Decomposition of Sparse Integer Programs via Nonlinear Edge Encodings and Column-and-Row Generation

Gustavo Angulo* Santanu S. Dey†

Abstract

A wide range of sparse integer programs admit a block structure in which subproblems interact through a small set of shared variables. Dualizing the linking equalities yields a decomposable Lagrangian relaxation, but generally introduces a duality gap. Recent work shows that this gap can be closed while preserving decomposability by dualizing exponentially large families of redundant nonlinear consistency constraints on the shared variables. We develop a computational framework for exploiting this idea without explicitly constructing the resulting exponentially large relaxation. Our framework combines nonlinear edge encodings of shared-variable consistency with a column-and-row generation (CRG) algorithm that generates local integer solutions by pricing and encoding constraints by separation. With complete encodings and exact separation, the framework recovers the exact relaxation while retaining independent optimization over the blocks. We introduce several encoding families and establish exponential separations among them: the Generalized family can require exponentially fewer constraints than the Vertex, Monomial, or Reflected families, yet can itself require exponentially many constraints on instances for which a single problem-specific encoding suffices. Computational experiments on decomposed stable-set and dominating-set instances show that CRG substantially outperforms a monolithic formulation across a range of tree topologies and coupling strengths. The results also show that richer encoding families need not perform better computationally, highlighting the choice of encoding as a central issue in effective decomposition.

Keywords: Sparse integer programming, Lagrangian decomposition, Dantzig–Wolfe decomposition, column-and-row generation, tree-structured optimization.

MSC: 90C10, 90C06, 90C46.

1 Introduction

A wide range of binary optimization problems arising in operations and engineering exhibit a block-sparse structure: each constraint involves only a small subset of the decision variables, and the variables can be grouped into blocks that interact through a limited number of shared variables; see the discussion in [8, 19]. This structure can be represented by the intersection graph of the constraint matrix [19, 20]. A tree decomposition of this graph [30] reveals a collection of subproblems organized on a tree, where adjacent subproblems interact only through shared variables [16].

Introducing a local copy of every variable in each block and enforcing consistency of the copies on the shared variables yields a block-structured formulation organized on a tree $T = (V, E)$,

$$\mathrm{OPT} := \min \left\{ \sum_{i \in V} f_i(\mathbf{x}^i) \ : \ \mathbf{x}^i \in X^i \ \ (i \in V), \ \ x_p^{e,i} = x_p^{e,j} \ \ (e = ij \in E, \ p \in Q^e) \right\}, \qquad (1)$$

*gangulo@uc.cl, Department of Industrial and Systems Engineering, Pontificia Universidad Católica de Chile, Santiago, Chile.

†sdey30@gatech.edu, H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, USA.

where X^i is a local feasible set associated with node $i \in V$, Q^e is the set of variables shared by the two blocks $i, j \in V$ incident to edge $e = ij \in E$, and the equalities enforce consistency of their local copies across the edges of the tree. More details are given in Section 2.

A leading source of such structure is two-stage and multistage stochastic binary optimization [11]. In the deterministic equivalent, each scenario defines a local block coupled to a separate first-stage block through the shared first-stage variables, so the consistency constraints $x_p^{e,i} = x_p^{e,j}$ in (1) impose nonanticipativity [31]. The tree decomposition $T = (V, E)$ is then a star, with the first-stage block at the center and the scenario blocks as leaves. Multistage problems produce deeper trees.

Decomposability against bound quality. Dualizing the coupling equalities $x_p^{e,i} = x_p^{e,j}$ in (1) decouples the formulation into one independent subproblem per block, an idea known as dual decomposition, variable splitting, or Lagrangian decomposition [14, 24, 13]. The resulting Lagrangian subproblems are solvable in parallel, and valid dual bounds are available at every iteration. The price of this decomposability is a duality gap: because the local sets are nonconvex, the Lagrangian dual generally falls short of OPT, and the resulting bound can be weak. This tension between decomposability and bound quality is one central obstacle in decomposition methods for integer programming. Methods that close the gap typically do so by giving up separability, by restricting the class of problems to which they apply, or by abandoning the availability of intermediate dual bounds.

This tension between decomposability and the duality gap can, however, be resolved by exploiting the fact that constraints that are redundant in the primal formulation need not be redundant in the Lagrangian dual. The paper [16] pioneered this approach, showing how carefully chosen redundant constraints can be used to strengthen the Lagrangian dual while preserving decomposability. For each edge $e = ij$ of the tree decomposition, the coupling constraints enforce $x_p^{e,i} = x_p^{e,j}$ for $p \in Q^e$. Since the local copies agree at every feasible solution, any function of the shared variables must also agree. Accordingly, for a chosen family of (nonlinear) functions $\mathcal{H}^e$, the formulation is augmented with redundant constraints of the form $\phi\big(\mathbf{x}^{e,i}\big) = \phi\big(\mathbf{x}^{e,j}\big)$ for $e = ij \in E$ and $\phi \in \mathcal{H}^e$. These constraints, henceforth referred to as encoding constraints, are dualized together with the original coupling constraints. The resulting relaxation remains separable across blocks while producing stronger dual bounds. The paper [16] shows that linear functions do not strengthen the dual and studies two nonlinear encoding families: monomial constraints, which enforce consistency of products of shared variables and induce the M-Lagrangian dual, and vertex-indicator constraints, which enforce consistency of all assignments of the shared variables and induce the V-Lagrangian dual. Both preserve decomposability and recover strong duality when enforced exhaustively on a tree decomposition. The resulting bound admits a Geoffrion-style primal characterization [22] as the value of a convex program obtained by replacing each local set with the convex hull of its lifted representation while leaving the coupling constraints explicit.

A zero-gap Lagrangian dual is not yet an algorithm. This resolves the tension in principle but not in computation, for two reasons. First, exact encoding requires exponentially many consistency constraints in the number of shared variables, making an explicit formulation of the full strengthened dual impractical. Second, when working with only a subset of the encoding constraints, the nonlinear Lagrangian dual is naturally optimized by subgradient or bundle methods, which return dual multipliers but no primal solution from which additional violated encoding constraints can be separated directly; recovering one is possible but requires additional machinery [4]. In practice, one must therefore work with a fixed, strict subfamily chosen before the solve begins;

see, for example, the computations in [16]. In the case of the M-Lagrangian dual, for example, this corresponds to an apriori truncation by degree, and the resulting bound depends on that truncation rather than adapting to the inequalities required by the particular instance.

This paper. We treat the primal characterization of the Lagrangian dual as the computational object and solve the corresponding convex program directly via Dantzig–Wolfe decomposition [17]. Columns correspond to feasible lifted integer solutions of the local blocks, while rows correspond to consistency constraints: the original equalities on shared variables together with the nonlinear encoding constraints induced by the chosen edge encodings. Since both the column set and the encoding constraints can be exponential in size, we generate columns by pricing and constraints by separation, yielding a column-and-row generation (CRG) framework.

A key consequence of working in the primal is that the encoding family enters only through a separation oracle, avoiding explicit instantiation of the exponentially large family of encoding constraints. Instead, violated constraints are identified dynamically and added to the master as needed, allowing the algorithm to converge to the relaxation defined by the complete family while materializing only a small fraction of it.

This yields, to the best of our knowledge, the first decomposition algorithm for integer programs that simultaneously has all three of the following properties. It is *exact*: with a complete encoding family and exact separation, the algorithm terminates with the optimal value of (1). It is *decomposable*: all subproblem work factors into one independent pricing problem per block, solvable in parallel, and a valid dual bound on OPT is available at the end of every outer iteration. And it is *structurally general*: it requires only that the block interaction graph be a tree and that the shared variables be binary. No convexity, continuous recourse, relatively complete recourse, or two-stage structure is assumed of the blocks.

Encoding choice is a modeling decision. Because the encoding family enters only through separation, the framework accommodates any family for which violated constraints can be identified. This leads to the question of how the choice of encoding family affects the method, which we investigate both theoretically and computationally. We organize the consistency encoding constraints through a general class of nonlinear edge encodings containing the Vertex and Monomial constructions of [16] together with two further families, Reflected and Generalized. Within this class we establish exponential separations: the Generalized family strictly dominates the other three, the Monomial and Reflected families are incomparable to one another, and even the Generalized family is not universal, since it can require exponentially many rows on a polynomially tractable instance that a single encoding outside the class closes. The proofs rest on classical polynomial threshold degree lower bounds [29] together with random restriction arguments.

Contributions. Therefore, the contributions of this paper are as follows.

1. *An exact, decomposable, and structurally general decomposition algorithm.* We develop a column-and-row generation (CRG) framework that computes the primal characterization of the encoding-strengthened Lagrangian bounds of [16]. The method combines Dantzig–Wolfe column generation with edge-based separation routines that dynamically identify and add violated encoding constraints, and it delivers the three properties stated above.

2. *A unified encoding framework.* We organize the generated consistency constraints through nonlinear edge encodings and introduce a Generalized encoding family that subsumes the Vertex and Monomial constructions of [16] and the new Reflected family.

3. *Theoretical limits of encoding families.* We establish that the Monomial, Reflected, and Vertex families can each require exponentially more rows than the Generalized family to close the gap, so the Generalized family strictly dominates all three. The Monomial and Reflected families are moreover not comparable to each other: one can require exponentially many rows where the other needs only polynomially many, and vice versa. Yet even the Generalized family is not universal: it can require exponentially many rows on a polynomially tractable instance for which a single encoding outside the above families closes the gap.

4. *Computational evaluation.* We evaluate the proposed encodings on decomposed stable-set and dominating-set instances over star, path, binary-tree, and random-tree topologies, three coupling levels, and both deterministic and stochastic objectives. Beyond the aggregate comparison, we analyze the structure of the generated rows and find that the families producing the most rows are not those producing the best bounds: Reflected and Generalized exhaust their row budget with dense, near-duplicate rows, while Monomial uses a fraction of the budget and gets more out of it.

Related decomposition approaches. We position our approach within the broader literature on decomposition for integer programming by comparing existing methods along three dimensions: exactness, decomposability, and structural generality.

- *Inexact methods*: Lagrangian and dual decomposition [14, 24] and progressive hedging [13] retain decomposability and intermediate dual bounds but leave a duality gap.

- *Methods requiring a master–recourse structure.* Benders-type methods achieve exactness by constructing an approximation of a recourse value function over the master variables. For two-stage stochastic integer programs with a pure binary first stage, the integer L-shaped method [7, 25, 3] is exact and decomposable. A substantial subsequent literature develops stronger cuts for stochastic integer programs, including split cuts derived in an extended space and projected into the Benders reformulation [12], as well as Lagrangian and strengthened Benders cuts [36, 15, 2, 18]; related constructions yield finite convergence for multistage problems with binary state variables. These methods are particularly close in spirit to ours: they also seek to close the gap arising from decomposition, and Lagrangian-cut generation invokes essentially the same local integer-optimization oracle that appears in our pricing step. Scenario decomposition [1] achieves exactness by a different mechanism, enumerating first-stage solutions rather than approximating a value function by cuts.

  The structural distinction is that these approaches exploit a directed master–recourse organization. For example, in multistage methods such as SDDiP, this takes the form of a sequential stagewise structure, with downstream decisions and value functions conditioned on the state inherited from earlier stages. In contrast, formulation (1) does not designate any block as a master and requires neither a stage ordering nor a recourse value function. Its blocks may be arranged on an arbitrary tree and interact symmetrically through shared variables. In particular, all block subproblems are independent and can be solved fully in parallel.

- *Non-decomposable approaches.* Augmented Lagrangian [21, 23] and semi-Lagrangian [6] duals attain strong duality at the expense of the decomposable structure, which [34] restores within an augmented Lagrangian framework through a mechanism distinct from the present one.

- *Other methods.* Nonserial dynamic programming [9] and LP formulations for polynomial optimization over tree-structured supports [10] achieve exactness on the same sparsity structure, but require an exponentially large object. In addition, nonserial dynamic programming

does not generally provide a useful bound if terminated before completion. In contrast, our method does not require an exponential-size formulation upfront and produces a valid dual bound after every completed outer iteration, providing meaningful information even when terminated before convergence.

On the mechanics, column-and-row generation builds on Dantzig–Wolfe decomposition [17, 27]: columns are generated by pricing, while the convexified master is progressively strengthened by dynamically generated coupling rows. Unlike branch-and-price [5], exactness is obtained without branching, through the generation of sufficiently strong encoding constraints while preserving blockwise pricing. The row-generation component plays a role analogous to a hierarchy of block-decomposable relaxations such as DRL* [28], and is related in motivation, though not in mechanism, to the lift-and-project, RLT, and Lasserre hierarchies [33, 26, 32].

Outline. Section 2 sets up the block-structured formulation, the edge encodings, and the convexified primal characterization. The column-and-row generation algorithm is developed in Section 3. Section 4 describes the four encoding strategies, and Section 5 formally establishes their strengths. Section 6 describes the test instances and computational setup, while results are discussed in Section 7. Finally, conclusions are drawn in Section 8.

2 Primal convexification by edge encodings

We consider a block-structured binary optimization problem whose blocks are organized by a tree $T = (V, E)$. Each node $i \in V$ represents a local feasible set X^i with local variables $\mathbf{x}^i$ and local cost function f_i. The local vector $\mathbf{x}^i$ contains the local copy of the variables that belong to block i, including the variables shared with neighboring blocks. For each edge $e = ij \in E$, let $Q^e = Q^{ij}$ denote the set of variables shared by blocks i and j. The nodes i and j are called the endpoints of e and Q^e is called the boundary of the edge e. We denote by $\mathbf{x}^{e,i}$ and $\mathbf{x}^{e,j}$ the boundary subvectors of $\mathbf{x}^i$ and $\mathbf{x}^j$, respectively, indexed by Q^e and written in a common order. Components are denoted by $x_p^{e,i}$ and $x_p^{e,j}$ for $p \in Q^e$. The decomposed formulation is

$$\begin{aligned} \mathrm{OPT} := \min \quad & \sum_{i \in V} f_i(\mathbf{x}^i) \\ \text{s.t.} \quad & \mathbf{x}^i \in X^i, \qquad i \in V, \\ & x_p^{e,i} = x_p^{e,j}, \quad e = ij \in E,\ p \in Q^e. \end{aligned} \tag{2}$$

Throughout, we assume that all variables appearing in coupling constraints are binary, that is, X^i includes the constraints that $x_p^{e,i} \in \{0,1\}$ for all $e = ij \in E,\ p \in Q^e$. Local variables that are not shared with any neighbor are not constrained to be binary.

The purpose of the strengthened Lagrangian reformulations is to add redundant constraints that are implied by the coupling equations $x_p^{e,i} = x_p^{e,j}$ in (2), but improve the Lagrangian dual bound. For each edge $e = ij \in E$, let $\mathcal{H}^e$ be an index set of *encoding functions* $\phi_h^e : \{0,1\}^{Q^e} \to \mathbb{R}$, $h \in \mathcal{H}^e$. The corresponding *encoding vector* associated with block i on edge $e = ij$ is denoted by $\mathbf{w}^{e,i}$, with scalar components $w_h^{e,i} = \phi_h^e(\mathbf{x}^{e,i})$, $h \in \mathcal{H}^e$. Analogously, block j has a vector $\mathbf{w}^{e,j}$ with entries $w_h^{e,j} = \phi_h^e(\mathbf{x}^{e,j})$. Since $\mathbf{x}^{e,i} = \mathbf{x}^{e,j}$ in every feasible solution of (2), the equalities $w_h^{e,i} = w_h^{e,j}$, $e = ij \in E$, $h \in \mathcal{H}^e$, are redundant for the original integer formulation. The resulting formulation

is:

$$\begin{aligned} \min \quad & \sum_{i \in V} f_i(\mathbf{x}^i) & & \\ \text{s.t.} \quad & \mathbf{x}^i \in X^i, & & i \in V, \\ & x_p^{e,i} = x_p^{e,j}, & & e = ij \in E,\ p \in Q^e, \\ & \phi_h^e(\mathbf{x}^{e,i}) = \phi_h^e(\mathbf{x}^{e,j}), & & e = ij \in E, h \in \mathcal{H}^e. \end{aligned} \tag{3}$$

The redundant constraint, when dualized, may strengthen the Lagrangian dual as discussed next [16].

