disj	Param Disj	Calc Disj Calc Cuts	Param Disj Calc Cuts	Param Disj Param Cuts
0.5	4	9.48	7.40	8.22	6.38	4.90
	16	26.36	22.52	19.29	14.80	16.46
	64	48.61	45.21	36.10	26.97	26.89
1.0	4	11.59	9.01	6.11	5.61	4.80
	16	26.21	25.03	15.18	13.83	12.91
	64	39.92	37.84	23.10	18.80	15.42
2.0	4	9.66	8.21	4.82	4.50	3.52
	16	25.10	22.23	12.03	11.56	10.15
	64	30.05	27.63	13.95	12.92	7.40

In Table 1, column 1 is the "Degree" perturbation averaged across instances and types of changes. Column 2, labeled "Terms", states the number of disjunctive terms used to run the experiments. The next column refer to the best

possible percent integrality gap closed one can obtain after calculating a new disjunction per instance (the same disjunction that is computed for the second experiment described above). The subsequent column provides the best possible percent integrality gap closed when the disjunction from the base instance is reused. The next three columns give the average results of the experiments explained earlier.

There are a few overarching trends amongst VPCs and their generating disjunctions regarding integrality gap closed in Table 1 that are 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 the disjunctions close at least as much gap than the cuts they generate, which follows from a property of disjunctive cut generation. Third, disjunctions generated in experiment 2 close more integrality gap than disjunctions generated in experiment 3, and the same relationship exists between the cuts they generate (VPCs from experiments 2 and 3, respectively). For a well-tuned solver like CBC, this is again a predictible 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. Further, we see VPCs generated from experiment 4 tend to not close as much integrality gap as VPCs generated from experiment 3. The instances in experiment 4 recycle their Farkas certificates generating VPCs whereas the instances in experiment 3 tailor their VPCs' Farkas certificates by solving PRLPs, which often results in stronger cuts.

Two anticipated trends that were absent from Table 1 were increased differences in ratio of integrality gap closed between disjunctions of experiments 2 and 3 as well as between the VPCs of experiments 2 and 3 as degree of perturbation increased. Instead, we see these ratios stay relatively constant. We would expect the VPCs and disjunctions of experiment 3 to close less integrality gap as perturbation increases since greater differences in problem structure lead to different disjunctions employed while solving them, causing a weakening of old disjunctions and farkas coefficients. As seen in Table 2, perhaps the smaller sample sizes for the smaller perturbations are cause for this, or the difference in perturbation is too little for measurable effect. Additionally, a profiling could be formed similar to figures 1 - 3 that may elucidate a more nuanced relationship.

The columns in Table 2 can be understood as follows. "Degree" and "Terms" take the same meaning as in Table 1. The columns beginning with "Expt" refer to the integrality gap closed at the root node from CBC's default cut generators plus the VPCs generated by the corresponding experiment. "Instances" refers to the number of MILP instances were solved for the each experiment for the given degree of perturbation and size of disjunction.

With respect to Table 1, a couple of similar results are apparent in Table 2. Even with the addition of default cuts generated at the root node, VPCs from experiment 2 still close more integrality gap than VPCs from experiment 4, and, when holding perturbation constant, VPCs close more integrality gap when they

Table 2: Average percent root integrality gap closed from CBC's default cut generators with different choice of VPCs added based on experiment run. VPCs parameterize best for large disjunctions and small perturbations.

Degree	Terms	Default	Param Disj Param Cuts	Calc Disj Calc Cuts	Instances
0.5	4	26.08	26.26	26.49	55
	16	28.68	32.46	34.24	50
	64	32.88	44.54	47.93	39
1.0	4	40.76	40.76	42.89	100
	16	43.86	47.37	48.57	92
	64	46.80	55.05	55.33	72
2.0	4	38.95	39.34	40.02	127
	16	44.29	45.38	47.43	104
	64	44.53	47.22	49.27	80

are generated against more disjunctive terms. Again, there appears to be little relationship between the degree of perturbation and effect on ratio of integrality gap closed when comparing VPCs from experiment 2 to experiment 4. Our understanding of why these results repeat is the same as before.

Table 2 also yields a couple of new trends. Instances with VPCs often see an improvement in root integrality gap closed compared to the same instances without VPCs. 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 ratio of gap closed between VPCs from experiments 2 and 4 is much smaller when applied along with root cuts, which suggests that default root cuts recover some of the refined relaxation lost from using VPCs from experiment 4 instead of from experiment 2. Additionally, there does exist an inverse relationship between perturbation and ratio of integrality gap closed, except it is between the root cuts of experiment 1 and experiment 4. We believe this to be the case for the same reasons we expected to see a similar trend in Table 1.

