

Network flow-based cuts for binary bilevel programs

Sebastián Vásquez^{*1}, Leonardo Lozano^{†2}, and Willem-Jan van Hoeve^{‡1}

¹Carnegie Mellon University

²University of Cincinnati

Abstract

Binary bilevel programs are challenging to solve due to the lack of compact single-level reformulations. Recent extended network flow-based reformulations have emerged as a promising approach, providing strong dual bounds. However, their increasing size can make them computationally prohibitive. We develop a cutting-plane framework that exploits these networks without requiring an extended-space formulation. We introduce two exact network flow-based reformulations and provide an efficient separation procedure for inequalities that certify the follower’s optimality. We further strengthen these inequalities through a tilting procedure that leverages the combinatorial structure of the networks. Computational experiments show that the proposed approach becomes increasingly advantageous as networks grow larger or more numerous. Moreover, incorporating the cuts into state-of-the-art bilevel solvers improves solution times and optimality gaps on BOBILib instances, closing eight more open instances than the baseline solvers. These results demonstrate the effectiveness of using network representations as separation oracles for binary bilevel optimization.

Keywords: Binary bilevel programming, cutting planes, network flows

Funding: Dr. Lozano was supported by AFOSR grant FA9550-25-1-0301.

1 Introduction

Bilevel optimization represents hierarchical interactions between two decision-makers, commonly referred to as the leader and the follower, who optimize distinct objective functions subject to individual and coupling constraints [Bracken and McGill, 1973]. The leader has complete knowledge of the follower’s problem, and only controls its own variables, while the follower selects an optimal response after observing the leader’s decisions. If the follower admits multiple optimal responses, under the optimistic setting the follower selects one that is most favorable to the leader, whereas under the pessimistic setting, it selects one that is least favorable. The overall problem consists in finding a leader decision that optimizes the leader’s objective while accounting for the follower’s optimal reaction.

For continuous bilevel programs, single-level reformulations can often be obtained by replacing the follower’s problem with its Karush-Kuhn-Tucker conditions or by exploiting strong duality. These reformulations typically yield nonlinear programs or mathematical programs with complementarity

^{*}savasque@andrew.cmu.edu

[†]leolozano@uc.edu

[‡]vanhoeve@andrew.cmu.edu

constraints [Fortuny-Amat and McCarl, 1981, Bard and Moore, 1990, Zare et al., 2019]. Such techniques, however, do not directly apply to problems with integer follower decisions, which makes integer bilevel programs substantially more difficult to reformulate and solve. State-of-the-art exact methods for integer bilevel optimization generally extend mixed-integer programming techniques by deriving and separating logic-based and intersection cuts [Fischetti et al., 2017, Tahernejad et al., 2020]. Other recent papers leverage network flow-based representations of the follower’s problem to obtain single-level reformulations and relaxations, and have achieved competitive results [Lozano et al., 2022, 2026, Vásquez et al., 2025]. The main source of computational overhead of the latter approaches is the impact of the size of the networks, as the reformulations are parametrized by the components of the network. Hence, if the network grows exponentially large, so will the reformulations.

In this work, we provide an alternative method to recover the objective value bound obtained with the decision diagram approach in Lozano et al. [2026] through valid inequalities. This allows to keep the single-level relaxation on the original variable space, improving its computational tractability. We first propose two new exact network flow-based single-level reformulations of the binary bilevel program. Then, we show how to separate the inequalities required to convexify the set of follower’s optimal responses. This, in turn, allows to design a cutting plane approach that inherits the properties proposed in Lozano et al. [2026] to yield dual bounds by using size-controlled networks. Finally, inspired by the cut strengthening techniques in Behle [2007] and Castro et al. [2022], we exploit the combinatorial structure of the networks to improve our cuts via tilting. We note that the proposed method is related to the one in Lefebvre and Schmidt [2025], where the dual of a Dantzig-Wolfe representation of the follower’s feasible solution set is used to build a bilinear single-level reformulation. In contrast, we work on an extended primal space induced by flow variables over networks. This allows to design exact linear reformulations, avoiding nonlinearities. Our method is also related to the one in Chen et al. [2025], which focuses on recovering a Dantzig-Wolfe reformulation bound via linear inequalities. However, in the bilevel setting, the feasible solution set is not explicitly represented, hindering the direct construction of outer approximations. The network flow encoding provides a more compact approximation of the follower’s optimal value, while exhibiting a combinatorial structure over which efficient graph algorithms can be executed to build and strengthen outer approximations of the feasible solution set.