Given a collection of encoding families $\mathcal{H} = \{\mathcal{H}^e\}_{e \in E}$, define the lifted local set

$$X^i_{\mathcal{H}} := \left\{ (\mathbf{x}^i, \mathbf{w}^i) \;\middle|\; \begin{array}{l} \mathbf{x}^i \in X^i, \\ \mathbf{w}^i = \{\mathbf{w}^{e,i} : e \in \delta(i)\}, \\ w_h^{e,i} = \phi_h^e(\mathbf{x}^{e,i}), \quad e \in \delta(i),\ h \in \mathcal{H}^e \end{array} \right\},$$

where $\delta(i)$ is the set of edges incident to i. The primal characterization of the encoding-strengthened Lagrangian dual is the following convex relaxation:

$$\begin{aligned} \mathrm{OPT}(\mathcal{H}) := \min \quad & \sum_{i \in V} \theta_i & & \\ \text{s.t.} \quad & (\mathbf{x}^i, \mathbf{w}^i, \theta_i) \in \mathrm{conv}\big\{(\mathbf{x}, \mathbf{w}, f_i(\mathbf{x})) : (\mathbf{x}, \mathbf{w}) \in X^i_{\mathcal{H}}\big\}, & & i \in V, \\ & x_p^{e,i} = x_p^{e,j}, & & e = ij \in E,\ p \in Q^e, \\ & w_h^{e,i} = w_h^{e,j}, & & e = ij \in E,\ h \in \mathcal{H}^e, \end{aligned} \tag{4}$$

where $w_h^{e,i} = w_h^{e,j}$ are the encoding constraints. It is straightforward to see that Problem (4) is a relaxation of (2). Since we use a minimization convention, this implies $\mathrm{OPT}(\mathcal{H})$ is a lower bound on OPT. Note that, by projection, the convexification in (4) can be restricted to shared variables only.

When the encoding family is suitably selected and the block structure satisfies the tree assumptions, this relaxation is exact [16]. In particular, the paper [16] introduces two specific encoding families, termed the M- and V-encodings, which we formally define in Section 4. The following theorem summarizes their main exactness result.

Theorem 2.1 ([16])**.** *Suppose that the block interaction graph is a tree. If the encoding family $\mathcal{H}$ is chosen as either the complete M-encoding or the V-encoding, then* $\mathrm{OPT}(\mathcal{H}) = \mathrm{OPT}$.

The Dantzig-Wolfe representation of (4) is obtained by writing each point of $\mathrm{conv}(X^i_{\mathcal{H}})$ as a convex combination of local integer columns. Let $\mathcal{P}^i$ be the set of feasible local columns of block i. A column $k \in \mathcal{P}^i$ consists of a local vector $\mathbf{x}^{i,k} \in X^i$, its restrictions (or subvector) $\mathbf{x}^{e,i,k}$ to the variables shared with neighboring blocks, for $e \in \delta(i)$, and the induced encoding entries $w_h^{e,i,k} = \phi_h^e(\mathbf{x}^{e,i,k})$ for $e \in \delta(i)$ and $h \in \mathcal{H}^e$. Let $c_{ik} := f_i(\mathbf{x}^{i,k})$ be the scalar cost of column k of

block i, and let λ_{ik} be its convex weight. The complete master problem is

$$\begin{aligned}
\min \quad & \sum_{i\in V}\sum_{k\in\mathcal{P}^i} c_{ik}\lambda_{ik} \\
\text{s.t.} \quad & \sum_{k\in\mathcal{P}^i} \lambda_{ik} = 1, && i \in V, \\
& \sum_{k\in\mathcal{P}^i} x_p^{e,i,k}\lambda_{ik} - \sum_{k\in\mathcal{P}^j} x_p^{e,j,k}\lambda_{jk} = 0, && e = ij \in E,\ p \in Q^e, \\
& \sum_{k\in\mathcal{P}^i} w_h^{e,i,k}\lambda_{ik} - \sum_{k\in\mathcal{P}^j} w_h^{e,j,k}\lambda_{jk} = 0, && e = ij \in E,\ h \in \mathcal{H}^e, \\
& \lambda_{ik} \geq 0, && i \in V,\ k \in \mathcal{P}^i.
\end{aligned} \tag{5}$$

The first constraints are the convexity constraints. The second group enforces consistency of the original shared variables. The third group enforces consistency of the selected edge encodings, that is, these are the encoding constraints. In exact formulations, either the column sets $\mathcal{P}^i$, the encoding sets $\mathcal{H}^e$, or both are exponentially large. The implementation therefore solves (5) by generating columns and encoding rows dynamically as discussed in the next section.

3 Column-and-row generation algorithm

At any iteration, the algorithm maintains a restricted set of columns $\mathcal{K}^i \subseteq \mathcal{P}^i$ for each block and a restricted set of encoding rows $\mathcal{C}^e \subseteq \mathcal{H}^e$ for each edge. The restricted master problem is obtained from (5) by replacing $\mathcal{P}^i$ by $\mathcal{K}^i$ and $\mathcal{H}^e$ by $\mathcal{C}^e$:

$$\begin{aligned}
z_{\mathrm{RMP}} := \min \quad & \sum_{i\in V}\sum_{k\in\mathcal{K}^i} c_{ik}\lambda_{ik} \\
\text{s.t.} \quad & \sum_{k\in\mathcal{K}^i} \lambda_{ik} = 1, && i \in V, \\
& \sum_{k\in\mathcal{K}^i} x_p^{e,i,k}\lambda_{ik} - \sum_{k\in\mathcal{K}^j} x_p^{e,j,k}\lambda_{jk} = 0, && e = ij \in E,\ p \in Q^e, \\
& \sum_{k\in\mathcal{K}^i} w_h^{e,i,k}\lambda_{ik} - \sum_{k\in\mathcal{K}^j} w_h^{e,j,k}\lambda_{jk} = 0, && e = ij \in E,\ h \in \mathcal{C}^e, \\
& \lambda_{ik} \geq 0, && i \in V,\ k \in \mathcal{K}^i.
\end{aligned} \tag{6}$$

The set $\mathcal{C}^e$ is initially empty or small, so that the first restricted master may contain only the original linear coupling rows. As the algorithm proceeds, new encoding rows are added to $\mathcal{C}^e$, and future columns receive their coefficients through the same functions ϕ_h^e.

3.1 Column generation for fixed encoding rows

Let α_i be the dual multiplier of the convexity constraint of block i. For each edge $e = ij \in E$, let π_p^e be the dual multiplier of the original coupling row for component $p \in Q^e$, and let μ_h^e be the dual multiplier of the generated encoding row indexed by $h \in \mathcal{C}^e$. We orient each edge as written, so the corresponding rows in (6) have the form "block i minus block j". Define

$$\tau_i^e := \begin{cases} +1, & \text{if } e = ij \text{ and } i \text{ is the first endpoint,} \\ -1, & \text{if } e = ji \text{ and } i \text{ is the second endpoint.} \end{cases}$$

For a fixed row set $\mathcal{C} = \{\mathcal{C}^e\}_{e \in E}$, the pricing problem for block i is

$$\rho_i := \min_{\mathbf{x}^i \in X^i} \left\{ f_i(\mathbf{x}^i) - \alpha_i \right. \\ \left. - \sum_{e \in \delta(i)} \tau_i^e \left(\sum_{p \in Q^e} \pi_p^e x_p^{e,i} + \sum_{h \in \mathcal{C}^e} \mu_h^e \phi_h^e(\mathbf{x}^{e,i}) \right) \right\}. \tag{7}$$

The value ρ_i is the minimum reduced cost among all columns of block i with respect to the current master. If $\rho_i < -\varepsilon_{\text{col}}$, an optimizer of (7) defines a new column and is added to $\mathcal{K}^i$. The column-generation phase for the current row set terminates when $\rho_i \geq -\varepsilon_{\text{col}}$ for every $i \in V$.

Solving the pricing problem once and adding any negative reduced-cost columns to the restricted master constitutes one *inner iteration*. When no negative reduced-cost column remains, (6) solves the full Dantzig-Wolfe master corresponding to the current encoding row set $\mathcal{C}$, thereby completing one *outer iteration*. Since this master is a relaxation of (1), its optimal objective value is a valid lower bound on OPT. The next outer iteration begins by separating violated encoding constraints and adding those that are identified, as described in the next section. The updated master is then re-solved via column generation, completing the outer iteration.

3.2 Row generation by encoding function separation

After column generation converges for the current encoding constraint set, the algorithm checks whether the current master solution violates any encoding equality that has not yet been added. For an edge $e = ij \in E$, the master solution assigns a nonnegative weight λ_{ik} to each column of block i, with the weights summing to one. Thus, the master solution can be viewed as a probability distribution over the columns of each block. For a signature $\mathbf{s} \in \{0,1\}^{Q^e}$ of the shared variables Q^e, let $\nu_{\mathbf{s}}^{e,i}$ denote the total weight of columns of block i whose shared variables take the value $\mathbf{s}$. Similarly, define $\nu_{\mathbf{s}}^{e,j}$ for block j. We refer to the collections $\nu^{e,i}$ and $\nu^{e,j}$ as the boundary distributions associated with edge e:

$$\nu_{\mathbf{s}}^{e,i} := \sum_{k \in \mathcal{K}^i \colon \mathbf{x}^{e,i,k} = \mathbf{s}} \lambda_{ik}, \qquad \nu_{\mathbf{s}}^{e,j} := \sum_{k \in \mathcal{K}^j \colon \mathbf{x}^{e,j,k} = \mathbf{s}} \lambda_{jk}, \qquad \mathbf{s} \in \{0,1\}^{Q^e}. \tag{8}$$

For a candidate encoding $h \in \mathcal{H}^e$, the violation of its consistency row is the difference $\Delta_h^e := \sum_{\mathbf{s} \in \{0,1\}^{Q^e}} \phi_h^e(\mathbf{s}) \left(\nu_{\mathbf{s}}^{e,i} - \nu_{\mathbf{s}}^{e,j} \right)$. If $|\Delta_h^e| > \varepsilon_{\text{row}}$ and $h \notin \mathcal{C}^e$, the algorithm adds the row

$$\sum_{k \in \mathcal{K}^i} \phi_h^e(\mathbf{x}^{e,i,k}) \lambda_{ik} - \sum_{k \in \mathcal{K}^j} \phi_h^e(\mathbf{x}^{e,j,k}) \lambda_{jk} = 0. \tag{9}$$

The separation oracle need not identify all violated rows. It may instead return only a subset, such as the most violated encoding rows on each edge or a fixed number of encoding rows per iteration.

3.3 Overall algorithm

The resulting procedure alternates between column generation (inner iteration) and row generation (outer iteration). A high-level description is given below.

1. Initialize nonempty column sets $\mathcal{K}^i$ for all $i \in V$ and initial encoding sets $\mathcal{C}^e$ for all $e \in E$. In the implementation, initial columns may be obtained from a monolithic solution, when available, by projecting it onto the blocks.

2. Solve the restricted master problem (6) and extract the dual multipliers α_i, π_p^e, and μ_h^e.

3. For each block $i \in V$, solve the pricing problem (7). Add every column whose reduced cost is smaller than $-\varepsilon_{\text{col}}$.

4. Repeat Steps 2–3 until no block generates a negative reduced-cost column.

5. For every edge $e = ij \in E$, compute the boundary distributions (8). Apply the selected separation strategy to find encodings $h \in \mathcal{H}^e \setminus \mathcal{C}^e$ with $|\Delta_h^e| > \varepsilon_{\text{row}}$.

6. Add the selected rows (9) to the master and update the pricing models so that the corresponding terms appear in (7).

7. Terminate if a full outer iteration adds neither rows nor columns in the corresponding inner iteration, or if an external stopping rule such as a time limit or relative gap tolerance is met. Otherwise, return to Step 2.

When the algorithm terminates because neither columns nor rows corresponding to encoding constraints are generated, then the current master solves exactly the relaxation defined by the generated encoding constraints. If the separation routine is exact for a complete encoding family which satisfies the assumptions for the strong-duality Theorem 2.1, this relaxation is exact and the resulting bound equals OPT. Otherwise, the algorithm returns the bound corresponding to the generated subset of encoding constraints.

4 Specific encoding strategies

The master problem and pricing framework are agnostic to the choice of encoding. An encoding strategy is specified by a family of functions ϕ_h^e, a separation oracle that identifies violated encoding constraints, and a pricing formulation that evaluates $\phi_h^e(\mathbf{x}^{e,i})$ in (7). We describe four strategies: Vertex, Monomial, Reflected, and Generalized.

4.1 Vertex strategy

The Vertex (or V) encoding was introduced in [16]. For each signature vector $\mathbf{s} \in \{0,1\}^{Q^e}$, define $\phi_{\mathbf{s}}^{V,e}(\mathbf{x}^{e,i}) := \mathbf{1}\{\mathbf{x}^{e,i} = \mathbf{s}\}$, where $\mathbf{1}\{\cdot\}$ is the indicator function. The corresponding encoding row is

$$\sum_{k \in \mathcal{K}^i} \mathbf{1}\{\mathbf{x}^{e,i,k} = \mathbf{s}\}\lambda_{ik} - \sum_{k \in \mathcal{K}^j} \mathbf{1}\{\mathbf{x}^{e,j,k} = \mathbf{s}\}\lambda_{jk} = 0,$$

or, equivalently, $\nu_{\mathbf{s}}^{e,i} = \nu_{\mathbf{s}}^{e,j}$. Thus, separation is immediate: compute the two marginal distributions on edge e and add any signatures for which $|\nu_{\mathbf{s}}^{e,i} - \nu_{\mathbf{s}}^{e,j}| > \varepsilon_{\text{row}}$.

In the pricing problem, the indicator $\mathbf{1}\{\mathbf{x}^{e,i} = \mathbf{s}\}$ can be represented by a binary variable $w_{\mathbf{s}}^{e,i}$. For a fixed signature $\mathbf{s}$, the standard linearization is

$$\begin{aligned}
&w_{\mathbf{s}}^{e,i} \le x_p^{e,i}, && p \in Q^e :\ s_p = 1,\\
&w_{\mathbf{s}}^{e,i} \le 1 - x_p^{e,i}, && p \in Q^e :\ s_p = 0,\\
&w_{\mathbf{s}}^{e,i} \ge \sum_{p \in Q^e:\, s_p = 1} x_p^{e,i} + \sum_{p \in Q^e:\, s_p = 0} (1 - x_p^{e,i}) - |Q^e| + 1.
\end{aligned}$$

4.2 Monomial strategy

The Monomial (or M) encoding was introduced in [16]. The Monomial strategy uses monomial encodings on the shared variables. For each subset $S \subseteq Q^e$, define $\phi_S^{\mathrm{M},e}(\mathbf{x}^{e,i}) := \prod_{p\in S} x_p^{e,i}$, with the convention that $\phi_\emptyset^{\mathrm{M},e} \equiv 1$. The row associated with S enforces equality of the corresponding "moment":

$$\sum_{k\in\mathcal{K}^i}\left(\prod_{p\in S} x_p^{e,i,k}\right)\lambda_{ik} - \sum_{k\in\mathcal{K}^j}\left(\prod_{p\in S} x_p^{e,j,k}\right)\lambda_{jk} = 0.$$

In the exact version of the Monomial strategy, separation is over all subsets $S \subseteq Q^e$. Given the current master solution, the most violated row in one direction can be found by solving

$$\max_{S\subseteq Q^e}\left\{\sum_{k\in\mathcal{K}^i}\lambda_{ik}w_S^{e,i,k} - \sum_{k\in\mathcal{K}^j}\lambda_{jk}w_S^{e,j,k}\right\}, \tag{10}$$

where $w_S^{e,i,k} := \prod_{p\in S} x_p^{e,i,k}$ and $w_S^{e,j,k} := \prod_{p\in S} x_p^{e,j,k}$. A symmetric problem with the two endpoints interchanged is also solved, that is maximizing $\sum_{k\in\mathcal{K}^j}\lambda_{jk}w_S^{e,j,k} - \sum_{k\in\mathcal{K}^i}\lambda_{ik}w_S^{e,i,k}$, and corresponding rows with violation are added.