Branch and Bound with Parameterized Cuts. Having analyzed the strength of VPCs, we turn our attention to their effect on branch and bound. The three comparisons of particular interest are how do the 4 experiments compare in terms of branch-and-bound nodes processed, LP iterations, and time to reach optimality. We present only the results with number of nodes; the others can be found in the appendix.

Figure 1 compares the 4 experiments in regards to the proportion of the test instances that are solved to optimality within a given number of nodes processed by branch-and-bound. Some trends in Table 2 reappear here, particularly measured performance improvements for small perturbations and large disjunctions as well as similar performance between experiments 2 and 4. It can be

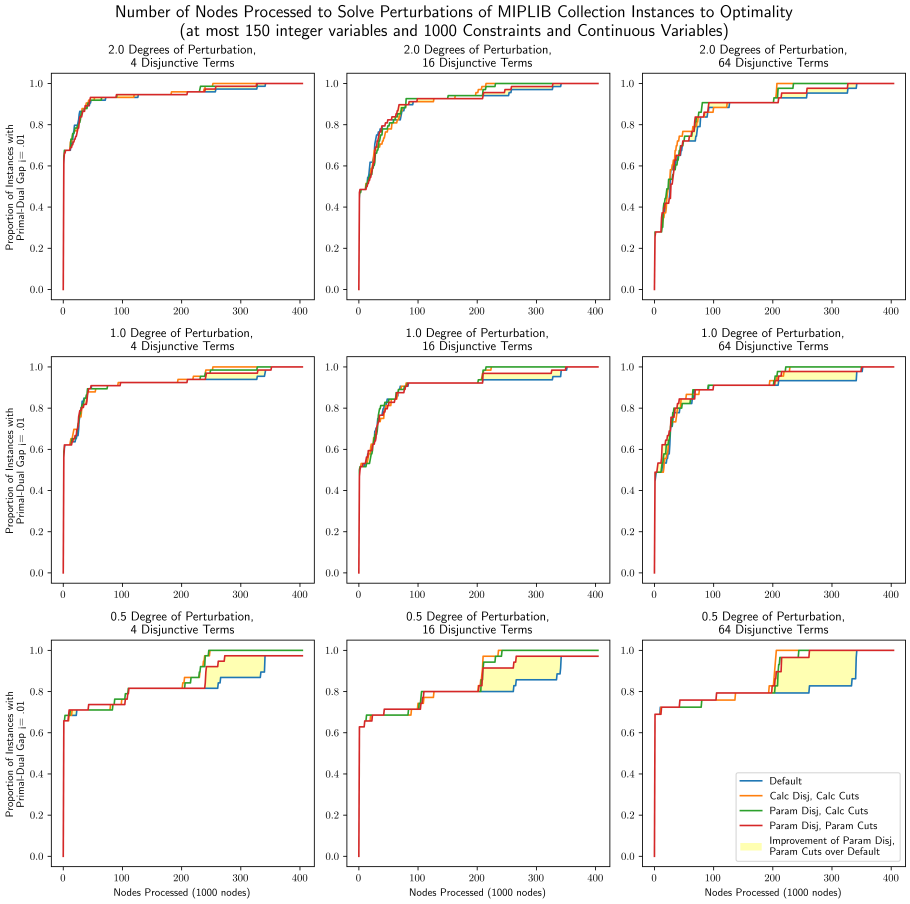

Fig. 1: Parameterizing VPCs for perturbations of small MIPLIB instances outperforms CBC with respect to number of branch-and-cut nodes processed to obtain optimality provided that the perturbations are sufficiently small and the disjunction used to initially generate the VPCs is sufficiently large.

hard to state with confidence the reasoning behind these improvements given the complexity of modern, production solvers. However, perhaps improvements are concentrated amongst the harder problems because VPCs' additional root node refinement prevent the solver from making detrimental branching decisions early in a solve, the impact of which becomes increasingly apparent as the size of the branch-and-bound tree grows.

Figure 3 shows a similar pattern to Figure 1 but with LP iterations. Given the correlation between the numbers of branch-and-bound nodes and LP iterations in a MILP solve, we expect the explanation for Figure 3 to echo that of Figure 1. For that reason, we append the display of Figure 3 to Appendix A.

