

Beyond Hand-Derived Inequalities: Decision Diagrams for Cut Generation in Binary Polynomial Optimization

Martin Cooper · Margarita Castro

Received: date / Accepted: date

Abstract We study cutting-plane generation for binary polynomial optimization (BPO), whose feasible region is the multilinear set of a hypergraph. Strong inequalities for this set—such as two-links, flowers, and odd β -cycles—are classically hand-derived for fixed support patterns. Instead, we propose a decision-diagram (DD) approach: for any chosen support, it separates a facet-defining cut in the local multilinear polytope and lifts it back to the original problem. We utilize a novel compact DD encoding based on a recursive formulation that represents only the vertex variables and implicitly encodes the hyperedge variables inside the state representation. From this DD encoding, we also obtain: (i) an extended formulation of the multilinear polytope, (ii) a width characterization via valid antichains that uncovers a new polynomially solvable class of hypergraphs, and (iii) a certificate for the facetness of the generated cuts. We explore three different support strategies for cut generation that reuse, expand, or partition local structures into section hypergraphs. Our empirical results show that our DD-based methodologies achieve a larger gap closure at the root node with fewer cuts than existing procedures and, in turn, markedly accelerate branch-and-bound procedures.

Keywords binary polynomial optimization · decision diagrams · cutting planes

Mathematics Subject Classification (2020) 90C09 · 90C10 · 90C57

M. Cooper
Department of Industrial and Systems Engineering, Pontificia Universidad Católica de Chile

M. Castro
Department of Industrial and Systems Engineering, Pontificia Universidad Católica de Chile
E-mail: margarita.castro@uc.cl

1 Introduction

Binary polynomial optimization (BPO) is a combinatorial problem that seeks a binary point that optimizes a multivariate polynomial function. The problem arises in many fields such as engineering, computer science, statistics, and physics [1, 6], and it generalizes well-studied special cases such as unconstrained binary quadratic optimization [27] and the maximum cut problem [2].

Formally, the problem can be encoded by a hypergraph $G = (V, E)$, in which each node $v \in V$ represents a binary variable (i.e., $z_v \in \{0, 1\}$) and each hyperedge $e \in E$ represents a monomial of the objective encoded by a binary variable (i.e., $z_e \in \{0, 1\}$). In what follows, we use bold notation $\mathbf{z}_v = (z_v)_{v \in V}$ and $\mathbf{z}_e = (z_e)_{e \in E}$ to refer to the vectors associated with vertex and hyperedge variables, respectively. Then, BPO minimizes a linear objective over the multilinear set $\mathcal{S}_G = \{(\mathbf{z}_v, \mathbf{z}_e) \in \{0, 1\}^{V \cup E} : z_e = \prod_{v \in e} z_v \ \forall e \in E\}$ of hypergraph G [17], that is,

$$\min \left\{ \sum_{v \in V} c_v z_v + \sum_{e \in E} c_e z_e : (\mathbf{z}_v, \mathbf{z}_e) \in \mathcal{S}_G \right\}, \quad (1)$$

where c_v and c_e are the costs associated with each vertex $v \in V$ and hyperedge $e \in E$, respectively.

This problem is commonly solved by linearizing $\mathcal{S}_G$ and strengthening the resulting relaxation with valid inequalities. Several important inequality families are tied to such hypergraph structures—including the two-link [13], flower [18], running-intersection [19], and odd β -cycle inequalities [16, 21]—which strengthen the standard linearization by capturing interactions among overlapping multilinear terms. At the same time, each family is hand-derived for a particular support pattern. This leaves open the question of how to generate strong inequalities once the relevant local interactions no longer fit a small catalog of named templates.

To address this, decision diagrams (DDs) offer a flexible alternative for generating such general cuts. A DD is a layered directed acyclic graph whose root-to-terminal paths encode the feasible solutions of a discrete set, compactly capturing discrete structures that a linear programming (LP) relaxation does not [5, 12, 25]. Building on early work that embedded DDs in a branch-and-cut framework [3, 4], Tjandraatmadja and van Hoeve [32] introduce *target cuts*: given a DD, they separate inequalities that are facet-defining for the convex hull of the points it encodes, so the strength of the cut is governed by how faithfully the DD models the relevant substructure. Other related DD-based schemes include an outer approximation procedure for non-linear problems [14] and the combinatorial cut-and-lift for binary problems [11], while specialized procedures have also been developed, for instance, to generate cuts for stochastic optimization [10, 28, 29].

This makes DDs an appealing engine for BPO: rather than deriving a valid inequality for each named support pattern, one can build a DD for a chosen support hypergraph and generate target cuts that expose its local interactions;

thus, yielding a single mechanism that is both stronger and more versatile than any fixed template. Moreover, we can show that the resulting cuts are facet-defining for the support hypergraph and, under certain conditions, facets of the original problem.

To generate such DD-based cuts, we present a novel compact DD encoding for BPO. Specifically, we present a recursive model over a given vertex order, in which vertex variables are explicitly represented as arcs in the diagram, while hyperedge variables are implicitly encoded within the problem’s state representation. Then, the DD is a compact state-space representation of such a recursive model, and the target cut procedure is adapted to account for the implicit hyperedge variables.

Moreover, we use the DD representation beyond cut generation to: (i) create a novel extended formulation for BPO, (ii) uncover new hypergraph structures that can be solved in polynomial time, and (iii) build certificates for facet-defining constraints. Specifically, we adapt the DD-based network flow formulation to obtain an extended formulation that accounts for both vertex and hyperedge variables. We also analyze the DD size (i.e., the node count) by characterizing the number of states in the recursive model via valid antichains. This allows us to prove that several known structures (e.g., flowers and cycles) can be represented by a DD of bounded size. Furthermore, we define a new family of hypergraphs that lead to polynomial-size DDs, and, as such, can be solved in polynomial time using, for example, the DD-based extended formulation. Lastly, we develop a novel procedure to determine whether a DD-based cut is facet-defining by computing its dimensionality over the support of the DD, and we use existing theoretical results [17] to prove that it is a facet in the original problem.

We instantiate this framework with three strategies for choosing the local support fed to the separator: a *fixed-support* baseline that reuses the supports of known literature inequalities, an *expanded-support* rule that grows a seed support by overlap, and a *partition-support* scheme that builds section hypergraphs from a vertex partition and is backed by zero-lifting guarantees. We evaluate these strategies on standard benchmarks from the literature (i.e., the LABS and VISION datasets), on which we observed a clear superiority of our procedures over existing methods. Specifically, expanded and, especially, partition supports close substantially more of the root gap with far fewer cuts—up to 92.4% of the gap on LABS and the full gap on VISION—and these stronger relaxations carry over to branch-and-bound (B&B), where the partition setting gives the best practical performance.

The remainder of the paper is as follows. Section 2 states the BPO linear formulation and the most commonly used cutting planes from the literature. Section 3 introduces the compact DD encoding of BPO and its extended formulation. Section 4 characterizes the size of the compact DD through valid antichains, analyzes the DD size for different hypergraph structures, and uncovers a new hypergraph family that can be solved in polynomial time. Section 5 describes the target-cut separation over our compact DD encoding, and Section 6 introduces support expansion and partition strategies. Section 7 de-

velops the dimension and facet-audit procedure. Lastly, Section 8 describes the computational experiments, and Section 9 concludes our work.

2 Binary polynomial optimization and existing cutting planes

We now present the integer linear programming (ILP) formulation commonly used in the literature to solve the BPO problem (1) and review existing cutting-plane approaches for tightening it. Recall that instances of BPO can be encoded by a hypergraph $G = (V, E)$, where we assume $|e| \geq 2$ for every $e \in E$, as linear terms are represented directly by the vertex variables. The standard linearization replaces the multilinear equalities encoded in $\mathcal{S}_G$ with two sets of linear inequalities, resulting in the following model [23, 24]:

$$\min \sum_{v \in V} c_v z_v + \sum_{e \in E} c_e z_e \quad (2a)$$

$$\text{s.t. } z_e \leq z_v \quad \forall e \in E, v \in e, \quad (2b)$$

$$z_e \geq \sum_{v \in e} z_v - |e| + 1 \quad \forall e \in E, \quad (2c)$$

$$z_v, z_e \in \{0, 1\} \quad \forall v \in V, e \in E. \quad (2d)$$

Note that the optimal value of problem (1) can be obtained by optimizing over the multilinear polytope, denoted as $MP_G = \text{conv}(\mathcal{S}_G)$, where $\text{conv}(\cdot)$ is the convex hull operator of a set. Similarly, model (2) optimizes over the same set, but by representing the multilinear set with linear inequalities. Relaxing (2d) with continuous variables yields the standard LP relaxation, denoted by MP_G^{LP} . This is the base relaxation used throughout this paper, which is later strengthened by cutting-plane algorithms.

2.1 Valid linear inequalities to strengthen MP_G^{LP}

It is well-known that MP_G^{LP} is a weak relaxation of MP_G , and the gap between them is driven by interactions among overlapping hyperedges [13, 17]. Thus, several works have identified special hypergraph structures to build valid inequalities that separate fractional points from MP_G . We now briefly review three families of such techniques that provide natural points of comparison for our proposed DD-based procedure. Moreover, we use such structures as a basis to construct more complex support hypergraphs for our own DD-based cuts (see Section 6). We refer the reader to the sources for all theoretical and algorithmic details.

Two-link inequalities [13]. A two-link is a pair of hyperedges $e, f \in E$ with $e \neq f$ and $e \cap f \neq \emptyset$. The corresponding inequality couples the two overlapping monomials $e, f \in E$ as follows:

$$\sum_{v \in f \setminus e} z_v + z_e - z_f \leq |f \setminus e|.$$

Flower inequalities [18]. A flower consists of a center hyperedge $f \in E$ and a nonempty set $T \subseteq E \setminus \{f\}$ of hyperedges denoted as petals. Each petal meets the center (i.e., $(e \cap f) \neq \emptyset \ \forall e \in T$) and the petal-center intersections are pairwise disjoint (i.e., $(e \cap f) \cap (e' \cap f) = \emptyset \ \forall e \neq e' \in T$). Then, the inequality is given by:

$$\sum_{v \in f \setminus \bigcup_{e \in T} e} z_v + \sum_{e \in T} z_e - z_f \leq \left| f \setminus \bigcup_{e \in T} e \right| + |T| - 1.$$

Intuitively, a flower generalizes a two-link by relating the center to several petals at once rather than one at a time. When the underlying support is γ -acyclic, the flower inequalities together with the standard linearization fully describe MP_G [18]. We note that running-intersection inequalities [19] are a strict generalization of the flower inequalities, differing only in the more general overlap patterns of (kite-free) β -acyclic supports. Therefore, we consider only flower inequalities in our empirical comparison.

Odd β -cycle inequalities [16]. Structurally, a β -cycle consists of distinct hyperedges $(e_1, \dots, e_k)$ and vertices $(v_1, \dots, v_k)$, indexed cyclically, such that $v_i \in e_i \cap e_{i+1}$ for every i , while nonconsecutive hyperedges in the sequence do not intersect. Odd β -cycle inequalities are associated with odd such cycles and require additional parity or sign data beyond this intersection pattern. We note that for our comparison, we use the regular odd β -cycle inequalities [16], not the simple odd β -cycle variant of Del Pia and Walter [21], which is a weakened version of the former.

3 Decision Diagrams for BPO

This section introduces the DD encoding used for BPO cut generation. In the context of integer and combinatorial optimization, DDs are graphical structures that encode the solution set of an optimization problem as paths. In what follows, we present the standard DD terminology [5, 12, 25].

Consider a combinatorial problem with n binary variables and feasible set $X_B \subseteq \{0, 1\}^n$. A binary DD $B = (N, A)$ is a layered directed acyclic graph that represents X_B as follows. Its node set is partitioned into layers $N = (N_1, \dots, N_{n+1})$, where $N_1 = \{r\}$ and $N_{n+1} = \{t\}$ contain the root and terminal node, respectively. Each transition from layer i to layer $i + 1$ corresponds to choosing the value of the i -th variable in the order given to the DD. Thus, an arc $a = (u, u') \in A$, with $u \in N_i$ and $u' \in N_{i+1}$, is labeled by a value $\text{val}(a) \in \{0, 1\}$. Each node has at most one outgoing arc with value 0 and at most one with value 1. Then, a root-terminal path $p = (a_1, \dots, a_n)$ encodes the binary vector $\mathbf{x}^p = (\text{val}(a_1), \dots, \text{val}(a_n)) \in \{0, 1\}^n$. Let $\mathcal{P}$ denote the set of all root-terminal paths; then we say that the DD is *exact* if $X_B = \{\mathbf{x}^p : p \in \mathcal{P}\}$, that is, each root-terminal path has a one-to-one correspondence with the solutions in X_B .