A compact integer programming model for (10) uses a binary variable z_p^e to indicate whether p is selected in S, and binary variables $u_k^{e,i}$ and $u_k^{e,j}$ to indicate whether the selected monomial is active in column k on the two endpoints. One such model is:

$$\begin{aligned}
\max\quad & \sum_{k\in\mathcal{K}^i}\lambda_{ik}u_k^{e,i} - \sum_{k\in\mathcal{K}^j}\lambda_{jk}u_k^{e,j} & & \\
\text{s.t.}\quad & u_k^{e,l} \le x_p^{e,l,k} + 1 - z_p^e, & & l\in\{i,j\},\ k\in\mathcal{K}^l,\ p\in Q^e,\\
& \sum_{p\in Q^e} x_p^{e,l,k} z_p^e \le \sum_{p\in Q^e} z_p^e - 1 + u_k^{e,l}, & & l\in\{i,j\},\ k\in\mathcal{K}^l,\\
& z_p^e \in \{0,1\}, & & p\in Q^e,\\
& u_k^{e,l} \in \{0,1\}, & & l\in\{i,j\},\ k\in\mathcal{K}^l.
\end{aligned}$$

In the pricing problem, once a subset S has been selected, the monomial is linearized by introducing $w_S^{e,i}$ and imposing

$$\begin{aligned}
& w_S^{e,i} \le x_p^{e,i}, && p\in S,\\
& w_S^{e,i} \ge \sum_{p\in S} x_p^{e,i} - |S| + 1. &&
\end{aligned}$$

4.3 Reflected strategy

The Reflected strategy applies the Monomial strategy to complements of the shared variables. For each subset $S \subseteq Q^e$, define $\phi_S^{\mathrm{R},e}(\mathbf{x}^{e,i}) := \prod_{p\in S}(1 - x_p^{e,i})$, with the convention that $\phi_\emptyset^{\mathrm{R},e} \equiv 1$. The associated row matches the mass of columns in which all variables in S are equal to zero:

$$\sum_{k\in\mathcal{K}^i}\left(\prod_{p\in S}(1 - x_p^{e,i,k})\right)\lambda_{ik} - \sum_{k\in\mathcal{K}^j}\left(\prod_{p\in S}(1 - x_p^{e,j,k})\right)\lambda_{jk} = 0.$$

Separation is identical to the Monomial separation problem after replacing every constant $x_p^{e,i,k}$ by $1-x_p^{e,i,k}$ and every $x_p^{e,j,k}$ by $1-x_p^{e,j,k}$. Thus the same integer programming separation model can be used on complemented boundary signatures. The modification of the pricing problem is analogous.

Connection to Monomial encoding. Observe the following standard result.

Lemma 4.1. *For every subset $S \subseteq Q^e$, we have that $\phi_S^{\mathrm{M},e}(x) = \sum_{U\subseteq S}(-1)^{|U|}\phi_U^{\mathrm{R},e}(x)$ and $\phi_S^{\mathrm{R},e}(x) = \sum_{U\subseteq S}(-1)^{|U|}\phi_U^{\mathrm{M},e}(x)$. Therefore, the Monomial and Reflected encoding families span the same vector space of functions on $\{0,1\}^{Q^e}$.*

Proof. Both identities follow by expanding $\prod_{p\in S}(1-(1-x_p))$ and $\prod_{p\in S}(1-x_p)$ and collecting terms by inclusion–exclusion. □

In other words, every monomial can be expressed as a linear combination of reflected monomials, and vice versa. Therefore, any term of the form $\phi_S^e(x)$ appearing in the objective function of the Lagrangian relaxation of (3), when using monomial encoding, can be represented exactly using reflected monomial terms. Since the monomial and reflected encoding families span the same function space on $\{0,1\}^{Q^e}$, Theorem 2.1 immediately yields the following corollary.

Corollary 4.2. *Suppose that the block interaction graph is a tree. If the encoding family $\mathcal{H}$ is chosen as the complete Reflected encoding, then* $\mathrm{OPT}(\mathcal{H}) = \mathrm{OPT}$.

4.4 Generalized strategy

The Generalized strategy generalizes all the above strategies. Let $S^+, S^- \subseteq Q^e$ be disjoint sets. Coordinates in S^+ get a monomial term, and coordinates in S^- get a reflected term, and all other coordinates are left free. Thus, the encoding is $\phi_{S^+,S^-}^{\mathrm{G},e}(\mathbf{x}^{e,i}) := \prod_{p\in S^+} x_p^{e,i}\prod_{p\in S^-}(1-x_p^{e,i})$, with the convention that $\phi_{\emptyset,\emptyset}^{\mathrm{G},e} \equiv 1$. This family contains the Monomial strategy when $S^- = \emptyset$ and the Reflected strategy when $S^+ = \emptyset$. If $S^+ \cup S^- = Q^e$, it recovers a full signature indicator of the Vertex strategy. Since these encoding functions generalize all the previous families, we obtain the following:

Corollary 4.3. *Suppose that the block interaction graph is a tree. If the encoding family $\mathcal{H}$ is chosen as the complete Generalized encoding, then* $\mathrm{OPT}(\mathcal{H}) = \mathrm{OPT}$.

A direct separation model uses binary variables $z_p^{e,+}$ and $z_p^{e,-}$ to indicate whether coordinate p is fixed to one or to zero. For each current column, introduce binary variables $u_k^{e,i}$ and $u_k^{e,j}$ indicating whether the partial assignment is satisfied. The model maximizing the violation from endpoint i to endpoint j is

$$
\begin{aligned}
\max\quad & \sum_{k\in\mathcal{K}^i}\lambda_{ik}u_k^{e,i} - \sum_{k\in\mathcal{K}^j}\lambda_{jk}u_k^{e,j} \\
\text{s.t.}\quad & u_k^{e,l} \le x_p^{e,l,k} + 1 - z_p^{e,+}, && l\in\{i,j\},\ k\in\mathcal{K}^l,\ p\in Q^e,\\
& \sum_{p\in Q^e} x_p^{e,l,k} z_p^{e,+} + \sum_{p\in Q^e}(1-x_p^{e,l,k})z_p^{e,-} \le \sum_{p\in Q^e}(z_p^{e,+}+z_p^{e,-}) - 1 + u_k^{e,l}, && l\in\{i,j\},\ k\in\mathcal{K}^l,\\
& z_p^{e,+} + z_p^{e,-} \le 1, && p\in Q^e,\\
& z_p^{e,+}, z_p^{e,-} \in \{0,1\}, && p\in Q^e,\\
& u_k^{e,l} \in \{0,1\}, && l\in\{i,j\},\ k\in\mathcal{K}^l.
\end{aligned}
$$

As in the Monomial strategy, the sign-reversed separation problem is also solved.

For pricing, once a partial pattern (S^+, S^-) has been selected, introduce a binary variable $w^{e,i}_{S^+,S^-}$ and impose

$$\begin{aligned} w^{e,i}_{S^+,S^-} &\le x^{e,i}_p, && p \in S^+, \\ w^{e,i}_{S^+,S^-} &\le 1 - x^{e,i}_p, && p \in S^-, \\ w^{e,i}_{S^+,S^-} &\ge \sum_{p\in S^+} x^{e,i}_p + \sum_{p\in S^-} (1 - x^{e,i}_p) - |S^+| - |S^-| + 1. \end{aligned}$$

5 Comparison of encoding rules

Recall the four encoding families on the shared variables $\mathbf{x} \in \{0,1\}^Q$ of an edge; we suppress the edge superscript here for simplicity. For disjoint $S^+, S^- \subseteq Q$, the *Generalized* encoding function is $\phi^G_{S^+,S^-}(\mathbf{x}) := \prod_{p\in S^+} x_p \prod_{p\in S^-}(1-x_p)$, and the remaining families are its special cases: the *Monomial* function $\phi_S(\mathbf{x}) := \prod_{p\in S} x_p$ has $S^- = \emptyset$, the *Reflected* function $\phi^R_S(\mathbf{x}) := \prod_{p\in S}(1-x_p)$ has $S^+ = \emptyset$, and the *Vertex* function, which is a point indicator $\phi_{\mathbf{s}}(\mathbf{x}) := \mathbf{1}\{\mathbf{x} = \mathbf{s}\}$, has $S^+ \cup S^- = Q$. Thus as discussed before, every Vertex, Monomial, or Reflected encoding function is a Generalized one, and the Generalized family contains the other three.

It is therefore straightforward to see that the above containment is inherited by the row counts of column-and-row generation.

Proposition 5.1. *On any instance, the least number of Generalized encoding rows that closes the gap is at most the least number of Vertex, of Monomial, and of Reflected encoding rows that closes the gap.*

Proposition 5.1 leaves two questions. Are the three sub-families themselves ordered, or can each beat another? And is the domination of the Generalized family strict, or is the Generalized family efficient on every tractable block? This section answers both, through the results stated next. In each, the instance is a single oriented edge whose two endpoint blocks admit exact linear-optimization oracles of size polynomial in the number of boundary variables and whose coupled set is the singleton $\{(0, \mathbf{0})\}$, so that OPT = 0; we call such an instance *tractable*. The constructions and proofs occupy Sections 5.2–5.4 and rest on the gadget of Section 5.1.

Theorem 5.2 (Monomial versus Reflected)**.** *For every $m \ge 4$ even, with $n = 4m^3$, there is a tractable instance on $n+1$ boundary variables on which the Reflected family closes the gap with $m = (n/4)^{1/3}$ rows, while every set of Monomial rows that closes the gap has cardinality at least $\frac{1}{2}\,2^{m/2} = 2^{\Omega(n^{1/3})}$.*

By the complementation symmetry of the gadget the two roles reverse.

Corollary 5.3 (Reflected versus Monomial)**.** *For every $m \ge 4$ even, with $n = 4m^3$, there is a tractable instance on $n+1$ boundary variables on which the Monomial family closes the gap with $m = (n/4)^{1/3}$ rows, while every set of Reflected rows that closes the gap has cardinality at least $\frac{1}{2}\,2^{m/2} = 2^{\Omega(n^{1/3})}$.*

With Proposition 5.1, these two place the Generalized family strictly above both sub-families.

Theorem 5.4 (Generalized strictly dominates Monomial and Reflected)**.** *For every $m \geq 4$ even, with $n = 4m^3$, there is a tractable instance on which the Monomial family requires $2^{\Omega(n^{1/3})}$ rows while the Generalized family closes the gap with $m = (n/4)^{1/3}$ rows, and a tractable instance on which the Reflected family requires $2^{\Omega(n^{1/3})}$ rows while the Generalized family closes the gap with $m = (n/4)^{1/3}$ rows.*

The Vertex family can also be dominated as sharply.

Theorem 5.5 (Generalized strictly dominates Vertex)**.** *For every $n \geq 6$ divisible by 3, there is a tractable instance on $n+1$ boundary variables on which the Generalized family closes the gap with two rows, while every set of Vertex rows that closes the gap has cardinality at least $2^{n/3} = 2^{\Omega(n)}$.*

Finally, the Generalized family, richest of the four, is itself not universal: it can be exponential on a tractable instance that a single encoding outside the family closes.

Theorem 5.6 (Generalized lower bound)**.** *For every $n \geq 12$ there is a tractable instance on $n+1$ boundary variables, the* parity instance*, such that every set of Generalized rows that closes the gap has cardinality at least $\frac{1}{2}(4/3)^{\lceil n/4 \rceil} = 2^{\Omega(n)}$.*

Proposition 5.7 (A single non-family encoding closes the gap)**.** *For the parity instance of Theorem 5.6, there is a nonlinear encoding outside the Generalized family such that adding the single consistency row closes the gap.*

Remark 5.8. The examples used in the proofs of this section also suggest a practical modeling principle. Whenever the coupling equalities imply simple restrictions involving variables that belong to different blocks, it is useful to translate those implications into local constraints before running column-and-row generation. Such constraints are redundant for the complete coupled formulation, but they may be nonredundant after local convexification and may substantially strengthen the pricing problems. In this sense, part of the modeling effort should be to expose as many cross-block implications as possible through local block descriptions, rather than leaving all of them to be discovered by the generated edge encodings.

5.1 A common gadget

All four separations live on one single-edge gadget, parametrized by a subset of the cube. Let the $\mathbf{z}$-coordinates be indexed by a finite set R, take an oriented edge $e = ij$ with boundary $Q = \{0\} \cup R$, and write a boundary signature as $\mathbf{x} := (\tau, \mathbf{z})$ with $\tau = s_0$ and $z_p = s_p$ for $p \in R$. Fix $F \subseteq \{0,1\}^R$ and write $\bar{F} := \{0,1\}^R \setminus F$. The two endpoint blocks are $X^i := \{(0,\mathbf{0})\} \cup \{(1,\mathbf{z}) : \mathbf{z} \in F\}$ and $X^j := \{(0,\mathbf{0})\} \cup \{(1,\mathbf{z}) : \mathbf{z} \in \bar{F}\}$, with local costs $f_i(\mathbf{x}^i) = -\tau$ and $f_j(\mathbf{x}^j) = -\tau$. We call this the *$F$-gadget*. Its value and its row requirement are completely captured by the following separation problem.

Lemma 5.9 (Gadget lemma)**.** *For the F-gadget the coupled set is the singleton $\{(0,\mathbf{0})\}$ and $\mathrm{OPT} = 0$. Moreover, for any encoding family $\mathcal{H}$ and any positive integer k, the following are equivalent:*

(a) there is a set $\mathcal{C}$ of k rows from $\mathcal{H}$ whose augmented Dantzig-Wolfe master certifies $\mathrm{OPT} = 0$ with no negative reduced-cost column;

(b) there are encoding functions $\phi_1, \dots, \phi_k \in \mathcal{H}$ and reals $\pi_0, (\pi_p)_{p\in R}, (\mu_h)_h$ such that $r(\mathbf{z}) := \Big(\pi_0 - \sum_{h=1}^k \mu_h \phi_h(0,\mathbf{0})\Big) + \sum_{p\in R} \pi_p z_p + \sum_{h=1}^k \mu_h \phi_h(1,\mathbf{z})$ satisfies

$$r(\mathbf{z}) \leq -1 \quad (\mathbf{z} \in F), \qquad r(\mathbf{z}) \geq 1 \quad (\mathbf{z} \in \bar{F}). \tag{11}$$

We call any r satisfying (11) *a* margin separator for F.

Proof. After imposing $\mathbf{x}^i = \mathbf{x}^j$ a point with $\tau = 1$ would require $\mathbf{z} \in F$ and $\mathbf{z} \in \bar{F}$ simultaneously, so $X^i \cap X^j = \{(0, \mathbf{0})\}$, whence OPT $= 0$.

Let $\mathcal{C}$ be a finite set of encoding rows. Let α_i, α_j be the dual multipliers of the two convexity constraints, π_p ($p \in Q$) those of the original coupling rows, and μ_h ($h \in \mathcal{C}$) those of the generated rows. Put $L(\tau, \mathbf{z}) := \pi_0 \tau + \sum_{p \in R} \pi_p z_p + \sum_{h \in \mathcal{C}} \mu_h \, \phi_h(\tau, \mathbf{z})$. Because the edge is oriented from i to j, the reduced cost of a candidate column is $\bar{c}_i(\tau, \mathbf{z}) = -\tau - \alpha_i - L(\tau, \mathbf{z})$ over X^i and $\bar{c}_j(\tau, \mathbf{z}) = -\tau - \alpha_j + L(\tau, \mathbf{z})$ over X^j.