Lastly, Figure 2 compares our 4 experiments in terms of the proportion of the test instances that are solved to optimality within a given amount of time. For length considerations, it has been appended to Appendix A. Important to note is that for large perturbations and small disjunctions, it appears that experiment 4 has a slight adverse effect on running time. Conversely, if the perturbation is small or the disjunction large, experiment 4 largely matches the running time of experiment 1, with an occurrence of significant time improvement for perturbations of 1 degree with disjunctions of 64 terms. Another takeaway is that experiments 2 and 3 tend to take more time than experiments 1 and 4 under any run setting. This is expected since experiments 2 and 3 require solving a PRLP, which can be a computationally expensive step.

5 Conclusion

We present a framework for reusing disjunctive cuts across a sequence of related instances. Our theoretical approach applies regardless of whether the objective or constraints change in the sequence, and it even allows for adding or deleting continuous variables. Computationally, we only perform limited experiments, and many open questions remain to be addressed as part of future research. For example, so far we have only run experiments with restrictions on the amount that any instance in a series can be perturbed from the original. In practice, this is a strong assumption, and as a series progresses, it may be sensible to regenerate a new batch of VPCs from a PRLP to maintain effectiveness of parameterizing VPCs. A useful future research endeavor could include "learning" when such a switch in generating VPCs should occur. Additionally, more study needs to take place to understand the size of disjunction to use when generating VPCs given that stronger cuts may be denser, which could hinder branching even if there is a significant bound improvement at the root node.

Bibliography

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

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

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

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

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

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

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

[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] 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. https://doi.org/10.1007/978-3-319-20086-6_14. URL https://doi.org/10.1007/978-3-319-20086-6_14.

[14] 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.

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

[16] 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.

[17] Aleksandr M. Kazachkov and Egon Balas. Monoidal strengthening of simple $\mathcal{V}$-polyhedral disjunctive cuts. In *Integer programming and combinatorial optimization*, volume 13904 of *Lecture Notes in Comput. Sci.*, pages 275–290. Springer, Cham, 2023. ISBN 978-3-031-32725-4; 978-3-031-32726-1. https://doi.org/10.1007/978-3-031-32726-1_20. URL https://doi.org/10.1007/978-3-031-32726-1_20.

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

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

[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. https://doi.org/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*, volume 2081 of *Lecture Notes in Comput. Sci.*, pages 348–360. Springer, Berlin, 2001. URL http://dx.doi.org/10.1007/3-540-45535-3_27.

[25] T.K. Ralphs and M. Güzelsoy. The SYMPHONY Callable Library for Mixed Integer Programming. In *Proceedings of the Ninth INFORMS Computing Society Conference*, pages 61–76, 2005. https://doi.org/10.1007/0-387-23529-9_5. URL http://coral.ie.lehigh.edu/~ted/files/papers/SYMPHONY04.pdf.

[26] 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.

[27] 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. https://doi.org/10.1287/ijoc.2020.0976. URL https://doi.org/10.1287/ijoc.2020.0976.

A Computational Experiments: Additional Details

A.1 Evaluation

As mentioned in Section 4, cuts are evaluated in two different ways, their strength and their effect on branch and bound. The strength of the cuts is assessed 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 applied. We define measure the quantity

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

A.2 Additional Results

A comparison of the 4 experiments in regards to the proportion of the test instances that are solved to optimality within a given amount of time is shown in Figure 2. Figure 3 illustrates a comparison of the 4 experiments in regards to the proportion of the test instances that are solved to optimality within a given number of LP iterations.