3.1 A compact DD encoding for BPO

We now introduce our compact DD representation of BPO. This DD encoding is based on a dynamic programming (DP) model for BPO that considers only vertex variables as state variables, with hyperedge variables represented implicitly in the states. The DP model leverages that z_e for $e \in E$ is completely defined by the assignment of its corresponding nodes since $z_e = \prod_{v \in e} z_v$.

We start by introducing some useful notation for the DP model. Consider a hypergraph $G = (V, E)$ that represents a BPO instance and a given vertex order $\sigma = (v_1, \dots, v_n)$, where $n = |V|$. For $i \in \{0, \dots, n\}$, let $U_0 = \emptyset$ and $U_i = \{v_1, \dots, v_i\}$, be the set of the first i vertex decisions based on σ . Then, we define the set of *active hyperedges* with respect to U_i as $\mathcal{A}_i = \{e \in E : e \cap U_i \neq \emptyset, e \not\subseteq U_i\}$, that is, the set of hyperedges that have some vertex in U_i , but not all of them. We say that a hyperedge $e \in E$ is *compatible* with the current partial assignment over the vertices in U_i if every assigned vertex of e has value one.

We define our DP model with n stages (i.e., one per vertex) such that the state variable at stage $i \in \{1, \dots, n\}$ corresponds to the vertex variable $z_{v_i} \in \{0, 1\}$ following the σ order. State s_i at stage $i \in \{1, \dots, n\}$ is a subset of active hyperedges $\mathcal{A}_i$ that are compatible with respect to the current partial assignment. Given $s_0 = \emptyset$ as the initial state, the transition function $\phi_i(s_{i-1}, z_{v_i})$ at stage $i \in \{1, \dots, n\}$ is given by

$$\phi_i(s_{i-1}, z_{v_i}) = \{e \in \mathcal{A}_i : (e \notin \mathcal{A}_{i-1} \text{ or } e \in s_{i-1}) \text{ and } (v_i \notin e \text{ or } z_{v_i} = 1)\}.$$

The first condition considers the previously active hyperedges and potentially newly active hyperedges at stage i , while the second condition updates compatibility after assigning v_i . Intuitively, this encoding keeps track of the $\mathbf{z}_e$ variables that can take the value one in each state (i.e., a compatible hyperedge) and ignores them if we have already assigned all vertices associated with a specific hyperedge. Formally, let $\mathcal{L}_i = \{e \in E : v_i \in e, e \subseteq U_i\}$ be the hyperedges whose last vertex in σ is v_i for every $i \in \{1, \dots, n\}$. For every arc that assigns z_{v_i} , a hyperedge $e \in \mathcal{L}_i$ has its value assigned: $z_e = 1$ if it is compatible before assigning v_i and $z_{v_i} = 1$; and $z_e = 0$ otherwise.

Given this DP model, the DD is built in a top-down fashion from the root (i.e., initial state s_0) by applying the transition function at each layer and merging nodes with equal states. Exactness with respect to $\mathcal{S}_G$ follows because the state keeps the needed information to determine which unfinished hyperedges can still attain value 1 under future assignments, and therefore to recover every vector $(\mathbf{z}_v, \mathbf{z}_e)$. See Example 1 for an illustrative example.

Example 1 Consider a BPO instance with vertices $\{v_1, v_2, v_3\}$ and hyperedges $e_1 = \{v_1, v_2\}$ and $e_2 = \{v_2, v_3\}$ and its associated compact DD as depicted in Figure 1. Nodes depict the states, solid black arcs are 1-arcs (i.e., $z_{v_i} = 1$) and dashed gray arcs are 0-arcs (i.e., $z_{v_i} = 0$). Given order $\sigma = (v_1, v_2, v_3)$, we have $U_1 = \{v_1\}$, $U_2 = \{v_1, v_2\}$, $U_3 = \{v_1, v_2, v_3\}$ and active hyperedge sets $\mathcal{A}_1 = \{e_1\}$, $\mathcal{A}_2 = \{e_2\}$, and $\mathcal{A}_3 = \emptyset$. Note that the states of the DD indeed

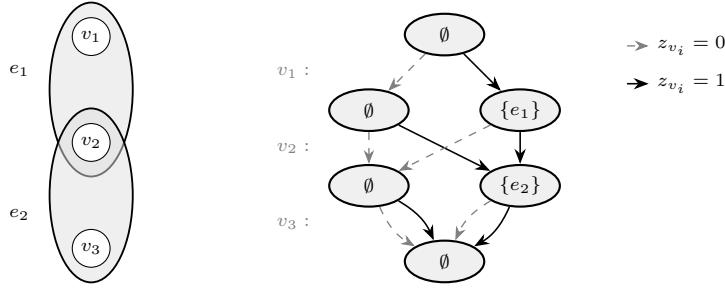


Fig. 1 BPO instance as a hypergraph (left) and its associated DD encoding (right).

encode the set of hyperedges $e \in E$ that could have $z_e = 1$ but have not yet been assigned. For instance, after assigning z_{v_1} , we have a state where either e_1 still has options of being selected (i.e., when $z_{v_1} = 1$) or not (i.e., when $z_{v_1} = 0$).

3.2 An extended formulation induced by the compact DD

A simple way to use an exact DD inside an LP formulation is through its network-flow representation [4]. This formulation assumes that every variable of the problem is explicitly represented in the DD (i.e., it corresponds to one of the decisions in each layer). In what follows, we adapt such a model to our compact DD encoding, in which hyperedge variables are implicitly represented.

Let $B = (N, A)$ be the compact DD associated with $G = (V, E)$ and assume vertex order $\sigma = (v_1, \dots, v_n)$. For an arc $a \in A$, let $s(a)$ and $t(a)$ denote its source and target nodes, respectively, and consider sets $A^{\text{out}}(u) = \{a \in A : s(a) = u\}$ and $A^{\text{in}}(u) = \{a \in A : t(a) = u\}$ to represent outgoing and incoming arcs of a node $u \in N$. Lastly, if $a \in A$ leaves layer N_i , then $\text{var}(a) = v_i$ is the vertex associated with arc a and $\text{val}(a) \in \{0, 1\}$ is its assigned value.

We now define the arcs that assign each variable (z_v, z_e) to the value one or zero. For a vertex $v \in V$, let $\Gamma_v^1 = \{a \in A : \text{var}(a) = v, \text{val}(a) = 1\}$ and $\Gamma_v^0 = \{a \in A : \text{var}(a) = v, \text{val}(a) = 0\}$ be the set of arcs associated with the $z_v = 1$ and $z_v = 0$ decisions, respectively. For hyperedge variables, recall that $\mathcal{L}_i$ is the set of hyperedges whose value can be deduced when v_i is assigned, for each $i \in \{1, \dots, n\}$. For each arc $a \in A^{\text{out}}(u)$ leaving a node $u \in N_i$ with state s_{i-1} , we define $\mathcal{E}_1(a) = \{e \in \mathcal{L}_i : e \in s_{i-1}, \text{val}(a) = 1\}$, and $\mathcal{E}_0(a) = \mathcal{L}_i \setminus \mathcal{E}_1(a)$ as the sets of hyperedges assigned value 1 and 0 on arc a , respectively. Lastly, for each $e \in E$, $\Gamma_e^1 = \{a \in A : e \in \mathcal{E}_1(a)\}$ and $\Gamma_e^0 = \{a \in A : e \in \mathcal{E}_0(a)\}$ are the set of arcs that define the z_e variable to 1 and 0, respectively.

To create the extended formulation for BPO based on the DD $B = (N, A)$, consider $y_a \geq 0$ a continuous flow variable associated with arc $a \in A$. Then,

the network flow model associated with B is:

$$\min \sum_{v \in V} c_v z_v + \sum_{e \in E} c_e z_e \quad (3a)$$

$$\text{s.t.} \quad \sum_{a \in A^{\text{out}}(u)} y_a - \sum_{a \in A^{\text{in}}(u)} y_a = 0 \quad \forall u \in N \setminus \{r, t\}, \quad (3b)$$

$$\sum_{a \in A^{\text{out}}(r)} y_a = \sum_{a \in A^{\text{in}}(t)} y_a = 1, \quad (3c)$$

$$z_j = \sum_{a \in \Gamma_j^1} y_a \quad \forall j \in V \cup E, \quad (3d)$$

$$y_a \geq 0 \quad \forall a \in A. \quad (3e)$$

Constraints (3b) and (3c) are the standard network-flow constraints over a DD, whose feasible flows are exactly the convex combinations of root-terminal paths; thus, they encode $\text{conv}(X_B)$. Constraints (3d) link the flow to the BPO variables: z_v accumulates the flow on the arcs that set $z_v = 1$, and z_e accumulates the flow on the arcs where the compact DD implicitly finalizes $z_e = 1$. Since every root-terminal path crosses exactly one arc of $\Gamma_j^1 \cup \Gamma_j^0$ for each $j \in V \cup E$, these identities are well defined, and the objective (3a) coincides with the BPO objective (2a). The validity of the model and the fact that the projection of (3b)–(3e) onto the $(\mathbf{z}_v, \mathbf{z}_e)$ variables equals MP_G follow directly from the compact DD encoding and Theorem 4.1 of [4].

4 DD size and special structures

This section analyzes the DD size and uncovers special structures for which its size remains bounded. In what follows, we use the width of a DD $B = (N, A)$ to measure its size. Specifically, the width of layer N_i for any $i \in \{1, \dots, n\}$ is the number of nodes in such layer (i.e., $|N_i|$), and the width of B is the maximum width over all layers, that is, $\max_{i \in \{1, \dots, n\}} |N_i|$.

While the compact DD encoding presented in Section 3 gives us an extended formulation for BPO, it is exponential in size for general BPO instances, and thus, impractical. Indeed, since the DD nodes correspond to states that store subsets of hyperedges and we have at most $n = |V|$ binary variables, their width is, in the worst case, $O(2^{n-1})$. Nonetheless, we can identify structural conditions under which the DD size remains small, and, as shown in Section 5, construct strong valid inequalities for these structures. In what follows, we use the set of compatible active hyperedges in a layer to characterize the size of a DD, which translates directly to DD width. We then use these results to identify hypergraph structures that can encode the BPO problem using bounded DD width and relate them to results in the literature.

4.1 Valid antichains and compact DD width

We now analyze the width of the compact DD relying on the fact that active hyperedges do not behave independently. Indeed, if one hyperedge contains another, then the compatibility of the larger one forces compatibility of the other. Thus, the number of reachable states in a layer can be less than $O(2^{|E|})$ for many hypergraphs.

In what follows, we use standard terminology for posets, chains, antichains, and ideals used in discrete mathematics [31]. A *poset* is a set equipped with a reflexive, antisymmetric, and transitive order. In our case, we consider set inclusions and illustrate these concepts with the poset $(2^{\{1,2,3\}}, \subseteq)$ as a running example. A *chain* is a subset of a poset in which each pair of elements is comparable, for example, $\{\{1\}, \{1, 2\}, \{1, 2, 3\}\}$ is a chain since $\{1\} \subseteq \{1, 2\} \subseteq \{1, 2, 3\}$. An *antichain* is a set of pairwise incomparable elements (e.g., $\{\{1, 2\}, \{2, 3\}\}$). An *ideal* is a downward-closed subset of a poset: whenever it contains an element, it also contains all smaller elements. Thus, if an ideal contains $\{1, 2, 3\}$, it must also contain every subset in the poset contained in $\{1, 2, 3\}$.