(a)⇒(b). The only rows with nonzero right-hand side are the two convexity constraints, each equal to 1, so a dual solution of objective value OPT $= 0$ has $\alpha_i + \alpha_j = 0$. Certification gives $\bar{c}_i, \bar{c}_j \geq 0$ throughout; at the origin $\bar{c}_i(0, \mathbf{0}) = -\alpha_i - L(0, \mathbf{0}) \geq 0$ and $\bar{c}_j(0, \mathbf{0}) = -\alpha_j + L(0, \mathbf{0}) \geq 0$, and adding them with $\alpha_i + \alpha_j = 0$ forces equality, so $-\alpha_i = \alpha_j = L(0, \mathbf{0})$. For $\mathbf{z} \in F$, $(1, \mathbf{z}) \in X^i$ yields $\bar{c}_i(1, \mathbf{z}) = -1 + L(0, \mathbf{0}) - L(1, \mathbf{z}) \geq 0$; for $\mathbf{z} \in \bar{F}$, $(1, \mathbf{z}) \in X^j$ yields $\bar{c}_j(1, \mathbf{z}) = -1 - L(0, \mathbf{0}) + L(1, \mathbf{z}) \geq 0$. Hence $r(\mathbf{z}) := L(1, \mathbf{z}) - L(0, \mathbf{0})$ satisfies (11). Substituting L and cancelling the constant gives the displayed form.

(b)⇒(a). Given r as in (b), read off duals: take μ_h, π_p, and π_0 as written, and set $\alpha_i = -L(0, \mathbf{0})$, $\alpha_j = L(0, \mathbf{0})$, so that $\alpha_i + \alpha_j = 0$ and the dual objective is 0. Then $\bar{c}_i(0, \mathbf{0}) = \bar{c}_j(0, \mathbf{0}) = 0$, while for $\mathbf{z} \in F$, $\bar{c}_i(1, \mathbf{z}) = -1 - r(\mathbf{z}) \geq 0$ by (11), and for $\mathbf{z} \in \bar{F}$, $\bar{c}_j(1, \mathbf{z}) = -1 + r(\mathbf{z}) \geq 0$. No column has negative reduced cost, so the rows certify OPT $= 0$. □

Lemma 5.9 reduces every claim below to a separation statement: an *upper bound* for a family is a short margin separator built from its encoding functions, and a *lower bound* is a proof that any margin separator built from its functions needs many of them.

We also record a complementation symmetry. Let $\kappa(\mathbf{z}) := \mathbf{1} - \mathbf{z}$ denote coordinatewise complementation on $\{0,1\}^R$. Then $\mathbf{z} \in \kappa(F)$ if and only if $\kappa(\mathbf{z}) \in F$, so the substitution $r \mapsto r \circ \kappa$ is a bijection between margin separators for F and margin separators for $\kappa(F)$ that preserves the number of encoding terms. On $\{0,1\}^R$, composition with κ maps affine functions to affine functions, exchanges the Monomial and Reflected functions ($\phi^M_S \circ \kappa = \phi^R_S$ and $\phi^R_S \circ \kappa = \phi^M_S$), maps $\phi^G_{S^+, S^-}$ to $\phi^G_{S^-, S^+}$, and maps the Vertex indicator of s to that of $\kappa(s)$. Therefore, for each family, the least number of terms from the family in a margin separator for F equals the least number of terms from the complemented family in a margin separator for $\kappa(F)$; combined with Lemma 5.9, this transfers row bounds between the two gadgets, with the Monomial and Reflected roles exchanged. Finally, since $\min\{\langle c, \mathbf{z}\rangle : \mathbf{z} \in \kappa(F)\} = \langle c, \mathbf{1}\rangle - \max\{\langle c, \mathbf{z}\rangle : \mathbf{z} \in F\}$, linear-optimization oracles transfer under κ, so the $\kappa(F)$-gadget is tractable whenever the F-gadget is.

5.2 Monomial versus Reflected versus Generalized

Fix $m \geq 4$ even. Set $w := 4m^2$ and $n := mw = 4m^3$, and index the $\mathbf{z}$-coordinates by m disjoint groups $G_1, \ldots, G_m$ of size w, writing z_{ab} for the b-th coordinate of group a ($a \in [m]$, $b \in [w]$). Consider the set-covering system $\sum_{b=1}^{w} z_{ab} \geq 1$ for $a \in [m]$, where $\mathbf{z} \in \{0,1\}^n$, and write $\mathrm{MP}(\mathbf{z}) = 1$ when $\mathbf{z}$ satisfies every covering inequality and $\mathrm{MP}(\mathbf{z}) = 0$ otherwise; equivalently, $\mathrm{MP}(\mathbf{z}) = 0$ exactly when some group is uncovered, $z_{G_a} = \mathbf{0}$. The *covering instance* is the F-gadget of Section 5.1 with $F = \{\mathbf{z} : \mathrm{MP}(\mathbf{z}) = 1\}$, the integer points of the covering system.

Write $\deg_{\pm}(g)$ for the *polynomial threshold degree* of a $\{0,1\}$-valued function g, the least degree of a polynomial p with $p(\mathbf{z}) > 0$ where $g(\mathbf{z}) = 1$ and $p(\mathbf{z}) < 0$ where $g(\mathbf{z}) = 0$. Let MP_k denote the group-covering function on k groups of $4k^2$ variables, that is $\mathrm{MP}_k(\mathbf{y}) = 1$ exactly when $\sum_{b=1}^{4k^2} y_{ab} \geq 1$ for every $a \in [k]$. We use the following bound (Minsky and Papert [29]).

Theorem 5.10 (Minsky–Papert, Section 3.2 [29])**.** *For every $k \geq 1$, $\deg_{\pm}(\mathrm{MP}_k) \geq k$.*

We also use that $\deg_{\pm}$ is monotone under coordinate fixing: if g is obtained from f by fixing some variables, then $\deg_{\pm}(g) \leq \deg_{\pm}(f)$, since any polynomial attaining $\deg_{\pm}(f)$ restricts to one for g of no larger degree.

The Monomial lower bound rests on the density estimate of Lemma 5.11 below, the Monomial-basis counterpart of Lemma 5.13. Both lower bounds use the following binomial tail estimate: for every integer $N \geq 1$,

$$\sum_{t \leq N/4} \binom{N}{t} \;\leq\; 4^{N/4}\left(\tfrac{4}{3}\right)^{3N/4} \;=\; 2^{N}\left(\tfrac{16}{27}\right)^{N/4}. \tag{12}$$

Indeed, each term of the expansion $1 = (\frac{1}{4}+\frac{3}{4})^N$ with $t \leq N/4$ satisfies $(\frac{1}{4})^t(\frac{3}{4})^{N-t} \geq (\frac{1}{4})^{N/4}(\frac{3}{4})^{3N/4}$, so keeping only these terms and dividing gives (12).

Lemma 5.11. *Let $r(\mathbf{z}) = l(\mathbf{z}) + \sum_{h=1}^{D} \mu_h\, \phi_h(\mathbf{z})$, where each $\phi_h(\mathbf{z}) = \prod_{p \in S_h} z_p$ is a Monomial encoding function and l has degree at most one. Suppose $r(\mathbf{z}) \leq -1$ whenever $\mathrm{MP}(\mathbf{z}) = 1$ and $r(\mathbf{z}) \geq 1$ whenever $\mathrm{MP}(\mathbf{z}) = 0$. Then $D \geq \frac{1}{2}\, 2^{m/2} = 2^{\Omega(n^{1/3})}$.*

Proof. Set $d := m/2$ and suppose, for contradiction, that $D < \frac{1}{2}\, 2^{d}$. Let $\mathcal{R}$ be the set of coordinate fixings that assign each variable z_{ab} one of two states, *free* or *fixed to* 0; then $|\mathcal{R}| = 2^{mw}$. For $\rho \in \mathcal{R}$ and any function f on $\{0,1\}^n$, let $f|_\rho$ denote its restriction to the face of ρ: the function of the free variables obtained by substituting the fixed values, so that $f|_\rho(\mathbf{z}) = f(\mathbf{z}')$ whenever $\mathbf{z}'$ completes ρ with the free values $\mathbf{z}$. Let F_a be the index set of free variables of group a. In particular, $r|_\rho$ is a polynomial in the free variables and, since ρ fixes variables to 0 only, $\mathrm{MP}|_\rho$ is again a group-covering indicator: $\mathrm{MP}|_\rho(\mathbf{z}) = 1$ exactly when $\sum_{b \in F_a} z_{ab} \geq 1$ for every $a \in [m]$.

A term ϕ_h with $|S_h| \geq d$ becomes the zero term under any ρ that fixes some variable of S_h to 0, and remains a term of degree $|S_h|$ only when every variable of S_h is free. The number of $\rho \in \mathcal{R}$ leaving ϕ_h nonzero is therefore $2^{mw-|S_h|} \leq 2^{mw-d}$. Summing over the at most D terms of degree at least d, the number of ρ under which any such term survives is at most $D\, 2^{mw-d} < \frac{1}{2}\, 2^{mw}$.

For a fixed group a, the number of ρ with $|F_a| < m^2$ is $2^{(m-1)w} \sum_{t<m^2} \binom{w}{t}$. Since $w = 4m^2$, the tail estimate (12) with $N = w$, so that $N/4 = m^2$, gives $\sum_{t<m^2} \binom{w}{t} \leq 2^w (16/27)^{m^2}$, and the count is therefore at most $2^{mw}(16/27)^{m^2}$. Over the m groups, at most $m\, 2^{mw}(16/27)^{m^2}$ fixings leave some group with fewer than m^2 free variables. We have that $m\,(16/27)^{m^2} < \frac{1}{4}$ for every $m \geq 2$: the value at $m = 2$ is $2\,(16/27)^4 < \frac{1}{4}$ and the ratio of consecutive values, $\frac{m+1}{m}(16/27)^{2m+1} \leq \frac{3}{2}(16/27)^5$, is below one. Therefore, the two counts sum to fewer than $(\frac{1}{2} + \frac{1}{4})\, 2^{mw} < 2^{mw}$. Hence some $\rho \in \mathcal{R}$ retains no term of degree at least d and gives every group at least m^2 free variables. Fix such a ρ.

We claim that $\mathrm{MP}_{m/2}$ is a fixing of $\mathrm{MP}|_\rho$. Define a fixing σ of the free variables as follows: in each group $a > m/2$, set one free variable to 1 and the remaining free variables of that group to 0; in each group $a \leq m/2$, fix all but $4(m/2)^2 = m^2$ of the free variables to 0, which is possible since $|F_a| \geq m^2$. Under σ, every group $a > m/2$ is covered outright, while a group $a \leq m/2$ is covered exactly when one of its m^2 surviving variables equals 1. Hence $(\mathrm{MP}|_\rho)|_\sigma = \mathrm{MP}_{m/2}$, the group-covering function on $m/2$ groups of $4(m/2)^2$ variables. By Theorem 5.10, $\deg_{\pm}(\mathrm{MP}_{m/2}) \geq m/2$, and since $\deg_{\pm}$ is monotone under fixing, $\deg_{\pm}(\mathrm{MP}|_\rho) \geq \deg_{\pm}\big((\mathrm{MP}|_\rho)|_\sigma\big) = \deg_{\pm}(\mathrm{MP}_{m/2}) \geq m/2$.

Every completion of ρ on the free variables is a point of $\{0,1\}^n$, so the hypotheses give $r|_\rho(\mathbf{z}) \leq -1$ where $\mathrm{MP}|_\rho(\mathbf{z}) = 1$ and $r|_\rho(\mathbf{z}) \geq 1$ where $\mathrm{MP}|_\rho(\mathbf{z}) = 0$. Thus $-r|_\rho$ is positive where $\mathrm{MP}|_\rho = 1$ and negative where $\mathrm{MP}|_\rho = 0$, so $\deg(r|_\rho) \geq \deg_{\pm}(\mathrm{MP}|_\rho) \geq m/2$. But $r|_\rho$ retains no term of degree at least d, and l has degree at most one, so $\deg(r|_\rho) \leq d - 1 < m/2$, a contradiction. Therefore $D \geq \frac{1}{2}\, 2^{d} = \frac{1}{2}\, 2^{m/2} = 2^{\Omega(n^{1/3})}$. □

Proof of Theorem 5.2. Tractability. Consider a linear objective on X^i. On the $\tau = 1$ part the constraints are the covering inequalities $\sum_{b=1}^{w} z_{ab} \geq 1$, one per group; since the groups occupy disjoint coordinates, the objective separates across groups and each group is optimized in $O(w)$ time by selecting its negative-cost variables and, if none is selected, the single variable of least cost. Comparing with the value of the origin gives the optimum in $O(n)$ time. On X^j the $\tau = 1$ part is $\bigcup_{a=1}^{m} \{\mathbf{z} : z_{G_a} = \mathbf{0}\}$, a union of m subcubes; optimizing over each and comparing with the origin gives the optimum in $O(mn)$ time. Both oracles are polynomial in n. Finally, by Lemma 5.9, the coupled set is the singleton $\{(0, \mathbf{0})\}$, so $\mathrm{OPT} = 0$ and the coupled set is trivial to optimize over.

Reflected upper bound. For each group $a \in [m]$ take the Reflected encoding function $\phi^R_{G_a}(\mathbf{z}) = \prod_{b=1}^{w}(1 - z_{ab})$, the indicator that group a is uncovered (all of its variables are 0). Then $\sum_a \phi^R_{G_a}(\mathbf{z})$ counts the uncovered groups, so it is 0 when $\mathbf{z}$ is feasible ($\mathrm{MP}(\mathbf{z}) = 1$) and at least 1 when $\mathbf{z}$ is infeasible ($\mathrm{MP}(\mathbf{z}) = 0$). Hence $r(\mathbf{z}) := -1 + 2\sum_{a=1}^{m} \phi^R_{G_a}(\mathbf{z})$ is a margin separator for F built from m Reflected functions, and by Lemma 5.9 these $m = (n/4)^{1/3}$ rows close the gap.

Monomial lower bound. By Lemma 5.9, a finite set $\mathcal{C}$ of Monomial rows certifies $\mathrm{OPT} = 0$ if and only if there is a margin separator for F, namely an affine function of $\mathbf{z}$ plus a μ-combination of the functions $\phi_h(1, \cdot)$, $h \in \mathcal{C}$, satisfying (11), that is $r(\mathbf{z}) \leq -1$ where $\mathrm{MP}(\mathbf{z}) = 1$ and $r(\mathbf{z}) \geq 1$ where $\mathrm{MP}(\mathbf{z}) = 0$. Fixing $\tau = 1$ in a Monomial encoding function yields a Monomial function of $\mathbf{z}$ or a constant absorbed into the affine part, so r has the form of Lemma 5.11 with at most $|\mathcal{C}|$ Monomial terms, and the displayed conditions are its hypotheses. Lemma 5.11 gives $|\mathcal{C}| \geq \frac{1}{2}\, 2^{m/2} = 2^{\Omega(n^{1/3})}$. □

Proof of Corollary 5.3. Let F be the feasible set of the covering instance of Theorem 5.2 and consider the $\kappa(F)$-gadget, where $\kappa(F)$ is the integer point set of the complemented system $\sum_{b=1}^{w}(1 - z_{ab}) \geq 1$, equivalently $\sum_{b=1}^{w} z_{ab} \leq w - 1$, $a \in [m]$ (no group is fully selected). By the complementation symmetry of Section 5.1, this gadget is tractable and its coupled set is the singleton $\{(0, \mathbf{0})\}$, and k rows from a family close its gap if and only if k rows from the complemented family close the gap of the covering instance. Since κ exchanges the Monomial and Reflected functions, the Reflected upper bound of Theorem 5.2 yields a Monomial upper bound of $(n/4)^{1/3}$ rows for the $\kappa(F)$-gadget, and its Monomial lower bound yields a Reflected lower bound of $\frac{1}{2}\, 2^{m/2}$. □

Proof of Theorem 5.4. On the covering instance the m Reflected functions of the upper bound are Generalized functions, so by Proposition 5.1 the Generalized family closes the gap with $m = (n/4)^{1/3}$ rows, whereas Theorem 5.2 forces $2^{\Omega(n^{1/3})}$ Monomial rows. On the instance of Corollary 5.3 the m Monomial functions are likewise Generalized functions, so the Generalized family closes the gap with $(n/4)^{1/3}$ rows, whereas the Reflected family requires $2^{\Omega(n^{1/3})}$. □

5.3 Vertex versus Generalized

Fix $n \geq 6$ divisible by 3, let $R = \{1, \ldots, n\}$, and partition R into three equal parts S^+, S^-, W of size $n/3$. Define the following special faces of the cube: $C_1 := \{\mathbf{z} : z_{S^+} = \mathbf{1},\ z_{S^-} = \mathbf{0}\}$, $C_2 := \{\mathbf{z} : z_{S^+} = \mathbf{0},\ z_{S^-} = \mathbf{1}\}$, each with the coordinates in W free, so $|C_1| = |C_2| = 2^{n/3}$. This *opposing-face instance* is the F-gadget with $F = C_1 \cup C_2$.