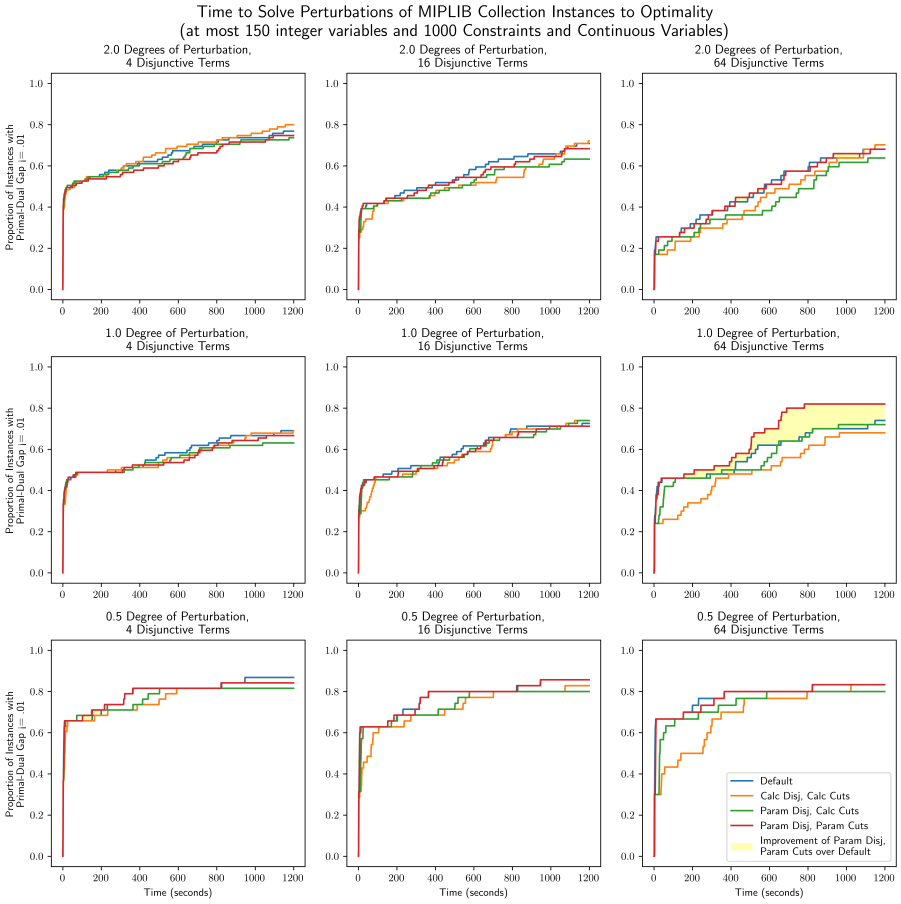

Fig. 2: Parameterizing VPCs for perturbations of small MIPLIB instances matches or outperforms CBC with respect to time to obtain optimality provided that the perturbations are sufficiently small and the disjunction used to initially generate the VPCs is sufficiently large.

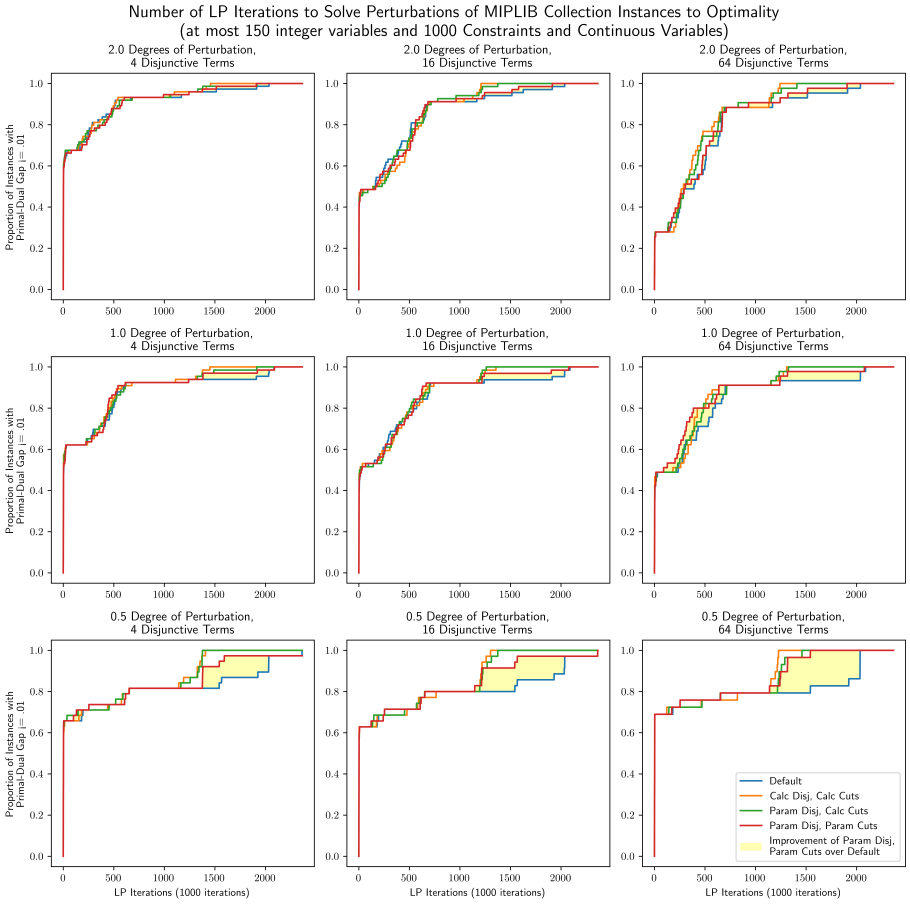

Fig. 3: Parameterizing VPCs for perturbations of small MIPLIB instances outperforms CBC with respect to number of LP iterations required to obtain optimality provided that the perturbations are sufficiently small and the disjunction used to initially generate the VPCs is sufficiently large.