Now consider a DD with a vertex order $\sigma = (v_1, \dots, v_n)$ and a layer N_i for some $i \in \{1, \dots, n\}$. For each active hyperedge $e \in \mathcal{A}_i$, we define its *reduced active hyperedge* by $\rho_i(e) = e \cap U_i$, that is, ignoring all vertices except the ones in U_i . Let $\mathcal{R}_i = \{\rho_i(e) : e \in \mathcal{A}_i\}$ be the set of distinct reduced active hyperedges at layer N_i , partially ordered by inclusion. If $R, R' \in \mathcal{R}_i$ are two reduced hyperedges such that $R \subseteq R'$, then compatibility of R' implies compatibility of R : if all assigned vertices in R' have value one, then the same holds for R . Hence, the compatible reduced hyperedges in any reachable state form an ideal of the poset $(\mathcal{R}_i, \subseteq)$, and that ideal is determined by its maximal elements, which form an antichain. Note that not every antichain encodes a reachable state because the union of the antichain may force additional reduced hyperedges to be compatible.

Given an antichain $\mathcal{C}$ of $(\mathcal{R}_i, \subseteq)$, we define $X(\mathcal{C}) = \bigcup_{R \in \mathcal{C}} R$ as the union of all reduced hyperedges in $\mathcal{C}$. We call $\mathcal{C}$ a *valid antichain* at layer N_i if

$$\forall R \in \mathcal{R}_i, R \subseteq X(\mathcal{C}) \implies \exists M \in \mathcal{C} \text{ such that } R \subseteq M.$$

In other words, an antichain is valid if every reduced hyperedge forced by setting the vertices in its union to one is already covered by one of its maximal elements. For example, if $\{1, 2\}, \{2, 3\}, \{1, 2, 3\} \in \mathcal{R}_i$, then $\{\{1, 2\}, \{2, 3\}\}$ is an antichain but not a valid one, because setting elements 1, 2, 3 to one also makes $\{1, 2, 3\}$ compatible. The valid antichain representing that assignment is instead $\{\{1, 2, 3\}\}$. Lemma 1 uses these definitions and observations to characterize the reachable states at layer N_i by valid antichains.

Lemma 1 *Consider a vertex order σ and a layer N_i of the compact DD $B = (N, A)$. The reachable states after assigning U_i variables are in bijection with the valid antichains of $(\mathcal{R}_i, \subseteq)$.*

Proof We first show that each reachable state in N_i corresponds to a valid antichain in $(\mathcal{R}_i, \subseteq)$. Let s_i be a reachable state and define $I(s_i) = \{\rho_i(e) : e \in s_i\} \subseteq \mathcal{R}_i$ as the set of reduced active hyperedges that are compatible in a reachable state s_i . Set $I(s_i)$ is downward closed because if $R' \in I(s_i)$ and $R \in \mathcal{R}_i$ satisfies $R \subseteq R'$, then all vertices of R also have value one under the current assignment, so R is compatible as well. Therefore $R \in I(s_i)$, and $I(s_i)$ is an ideal.

Since every ideal is determined by its maximal elements, let us define $\text{Max}(I(s_i)) = \{R \in I(s_i) : \nexists R' \in I(s_i) \text{ with } R \subsetneq R'\}$ as the set of maximal elements, which in turn form an antichain. Note that the elements in $\text{Max}(I(s_i))$ are also valid: if $R \in \mathcal{R}_i$ satisfies $R \subseteq X(\text{Max}(I(s_i)))$, then all vertices of R have value one, so R is compatible and hence belongs to $I(s_i)$. Because $I(s_i)$ is an ideal, R is contained in some maximal element of $I(s_i)$.

Now, we show the other direction by assuming that $\mathcal{C}$ is a valid antichain. Consider the assignment on U_i that sets the vertices in $X(\mathcal{C})$ to one and the remaining vertices of U_i to zero. Under this assignment, a reduced active hyperedge $R \in \mathcal{R}_i$ is compatible if and only if $R \subseteq X(\mathcal{C})$. By validity, every such R is contained in some member of $\mathcal{C}$. Hence, the induced state is $s_i(\mathcal{C}) = \{e \in \mathcal{A}_i : \rho_i(e) \subseteq M \text{ for some } M \in \mathcal{C}\}$. The maximal elements of the corresponding ideal are exactly the members of $\mathcal{C}$. This gives the claimed bijection. $\square$

Counting antichains in a general poset is #P-complete [30], so valid antichains are not themselves a practical algorithmic primitive. They are, however, a useful structural guide: good orders are those that make reduced active hyperedges comparable, or even identical, as often as possible.

We now use this notation to characterize a new family of hypergraphs that lead to polynomial-size DDs (see Theorem 1), and, in turn, BPO instances that can be solved in polynomial time with, for instance, the extended formulation (3). To do so, let $\text{VA}_i(\sigma)$ denote the set of valid antichains of $(\mathcal{R}_i, \subseteq)$ at layer N_i under order σ . By Lemma 1, the compact-DD width at layer N_i is $|\text{VA}_{i-1}(\sigma)|$. Therefore, the maximum width induced by σ is $W(\sigma) = \max_i |\text{VA}_i(\sigma)|$, and the valid-antichain width of G is $\text{VAW}(G) = \min_\sigma W(\sigma)$.

Theorem 1 *Let $\mathcal{H}$ be a class of hypergraphs such that, for every $G = (V, E) \in \mathcal{H}$, there exists a vertex order σ satisfying $W(\sigma) \leq q(|V|, |E|)$ for some polynomial q . Then the compact DD encoding for G under σ has polynomial size. Moreover, if $q(|V|, |E|)$ is a constant, then the compact DD has $O(|V|)$ nodes and arcs.*

Proof By Lemma 1, each layer N_i contains exactly $|\text{VA}_{i-1}(\sigma)|$ reachable states, for each $i \in \{1, \dots, n\}$ ($n = |V|$). Therefore, the number of non-terminal nodes is

$$\sum_{i=1}^n |\text{VA}_{i-1}(\sigma)| \leq |V| \max_{i \in \{0, \dots, n-1\}} |\text{VA}_i(\sigma)| = |V| W(\sigma) \leq |V| \cdot q(|V|, |E|),$$

making the node count polynomial. Note that if $q(|V|, |E|) = q \in \mathbb{R}^+$ is a constant, the node and arc count is $O(|V|)$ because each node has at most two outgoing arcs. $\square$

4.2 Small exact DDs for literature structures

We now show that there exist well-known hypergraph structures with a bounded compact DD encoding. We use the previously introduced valid antichain characterization to find appropriate vertex orders that satisfy the conditions in Theorem 1. Specifically, we show that flowers and cycles have this property, as stated in Propositions 1 and 2, respectively. We focus on these two structures since flower [18] and β -cycle [16, 21] inequalities are based on them and are, thus, also attractive to our DD-based cutting plane approach. In what follows, a cycle hypergraph is one whose hyperedges can be ordered cyclically so that each hyperedge intersects only its two cyclic neighbors.

Proposition 1 *Let $G = (V, E)$ be a flower with center hyperedge e_0 and petals $e_1, \dots, e_m$. Assume the petals are pairwise disjoint outside the center and that their intersections with the center are pairwise disjoint. Then, $\text{VAW}(G) \leq 4$.*

Proof We first define a vertex order σ in which the vertices in the petals are considered first, followed by the vertices at the center. Formally, consider an arbitrary petal order and, for each petal e_j , the order first uses the vertices in $e_j \cap e_0$, in any order, and then the vertices in $e_j \setminus e_0$, again in any order. After all petal vertices have been ordered, order any remaining vertices of $e_0 \setminus \bigcup_{j=1}^m e_j$ arbitrarily. Because the petals are pairwise disjoint outside the center and their intersections with the center are pairwise disjoint, this order ensures that at any layer there is at most one active petal together with the center hyperedge e_0 .

Now consider a layer in which the active petal is e_j . While the order is processing the vertices of $e_j \cap e_0$, every assigned vertex of e_j is also an assigned vertex of the center, so $\rho_i(e_j) \subseteq \rho_i(e_0)$. Therefore, the set of valid antichains is $\{\emptyset, \{\rho_i(e_j)\}, \{\rho_i(e_0)\}\}$, so there are at most three reachable states.

Then, consider a layer where we assign vertices of $e_j \setminus e_0$. The reduced petal $\rho_i(e_j)$ contains assigned vertices outside the center. In general, $\rho_i(e_j)$ and $\rho_i(e_0)$ are then incomparable. Since no other petal is active, the set of valid antichains is $\{\emptyset, \{\rho_i(e_j)\}, \{\rho_i(e_0)\}, \{\rho_i(e_j), \rho_i(e_0)\}\}$. Thus, there are at most four states while a petal is active. Finally, after all petals have closed, only the center hyperedge can remain active, giving at most two reachable states. Therefore, every layer has at most four valid antichains. By Lemma 1, the compact DD constructed from this order has width at most 4, hence $\text{VAW}(G) \leq 4$. $\square$

Proposition 2 *Let $G = (V, E)$ be a cycle. Then, $\text{VAW}(G) \leq 6$.*

Proof First, we construct the vertex order in accordance with the cyclic nature of the hyperedges. Assume that $G = (V, E)$ has $|E| = L + 1$ hyperedges,

choose an arbitrary hyperedge e_0 of the cycle, and let e_L be one of its two cyclic neighbors. Let $e_1, e_2, \dots, e_{L-1}$ be the sequence of remaining hyperedges obtained by walking around the cycle in the other direction, so that e_{L-1} is adjacent to e_L . Then, the vertex order σ is constructed as follows. First, consider all vertices of e_0 and then, for $h = 1, \dots, L-1$, assign all not-yet-assigned vertices of e_h . Finally, consider the remaining vertices of e_L . Ties inside each hyperedge can be broken arbitrarily.

We now characterize the reachable states in each layer using valid antichains. When the vertices of e_0 are being assigned, the only possible active hyperedges are e_0 and its two neighbors, e_L and e_1 . Moreover, we have the active hyperedge relations $\rho_i(e_L) \subseteq \rho_i(e_0)$, and $\rho_i(e_1) \subseteq \rho_i(e_0)$, because every assigned vertex of e_L or e_1 belongs to e_0 at this stage. Then, the set of possible valid antichains is $\{\emptyset, \{\rho_i(e_L)\}, \{\rho_i(e_1)\}, \{\rho_i(e_L), \rho_i(e_1)\}, \{\rho_i(e_0)\}\}$. Thus, by Lemma 1, the maximum width in these first layers is at most 5.

Now consider a layer in which the order assigns the remaining vertices of e_h , with $1 \leq h < L-1$. At this point, only the unresolved neighbor e_L , the current hyperedge e_h , and the next hyperedge e_{h+1} may be active. Thus, the only containment relation that can arise is $\rho_i(e_{h+1}) \subseteq \rho_i(e_h)$ because the only assigned vertices of e_{h+1} are those already assigned inside e_h . Therefore, the set of possible valid antichains is $\{\emptyset, \{\rho_i(e_L)\}, \{\rho_i(e_{h+1})\}, \{\rho_i(e_L), \rho_i(e_{h+1})\}, \{\rho_i(e_h)\}, \{\rho_i(e_L), \rho_i(e_h)\}\}$. Thus, by Lemma 1, there are at most six reachable states in such a layer.

Finally, if the current vertex corresponds to a hyperedge adjacent to e_L (i.e., e_{L-1}), at most two reduced active hyperedges remain relevant, namely those of e_{L-1} and e_L . If they are incomparable, the set of possible valid antichains are $\{\emptyset, \{\rho_i(e_L)\}, \{\rho_i(e_{L-1})\}, \{\rho_i(e_L), \rho_i(e_{L-1})\}\}$, and if one contains the other there are fewer. Hence, in these final layers there are at most four reachable states.

Taken together, the three cases imply that every layer has at most six valid antichains. By Lemma 1, the compact DD constructed from this order has width at most 6, hence $\text{VAW}(G) \leq 6$. $\square$

Example 2 illustrates the vertex order σ defined for each hypergraph structure and the corresponding compact DD of bounded width. Also, we note that while Proposition 2 proves that cycle hypergraphs allow a compact DD encoding of bounded size, it remains open whether β -acyclic hypergraph classes admit polynomial-size compact DDs under suitable orders. This question is distinct from the known knowledge-compilation result that bounded incidence treewidth yields a polynomial-size d-DNNF representation and polynomial-size extended formulations for BPO [9]. As shown in the following subsection, incidence-treewidth arguments do not directly carry over to our compact DD construction.