Proof of Theorem 5.5. Tractability. Over the $\tau = 1$ part of X^i optimize a linear objective over C_1 and over C_2 separately, each a subcube, and take the better; over X^j optimize over the full cube and, if the optimum lands in $C_1 \cup C_2$, flip the cheapest coordinate that leaves $C_1 \cup C_2$. Comparing with the origin gives both oracles in time $O(n)$.

Generalized upper bound. The functions $\phi^G_{S^+,S^-}$ and $\phi^G_{S^-,S^+}$ are the indicators of C_1 and C_2, each a Generalized encoding function carrying $n/3$ uncomplemented and $n/3$ complemented coordinates.

Since C_1 and C_2 are disjoint, $r(\mathbf{z}) := 1 - 2\phi^G_{S^+,S^-}(\mathbf{z}) - 2\phi^G_{S^-,S^+}(\mathbf{z})$ equals -1 on $C_1 \cup C_2 = F$ and $+1$ on $\bar{F}$, hence is a margin separator with two Generalized functions, and Lemma 5.9 closes the gap with these two rows. Neither function is Monomial, Reflected, or Vertex: each mixes uncomplemented and complemented coordinates and has support $2n/3 < n$.

Vertex lower bound. We show that closing the gap with Vertex rows needs at least $2^{n/3}$ of them.

A Vertex row uses a point indicator $\phi_{\mathbf{s}}(\mathbf{z}) = \mathbf{1}\{\mathbf{z} = \mathbf{s}\}$, which is 1 at the single signature $\mathbf{s}$ and 0 everywhere else.

By Lemma 5.9, if a set $\mathcal{C}$ of Vertex rows closes the gap, there is a margin separator $r(\mathbf{z}) = \ell(\mathbf{z}) + \sum_h \mu_h \phi_h(1, \mathbf{z})$ with ℓ affine. The restriction at $\tau = 1$ of a Vertex indicator $\phi_{\mathbf{s}}$, $\mathbf{s} \in \{0,1\}^Q$, is $\mathbf{1}\{\mathbf{z} = \mathbf{s}_R\}$ if $s_0 = 1$ and the constant 0 if $s_0 = 0$. Hence, r can be written as $r(\mathbf{z}) = \ell(\mathbf{z}) + \sum_h \mu_h \mathbf{1}\{\mathbf{z} = \mathbf{s}_h\}$, and satisfies $r(\mathbf{z}) \le -1$ for $\mathbf{z} \in F$ and $r(\mathbf{z}) \ge 1$ for $\mathbf{z} \in \bar{F}$.

Let $E = \{\mathbf{s}_h\}$ be the set of signatures used by these indicators, so $|E| \le |\mathcal{C}|$. At any point $\mathbf{z} \notin E$ all indicators vanish, so $r(\mathbf{z}) = \ell(\mathbf{z})$ there. The requirements on r therefore become requirements on the affine function alone: $\ell(\mathbf{z}) \le -1$ for $\mathbf{z} \in F \setminus E$ and $\ell(\mathbf{z}) \ge 1$ for $\mathbf{z} \in \bar{F} \setminus E$. That is, ℓ must be negative on the two tails C_1, C_2 and positive in the middle $\bar{F}$, except at the points of E, which the indicators are free to correct. It thus suffices to show that any affine ℓ violates these sign requirements at no fewer than $2^{n/3}$ points; each such point must lie in E, giving $|\mathcal{C}| \ge |E| \ge 2^{n/3}$.

Fix the coordinates in W to some value $\mathbf{y} \in \{0,1\}^W$, and consider the four signatures that share this $\mathbf{y}$: $\mathbf{u}_y := (\mathbf{1}_{S^+}, \mathbf{0}_{S^-}, \mathbf{y})$, $\mathbf{v}_y := (\mathbf{0}_{S^+}, \mathbf{1}_{S^-}, \mathbf{y})$, $\mathbf{m}_y := (\mathbf{1}_{S^+}, \mathbf{1}_{S^-}, \mathbf{y})$, $\mathbf{m}'_y := (\mathbf{0}_{S^+}, \mathbf{0}_{S^-}, \mathbf{y})$. Here $\mathbf{u}_y \in C_1$ and $\mathbf{v}_y \in C_2$, so both lie in F and ℓ should be ≤ -1 on them; while $\mathbf{m}_y, \mathbf{m}'_y \in \bar{F}$, so ℓ should be ≥ 1 on them.

Coordinate by coordinate, $\mathbf{u}_y + \mathbf{v}_y = \mathbf{m}_y + \mathbf{m}'_y$. Since ℓ is affine, we have $\ell(\mathbf{u}_y) + \ell(\mathbf{v}_y) = \ell(\mathbf{m}_y) + \ell(\mathbf{m}'_y)$. If ℓ had the correct sign at all four points, the left side would be at most -2 and the right side at least $+2$, which is impossible. Hence ℓ has the wrong sign at one of the four points, and that point lies in E.

The four points all carry the same W-block $\mathbf{y}$, so the quadruples for different $\mathbf{y}$ are disjoint. There are $2^{n/3}$ choices of $\mathbf{y}$, each forcing a distinct point into E. Therefore $|E| \ge 2^{n/3}$, and so $|\mathcal{C}| \ge 2^{n/3}$. □

5.4 The Generalized family is not universal

We begin by introducing some definitions and the tool used in the proof. A real polynomial p *sign-represents* a function $g\colon \{0,1\}^n \to \{-1,+1\}$ if $\operatorname{sign} p(\mathbf{x}) = g(\mathbf{x})$ for every $\mathbf{x} \in \{0,1\}^n$. The *sign degree* of g is the minimum degree of any polynomial that sign-represents g.

Theorem 5.12 (Minsky–Papert, Section 3.1 [29])**.** *The sign degree of the parity function $\mathbf{x} \mapsto -(-1)^{|\mathbf{x}|}$ on $n \ge 1$ variables equals n, where $|\mathbf{x}| := \sum_{q=1}^n x_q$ denotes the Hamming weight of $\mathbf{x} \in \{0,1\}^n$.*

Lemma 5.13. *Let $n \ge 12$ and let $r(\mathbf{z}) = l(\mathbf{z}) + \sum_{h=1}^m \mu_h \phi_h(\mathbf{z})$, where each ϕ_h is a Generalized encoding function and l has degree at most one. Suppose $r(\mathbf{z}) \le -1$ for $|\mathbf{z}|$ even and $r(\mathbf{z}) \ge 1$ for $|\mathbf{z}|$ odd. Then $m \ge \frac{1}{2}(4/3)^{\lceil n/4 \rceil} = 2^{\Omega(n)}$.*

Proof of Lemma 5.13. The hypothesis says exactly that $\operatorname{sign} r(\mathbf{z}) = -(-1)^{|\mathbf{z}|}$ with $|r(\mathbf{z})| \ge 1$ for all $\mathbf{z} \in \{0,1\}^n$; in particular, r sign-represents parity. Let $S_h \subseteq \{1, \dots, n\}$ be the support of ϕ_h (i.e., the set of coordinates $S^+ \cup S^-$ in its definition), so that $\deg \phi_h = |S_h|$.

Set $d = \lceil n/4 \rceil \ge 2$. Let $m_{\ge d}$ denote the number of rows ϕ_h with $|S_h| \ge d$. We will prove that $m_{\ge d} \ge \frac{1}{2}(4/3)^d$. Since $m \ge m_{\ge d}$, this proves the result.

Recall that each row ϕ_h may be written as $\phi_h(\mathbf{z}) = \prod_{p \in S_h} \mathbf{1}\{z_p = a_p\}$ for $a_p \in \{0,1\}$. We say a fixed coordinate p *annihilates* ϕ_h if its assigned value forces this product to be identically 0, that is, if the factor $\mathbf{1}\{z_p = a_p\}$ becomes the constant 0.

Consider the set $\mathcal{R}$ of coordinate labelings that assign to each coordinate $p \in \{1, \ldots, n\}$ one of four labels, free_0, free_1, fix_0, or fix_1, so that $|\mathcal{R}| = 4^n$. A labeling ρ acts as a restriction: coordinates labeled free_0 or free_1 remain free, and the others are fixed to the indicated bit; the two free labels are duplicates whose only purpose is to make the four labels equally numerous. The fixed coordinates become constants, $r|_\rho$ is a polynomial in the $n' = n'(\rho)$ free coordinates, and we call the corresponding subcube the *face*.

Fix a row ϕ_h with support of size $k \geq d$ and a support coordinate p, whose factor is $\mathbf{1}\{z_p = a_p\}$. Among the four labels of p, exactly one, namely fix_{1-a_p}, annihilates ϕ_h. Since a labeling chooses one label per coordinate, the number of $\rho \in \mathcal{R}$ under which ϕ_h is annihilated by none of its k support coordinates is the product of the per-coordinate counts, three at each support coordinate and four elsewhere: $3^k 4^{n-k} \leq (3/4)^d 4^n$.

Suppose, for contradiction, that $m_{\geq d} < \frac{1}{2}(4/3)^d$. Since $(4/3)^d(3/4)^d = 1$, summing the survival counts over the rows with $|S_h| \geq d$ shows that fewer than $\frac{1}{2}\,4^n$ labelings leave some such row not annihilated.

Next, a labeling with free set of size t is determined by the free set, one of two free labels on each free coordinate, and one of two bits on each fixed coordinate, so the number of ρ with $n' < n/4$ is $2^n \sum_{t<n/4} \binom{n}{t}$. The tail estimate (12) with $N = n$ gives $\sum_{t<n/4} \binom{n}{t} \leq 2^n (16/27)^{n/4}$, so at most $4^n(16/27)^{n/4}$ labelings satisfy $n' < n/4$, and $(16/27)^{n/4} \leq (16/27)^3 < \frac{1}{4}$ for $n \geq 12$. The two counts sum to fewer than $(\frac{1}{2} + \frac{1}{4})\,4^n < 4^n$ labelings, so some $\rho \in \mathcal{R}$ annihilates every row with $|S_h| \geq d$ and satisfies $n' \geq n/4$. Fix such a ρ.

On its face, $r|_\rho$ is the affine part plus rows with $|S_h| < d$, so $\deg(r|_\rho) \leq d - 1 < n/4 \leq n'$. Every assignment to the free coordinates, together with the bits fixed by ρ, is a point of $\{0,1\}^n$, so each value of $r|_\rho$ on the face is a value of r; in particular $r|_\rho(\mathbf{z}) \leq -1$ or $r|_\rho(\mathbf{z}) \geq 1$ at every point of the face. Moreover, fixing the assigned coordinates changes the parity of a point only by the constant amount contributed by those bits, so on the face $r|_\rho$ sign-represents the parity of the n' free coordinates, up to a single global sign determined by the assigned bits. By Theorem 5.12, sign-representing parity on n' variables requires degree n', contradicting $\deg(r|_\rho) < n'$. Therefore $m \geq m_{\geq d} \geq \frac{1}{2}(4/3)^{\lceil n/4 \rceil} = 2^{\Omega(n)}$. □

Proof of Theorem 5.6. Let $E_n := \{\mathbf{z} \in \{0,1\}^n : \sum_{p=1}^n z_p \text{ is even}\}$ and $O_n := \{\mathbf{z} \in \{0,1\}^n : \sum_{p=1}^n z_p \text{ is odd}\}$. The *parity instance* is the F-gadget of Section 5.1 with $F = E_n$ (so $\bar{F} = O_n$); that is, the edge $e = ij$ carries the two endpoint blocks $X^i := \{(0, \mathbf{0})\} \cup \{(1, \mathbf{z}) : \mathbf{z} \in E_n\}$ and $X^j := \{(0, \mathbf{0})\} \cup \{(1, \mathbf{z}) : \mathbf{z} \in O_n\}$, with local costs $f_i(\mathbf{x}^i) = -\tau$ and $f_j(\mathbf{x}^j) = -\tau$.

Tractability. Given a linear objective, optimizing over the $\tau = 1$ part of X^i reduces to optimizing over the even-parity vectors; this is done by taking the coordinatewise best vector and, if its parity is wrong, flipping a coordinate of minimum loss, and then comparing with the value of the origin. The same argument applies to X^j with odd parity. Thus, each endpoint block admits an $\mathcal{O}(n)$ linear-optimization oracle. By Lemma 5.9, the coupled set is the singleton $X^i \cap X^j = \{(0, \mathbf{0})\}$, so $\text{OPT} = 0$ and the coupled set is trivial to optimize over.

Generalized lower bound. By Lemma 5.9, a finite set $\mathcal{C}$ of Generalized encoding rows certifies $\text{OPT} = 0$ if and only if there is a margin separator for $F = E_n$, namely an affine function of $\mathbf{z}$ plus a μ-combination of the functions $\phi_h(1, \cdot)$, $h \in \mathcal{C}$, satisfying (11), that is $r(\mathbf{z}) \leq -1$ for $\mathbf{z} \in E_n$ and $r(\mathbf{z}) \geq 1$ for $\mathbf{z} \in O_n$. Fixing $\tau = 1$ in a Generalized encoding function yields either a constant, which is absorbed into the affine part, or again a Generalized encoding function of $\mathbf{z}$. Hence r has exactly the form treated in Lemma 5.13, with at most $|\mathcal{C}|$ Generalized terms, and the displayed conditions

are exactly its hypothesis. Lemma 5.13 then forces $|\mathcal{C}| \geq \frac{1}{2}(4/3)^{\lceil n/4 \rceil} = 2^{\Omega(n)}$ and completes the proof. □

For the same instance, define the parity encoding $\phi^{\mathrm{par}}(\tau, \mathbf{z}) := \tau(-1)^{\sum_{p=1}^{n} z_p}$. Thus $\phi^{\mathrm{par}}(0, \mathbf{0}) = 0$, while $\phi^{\mathrm{par}}(1, \mathbf{z}) = 1$ for $\mathbf{z} \in E_n$ and $\phi^{\mathrm{par}}(1, \mathbf{z}) = -1$ for $\mathbf{z} \in O_n$.

Proof of Proposition 5.7. Set $r(\mathbf{z}) := -\phi^{\mathrm{par}}(1, \mathbf{z})$, so that $r(\mathbf{z}) = -1$ for $\mathbf{z} \in E_n$ and $r(\mathbf{z}) = 1$ for $\mathbf{z} \in O_n$. This is a margin separator for $F = E_n$ in the sense of (11), built from the single encoding function ϕ^{par} (with no affine part). By Lemma 5.9, applied to the single-function family $\{\phi^{\mathrm{par}}\}$, the one row ϕ^{par} certifies OPT = 0, that is, it closes the gap. By Theorem 5.6, every set of Generalized rows closing the gap of this instance has cardinality at least $\frac{1}{2}(4/3)^{\lceil n/4 \rceil} > 1$ for $n \geq 12$, whereas the parity row closes it alone. Hence the parity row is not affinely equivalent to any single Generalized row, nor to any subexponential collection of them. □

6 Test instances and computational setup

6.1 Problem type

Stable-set instances. For the stable-set family, each block $i \in V$ contains a local conflict graph $G_i = (U_i, F_i)$, where U_i is the set of local binary variables and F_i is the set of local conflict edges. A local feasible solution is a stable set of G_i: $X^i_{\mathrm{SS}} := \left\{\mathbf{x}^i \in \{0,1\}^{U_i} : \; x^i_u + x^i_v \leq 1 \quad \forall \{u, v\} \in F_i\right\}$.