Example 2 Figure 2 depicts the vertex orders of Propositions 1 and 2 and the resulting bounded-width compact DDs. Figure 2(a) illustrates the two hypergraphs: on top a flower with center $e_0 = \{v_1, v_3, v_5\}$ and petals $e_1 = \{v_1, v_2\}$ and $e_2 = \{v_3, v_4\}$, and on bottom a cycle with hyperedges $e_0 = \{v_1, v_2, v_3\}$,

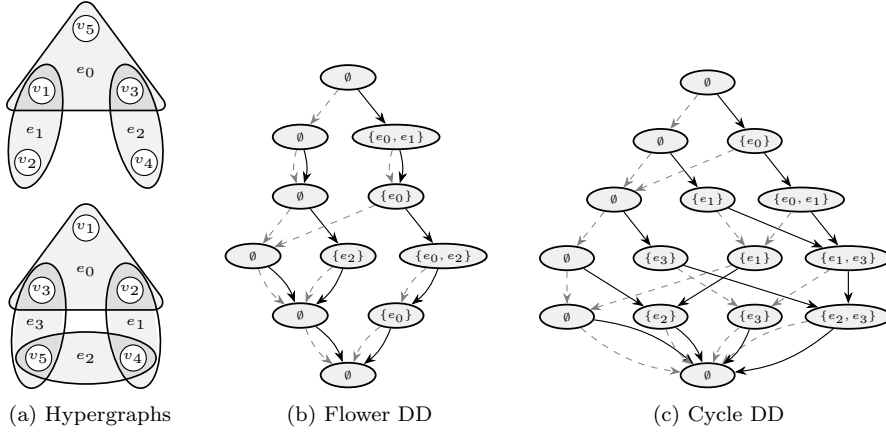


Fig. 2 Bounded-width compact DDs for a flower and a cycle hypergraph, described in Example 2: (a) the flower and cycle hypergraphs, (b) the flower DD of width 3, and (c) the cycle DD of width 4.

$e_1 = \{v_2, v_4\}$, $e_2 = \{v_4, v_5\}$, and $e_3 = \{v_3, v_5\}$. Note that vertices are numbered according to the vertex order of each DD. In both cases, the compact DD width is bounded and slightly smaller than the worst-case scenario proved in the corresponding propositions.

4.3 Valid antichains, pathwidth, and incidence treewidth

Researchers have also characterized the hypergraph through *incidence treewidth* to build polynomial-size extended formulations for BPO using a d-DNNF representation [9]. We now show that the valid antichains characterized here are indeed incomparable with the incidence treewidth in Proposition 3. Recall that treewidth is the minimum, over all tree decompositions, of the largest bag size minus one [22]. The incidence treewidth of a hypergraph is the treewidth of its incidence graph.

Before proving the incomparability, we recall that for ordinary graphs (i.e., rank-two hypergraphs) the valid-antichain width is governed exactly by a classical width parameter. For a graph $G = (V, E)$ and an order $\sigma = (v_1, \dots, v_n)$ with prefixes $U_i = \{v_1, \dots, v_i\}$, let $b_i(\sigma) = |\{v \in U_i : \exists w \in V \setminus U_i \text{ with } \{v, w\} \in E\}|$ be the number of *boundary* vertices at U_i . The *vertex separation* of σ is $vs_\sigma(G) = \max_{i \in \{1, \dots, n\}} b_i(\sigma)$, and the *vertex separation number* is $vs(G) = \min_\sigma vs_\sigma(G)$. By a classical result, the vertex separation number coincides with pathwidth (i.e., a number that shows how close a graph is to being a path), that is, $vs(G) = pw(G)$ [26].

Corollary 1 *For every ordinary graph $G = (V, E)$, $VAW(G) = 2^{vs(G)} = 2^{pw(G)}$.*

Proof Fix an order σ . At any prefix U_i , each active edge has exactly one end-point in U_i and one outside, so every reduced active edge is a singleton $\{v\}$ with v a boundary vertex, and these singletons are pairwise incomparable. Hence every subset of the $b_i(\sigma)$ boundary singletons is a valid antichain, giving $|\text{VA}_i(\sigma)| = 2^{b_i(\sigma)}$. Since $x \mapsto 2^x$ is increasing, $W(\sigma) = \max_i 2^{b_i(\sigma)} = 2^{\text{vs}_\sigma(G)}$, and minimizing over σ yields $\text{VAW}(G) = 2^{\text{vs}(G)}$. The identity $\text{vs}(G) = \text{pw}(G)$ [26] gives the last equality. $\square$

We note that this identity is specific to rank-two hypergraphs; for hypergraphs of higher rank, valid-antichain width departs from pathwidth, as the converse direction below shows.

Proposition 3 *The valid antichain characterization is incomparable to the incidence treewidth.*

Proof We first show that bounded incidence treewidth does not imply bounded VAW. Consider $T_h = (V, E)$, the complete binary tree of height h , where each edge has exactly two vertices. Since T_h is a tree, its treewidth is one. Moreover, the incidence graph of T_h is obtained by subdividing every edge of T_h once, so each original edge becomes an incidence node adjacent to its two endpoints; the incidence graph is therefore also a tree and has treewidth one. On the other hand, complete binary trees have unbounded pathwidth as h grows [22]. Hence, by Corollary 1, $\text{VAW}(T_h) = 2^{\text{pw}(T_h)}$ is also unbounded. Thus, the family $\{T_h\}_{h \geq 1}$ has bounded incidence treewidth but unbounded VAW.

We now show the converse also fails. Let $A = \{a_1, \dots, a_p\}$ and $B = \{b_1, \dots, b_q\}$ be sets of vertices, and define a hypergraph $G = (V, E)$ with $V = A \cup B$, and $E = \{A \cup \{b\} : b \in B\}$. Order all vertices of A first and then all vertices of B . At any prefix U_i of this order, all active hyperedges have the same reduced active hyperedge. Therefore, there are at most two valid antichains, the empty one and the singleton containing that common reduced hyperedge, so $|\text{VA}_i(\sigma)| \leq 2$. On the other hand, the incidence graph contains a complete bipartite graph between A and the hyperedges, so its incidence treewidth is unbounded as the number of vertices grows. $\square$

5 Target cuts over compact DDs

We now present the cut generator linear program (CGLP) that creates valid inequalities using our compact DD encoding introduced in Section 3. As previously mentioned, there are several cut generation procedures specifically designed to separate a fractional point from a convex set encoded as a DD [3, 4, 11, 14]. In this work, we opt to use DD-based target cuts [32], due to their simple implementation and their guarantee of facet-defining inequalities.

Broadly, given a fractional point $\bar{\mathbf{z}}$ to separate and a fixed interior point ω of the polytope represented by the DD, the target cuts procedure [7] searches over the translated polar of that polytope for the valid inequality of maximum violation in the direction from ω to $\bar{\mathbf{z}}$. The correspondence between DDs and

polarity allows this polar to be optimized compactly over the DD network flow model [32], and whenever the returned solution is an extreme point of the polar, the associated inequality is facet-defining for the represented polytope. As with the extended formulation of Section 3, the CGLP of [32] assumes that every variable appearing in the cut is represented explicitly in the DD. Our compact DD, however, branches only on the vertex variables $\mathbf{z}_v$ and recovers the hyperedge variables $\mathbf{z}_e$ implicitly along its states, so the standard target-cut CGLP does not apply directly. We now present the extension to our compact DD encoding.

In what follows, let $B = (N, A)$ be the compact DD encoding for a hypergraph $G = (V, E)$, so B is exact for $\mathcal{S}_G$ and therefore represents MP_G . Recall from Section 3 that, for each arc $a \in A$ leaving layer N_i ($i \in \{1, \dots, n\}$), $\text{var}(a) = v_i$ is the associated vertex, $\text{val}(a) \in \{0, 1\}$ its assigned value, and $\mathcal{E}_1(a)$ the set of hyperedges that take value one on arc a . Throughout, let $\boldsymbol{\omega} \in \text{int}(MP_G)$ be a fixed interior point; since MP_G is full-dimensional [17] and $\boldsymbol{\omega}$ lies in its interior, the translated polar $(MP_G - \boldsymbol{\omega})^\circ$ is bounded and the CGLP below attains its optimum.

Consider continuous variables $\mathbf{u} \in \mathbb{R}^{V \cup E}$ and $\boldsymbol{\theta} \in \mathbb{R}^N$ and define arc sets associated with 0 and 1 assignments as $A^1 = \{a \in A : \text{val}(a) = 1\}$ and $A^0 = \{a \in A : \text{val}(a) = 0\}$, respectively. Given a fractional point $\bar{\mathbf{z}} = (\bar{\mathbf{z}}_v, \bar{\mathbf{z}}_e) \in \mathbb{R}^{V \cup E}$, the target-cut CGLP for our compact DD encoding is as follows, where r and t denote the root and terminal nodes of B , respectively:

$$\max \quad \mathbf{u}^\top (\bar{\mathbf{z}} - \boldsymbol{\omega}) \quad (4a)$$

$$\text{s.t.} \quad \theta_j \leq \theta_i - u_{\text{var}(a)} - \sum_{e \in \mathcal{E}_1(a)} u_e \quad \forall a = (i, j) \in A^1, \quad (4b)$$

$$\theta_j \leq \theta_i \quad \forall a = (i, j) \in A^0, \quad (4c)$$

$$\theta_r = 1 + \mathbf{u}^\top \boldsymbol{\omega}, \quad (4d)$$

$$\theta_t = 0. \quad (4e)$$

Constraints (4b)–(4d) encode the translated polar. Note that an arc $a \in A$ with $\text{val}(a) = 1$ (i.e., inequalities (4b)) contributes the coefficient of the assigned vertex and the coefficients of all hyperedges assigned to one on that arc; while the remaining arcs contribute nothing. Summing the arc contributions along an r – t path recovers $\mathbf{u}^\top \mathbf{z}$ for point $\mathbf{z}$, so the model enforces $\mathbf{u}^\top (\mathbf{z} - \boldsymbol{\omega}) \leq 1$ for every point of MP_G .

Model (4) has $O(|A|)$ constraints and $|V| + |E| + |N|$ variables, so its size is polynomial in the size of B ; together with the structural results of Section 4, a bounded valid-antichain width yields polynomial-time separation. As in the target-cut framework, if the optimal value of (4) is at most 1, then the DD yields no violated target cut. Otherwise, the resulting inequality $\mathbf{u}^\top \mathbf{z} \leq 1 + \mathbf{u}^\top \boldsymbol{\omega}$ is valid for the represented polytope MP_G and cuts off $\bar{\mathbf{z}}$.

Since multilinear polytopes are full-dimensional for every hypergraph [17], the translated polar has the usual facet correspondence [7]. If the solution

returned by (4) is an extreme point of $(MP_G - \omega)^\circ$, then the associated inequality corresponds to a facet of the represented polytope MP_G . In particular, a unique optimum is sufficient for this conclusion. If the optimum is degenerate, the returned inequality remains valid, but facetness is not automatic unless an extreme optimal solution is selected.

6 Local supports and expansions for cut generation

As mentioned in Section 4, the compact DD encoding can be exponential in size for generic BPO instances. Thus, constructing such a DD and generating cuts with it can be impractical. With that in mind, we develop three different procedures that build compact DDs for portions of the problem (i.e., local supports for a hypergraph $G = (V, E)$) with the aim of returning strong valid inequalities for the original problem.

Formally, a *local support* for $G = (V, E)$ is a hypergraph $G_S = (V_S, E_S)$ chosen from G , typically with $V_S \subseteq V$ and $E_S \subseteq E$, which are also known as *partial hypergraphs* [17]. If E_S has all the hyperedges that are subsets of V_S (i.e., $E_S = \{e \in E : e \subseteq V_S\}$), then we say that G_S is a *section hypergraph* of G induced by V_S . Once a support G_S has been chosen, we build a compact DD for it and use the target-cut CGLP introduced in Section 5 to create valid inequalities. The generated cut can be lifted back to the full variable space by *zero lifting* [17] (i.e., setting coefficients of variables outside $V_S \cup E_S$ to zero).

We now describe three alternatives for creating this local support. To summarize, the *fixed-support* strategy considers supports from known valid inequalities in the literature (i.e., two-links, flowers, and β -cycles) and serves as a validation benchmark. The *expanded-support* strategy expands over fixed-supports by considering additional edges with the goal of exposing nearby interactions that the literature support alone omits. Lastly, the *partition-support* strategy partitions the set of vertices and builds section hypergraphs induced by each partition to leverage existing theoretical results.

Fixed-support strategy. For a given fractional point $\bar{\mathbf{z}}$, this strategy builds compact DDs based on a known violated valid inequality. Specifically, we heuristically search for violated two-link, flower, and β -cycle inequalities and create a DD for the corresponding local support for any such violated cuts. To do so, we work in the edge-overlap graph of G , whose nodes are the hyperedges and adjacency is defined by nonempty overlap. Starting from a set of sampled edges, we build small-overlap neighborhoods and mine candidate local supports within those neighborhoods. A candidate's support is passed to the DD separator only if its associated literature inequality is violated at the current relaxation point $\bar{\mathbf{z}}$. We note that, given its nature, this strategy is merely a validation benchmark for our DD-based CGLP.

Expanded-support strategy. This strategy is an extension of our fixed-support approach, with the aim of exposing nearby interactions that are not visible

Algorithm 1 EXPANDED SUPPORT STRATEGY

Require: Hypergraph $G = (V, E)$, support $G_S = (V_S, E_S)$, hyperedge cap M
 1: **while** $|E_S| < M$ and there exists an $e' \in E \setminus E_S$ that satisfies $|e' \cap V_S| \geq 2$ **do**
 2: Choose an eligible e' of maximum overlap with E_S , that is,

$$e' := \arg \max_{e'' \in E} \left\{ \sum_{e \in E_S} |e'' \cap e| : e'' \in E \setminus E_S, |e'' \cap V_S| \geq 2 \right\},$$

3: Update $E_S \leftarrow E_S \cup \{e'\}$ and $V_S \leftarrow \bigcup_{e \in E_S} e$
 4: **return** expanded local support $G_S = (V_S, E_S)$

from the support of simple structures (i.e., two-links, flowers, and β -cycles) alone. The strategy is motivated by the inducing-support conditions that appear in zero-lifting results for multilinear polytopes [17]: supports that omit overlapping hyperedges may miss relevant overlap information. Expansion is therefore a heuristic approach for incorporating more of this local overlap structure (i.e., adjacent vertices and hyperedges) into the local support, with the goal of obtaining stronger cuts.

Algorithm 1 summarizes the expanded-support strategy. The procedure receives an initially violated local support $G_S = (V_S, E_S)$ of G (found by the heuristic used in the previous strategy) and a maximum number of hyperedges M that are allowed in the expanded support. It then iteratively selects an eligible hyperedge that overlaps the current vertex set and maximizes overlap with the current hyperedges (lines 1–2) and updates the local support accordingly (line 3). Once the hyperedge cap is reached, we create the DD on the resulting support, find a valid inequality, and zero-lift it to the original variable space.

Partition-support strategy. Unlike the other two alternatives, this strategy builds a set of local supports based on a given vertex partition, without relying on the simple local structures from known valid inequalities. For any fractional point $\bar{\mathbf{z}}$, this strategy iterates over the pre-defined set of local supports to create one DD-based valid inequality for each local support G_S where $\bar{\mathbf{z}} \notin MP_{G_S}$. The strategy is designed such that we can: (i) test whether generic local supports can render useful cuts, and (ii) leverage the theoretical results for section hypergraphs to obtain facet-defining inequalities under suitable conditions.

Algorithm 2 summarizes the procedure for creating the local supports. First, we arbitrarily partition the set of vertices V into H disjoint sets $\{V_1, \dots, V_H\}$. Then, for each vertex group V_h ($h \in \{1, \dots, H\}$), we create a local support following the *contained-edge rule*, that is, consider all hyperedges that are contained in V_h (i.e., $E_h = \{e \in E : e \subseteq V_h\}$). Then, by construction, each local support $G_h = (V_h, E_h)$ is a section hypergraph of G [17].

The main benefit of building a section hypergraph $G_h = (V_h, E_h)$ is that, if V_h is *inducing* (i.e., every $e \in E$ with $|e \cap V_h| \geq 2$ satisfies $e \cap V_h \in E$) and the resulting inequality is facet-defining for MP_{G_h} , then its zero-lifting is facet-defining for MP_G whenever it is maximal [17]. Since our DD-based CGLP (4)

Algorithm 2 PARTITIONSUPPORTSTRATEGY

Require: Hypergraph $G = (V, E)$, and vertex partition $\{V_1, \dots, V_H\}$

- 1: **for** $h \in \{1, \dots, H\}$ **do**
- 2: Select edges with the contained-edge rule: $E_h \leftarrow \{e \in E : e \subseteq V_h\}$
- 3: Create local support $G_h = (V_h, E_h)$
- 4: **return** set of local supports $\{G_1, \dots, G_H\}$

can return facets of the local polytope MP_{G_h} (Section 5), section-hypergraph supports are precisely the setting in which these facets can be zero-lifted to facets of the full instance.

We note that in our implementation, the fixed-support and expanded-support strategies are repeatedly called at the fractional point $\bar{\mathbf{z}}$. Thus, local supports are selected dynamically, and their corresponding DDs are created in each iteration. In contrast, the partition-support strategy builds the set of local supports and their corresponding DDs at the beginning of the optimization process and reuses them whenever we want to separate a fractional point.

7 Dimension and facet audit for DD cuts

We now present a novel approach for computing the dimension of the face induced by a valid inequality for the polytope encoded by a DD. We present the procedure in general terms and then extend it to the compact DD encoding introduced in Section 3. We then show how to use this procedure to obtain facet-defining certificates for our cuts, which we employ in Section 8 as a quality measurement.

We now present the notation used throughout this section. Consider a set $Q \subseteq \{0, 1\}^n$ and let $\dim(Q)$ be its dimension. We say that an inequality $\boldsymbol{\pi}^\top \mathbf{z} \leq \pi_0$ is *valid* for $\text{conv}(Q)$ if it holds for every $\mathbf{z} \in Q$, and its associated face is $F(\boldsymbol{\pi}) = \{\mathbf{z} \in \text{conv}(Q) : \boldsymbol{\pi}^\top \mathbf{z} = \pi_0\}$. If $F(\boldsymbol{\pi}) \neq \emptyset$ and $\dim(F(\boldsymbol{\pi})) = \dim(Q) - 1$, then $F(\boldsymbol{\pi})$ is a *facet* of $\text{conv}(Q)$.

In what follows, we introduce vector notation to represent the value associated with an arc, which is amenable to our DD-based dimension certificate procedure. Let $B = (N, A)$ be a DD representing Q . Recall from Section 3 that $\text{var}(a) \in V$ and $\text{val}(a) \in \{0, 1\}$ are the vertex and value associated with arc $a \in A$, respectively. Then, define the vector contribution of an arc $a \in A$ as $\chi(a) = \mathbf{e}^{\text{var}(a)}$ if $\text{val}(a) = 1$ and $\chi(a) = \mathbf{0}$ otherwise, where $\mathbf{e}^q$ denotes the q -th canonical (unit) vector ($q \leq n$) and $\mathbf{0}$ the zero vector of $\mathbb{R}^n$. Then, for a valid inequality $\boldsymbol{\pi}^\top \mathbf{z} \leq \pi_0$ of Q , we define the *scalar arc length* of arc $a \in A$ as $\ell_\pi(a) = \boldsymbol{\pi}^\top \chi(a)$, where $\ell_\pi(a) = \pi_{\text{var}(a)}$ if $\text{val}(a) = 1$ and $\ell_\pi(a) = 0$ otherwise.

Throughout this section, lowercase p denotes a *partial path* (i.e., a sequence of arcs) over $B = (N, A)$ from root node r to a given node $u \in N$, while uppercase P denotes a complete r - t path. Then, for any partial path p , define $\mathbf{z}(p) = \sum_{a \in p} \chi(a)$ as its vector encoding, i.e., a vector that encodes all the

Algorithm 3 DD-DIMENSIONCERTIFICATE**Require:** Exact DD $B = (N, A)$, valid inequality $\pi^\top \mathbf{z} \leq \pi_0$ with $F(\pi) \neq \emptyset$ **Ensure:** Dimension of $F(\pi)$

```

1: Create the sub-diagram  $B^\pi = (N^\pi, A^\pi)$  induced by  $A^\pi$ 
2: Set  $\text{rep}[u] \leftarrow \text{undefined}$  for all  $u \in N^\pi$ ,  $\text{rep}[r] \leftarrow \mathbf{0}$ , and  $\mathcal{M} \leftarrow \emptyset$ 
3: for each layer  $i \in \{1, \dots, n\}$  do
4:   for each arc  $a = (u, u') \in A_i^\pi$  do
5:      $\text{cand} \leftarrow \text{rep}[u] + \chi(a)$ 
6:     if  $\text{rep}[u']$  is undefined then
7:        $\text{rep}[u'] \leftarrow \text{cand}$ 
8:     else
9:       add  $\text{cand} - \text{rep}[u']$  to  $\mathcal{M}$ 
10: return  $\text{rank}(\mathcal{M})$ 

```

decisions for arcs $a \in p$ but sets to zero all decisions after the last arc represented in p . Note that if $p = P$ (i.e., an r - t path) then $z(P)$ is the binary point encoded by P .

7.1 Tight arcs and dimension certificate

We now present our DD-based dimension certificate for a DD $B = (N, A)$ that exactly represents Q and a valid inequality $\pi^\top \mathbf{z} \leq \pi_0$ with $F(\pi) \neq \emptyset$. The overall idea is to work over the arcs and paths in B where $\pi^\top \mathbf{z} = \pi_0$ and then collect a set of difference vectors whose rank equals the dimension of $F(\pi)$.

First, we identify the arcs in B that are associated with the binary points in $F(\pi)$, which we call *tight arcs*. Consider top-down and bottom-up longest-path values for any node $u \in N$ as $L^\uparrow(\pi, u) = \max \left\{ \sum_{a \in p} \ell_\pi(a) : p \text{ is an } r\text{-}u \text{ path} \right\}$ and $L^\downarrow(\pi, u) = \max \left\{ \sum_{a \in p} \ell_\pi(a) : p \text{ is a } u\text{-}t \text{ path} \right\}$, respectively [32, 11]. Then, we say that an arc $a = (u, u')$ is a tight arc if it belongs to at least one tight r - t path, that is, if $L^\uparrow(\pi, u) + \ell_\pi(a) + L^\downarrow(\pi, u') = \pi_0$, so that the longest path traversing arc a has length π_0 . For each stage $i \in \{1, \dots, n\}$, let A_i^π be the set of outgoing arcs in layer N_i that are tight, and let $A^\pi = \bigcup_{i=1}^n A_i^\pi$ be the set of tight arcs. Lastly, let $\mathcal{P}_\pi(r, u)$ denote the set of r - u paths that are prefixes of at least one tight r - t path.

Our DD-based dimension certificate algorithm, summarized in Algorithm 3, uses the sub-diagram $B^\pi = (N^\pi, A^\pi)$ of B induced by the tight arcs A^π (i.e., $N^\pi = \{r\} \cup \{u \in N : \exists a \in A^\pi \text{ s.t. } a \in A^{\text{in}}(u)\}$) (line 1). The algorithm traverses B^π starting at the root r and maintains, for each node $u \in N^\pi$, a representative vector $\text{rep}[u]$ equal to the encoding $z(p_u)$ of a single tight prefix path from r to u . It starts from $\text{rep}[r] = \mathbf{0}$ and processes the tight arcs layer by layer, so that $\text{rep}[u]$ is already available when the outgoing arcs of u are examined. When a tight arc $a = (u, u')$ reaches u' for the first time, the algorithm fixes $\text{rep}[u'] = \text{rep}[u] + \chi(a)$, extending the prefix chosen for u along a (lines 5-7); these first-visit arcs form a spanning tree of B^π rooted at r . Every other tight arc into an already-represented node u' yields an alternative tight

prefix reaching u' , and the discrepancy $\text{cand} - \text{rep}[u'] = (\text{rep}[u] + \chi(a)) - \text{rep}[u']$ between the two encodings is added to the matrix $\mathcal{M}$ (lines 5 and 9). As shown by Lemma 2 and Theorem 2, each such vector is a difference of two tight r - t path vectors, and these differences span their entire linear space; hence the algorithm returns $\text{rank}(\mathcal{M})$, which equals the dimension of $F(\pi)$. The proof uses the concatenation operation $\circ$ to augment a path: if $p' = (a_1, \dots, a_k)$ is a partial path with k arcs, then $p = p' \circ a = (a_1, \dots, a_k, a)$.