Arranging the blocks above on a tree $T = (V, E)$, the full coupled stable-set instance is then:

$$\begin{aligned} \max \quad & \sum_{i \in V} \sum_{v \in U_i} x^i_v \\ \text{s.t.} \quad & \mathbf{x}^i \in X^i_{\mathrm{SS}}, \quad i \in V, \\ & x^{e,i}_p = x^{e,j}_p, \quad e = ij \in E, \; p \in Q^e. \end{aligned}$$

The local conflict constraints remain inside the blocks, while the equalities identify selected variables across adjacent blocks.

In [16] it was shown that, for packing IPs, requiring the projection of the feasible region onto the boundary variables to agree across every edge yields convergence of the monomial method, with explicit approximation guarantees for subfamilies of monomial encodings of bounded degree. Moreover, following Remark 5.8, we introduce a simple conflict-propagation step along the coupling edges before constructing the local optimization models. Consider an edge $e = ij \in E$ and two boundary variables $p, q \in Q^e$. If the corresponding variables are adjacent in the local conflict graph of block i, then the same conflict is inserted between the paired boundary variables of block j, and vice verse. This propagation is applied in both directions over all block-tree edges until no new local conflict edge is added. This conflict resolution achieves that the projections of the two blocks onto the shared boundary Q^e agree. The preprocessing does not change the feasible solutions of the complete coupled instance, but it strengthens the individual block models.

Dominating-set instances. For the dominating-set family, each block $i \in V$ again contains a local graph $G_i = (U_i, F_i)$. For $v \in U_i$, let $N_i(v) := \{u \in U_i : \{u, v\} \in F_i\}$ be the open neighborhood of v inside block i. A local feasible solution is a dominating set of G_i: $X^i_{\mathrm{DS}} := \left\{\mathbf{x}^i \in \{0,1\}^{U_i} : x^i_v + \sum_{u \in N_i(v)} x^i_u \geq 1 \quad \forall v \in U_i\right\}$.

Arranging the blocks above on a tree $T = (V, E)$, the full coupled dominating-set instance is then:

$$\begin{aligned} \min \quad & \sum_{i \in V} \sum_{v \in U_i} x_v^i \\ \text{s.t.} \quad & \mathbf{x}^i \in X^i_{\mathrm{DS}}, \quad i \in V, \\ & x_p^{e,i} = x_p^{e,j}, \quad e = ij \in E,\ p \in Q^e. \end{aligned}$$

As in the stable-set case, the only constraints coupling different blocks are the equality constraints on the selected boundary variables.

For the dominating-set instances, we use the same principle as in Remark 5.8: implications induced by the coupling equalities are made explicit whenever they can be expressed as local constraints. Here, this is achieved by a simple neighborhood-dominance preprocessing step along each coupling edge. For an edge $e = ij \in E$ and a boundary variable $p \in Q^e$, let $N_i(p)$ and $N_j(p)$ denote the open neighborhoods of the paired local variables in blocks i and j. If $N_j(p)$ lies entirely in the boundary of e, it can be translated through the coupling map to block i. Whenever the translated neighborhood is a strict subset of $N_i(p)$, we replace the local neighborhood of p in block i by this smaller neighborhood. In terms of domination constraints, this replaces $x_p^{e,i} + \sum_{u \in N_i(p)} x_u^i \geq 1$ by the stronger local inequality $x_p^{e,i} + \sum_{u \in N'_j(p)} x_u^i \geq 1$, where $N'_j(p) \subsetneq N_i(p)$ is the neighborhood of the paired variable in block j, translated to the local indexing of block i. The procedure is applied in both directions over all block-tree edges until no neighborhood can be reduced further. As in the stable-set case, the preprocessing preserves the feasible set of the full coupled formulation, while strengthening the local block descriptions used by the decomposition algorithm.

6.2 Coupling equalities and topologies of $T = (V, E)$

A coupling-density parameter specifies how many equality constraints are placed on each edge of the block tree. We denote this number by q. For each edge $e = ij \in E$, we select two ordered lists of local indices, one in block i, $I^{e,i} = (\iota_1^{e,i}, \ldots, \iota_q^{e,i})$, and one in block j, $I^{e,j} = (\iota_1^{e,j}, \ldots, \iota_q^{e,j})$, and impose the pairwise equalities $x^i_{\iota_p^{e,i}} = x^j_{\iota_p^{e,j}}$, $p = 1, \ldots, q$. Equivalently, $Q^e = \{1, \ldots, q\}$ indexes the coupling positions on edge e, and $x_p^{e,i}$ denotes the local variable $x^i_{\iota_p^{e,i}}$. The choice of the ordered index lists depends on the block-tree topology. Fix $n := |U_i|$ as the number of local variables in all blocks i.

Star. The star topology has center block 1 and edges $e = 1j$, $j = 2, \ldots, |V|$. For every such edge, the same first q variables of the center block are coupled with the first q variables of the leaf block: $I^{1j,1} = (1, \ldots, q)$, $I^{1j,j} = (1, \ldots, q)$, for $j = 2, \ldots, |V|$. Thus the same center variables are reused across all incident edges. This creates the usual two-stage structure in which all leaf blocks share the same first-stage variables.

Path. For the path topology, the edges are $e = i(i+1)$, $i = 1, \ldots, |V| - 1$. The last q variables of block i are coupled with the first q variables of block $i + 1$: $I^{i(i+1),i} = (n - q + 1, \ldots, n)$, $I^{i(i+1),i+1} = (1, \ldots, q)$. Hence, for each $p = 1, \ldots, q$, the equality is $x^i_{n-q+p} = x_p^{i+1}$. This convention makes the right boundary of one block coincide with the left boundary of the next block.

Binary tree. For the binary-tree topology, with the natural ordering of nodes, each parent block i is connected to its children $2i$ and $2i + 1$, whenever these indices are present. For each parent–

child edge $e = ij$, the last q variables of the parent are coupled with the first q variables of the child: $I^{ij,i} = (n - q + 1, \ldots, n)$, $I^{ij,j} = (1, \ldots, q)$. Thus, if a block has two children, the same last q parent variables are coupled to both children. This produces a branching analogue of the path construction found in multi-stage stochastic programming problems.

Random tree. For the random-tree topology, the block tree is sampled randomly on a prescribed number of blocks. For each sampled edge $e = ij$, we choose q distinct local indices from block i and q distinct local indices from block j, uniformly at random and independently across the two endpoints, i.e., $I^{e,i} \subseteq U_i$, $I^{e,j} \subseteq U_j$, $|I^{e,i}| = |I^{e,j}| = q$. The order of the sampled lists determines the pairing of the equality constraints. The sampling is performed separately for each edge, so a local variable of a high-degree block may be selected in more than one incident coupling edge.

6.3 Instance generation

All computational experiments use instances with $|V| = 15$ blocks. Each block contains 100 local binary variables. For both the stable-set and dominating-set families, the local graph in each block is generated by sampling 500 edges uniformly without replacement from the complete graph on 100 nodes. Thus, each local graph is a sparse random graph with the same number of vertices and edges. The random seed of each block is derived from the global instance seed and the block index, so that different blocks are generated independently but reproducibly.

For each problem family, we consider three coupling sizes $q \in \{20, 30, 40\}$. For each value of q, we generate instances on the four deterministic or random block-tree topologies described above: star, path, binary tree, and random tree. In addition, we generate stochastic variants for the star and binary-tree topologies.

The non-stochastic instances use the same objective scaling in all blocks. In the stochastic variants, the objective coefficients are scaled to mimic repeated or inherited decisions across stages. For the star topology, the central block is assigned weight $|V| - 1$, while every leaf block has weight 1. For the binary-tree topology, the weight of a block depends on its depth. With $|V| = 15$, the tree has four levels; a block at level t receives weight 2^{4-t}, so the root has weight 8, its children have weight 4, the next level has weight 2, and the leaves have weight 1.

We use five independent seeds for each combination of problem family, topology, coupling size, and stochastic flag. The stochastic flag is only used for the star and binary-tree topologies. Thus, for each problem family and each coupling size q, the test set contains four non-stochastic topology classes and two stochastic topology classes.

6.4 Computational setup

All algorithms were implemented in Python 3.13 using Gurobi 13.0.1 as the MIP solver. The experiments were run on Amazon EC2 `c8a.4xlarge` instances with 16 vCPUs and 32 GB of RAM. Each run was assigned a time limit of 1800 seconds. The monolithic model had an additional 26 GB soft memory limit to avoid crashing. Instances that reached either of these limits were tagged with status `TimeLimit` and `MemoryLimit`, respectively.

The experiments solve one benchmark instance at a time. For the CRG algorithms, the pricing phase exploits the block structure by solving the 15 block pricing subproblems in parallel, using one Gurobi thread per pricing model. The monolithic formulation is solved with 16 Gurobi threads. The auxiliary separation MIPs used by the Monomial, Reflected, and Generalized strategies are also solved with 16 Gurobi threads.

Each CRG run is initialized from a short monolithic solve. Specifically, the monolithic formulation is solved with a deterministic Gurobi work limit of 10, which suffices to find a feasible solution. Its restriction to each block is used to create one initial column per block, and its objective value initializes the primal bound. We also solve the LP relaxation of the monolithic model and use its objective value as the initial dual bound.

The restricted master problem is solved as a linear program during column generation. A new column is added when its reduced cost is smaller than $-\varepsilon_{\text{price}} = -10^{-4}$. After column generation terminates for the current set of generated encoding rows, the current master solution is aggregated by boundary signatures and the selected separation strategy is applied on each block-tree edge. The default row-separation tolerance is $\varepsilon_{\text{cut}} = 10^{-6}$ for the Vertex, Monomial, Reflected, and Generalized strategies.

The number of generated rows per separation call is controlled by a factor parameter. For the Vertex strategy, we use factor 1.0, so at most q violated signature equalities are added per edge and separation call, where q is the number of coupling equalities per edge. For the Monomial, Reflected, and Generalized strategies, we use factor 0.5; the corresponding auxiliary separation MIP stores up to $0.5q$ improving solutions from the Gurobi solution pool in each orientation of the edge.

We also consider a hybrid CRG strategy that combines the Monomial and Vertex separation mechanisms. The algorithm starts with the Monomial strategy, which searches for violated monomial equalities through the auxiliary separation problem. This phase is used for the first five row-generation rounds, adding up to $0.5q$ M-type rows per edge and orientation in each separation call. After this initial phase, the algorithm switches permanently to the Vertex strategy, which separates violated signature equalities and adds up to q V-type rows per edge and separation call. Rows generated during the Monomial phase remain active after the switch and continue to contribute dual terms to the pricing problems.

The CRG algorithm stops with status `Optimal` when no columns and no rows are added. It also stops early with status `Gap_Closed` when the relative gap between the current primal and dual bounds falls below 10^{-6}. Because the tested objectives are integer-valued, we also declare convergence with status `Integer_Gap_Closed` when the primal and dual bounds have the same integer ceiling up to a tolerance of 10^{-6}. After each row-generation round, a five-second MIP heuristic is run on the current master with binary column variables. At the end of the CRG run, the final generated master is also solved with binary column variables to update the best primal bound.

For the star topology, we also implemented two benchmark decomposition methods: the integer L-shaped method [25, 3] and scenario decomposition [1]. The integer L-shaped master problem is solved with 16 Gurobi threads, while its 14 scenario subproblems are solved in parallel using one Gurobi thread each. Scenario decomposition solves 15 block subproblems in parallel, again using one Gurobi thread per subproblem. Thus, both implementations use the available threads in a manner comparable to the CRG algorithms, while respecting the different subproblem structure of each method.

7 Computational results

This section evaluates the computational behavior of the proposed CRG variants on the stable-set and dominating-set test beds. The benchmark contains 180 instances, 90 for each problem. Each instance is specified by a decomposition topology, a coupling level, a stochastic flag when applicable, and a random seed. The common benchmark compares the five CRG variants against the monolithic formulation on all instances. The integer L-shaped and scenario-decomposition

baselines are reported separately because they are available only for the star topology.

We use the following status convention throughout the section. Runs with status `Optimal`, `Integer_Gap_Closed`, or `Gap_Closed` are treated as solved, and their final gap is set to zero. Runs with status `TimeLimit` or `MemoryLimit` are treated as unsolved and assigned a censored time limit of 1800 seconds. The reported average gap, denoted by *Gap*, therefore averages the final gaps after setting solved runs to zero. The column *Unsolved gap* reports the average gap restricted to runs that did not close. All gaps are reported in percentage points.

The primary ordering criterion is lexicographic: number of solved runs, average censored time, and final gap. This criterion is used to order the solvers within each problem in Table 1. Under this ordering, a method that closes more runs is preferred even if another method produces a smaller average residual gap on the runs that remain unsolved.

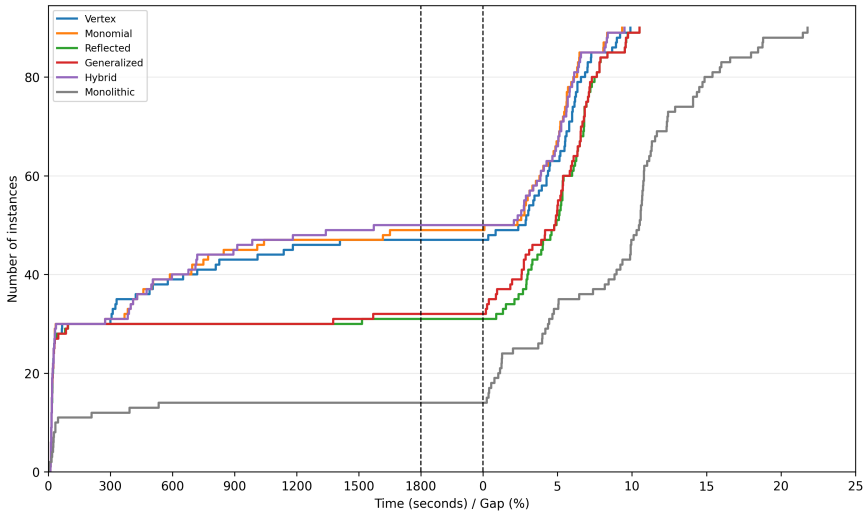
Table 1: Common benchmark over all topologies.

Problem	Solver	Solved	Time (s)	Time solved (s)	Gap (%)	Unsolved gap (%)
Dominating set	Vertex	19/90	1607	888	6.22	7.88
Dominating set	Hybrid	14/90	1682	1044	5.96	7.06
Dominating set	Monomial	13/90	1683	990	6.04	7.06
Dominating set	Generalized	4/90	1758	849	7.84	8.21
Dominating set	Reflected	4/90	1759	886	7.95	8.32
Dominating set	Monolithic	0/90	1800	–	13.52	13.52
Stable set	Hybrid	50/90	963	293	2.31	5.20
Stable set	Monomial	49/90	975	285	2.30	5.06
Stable set	Vertex	47/90	993	255	2.59	5.41
Stable set	Generalized	32/90	1201	115	3.51	5.44
Stable set	Reflected	31/90	1205	72	3.71	5.66
Stable set	Monolithic	14/90	1535	100	8.11	9.60

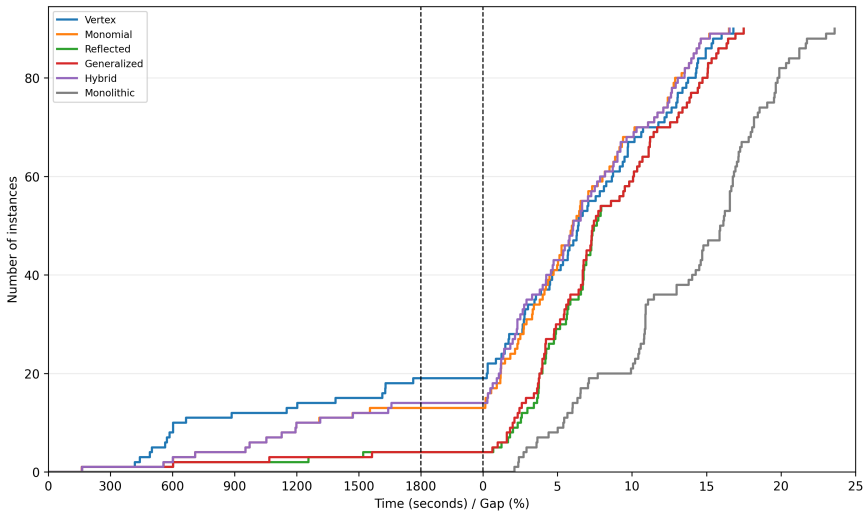
Table 1 shows a clear separation between the two problem classes. The stable-set instances are substantially easier: the best methods close slightly more than half of the runs, whereas the best dominating-set method closes only 19 out of 90 runs. For the stable-set benchmark, Hybrid has the best lexicographic performance, followed closely by Monomial and Vertex. Monomial obtains the smallest average gap, but Hybrid closes one additional run and has the smallest average censored time. For the dominating-set benchmark, Vertex is the best method under the lexicographic criterion because it closes the largest number of runs and has the smallest average censored time. However, Hybrid and Monomial provide stronger residual bounds on the unsolved runs. Thus, Vertex is the most effective method when the objective is to close runs, while Hybrid and Monomial are preferable when the quality of the final bound is the main criterion.

The monolithic formulation is not competitive on this benchmark. It closes only 14 stable-set runs and no dominating-set runs, and it has the largest average gaps in both problem classes. Its poor behavior is especially pronounced for the dominating-set instances, where all monolithic runs reach the time or memory limit.

Figure 1 gives the distributional view behind the aggregate statistics. The left side of each profile orders solved instances by running time, whereas the right side orders unsolved instances by final gap. In the stable-set profile (Figure 1a), the CRG variants dominate the monolithic baseline by closing many more instances and by producing smaller residual gaps when they do not close. The relative ordering among Hybrid, Monomial, and Vertex is close, which is consistent with Table 1. In the dominating-set profile (Figure 1b), the separation between Vertex and the other CRG variants



(a) Stable set



(b) Dominating set

Figure 1: Common performance profiles. The left side of each panel orders solved instances by running time; the right side orders unsolved instances by final gap.

is mainly on the solved side of the plot, while Hybrid and Monomial remain competitive on the gap side. This confirms that no single CRG variant dominates the others uniformly: the best method depends on whether the priority is closing instances or obtaining the strongest residual bound.

7.1 Iteration counts and convergence

Table 2 reports average iteration counts for the five CRG variants on the common benchmark. Outer iterations correspond to rounds in which the master is strengthened by newly generated cuts. Inner iterations correspond to pricing iterations and therefore capture most of the repeated subproblem work. The ratio *Inner/outer* gives a coarse measure of how much pricing work is performed per outer round.

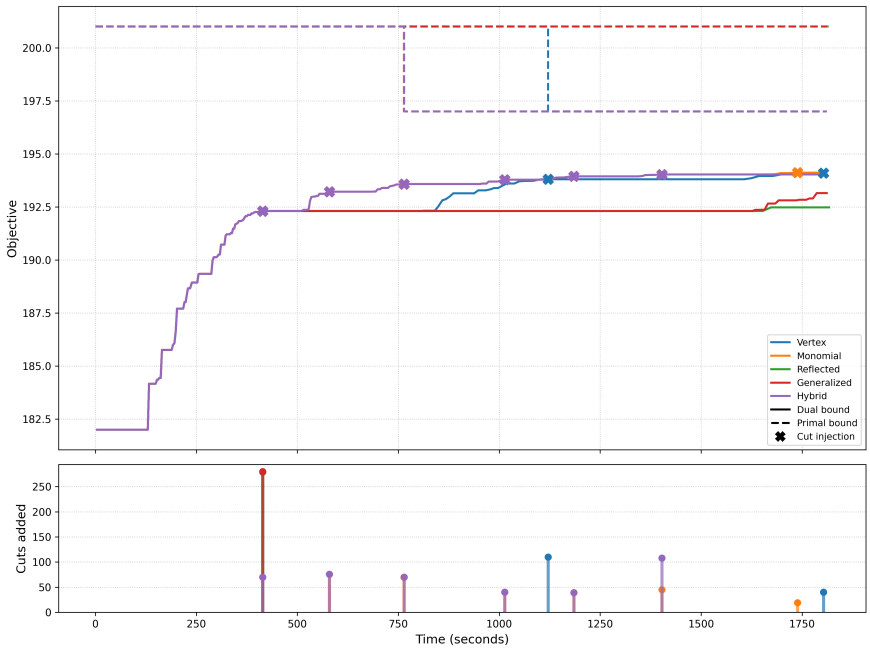
Table 2: Average CRG iteration counts.

Problem	Solver	Solved	Outer it.	Inner it.	Inner/outer	Cuts
Dominating set	Vertex	19/90	3.0	365.9	129.0	788
Dominating set	Hybrid	14/90	6.3	490.8	79.6	516
Dominating set	Monomial	13/90	6.9	506.2	76.3	325
Dominating set	Generalized	4/90	2.1	405.8	198.2	400
Dominating set	Reflected	4/90	2.0	396.3	196.4	390
Stable set	Hybrid	50/90	6.2	408.3	60.7	976
Stable set	Monomial	49/90	7.7	444.8	55.1	431
Stable set	Vertex	47/90	4.4	345.0	73.5	1338
Stable set	Generalized	32/90	2.7	337.5	118.2	566
Stable set	Reflected	31/90	2.5	333.4	123.9	509

The iteration statistics reveal two complementary patterns. First, Vertex usually uses fewer outer iterations than Monomial and Hybrid. On the dominating-set benchmark, Vertex performs only 3.0 outer iterations on average, compared with 6.3 for Hybrid and 6.9 for Monomial. Second, Monomial and Hybrid spread the work over more outer rounds and have lower inner-per-outer ratios. This behavior is consistent with their stronger residual gaps on unsolved dominating-set instances. Generalized and Reflected complete relatively few outer rounds that require many inner iterations per outer round and do not obtain comparable closure rates or gaps.

Figure 2 illustrates these iteration patterns on two binary-tree instances. The top panel shows a stable-set instance with stochastic coupling, and the bottom panel shows a deterministic dominating-set instance with the same topology and coupling level. In each panel, solid curves report the dual bound, dashed step curves report the incumbent primal bound, markers indicate the end of outer iterations, and the lower subpanel records the cuts added at those rounds. The stable-set instance (Figure 2a; maximization form) is a nontrivial case in which Vertex closes first, Monomial and Hybrid close later, and Generalized and Reflected reach the time limit. The dominating-set instance (Figure 2b; minimization form) is harder: no CRG method closes, so the comparison is driven by the final residual gap. The trajectories show how the dual bound evolves between cut-injection rounds and how primal improvements occur only intermittently. In both cases, the methods that generate the most cuts are not necessarily the methods with the best final outcome, which motivates a separate look at where and how many cuts are generated.

(a) Stable set



(b) Dominating set

Figure 2: Representative convergence trajectories.

7.2 Effect of topology and coupling

The aggregate comparison hides a strong dependence on the structure of the instance. Table 3 reports the effect of topology and coupling on the common benchmark. The star topology is the easiest topology for both problems, but the effect is much stronger for the stable-set instances. On the stable-set benchmark, the common solvers (the five CRG variants and the monolithic formulation) close 164 out of 180 star runs, whereas the non-star topologies are much harder. On the dominating-set benchmark, even the star topology remains difficult: only 35 out of 180 common solver runs close.

Coupling is the most visible difficulty driver. For the dominating-set problem, increasing the coupling level from 20 to 30 reduces the closure rate from 47/180 to 6/180, and coupling level 40 leaves only one closed run among 180. The stable-set instances are less sensitive in terms of closure counts because the star instances remain easy, but the average gap increases monotonically with the coupling level.

The stochastic flag has a milder effect than topology or coupling. Restricting the comparison to the topologies where both deterministic and stochastic variants are present, stochasticity is nearly neutral for the stable-set problem. For the dominating-set problem, stochastic instances are easier on average, mostly because the deterministic binary-tree cases are particularly hard.

Table 3: Effect of topology, coupling, and the stochastic flag on the common benchmark.

Factor	Problem	Value	Solved	Time (s)	Gap (%)	Unsolved gap (%)
Topology	Dominating set	star	35/180	1598	3.87	4.80
Topology	Dominating set	path	10/90	1752	10.59	11.92
Topology	Dominating set	binary tree	6/180	1783	9.36	9.69
Topology	Dominating set	random tree	3/90	1776	10.47	10.83
Topology	Stable set	star	164/180	186	0.17	1.91
Topology	Stable set	binary tree	37/180	1578	5.33	6.71
Topology	Stable set	path	15/90	1626	5.69	6.82
Topology	Stable set	random tree	7/90	1719	5.84	6.33
Coupling	Dominating set	20	47/180	1569	4.42	5.98
Coupling	Dominating set	30	6/180	1777	7.72	7.98
Coupling	Dominating set	40	1/180	1799	11.63	11.69
Coupling	Stable set	20	105/180	961	1.88	4.50
Coupling	Stable set	30	58/180	1265	3.93	5.80
Coupling	Stable set	40	60/180	1210	5.46	8.19
Stochastic	Dominating set	true	33/180	1624	5.87	7.18
Stochastic	Dominating set	false	8/180	1757	7.36	7.71
Stochastic	Stable set	true	103/180	854	2.78	6.49
Stochastic	Stable set	false	98/180	910	2.72	5.98

7.3 Star topology and specialized baselines

The integer L-shaped and scenario-decomposition baselines are meaningful only on the star topology. Therefore, they are excluded from the common performance profiles and reported separately in Table 4. On stable-set star instances, all CRG variants close all runs, but the two specialized baselines are substantially faster. Integer L-shaped closes all 30 runs in 4.6 seconds on average,

and scenario decomposition closes all 30 runs in 7.0 seconds on average. The fastest CRG methods require about 20 seconds on average.

The same qualitative conclusion does not fully carry over to the dominating-set star instances. Integer L-shaped remains the strongest method, closing 20 out of 30 runs. Scenario decomposition closes fewer runs than Integer L-shaped, although its average gap is slightly smaller. Among the CRG methods, Vertex is the strongest by the lexicographic criterion, while Hybrid and Monomial are competitive in terms of final gap.

Overall, these comparisons are encouraging for CRG: although it does not exploit the specialized master–recourse structure available on star instances, its performance remains competitive with methods designed specifically for that setting, particularly on the harder dominating-set instances.

Table 4: Star-topology comparison with specialized baselines.

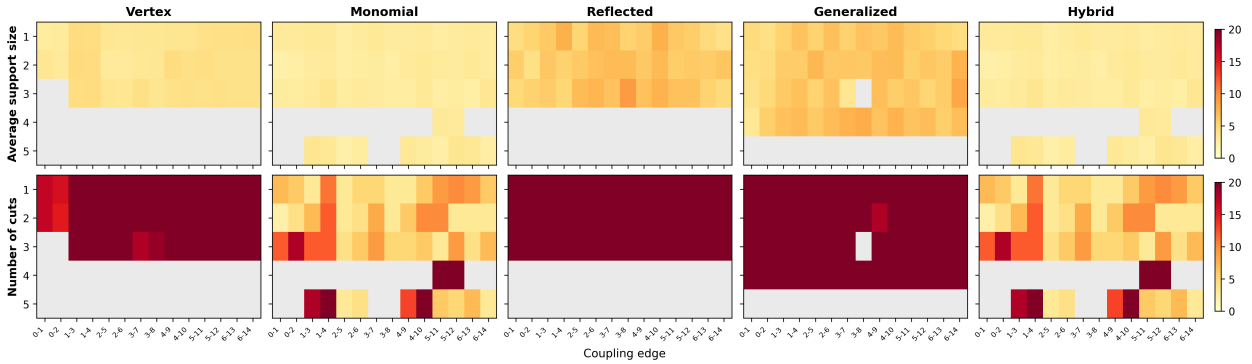
Problem	Solver	Solved	Time (s)	Gap (%)	Unsolved gap (%)
Dominating set	Integer L-shaped	20/30	740	2.08	6.24
Dominating set	Scenario decomp.	12/30	1298	1.97	3.28
Dominating set	Vertex	11/30	1327	2.50	3.95
Dominating set	Hybrid	9/30	1550	2.46	3.52
Dominating set	Monomial	7/30	1560	2.63	3.44
Dominating set	Generalized	4/30	1673	4.01	4.62
Dominating set	Reflected	4/30	1678	4.02	4.64
Dominating set	Monolithic	0/30	1800	7.58	7.58
Stable set	Integer L-shaped	30/30	5	0.00	–
Stable set	Scenario decomp.	30/30	7	0.00	–
Stable set	Monomial	30/30	20	0.00	–
Stable set	Hybrid	30/30	20	0.00	–
Stable set	Vertex	30/30	22	0.00	–
Stable set	Reflected	30/30	24	0.00	–
Stable set	Generalized	30/30	24	0.00	–
Stable set	Monolithic	14/30	1006	1.02	1.91

7.4 Cut-generation behavior

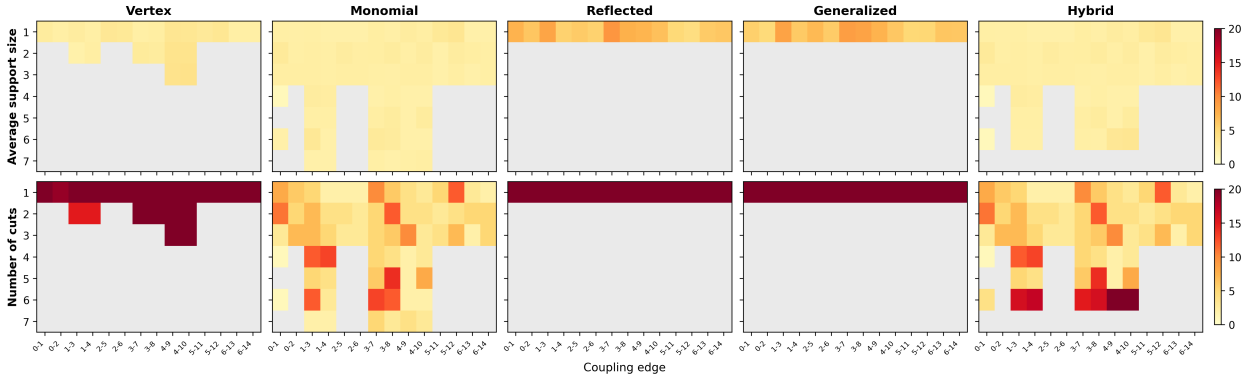
The preceding tables and performance profiles compare the final outcomes of the algorithms. Figure 3 instead illustrates how the CRG variants generate the rows used to enforce consistency between adjacent blocks. Each row of a heatmap corresponds to an outer iteration and each column to a coupling edge. The top panels report the average support size of the signatures generated on each edge, whereas the bottom panels report the corresponding number of cuts.

Figure 3b shows a representative hard dominating-set instance on a binary tree with coupling level 20. None of the CRG variants closes this instance, so their final residual gaps provide the relevant secondary comparison. Generalized and Reflected generate many relatively dense cuts in their first outer iterations, but this activity does not translate into a comparably strong final bound. Monomial and Hybrid distribute their cut generation over more outer iterations and finish with smaller residual gaps. Vertex is more aggressive and closes more dominating-set instances overall, although this example also illustrates that the number of closed instances and the quality of the residual bound need not induce the same ranking.

Figure 3a presents a stable-set instance for which the algorithms exhibit different closing behavior. Vertex closes first, followed by Monomial and Hybrid, whereas Generalized and Reflected reach



(a) Stable set, binary tree, $q = 20$, stochastic



(b) Dominating set, binary tree, $q = 20$, deterministic

Figure 3: Cuts and average signature support size per outer iteration and coupling edge.

the time limit. The latter two methods again generate a large number of dense signatures early in the solution process. Thus, the heatmaps suggest that generating more cuts, or cuts with larger supports, does not by itself produce faster convergence. The structural diversity of the generated signatures also appears to be relevant.