Lemma 2 *In Algorithm 3, every representative vector $\text{rep}[u]$, with $u \in N^\pi$, equals $z(p_u)$ for some partial path $p_u \in \mathcal{P}_\pi(r, u)$. Moreover, for every node $u \in N^\pi$ and partial path $p \in \mathcal{P}_\pi(r, u)$, $z(p) - \text{rep}[u] \in \text{span}(\mathcal{M})$.*

Proof We proceed by induction over the layers. The claim is immediate at the root r , where $\text{rep}[r] = \mathbf{0}$, which corresponds to $z(r)$ and $\mathbf{0} \in \text{span}(\mathcal{M})$ for any matrix $\mathcal{M}$. Then, consider the hypothesis true for all nodes in layer N_i^π , and take any arc $a = (u, u') \in A^\pi$ with $u \in N_i^\pi$, $p = p' \circ a$, with $p' \in \mathcal{P}_\pi(r, u)$. Then, $z(p) = z(p') + \chi(a)$, and, by the induction hypothesis, $z(p') - \text{rep}[u] \in \text{span}(\mathcal{M})$.

If $\text{rep}[u']$ is undefined, the algorithm sets $\text{rep}[u'] = \text{rep}[u] + \chi(a) = z(p_u \circ a)$, with $p_u \circ a \in \mathcal{P}_\pi(r, u')$, which establishes the first claim for u' . Hence $z(p) - \text{rep}[u'] = z(p') + \chi(a) - (\text{rep}[u] + \chi(a)) = z(p') - \text{rep}[u]$, which belongs to $\text{span}(\mathcal{M})$ by the induction hypothesis.

If $\text{rep}[u']$ is already defined, the algorithm adds $\delta_a = \text{rep}[u] + \chi(a) - \text{rep}[u']$ to $\mathcal{M}$. Therefore,

$$\begin{aligned} z(p) - \text{rep}[u'] &= z(p') + \chi(a) - \text{rep}[u'] \\ &= (z(p') - \text{rep}[u]) + (\text{rep}[u] + \chi(a) - \text{rep}[u']) = (z(p') - \text{rep}[u]) + \delta_a, \end{aligned}$$

which belongs to $\text{span}(\mathcal{M})$ since it is the sum of two vectors in $\text{span}(\mathcal{M})$. $\square$

Theorem 2 *Algorithm 3 returns the dimension of $F(\pi)$ over $\text{conv}(Q)$.*

Proof Since DD B exactly represents Q , the extreme points of $F(\pi)$ are exactly the vectors $z(P)$ for tight r - t paths $P \in \mathcal{P}_\pi(r, t)$; thus $F(\pi) = \text{conv}(\{z(P) : P \in \mathcal{P}_\pi(r, t)\})$. By Lemma 2, $z(P) - \text{rep}[t] \in \text{span}(\mathcal{M})$ for every such P . Therefore, we conclude that $\dim(F(\pi)) \leq \text{rank}(\mathcal{M})$.

For the reverse inclusion, we show that every vector added to $\mathcal{M}$ is the difference of two tight r - t path vectors. Consider a vector $\text{rep}[u] + \chi(a) - \text{rep}[u']$ added to $\mathcal{M}$ while processing an arc $a = (u, u')$. By Lemma 2, $\text{rep}[u] = z(p_u)$ for some $p_u \in \mathcal{P}_\pi(r, u)$ and $\text{rep}[u'] = z(p_{u'})$ for some $p_{u'} \in \mathcal{P}_\pi(r, u')$. Now choose any tight u' - t path q and consider $P' = p_u \circ a \circ q$ and $P = p_{u'} \circ q$.

Recall that any path in $\mathcal{P}_\pi(r, w)$ has length $L^\uparrow(\pi, w)$, since $w \in N^\pi$ lies on a tight path. Therefore, both P' and P are tight r - t paths because p_u and $p_{u'}$ have lengths $L^\uparrow(\pi, u)$ and $L^\uparrow(\pi, u')$, q is tight, and a is a tight arc. Then,

$$\begin{aligned} z(P') - z(P) &= z(p_u) + \chi(a) + z(q) - z(p_{u'}) - z(q) \\ &= \text{rep}[u] + \chi(a) - \text{rep}[u']. \end{aligned}$$

The common suffix cancels, so the vector added to $\mathcal{M}$ by Algorithm 3 (line 9) is a difference of two tight r - t path vectors. Hence $\text{span}(\mathcal{M})$ is contained in

the linear span of all differences of $z(P)$ for $P \in \mathcal{P}_\pi(r, t)$. Together with the first inclusion, this gives $\dim(F(\pi)) = \text{rank}(\mathcal{M})$. $\square$

We now adapt Algorithm 3 to the compact DD encoding for BPO. Specifically, given a compact DD $B = (N, A)$ for a hypergraph $G = (V, E)$, we need to adjust the definitions of $\chi(a)$ for $a \in A$ to account for the hyperedge variables. Recall, from Section 3, that $\mathcal{E}_1(a)$ is the set of hyperedge variables $\mathbf{z}_e$ assigned value one on arc $a \in A$. Then, for an arc $a \in A$, we redefine $\chi(a) = \mathbf{e}^{\text{var}(a)} + \sum_{f \in \mathcal{E}_1(a)} \mathbf{e}^f$ when $\text{val}(a) = 1$ and $\chi(a) = \mathbf{0}$ otherwise. Thus, $\chi(a)$ is a vector of dimension $|V| + |E|$ that accounts for both vertex and hyperedge variables assigned in such an arc. The rest of the definitions and steps of the algorithm (e.g., $\ell_\pi(a)$) follow from the new $\chi(a)$ definition. Note that since MP_G has full dimension (i.e., $\dim(MP_G) = |V| + |E|$) [17], any valid inequality is facet-defining for MP_G if Algorithm 3 returns $|V| + |E| - 1$.

7.2 The I_K certificate and zero lifting

Algorithm 3 provides a global facet test for MP_G when applied to the hypergraph instance $G = (V, E)$; however, it is impractical for large instances. We therefore use the certificate over local supports used for cut generation (see Section 5), and the zero-lifting theory [17]. The resulting test is conservative: if it succeeds, the cut is certified to be a global facet of MP_G ; if it fails, the global status is unknown.

Del Pia and Khajavirad [17] show that when a local support is an inducing section hypergraph, a zero-lifted local facet remains facet-defining for the larger multilinear polytope if and only if it is maximal for the larger hypergraph. Their Corollary 6 can be used as a conservative certificate even when the original support is not inducing by completing the support with the required intersection hyperedges, auditing the cut on this completed support, and then checking maximality.

For a valid inequality $\pi^\top \mathbf{z} \leq \pi_0$, consider $K = G(\pi)$ to be its support hypergraph, in which its hyperedge set is $E_K = \{e \in E : \pi_e \neq 0\}$, and its vertex set is $V_K = \{v \in V : \pi_v \neq 0\} \cup \{v \in V : \exists e \in E \text{ with } v \in e, \pi_e \neq 0\}$. Let G_K be the section hypergraph of the full instance $G = (V, E)$ induced by V_K , that is, $V_{G_K} = V_K$ and $E_{G_K} = \{e \in E : e \subseteq V_K\}$. Then, define the intersection completion hypergraph as $I_K = (V_K, E_{I_K})$ with hyperedge set $E_{I_K} = E_{G_K} \cup \{e \cap V_K : e \in E, |e \cap V_K| \geq 2\}$, where duplicate hyperedges are removed and hyperedges in $E_{I_K} \setminus E_{G_K}$ correspond to auxiliary certificate hyperedges. To audit the cut on I_K , we zero-extend π by assigning a coefficient of zero to every auxiliary hyperedge in $E_{I_K} \setminus E_K$.

The remaining ingredient is the maximality check. Maximality is violated if there exists a decomposition obtained by multiplying the local inequality by a monomial involving variables outside the selected subhypergraph and then linearizing the resulting expressions. This test is performed first, before the remaining certificate checks, using the finite criterion of Del Pia and Khajavirad [17], based on their Proposition 7 and Lemma 2. If the maximality

Algorithm 4 GLOBALFACETAUDITVIAIK

Require: Full hypergraph $G = (V, E)$, cut $\pi^\top \mathbf{z} \leq \pi_0$
Ensure: Certification status for global facetness

- 1: **if** Cut is non-maximal **then**
- 2: **return** non-maximal; classify the cut as a global nonfacet
- 3: Build the compact DD B_{I_K} over I_K
- 4: $d \leftarrow \text{DD-DIMENSIONCERTIFICATE}(B_{I_K}, \pi^\top \mathbf{z} \leq \pi_0)$
- 5: **if** $d = |V_K| + |E_{I_K}| - 1$ **then**
- 6: **return** certified global facet by the I_K certificate
- 7: **if** Exact global DD is computationally feasible **then**
- 8: Build the compact exact global DD B_G
- 9: $d_G \leftarrow \text{DD-DIMENSIONCERTIFICATE}(B_G, \pi^\top \mathbf{z} \leq \pi_0)$
- 10: **if** $d_G = |V| + |E| - 1$ **then**
- 11: **return** certified global facet by direct global audit
- 12: **else**
- 13: **return** certified global nonfacet by direct global audit
- 14: **return** unknown global status

test fails, the candidate inequality is rejected as a facet of the full multilinear polytope; otherwise, the audit proceeds to the remaining certificate checks.

Algorithm 4 presents the complete procedure. It first checks the maximality (line 1); if the inequality is non-maximal, it is classified as a global nonfacet for this audit. If maximality passes, the algorithm builds the DD B_{I_K} and runs Algorithm 3 on it (lines 3–4). The rank test (line 5) checks the dimension of the cut’s face; if it equals $|V_K| + |E_{I_K}| - 1$, the cut is certified as a global facet by the I_K certificate. If this local certificate fails, the algorithm attempts a direct global audit when the exact DD for the complete problem can be built. The final rank test (line 10) certifies either global facetness or global nonfacetness. If the global DD is not computationally feasible, the global status remains unknown.

8 Computational results

This section presents the empirical evaluation of our DD-based cut generation procedures. We first focus on the quality of the generated cuts at the root node, in terms of gap reduction and computational efficiency, and compare with alternatives in the literature. Then, we test the efficiency of our procedures for solving instances from the literature to optimality using a B&B mechanism with cuts added at the root node, and compare them with existing alternatives.

8.1 Experimental setup

We consider two datasets commonly used in the literature: LABS and VISION. The LABS instances are the low-autocorrelation binary sequence instances distributed in POLIP and MINLPLib [15, 8, 21]. Specifically, we consider instances with fewer than 5000 multilinear terms and discard the `autocorr_bern20-03`

instance (which solves at the root node), for a total of 22 instances. The VISION benchmark consists of the 45 image-restoration instances introduced in the BPO literature [20], with image sizes 10×10 , 10×15 , and 15×15 .

Our experimental comparison considers four families of cut-generation routines (i.e., LT, FS, EX, and PT), all of which are applied at the root node, as follows. LT denotes the literature-support baseline, which separates the violated two-link, flower, and odd β -cycle inequalities of length 3 found by well-known separators. FS corresponds to the DD separation on the same fixed supports used by LT to test whether the DD separator can recover comparable or stronger cuts from those supports. EXM denotes DD separation on expanded literature supports, where $M \in \{4, 5, 6, 7, 8\}$ is the hyper-edge cap in Algorithm 1. Lastly, PT q is the partition-support strategy, where $q \in \{0, 1, 2, 3, 4\}$ is the partition size parameter. Specifically, the approach considers $r = \max_{e \in E} |e|$ to be the rank of the instance hypergraph, and each PT q strategy considers a vertex partition generated with target size $r + q$, inducing a section hypergraph as described in Algorithm 2.

All DD-based CGLPs use the same interior point ω . For each local support hypergraph $G_S = (V_S, E_S)$, we set $\omega_v = \frac{1}{2}$ for each $v \in V_S$ and $\omega_e = 2^{-|e|}$ for each $e \in E_S$.

Our methodology is implemented in Python 3.11.15 using Gurobi 13.0.1, with a 3600s time limit across all experiments. We use a single-thread (i.e., `Threads=1`), set `PreCrush=1` to allow user cuts to be translated through pre-solve, and deactivate native cuts in Gurobi (i.e., `Cuts=0`, `MIPSepCuts=0`, and `CutPasses=0`) to avoid interference of external cuts in our comparisons. The experiments were run on the *Ingeniería UC Cluster* using SLURM, equipped with an AMD EPYC 9354 processor and 256 GB of RAM.

A separation round solves the current LP relaxation, searches for local supports under the selected strategy, and adds at most $\lceil 0.05|E| \rceil$ violated cuts. For neighborhood-based strategies, the initial neighborhood contains 300 hyperedges and is expanded by 30 hyperedges whenever no candidate support is found, until the search budget is exhausted.

In all experiments, cuts are added to the root node based on the following criteria. Separation stops when a round finds no cuts; for PT, up to five consecutive no-cut rounds are allowed before stopping. Separation also stops if the average LP improvement per cut over the last 50 rounds is less than 10% of the historical average LP improvement per cut, or if a round time exceeds 2.5 times the median separation time over the last 20 rounds. In particular, the last stopping criterion is used to avoid adding uninformative cuts, which, in turn, increase computational time and model size.

8.2 Root relaxation and facet audit

We first evaluate the efficiency of the proposed DD-based cuts across different strategies and relative to cuts from the literature. Moreover, we use the facet

Table 1 Root-relaxation performance on tested instances.

Dataset	Method	Gap	C%	Time	CM	C% / 1000
LABS	LT	437.15	60.97	47.02	3412.4	17.87
LABS	FS	488.16	57.63	51.61	2801.4	20.57
LABS	EX4	265.42	75.58	42.10	2092.7	36.12
LABS	EX5	214.52	78.65	42.56	1892.8	41.55
LABS	EX6	216.52	78.49	49.09	1615.8	48.57
LABS	EX7	205.16	79.28	50.30	1670.5	47.46
LABS	EX8	224.29	78.06	51.01	1532.7	50.93
LABS	PT0	57.09	91.31	32.22	2202.5	41.46
LABS	PT1	51.97	91.82	35.25	1677.7	54.73
LABS	PT2	45.37	92.33	24.65	1175.5	78.55
LABS	PT3	45.46	92.36	30.67	914.4	101.01
LABS	PT4	43.28	92.40	28.92	686.5	134.59
VISION	LT	5.86	95.82	21.25	1211.4	79.10
VISION	FS	9.17	93.44	13.13	1021.8	91.45
VISION	EX4	9.68	93.15	17.67	862.9	107.95
VISION	EX5	2.06	98.53	22.41	855.3	115.21
VISION	EX6	0.55	99.59	16.43	548.7	181.52
VISION	EX7	0.73	99.50	19.03	597.8	166.44
VISION	EX8	1.69	98.83	21.00	607.3	162.74
VISION	PT0	0.00	100.00	5.14	675.4	148.06
VISION	PT1	0.04	99.98	5.14	662.9	150.82
VISION	PT2	0.02	99.99	5.28	624.6	160.07
VISION	PT3	0.19	99.87	5.81	662.6	150.73
VISION	PT4	0.01	99.99	6.20	646.4	154.69

dimension certificate from Section 7 to check the quality of the generated cuts and determine the portion of them that are facet-defining.

Table 1 reports the root-relaxation performance across all alternatives. Column “Gap” is the average root gap left after separation, while “C%” reports the average percentage of the initial root gap closed by the generated cuts [21]: for each instance, if SR is the root objective value before adding our cuts, DB is the root objective value after separation, and $PRIMAL$ is the reference integer objective value, then $C\% = 100 \cdot \frac{DB - SR}{PRIMAL - SR}$. Column “Time” is the mean root-separation time, while “CM” measures the average number of cuts generated, and “C% / 1000” reports the mean percentage of gap closed divided by the mean number of cuts in thousands.

We first note that LT and FS have similar performance in both datasets, which is the intended validation outcome. This result shows that when the DD separator is given the same structural information as the literature separator, it recovers comparable root behavior. The smaller number of cuts in FS is likely due to cut collisions: if two literature structures share part of their support, DD separation may yield the same inequality over the shared region, so fewer distinct cuts are added.

In contrast, we observe significant improvement when using the DDs to represent more complex structures. On LABS, the PT family dominates: PT4 gives the smallest final root gap, closing 92.40% of the initial gap with only

686.5 cuts on average, and it also has the largest Gap closed per 1000 cuts value. The EX family also improves substantially over LT: EX4 reduces the final average gap from 437.15 to 265.42, and EX7 yields the smallest EX final gap. On VISION, the result is even sharper: PT0 closes 100.00% of the root gap across all tested instances, yielding perfect root relaxation on this benchmark set under our experimental configuration. The EX methods also nearly close the gap, with EX6 reaching 99.59%, but PT0 obtains the tightest relaxation with the fastest average separation time.

We also observed that stronger supports change the nature of the generated cuts. In the EX family, increasing the expansion generally reduces the number of cuts while increasing the gap closed per 1000 cuts. This suggests that richer local DDs produce stronger individual inequalities, leaving fewer violated structures available after each round. The same pattern is even clearer in the PT family on LABS: as the q parameter increases, the method produces fewer cuts but achieves higher efficiency. Thus, the larger local DDs are not merely producing more inequalities; they are producing more informative inequalities. Moreover, we observed that the computation time (i.e., column “Time”) of all DD-based strategies is similar, reinforcing the practical benefits of DD-based cut generation strategies.

To further analyze the quality of the generated cuts, we apply our DD-based dimension certificate procedure (i.e., Algorithms 3 and 4) to all gener-

Table 2 Facet-audit summary on tested instances.

Dataset	Method	Max %	I_K F%	I_K R%	G F%	GK %
LABS	LT	27.4	0.0	53.3	0.0	85.7
LABS	FS	22.4	0.0	53.5	0.0	91.3
LABS	EX4	38.3	31.6	90.9	31.6	97.5
LABS	EX5	67.0	64.2	92.6	64.2	98.7
LABS	EX6	99.8	97.8	92.9	97.8	99.6
LABS	EX7	99.8	98.3	93.0	98.3	99.2
LABS	EX8	99.8	98.3	93.0	98.3	99.7
LABS	PT0	92.4	92.4	93.3	92.4	100.0
LABS	PT1	95.0	94.9	95.0	94.9	100.0
LABS	PT2	96.7	96.6	96.6	96.6	100.0
LABS	PT3	96.6	96.5	97.2	96.5	100.0
LABS	PT4	96.8	96.6	97.7	96.6	99.9
VISION	LT	60.0	44.0	74.8	44.0	87.5
VISION	FS	57.7	51.8	79.6	51.8	95.0
VISION	EX4	52.4	49.1	90.2	49.1	97.3
VISION	EX5	78.6	52.6	79.0	52.7	78.9
VISION	EX6	93.0	65.0	86.0	66.3	77.4
VISION	EX7	100.0	59.5	88.1	67.2	67.5
VISION	EX8	100.0	60.4	88.4	68.1	68.3
VISION	PT0	86.7	52.5	88.9	59.1	72.5
VISION	PT1	93.0	53.1	88.5	60.6	67.9
VISION	PT2	95.5	57.1	88.8	64.5	69.3
VISION	PT3	94.2	55.8	88.9	62.7	69.2
VISION	PT4	96.2	58.8	89.3	65.8	70.0

ated cuts and report the results in Table 2. Column “Max %” is the percentage of audited cuts whose support is maximal in the selected local hypergraph. Column “ I_K F%” is the percentage certified as facet-defining for the local I_K relaxation by Algorithm 4. Column “ I_K R%” measures the average ratio between the certified face dimension and the target local dimension, and “G F%” is a conservative lower bound on the percentage of cuts certified as globally facet-defining. Lastly, “GK %” reports the percentage of audited cuts for which the global status could be resolved within the audit budget. If the I_K test does not certify facetness, a global audit is attempted only when the corresponding global DD can be built and audited within a 60s limit; otherwise, the global status is left unresolved. The diagnostic audit time is excluded from the reported algorithmic wall-clock times.

Table 2 supports the results reported in Table 1. On LABS, LT and FS generate many cuts, but essentially none are certified as I_K facets. By contrast, EX6 and larger expansions produce cuts that are almost always maximal and overwhelmingly I_K -facet-defining, while the PT variants achieve similarly strong certification rates. On VISION, the audit gives a more nuanced but still positive picture. The strongest expanded supports, especially EX7 and EX8, reach 100.0% maximality and have the largest global-certification lower bounds among the EX methods, but their global status is often unresolved within the conservative 60s audit budget. The PT methods have lower local facet rates than on LABS, but still certify a large fraction of cuts and, as Table 1 shows, PT0 closes the root gap completely on the tested VISION instances. Taken together, the root and audit results support the main claim of the paper: DD separation is most useful not as another named family of inequalities, but as a black-box mechanism that can take literature-derived structures, expand the support seen by the separator, and produce stronger inequalities than the original structures alone.

8.3 B&B performance

We next evaluate whether the stronger root relaxations translate into better B&B performance. To ensure a clean comparison, these experiments consider cuts added at the root node by a cutting-plane procedure; that is, root separation is performed on an explicit LP relaxation before tree search begins. Thus, all found cuts are added to this LP model, and we then run Gurobi’s B&B procedure over its MILP version.

Table 3 reports B&B performance of all procedures except FS, which was evaluated only at the root node as a validation benchmark. In the table, column “Solved” gives the number of instances solved within the 3600s limit. Column “Time” corresponds to the average solve time computed only over solved runs, “Nodes” reports the mean number of B&B nodes, and “Gap %” is the mean final gap percent computed only over instances that hit the time limit. The last column repeats the root gap closed (i.e., “C%”), allowing us to compare root strength against downstream tree-search behavior.

Table 3 Branch-and-bound performance on tested instances.

Dataset	Method	Solved	Time	Nodes	Gap %	C%
LABS	LT	11	711.49	1717587	106.38	60.97
LABS	EX4	10	466.93	1412006	60.54	75.58
LABS	EX5	10	548.47	1768355	58.04	78.65
LABS	EX6	10	575.08	1779272	49.98	78.49
LABS	EX7	11	687.33	1667117	54.06	79.28
LABS	EX8	10	520.01	1771057	52.00	78.06
LABS	PT0	11	170.17	150556	37.17	91.31
LABS	PT1	11	492.98	72437	51.48	91.82
LABS	PT2	11	689.01	47030	69.64	92.33
LABS	PT3	10	496.79	79766	47.30	92.36
LABS	PT4	10	527.26	57131	44.09	92.40
VISION	LT	40	71.79	96337	2.41	95.82
VISION	EX4	36	218.73	573610	4.61	93.15
VISION	EX5	45	31.16	1342	–	98.53
VISION	EX6	45	17.36	2	–	99.59
VISION	EX7	45	19.99	4	–	99.50
VISION	EX8	45	22.05	6	–	98.83
VISION	PT0	45	5.76	1	–	100.00
VISION	PT1	45	5.89	2	–	99.98
VISION	PT2	45	6.03	1	–	99.99
VISION	PT3	45	7.08	1	–	99.87
VISION	PT4	45	7.35	1	–	99.99

On LABS instances, PT0 is the strongest practical B&B setting in terms of instances solved, time, and gap. This is notable because PT0 is not the strongest PT configuration at the root. The result suggests that the cuts that close the most root gap are not necessarily the cuts that produce the best B&B behavior. In particular, this result can be explained by the fact that PT0 yields significantly more cuts than the other PT strategies (see Table 1), which, in turn, can help the B&B algorithm prove optimality. Nonetheless, we note that all PT strategies perform quite well when compared to the EX strategies and the LT baseline.