To examine this issue systematically, we use the signatures recorded in the cut-generation logs. Recall from Section 6.2 that every coupling edge e carries q pairs of boundary variables, indexed by $Q^e = \{1, \dots, q\}$, and from Section 4 that a generated Monomial or Reflected row is specified by a nonempty support $S \subseteq Q^e$, a Generalized row by a pair of disjoint sets $S^+, S^- \subseteq Q^e$, and a Vertex row by a full assignment $\mathbf{s} \in \{0,1\}^{Q^e}$. We call this defining data, together with the edge, the *signature* of the row, and we refer to the elements of S^+ and S^- as the positive and negative *literals* of a Generalized row. The diversity analysis below concerns the three set-based encodings.

The *literal set* of a generated row h is $\mathcal{L}(h) := S$ for a Monomial or Reflected row with support S, and $\mathcal{L}(h) := \{(p,+) : p \in S^+\} \cup \{(p,-) : p \in S^-\}$ for a Generalized row, so that Monomial and Reflected literal sets are unsigned while Generalized literal sets retain signs. The *degree* of a row is $|\mathcal{L}(h)|$ and its *density* is $|\mathcal{L}(h)|/q$. For two rows h and h' generated on the same edge, we measure structural dissimilarity through the Jaccard index [35]

$$J(h,h') := 1 - \frac{|\mathcal{L}(h) \cap \mathcal{L}(h')|}{|\mathcal{L}(h) \cup \mathcal{L}(h')|} \in [0,1].$$

A pair is classified as *nearly duplicate* when $J(h,h') \le 0.2$, or equivalently when at least 80% of their combined literals coincide. A pair is *nested* when one of its literal sets is contained in the

Table 5: Signature diversity and effective row-budget utilization in the first cut-producing outer iteration.

Problem	Encoding	Budget share	Density	Jaccard	Near dup.	Nested
Stable set	Monomial	0.196	0.094	0.705	0.015	0.193
Stable set	Reflected	0.873	0.257	0.544	0.178	0.230
Stable set	Generalized	0.887	0.249	0.571	0.163	0.210
Dominating set	Monomial	0.140	0.077	0.738	0.002	0.127
Dominating set	Reflected	0.907	0.316	0.531	0.229	0.211
Dominating set	Generalized	0.911	0.311	0.537	0.219	0.203

other.

Table 5 summarizes these statistics for the first outer iteration in which each encoding generates cuts. Recall that the separation problem is solved in both orientations of each coupling edge and may produce up to $0.5q$ rows per orientation, giving a total per-edge row budget of q. For each active instance–edge pair, "Budget share" is the number of generated signatures divided by q, while "Density" is the average signature support size divided by q. The remaining statistics are averages over signatures generated on the edge. These edge-level quantities are then averaged over active edges within each instance and finally across instances. This hierarchical aggregation gives equal weight to each instance, while normalizing by q makes the budget and density columns comparable across coupling levels. An edge contributes only if it generates at least one signature.

The differences are substantial. On an active edge in the first iteration, Monomial uses on average only 19.6% of the available row budget for stable set and 14.0% for dominating set. Reflected and Generalized use between 87.3% and 91.1%, indicating that their separation procedures typically return close to the maximum permitted number of rows. Their signatures are also much denser: the average Monomial density is below 0.10, whereas the average density of Reflected and Generalized ranges from 0.25 to 0.32.

Despite using a much smaller fraction of the row budget and producing sparser signatures, Monomial exhibits a substantially larger average pairwise Jaccard distance, 0.71–0.74 compared with 0.53–0.57. Moreover, its average fraction of nearly duplicate pairs is below 2% for both problem classes, whereas between 16% and 23% of the Reflected and Generalized pairs are nearly duplicate. Thus, the additional rows produced by Reflected and Generalized are not proportionally more diverse; their high budget utilization is associated with denser and more mutually similar signatures.

The nesting statistic provides more qualified evidence. For dominating set, nested pairs are less frequent under Monomial, with a fraction of 0.13 compared with 0.20–0.21. For stable set, the differences are more modest, with fractions between 0.19 and 0.23. The most consistent distinction is therefore the combination of higher budget utilization, higher density, smaller pairwise Jaccard distance, and a larger fraction of nearly duplicate pairs under Reflected and Generalized.

These observations are consistent with the behavior of the encoding functions on sparse columns. For Monomial, the set of columns on which $\phi_S^{M,e}$ evaluates to one can only shrink when a coordinate is added to S. If the columns are sparse, high-degree monomials tend to evaluate to zero on almost every column in both adjacent blocks. Such rows are unlikely to exhibit a large disagreement, so useful Monomial supports tend to be small.

Reflected applies the same construction to the complements of the shared variables. When the original columns are sparse, their complements are dense. Thus, a coordinate may be added to a Reflected support without changing the evaluation of the row on any active column: if $x_p^{e,i,k} = 0$

Table 6: Overlap between Generalized and Reflected signatures.

Problem	Instances	Negative only	Exact match	Nearest dist.	Nearest neg.
Stable set	74	0.736	0.682	0.145	0.055
Dominating set	77	0.839	0.574	0.129	0.060

for every active column $k \in \mathcal{K}^i$ with $\phi_S^{\mathrm{R},e}(\mathbf{x}^{e,i,k}) = 1$, then $\phi_{S\cup\{p\}}^{\mathrm{R},e}$ and $\phi_S^{\mathrm{R},e}$ coincide on every active column of the block. The separation problem can therefore admit several high-degree supports that are distinct as sets but induce identical or nearly identical rows on the current restricted master. The Generalized family contains the Reflected family as the special case $S^+ = \emptyset$ and may inherit this behavior.

The comparison between Generalized and Reflected signatures supports this interpretation. Table 6 compares the rows generated by the two methods on the same instance, coupling edge, and first outer iteration. A Generalized signature is called *negative-only* when $S^+ = \emptyset$ and $S^- \neq \emptyset$; its unsigned support is then S^-. The column "Negative-only" reports the fraction of Generalized signatures of this form, while "Exact match" reports the fraction of negative-only signatures whose unsigned support coincides with the support of a Reflected row generated on the same edge. The last two columns report the mean Jaccard distance from each unsigned Generalized support to its closest Reflected support, first over all Generalized signatures and then over the negative-only signatures. Entries are averaged over signatures within each edge, over edges within each instance, and finally across instances. An edge contributes to a column only when the corresponding quantity is defined.

The fraction of first-iteration Generalized signatures that are negative-only is 0.736 for stable set and 0.839 for dominating set. Among the negative-only signatures, the mean fraction whose unsigned support exactly matches that of a Reflected row generated on the same edge is 0.682 and 0.574, respectively. More generally, negative-only signatures remain close to the Reflected supports: their mean nearest-neighbor Jaccard distance is 0.055 for stable set and 0.060 for dominating set, compared with 0.145 and 0.129 when all Generalized signatures are included. Thus, in the first iteration, Generalized behaves predominantly, though not exclusively, like Reflected at the level of unsigned supports; signatures containing positive literals tend to use supports farther from those generated by Reflected.

Overall, the stored signatures provide empirical support for the mechanism suggested by the heatmaps. Monomial generates fewer and sparser rows, but their signatures are more dispersed in literal space. Reflected and Generalized generate many dense rows concentrated around common patterns, and Generalized largely reproduces the negative-literal structure of Reflected. This helps explain why their high cut counts do not yield a corresponding improvement in convergence.

8 Discussion

This paper develops a computational realization of strengthened Lagrangian dual for tree-structured integer programs. The main idea is to work with the primal convex characterization of the strengthened dual and to generate both of its potentially exponential objects dynamically: feasible local solutions are generated as columns through blockwise pricing, while consistency constraints using nonlinear encoding functions are generated as rows through separation. With a complete exact encoding family, this gives an exact algorithm; moreover before convergence, completed outer iterations provide valid dual bounds. Thus, exactness does not require explicitly constructing the

exponentially large formulation apriori, and local optimization remains decomposed across the blocks throughout the algorithm.

Computationally, CRG consistently outperforms the monolithic formulation on stable-set and dominating-set benchmarks, closing more instances and producing stronger bounds, while applying without modification across star, path, binary-tree, and random-tree topologies. On the star instances, specialized integer L-shaped and scenario-decomposition methods are generally marginally faster, as expected from methods designed specifically for a master–recourse structure. Nevertheless, CRG remains competitive, particularly on the harder dominating-set instances, despite making no use of a distinguished first-stage block or recourse value function. The intended role of the framework is therefore not to replace specialized methods when additional structure is available, but to provide a common decomposition mechanism when it is not.

The experiments suggest that interface size is an important driver of difficulty. Performance deteriorates as the number of shared variables increases, especially for dominating set. This is consistent with the formulation: local complexity is handled independently inside the pricing oracles, whereas the master must progressively recover the information lost by separating the blocks across their shared boundaries.

This leads to perhaps the most important conclusion concerning the choice of encoding. The theoretical results show that Generalized encodings can require exponentially fewer rows than Vertex, Monomial, or Reflected encodings on suitable instances, while even the Generalized family can require exponentially many rows when a single problem-specific encoding suffices. The computations reveal a complementary phenomenon: greater expressive power does not necessarily translate into a better algorithm. Generalized and Reflected separation frequently produces many dense and closely related rows, whereas Monomial generates substantially fewer, sparser, and more diverse rows. Consequently, the theoretically richer Generalized family performs worse than the simpler Monomial and Vertex strategies on the present benchmarks. What matters computationally is therefore not only whether a family can eventually close the gap, but whether separation can identify a small collection of informative rows whose strengthening effect justifies the additional complexity they introduce into subsequent pricing problems.

These observations suggest several directions for future research. First, encoding selection itself should be adaptive. The Hybrid results already show that changing the encoding strategy during the computation can be beneficial. More sophisticated schemes could use the current boundary distributions, violations, dual multipliers, or the observed effectiveness and redundancy of previous rows to choose which encoding family to search and which rows to retain. Second, there is considerable potential in discovering problem-specific encodings. The parity example demonstrates that the difference between a generic family and an appropriate problem-specific encoding can be exponential.

More broadly, the results highlight the information exchanged across block boundaries as a key determinant of decomposition performance. Nonlinear edge encodings capture this information, but different encodings can require dramatically different numbers of constraints to recover exactness.

References

[1] Shabbir Ahmed. A scenario decomposition algorithm for 0–1 stochastic programs. *Operations Research Letters*, 41(6):565–569, 2013.

[2] Shabbir Ahmed, Filipe Goulart Cabral, and Bernardo Freitas Paulo da Costa. Stochastic Lipschitz dynamic programming. *Mathematical Programming*, 191(2):755–793, 2022.

[3] Gustavo Angulo, Shabbir Ahmed, and Santanu S. Dey. Improving the integer L-shaped method. *INFORMS Journal on Computing*, 28(3):483–499, 2016.

[4] Kurt M. Anstreicher and Laurence A. Wolsey. Two "well-known" properties of subgradient optimization. *Mathematical Programming*, 120(1):213–220, 2009.

[5] 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. *Operations Research*, 46(3):316–329, 1998.

[6] César Beltran, Claude Tadonki, and Jean-Philippe Vial. *Semi-Lagrangian Relaxation*. HEC Genève, Geneva, Switzerland, 2004.

[7] Jacobus F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962.

[8] Martin Bergner, Alberto Caprara, Alberto Ceselli, Fabio Furini, Marco E. Lübbecke, Enrico Malaguti, and Emiliano Traversi. Automatic Dantzig–Wolfe reformulation of mixed integer programs. *Mathematical Programming*, 149(1–2):391–424, 2015.

[9] Umberto Bertelè and Francesco Brioschi. On non-serial dynamic programming. *Journal of Combinatorial Theory, Series A*, 14(2):137–148, 1973.

[10] Daniel Bienstock and Gonzalo Muñoz. LP formulations for polynomial optimization problems. *SIAM Journal on Optimization*, 28(2):1121–1150, 2018.

[11] John R. Birge and François Louveaux. *Introduction to Stochastic Programming*. Springer, New York, 2nd edition, 2011.

[12] Merve Bodur, Sanjeeb Dash, Oktay Günlük, and James Luedtke. Strengthened Benders cuts for stochastic integer programs with continuous recourse. *INFORMS Journal on Computing*, 29(1):77–91, 2017.

[13] Natashia Boland, Jeffrey Christiansen, Brian Dandurand, Andrew Eberhard, Jeff Linderoth, James Luedtke, and Fabricio Oliveira. Combining progressive hedging with a Frank–Wolfe method to compute Lagrangian dual bounds in stochastic mixed-integer programming. *SIAM Journal on Optimization*, 28(2):1312–1336, 2018.

[14] Claus C. Carøe and Rüdiger Schultz. Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24(1–2):37–45, 1999.

[15] Rui Chen and James Luedtke. On generating Lagrangian cuts for two-stage stochastic integer programs. *INFORMS Journal on Computing*, 34(4):2332–2349, 2022.

[16] Diego Cifuentes, Santanu S. Dey, and Jingye Xu. Lagrangian dual for integer optimization with zero duality gap that admits decomposition. In *Integer Programming and Combinatorial Optimization (IPCO 2025)*, Lecture Notes in Computer Science, pages 184–198. Springer, 2025.

[17] George B. Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960.

[18] Haoyun Deng and Weijun Xie. On the ReLU Lagrangian cuts for stochastic mixed integer programming. *arXiv preprint arXiv:2411.01229*, 2024.

[19] Santanu S. Dey, Marco Molinaro, and Qianyi Wang. Analysis of sparse cutting planes for sparse MILPs with applications to stochastic MILPs. *Mathematics of Operations Research*, 43(1):304–332, 2018.

[20] Yuri Faenza, Gonzalo Muñoz, and Sebastian Pokutta. New limits of treewidth-based tractability in optimization. *Mathematical Programming*, 191(2):559–594, 2022.

[21] Mohammad Javad Feizollahi, Shabbir Ahmed, and Andy Sun. Exact augmented Lagrangian duality for mixed integer linear programming. *Mathematical Programming*, 161(1–2):365–387, 2017.

[22] Arthur M. Geoffrion. Lagrangean relaxation for integer programming. In *Approaches to Integer Programming*, pages 82–114. Springer, Amsterdam, Netherlands, 2009.

[23] Xiaoyi Gu, Shabbir Ahmed, and Santanu S. Dey. Exact augmented Lagrangian duality for mixed integer quadratic programming. *SIAM Journal on Optimization*, 30(1):781–797, 2020.

[24] Monique Guignard and Siwhan Kim. Lagrangean decomposition: A model yielding stronger Lagrangean bounds. *Mathematical Programming*, 39(2):215–228, 1987.

[25] Gilbert Laporte and François V. Louveaux. The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13(3):133–142, 1993.

[26] Jean B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.

[27] Marco E. Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

[28] Michel Minoux and Hacene Ouzia. DRL*: A hierarchy of strong block-decomposable linear relaxations for 0–1 MIPs. *Discrete Applied Mathematics*, 158(18):2031–2048, 2010.

[29] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, 1969.

[30] Neil Robertson and Paul D. Seymour. Graph minors. II. Algorithmic Aspects of Tree-Width. *Journal of Algorithms*, 7(3):309–322, 1986.

[31] R. Tyrrell Rockafellar and Roger J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16(1):119–147, 1991.

[32] Thomas Rothvoß. The Lasserre hierarchy in approximation algorithms. *Lecture Notes for the MAPSP*, pages 1–25, 2013.

[33] Hanif D. Sherali and Warren P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3(3):411–430, 1990.

[34] Kaizhao Sun, Mou Sun, and Wotao Yin. Decomposition methods for global solution of mixed-integer linear programs. *SIAM Journal on Optimization*, 34(2):1206–1235, 2024.

[35] Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, and Vipin Kumar. *Introduction to Data Mining*. Pearson, 2nd edition, 2019.

[36] Jikai Zou, Shabbir Ahmed, and Xu Andy Sun. Stochastic dual dynamic integer programming. *Mathematical Programming*, 175(1):461–502, 2019.

Statements and declarations

Competing Interests

The authors have no relevant financial or non-financial interests to disclose.

Data Availability

Code and results are available at `https://github.com/ganguloo/block-lagrangian`.