On VISION, the result obtained at the root node essentially carries over to B&B. PT strategies solve all 45 instances at the root node and yield the fastest solve times. This is consistent with the root table, where PT0 closes 100.00% of the root gap on VISION, and the other alternatives more than 99.8%. The EX methods also perform strongly once the expansion is large enough: EX5–EX8 solve all instances, and EX6 is the fastest expanded-support method. Overall, the B&B results reinforce the main computational message: DD support expansion improves performance, but the best strategy is not necessarily the one that maximizes root closure.

Figure 3 summarizes the same behavior across the two datasets. The plot depicts only two strategies per family (i.e., EX4, EX7, PT0, PT4) to avoid overcrowding. The left part of the plot shows the cumulative number of solved instances over time, while the right part is a cumulative gap plot for the instances that reach the time limit. Once again, we see that the PT strategies

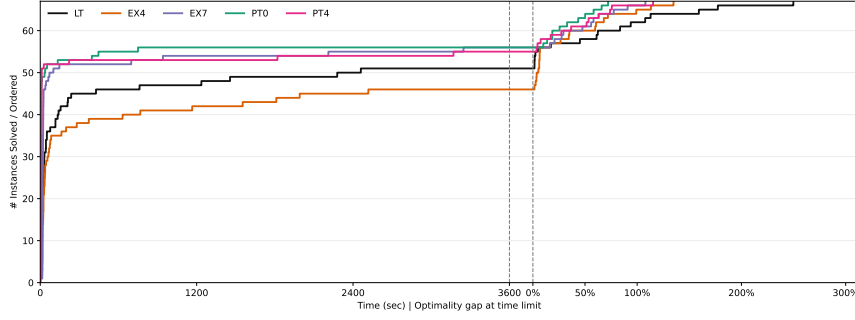


Fig. 3 B&B profile for representative methods.

and the larger EX excel in both runtime and optimality gap compared to our baseline LT. Also, it is interesting to see that EX4 has poor performance, which is mostly explained by the fact that the underlying DDs do not represent structures as rich as those of the other alternatives, thus, providing weaker cuts as shown in Tables 1 and 2.

8.4 Discussion

These results show that the main gains from our procedures stem from strengthening the support prior to separation. Expanded supports improve substantially over the fixed-support baseline as M increases, and the partition supports are the strongest family overall. On LABS, the PT family yields the best root gaps and the best B&B behavior among the tested DD-based strategies; in particular, PT0 achieves the fastest solved-instance time and the smallest remaining gap among the unsolved instances, while deeper partition levels close slightly more of the root gap but do not translate into better B&B performance. On VISION, the same pattern is even sharper: PT0 closes the full root gap on all tested instances and gives the fastest solve times. On VISION, EX6 is the strongest configuration among the EX variants, but the partition supports dominate the overall operational comparison.

In a nutshell, DDs provide a mechanism for turning a chosen support into a separation oracle. When that support is the classical literature structure, the method validates itself by producing cuts of comparable strength. When the support is expanded or repartitioned, the same mechanism can generate stronger cuts than the original structure alone could. This supports the interpretation of DDs as a black-box strengthening device: rather than deriving a new family of inequalities by hand for every enlarged structure, we specify the structure and let the DD-based target-cut model separate over it.

9 Conclusions

This paper studied binary polynomial optimization (BPO) through the lens of its multilinear polytope and developed a decision-diagram (DD) framework for generating cutting planes over local structures. We presented a compact DD encoding of the multilinear set that considers only vertex variables and implicitly recovers the hyperedge variables along states and arcs. This encoding yields an extended network-flow formulation of MP_G and supports a target-cut separation model adapted to the implicitly assigned hyperedge variables. We analyzed the size of this construction via a valid-antichain characterization of its width, which allowed us to identify hypergraph structures—including flowers and cycles—that admit bounded-width DDs, as well as a new family of hypergraphs for which BPO is solvable in polynomial time. Building on this machinery, we proposed three strategies for choosing the local support fed to the separator: a fixed-support baseline that mirrors known inequalities in the literature, an expanded-support rule that grows a seed by overlap, and a partition-support scheme based on section hypergraphs. Finally, we introduced a novel DD-based dimension and facet-audit procedure to certify the polyhedral quality of the generated cuts.

Computationally, on the LABS and VISION datasets, the fixed-support DD separator reproduces the behavior of the corresponding literature cuts, confirming that the target-cut model recovers comparable strength when given the same local information. The main gains, however, come from strengthening the support before separation: expanded and, especially, partition supports close substantially more of the root gap with far fewer cuts—on LABS the partition family closes up to 92.4% of the root gap, and on VISION it closes the gap almost completely on every tested instance—and these stronger relaxations translate into a faster branch-and-bound, with the partition setting solving all VISION instances and giving the best practical performance on LABS. The facet audit corroborates this picture: strengthened supports yield cuts that are far more often maximal and facet-defining than those from the literature. Taken together, the results support the central message of the paper: a DD is best viewed not as another hand-derived family of inequalities, but as a black-box strengthening device that turns any chosen support into a facet-producing separation oracle.

Several limitations remain. The current implementation generates cuts only at the root, uses offline support choices, and relies on heuristic selection of structure. Future work should integrate the separator into a full branch-and-cut routine, combine DD cuts with the solver’s native cuts, and develop adaptive criteria for selecting supports during the solve. Another promising direction is to use the relatively cheap I_K -facet audit as a cut-selection tool, prioritizing cuts that are not only violated but also locally facet-defining. Finally, the randomized nature of the PT family suggests studying multi-seed or adaptive partitioning schemes that preserve PT0’s strong performance while improving robustness.

More broadly, the same approach could be applied to other binary optimization problems where strong inequalities arise from small but meaningful supports and, thus, enhance modern solvers such as Gurobi. Moreover, our procedure can be seen as a research tool in its own right. Specifically, it can help researchers identify interesting problem structures (i.e., for BPO or other problems), obtain strong cuts from them, and then infer properties and develop algorithms to generate those same cuts without needing to create a DD.

Acknowledgements This research was partially supported by the National Center for Artificial Intelligence CENIA FB210017 (Basal ANID), and the *Agencia Nacional de Investigación y Desarrollo* of Chile under grant ANID-FONDECYT-1261833.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. Barahona, F., Grötschel, M., Jünger, M., Reinelt, G.: An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research* **36**(3), 493–513 (1988)
2. Barahona, F., Mahjoub, A.R.: On the cut polytope. *Mathematical Programming* **36**(2), 157–173 (1986)
3. Becker, B., Behle, M., Eisenbrand, F., Wimmer, R.: Bdds in a branch and cut framework. In: *International Workshop on Experimental and Efficient Algorithms*, pp. 452–463. Springer (2005)
4. Behle, M.: Binary decision diagrams and integer programming. Ph.D. thesis, Saarland University (2007)
5. Bergman, D., Cire, A.A., van Hoeve, W.J., Hooker, J.: *Decision Diagrams for Optimization*. Springer (2016). DOI 10.1007/978-3-319-42849-9
6. Boros, E., Hammer, P.L.: Pseudo-Boolean optimization. *Discrete Applied Mathematics* **123**(1-3), 155–225 (2002)
7. Buchheim, C., Liers, F., Oswald, M.: Local cuts revisited. *Operations Research Letters* **36**(4), 430–433 (2008)
8. Bussieck, M.R., Drud, A.S., Meeraus, A.: MINLPLib—a collection of test models for mixed-integer nonlinear programming. *INFORMS Journal on Computing* **15**(1), 114–119 (2003)
9. Capelli, F., Del Pia, A., Di Gregorio, S.: A knowledge compilation take on binary polynomial optimization. *Mathematical Programming* pp. 1–42 (2026)
10. Casassus, N., Castro, M., Angulo, G.: A decision diagram approach for the parallel machine scheduling problem with chance constraints. *arXiv preprint arXiv:2504.20889* (2025)
11. Castro, M.P., Cire, A.A., Beck, J.C.: A combinatorial cut-and-lift procedure with an application to 0–1 second-order conic programming. *Mathematical Programming* **196**(1–2), 115–171 (2022). DOI 10.1007/s10107-021-01699-y
12. Castro, M.P., Cire, A.A., Beck, J.C.: Decision diagrams for discrete optimization: A survey of recent advances. *INFORMS Journal on Computing* **34**(4), 2271–2295 (2022). DOI 10.1287/ijoc.2022.1170
13. Crama, Y., Rodríguez-Heck, E.: A class of valid inequalities for multilinear 0–1 optimization problems. *Discrete Optimization* **25**, 28–47 (2017). DOI 10.1016/j.disopt.2017.02.001

14. Davarnia, D., van Hoeve, W.J.: Outer approximation for integer nonlinear programs via decision diagrams. *Mathematical Programming* **187**(1), 111–150 (2021). DOI 10.1007/s10107-020-01475-4
15. Del Pia, A.: POLIP: Polynomial integer programming library. <https://sites.google.com/site/delpiaprojects/polip>. Accessed 2026-06-07
16. Del Pia, A., Di Gregorio, S.: Chvátal rank in binary polynomial optimization. *INFORMS Journal on Optimization* **3**(4), 315–349 (2021). DOI 10.1287/ijoo.2019.0049
17. Del Pia, A., Khajavirad, A.: A polyhedral study of binary polynomial programs. *Mathematics of Operations Research* **42**(2), 389–410 (2017). DOI 10.1287/moor.2016.0804
18. Del Pia, A., Khajavirad, A.: The multilinear polytope for acyclic hypergraphs. *SIAM Journal on Optimization* **28**(2), 1049–1076 (2018). DOI 10.1137/16M1095998
19. Del Pia, A., Khajavirad, A.: The running intersection relaxation of the multilinear polytope. *Mathematics of Operations Research* **46**(3), 1008–1037 (2021). DOI 10.1287/moor.2021.1121
20. Del Pia, A., Khajavirad, A., Sahinidis, N.V.: On the impact of running-intersection inequalities for globally solving polynomial optimization problems. *Mathematical Programming Computation* **12**(2), 165–191 (2020). DOI 10.1007/s12532-019-00169-z
21. Del Pia, A., Walter, M.: Simple odd beta-cycle inequalities for binary polynomial optimization. *Mathematical Programming* **206**, 203–238 (2024). DOI 10.1007/s10107-023-01992-y
22. Diestel, R.: *Graph Theory*, 5 edn. Springer (2017). DOI 10.1007/978-3-662-53622-3
23. Fortet, R.: L’algèbre de Boole et ses applications en recherche opérationnelle. *Trabajos de Estadística* **11**(2), 111–118 (1960). DOI 10.1007/BF03006558
24. Glover, F., Woolsey, E.: Converting the 0-1 polynomial programming problem to a 0-1 linear program. *Operations Research* **22**(1), 180–182 (1974)
25. van Hoeve, W.J.: An introduction to decision diagrams for optimization. In: *Tutorials in Operations Research*. INFORMS (2024)
26. Kinnersley, N.G.: The vertex separation number of a graph equals its path-width. *Information Processing Letters* **42**(6), 345–350 (1992). DOI 10.1016/0020-0190(92)90234-M
27. Kochenberger, G., Hao, J.K., Glover, F., Lewis, M., Lü, Z., Wang, H., Wang, Y.: The unconstrained binary quadratic programming problem: a survey. *Journal of Combinatorial Optimization* **28**(1), 58–81 (2014)
28. Lozano, L., Smith, J.C.: A binary decision diagram based algorithm for solving a class of binary two-stage stochastic programs. *Mathematical Programming* **191**(1), 381–404 (2022)
29. MacNeil, M., Bodur, M.: Leveraging decision diagrams to solve two-stage stochastic programs with binary recourse and logical linking constraints. *European Journal of Operational Research* **315**(1), 228–241 (2024)
30. Provan, J.S., Ball, M.O.: The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing* **12**(4), 777–788 (1983). DOI 10.1137/0212053
31. Stanley, R.P.: *Enumerative Combinatorics*, vol. 1, 2 edn. Cambridge University Press (2012). DOI 10.1017/CBO9781139058520
32. Tjandraatmadja, C., van Hoeve, W.J.: Target cuts from relaxed decision diagrams. *INFORMS Journal on Computing* **31**(2), 285–301 (2019). DOI 10.1287/ijoc.2018.0830