Contributions. Our first contribution is the design of two network flow-based single-level reformulations of binary bilevel programs. They replace the follower’s optimality condition with a family of valid inequalities, for which we also provide options for their separation. Our second contribution is the strengthening of these inequalities via tilting. Inspired by the work of Castro et al. [2022], we leverage the combinatorial structure of the networks to provide a relatively efficient way to tilt valid cuts. Lastly, our third contribution is a comprehensive computational study of the performance of the new cut families. In particular, we show that our cuts are especially effective for large networks or in a decomposition setting with multiple independent networks, and assess their portability to benchmark BOBILib instances by incorporating them into state-of-the-art solvers.

Organization. The remainder of the paper is organized as follows. In Section 2, we introduce relevant definitions, notation, and previous work on network flow-based reformulations for binary bilevel optimization. In Section 3, we present our new reformulations and cuts, and explain how the latter can be separated. In Section 4, we describe how to use the networks to strengthen cuts. In Section 5, we present numerical results assessing the effectiveness of our cuts. Finally, we conclude the work in Section 6.

2 Preliminaries

We consider the binary linear bilevel program

$$\min_{x,y} c_L^\top x + c_F^\top y \quad (1a)$$

$$\text{s. t. } Ax + By \leq a \quad (1b)$$

$$x \in \{0, 1\}^{n_L} \quad (1c)$$

$$y \in \mathcal{R}(x), \quad (1d)$$

where

$$\mathcal{R}(x) := \arg \min_y \{d^\top y : Dy \leq b - Cx, y \in \mathcal{Y} \subseteq \mathbb{R}^{n_F}\}$$

is the follower's optimal response set, $\mathbb{Y}(x) = \{y \in \mathcal{Y} : Dy \leq b - Cx\}$ is the parametric follower's feasible solution set, and $\text{dom } \mathbb{Y} = \{x \in \{0, 1\}^{n_L} : \mathbb{Y}(x) \neq \emptyset\}$ is the set of binary leader decisions that render the follower's problem feasible. Note that program (1) can also model bounded integer leader variables x through binary expansion. After applying a linear transformation to make the leader variables nonnegative, we introduce a new variable $x^j \in \{0, 1\}^{\lfloor \log(u_j) \rfloor + 1}$ for each $j \in [n_L]$ such that $x_j \in \mathbb{Z}$, where $u_j > 0$ is the upper bound on x_j after the transformation. This way, by replacing x_j with $\sum_{i=0}^{\lfloor \log(u_j) \rfloor} 2^i x_i^j$, we can retrieve a binary bilevel program of the form (1).

We refer to function $\phi(x) = \min\{d^\top y : y \in \mathbb{Y}(x)\}$ as the follower's value function, which computes the optimal value of the follower's problem for a given vector x . We assume that if $x \notin \text{dom } \mathbb{Y}$, then $\phi(x) = +\infty$. The *High-Point Relaxation* (HPR) corresponds to the program obtained from relaxing (1d) from (1), *i.e.*, $\min\{c_L^\top x + c_F^\top y : (x, y) \in \mathcal{H}\}$, where $\mathcal{H} := \{(x, y) \in \{0, 1\}^{n_L} \times \mathcal{Y} : Ax + By \leq a, Cx + Dy \leq b\}$. Therefore, program (1) can be reformulated as

$$\min\{c_L^\top x + c_F^\top y : (x, y) \in \mathcal{H}, d^\top y \leq \phi(x)\}, \quad (2)$$

which is known as the *value function reformulation*. We denote as $\mathcal{F} := \{(x, y) \in \mathcal{H} : d^\top y \leq \phi(x)\}$ the set of feasible solutions to (2), also known as *bilevel-feasible set*.

Example 1. As a running example, consider the integer bilevel program in Figure 1a with follower's optimal response set $\mathcal{R}(x)$ in Figure 1b. Figure 2a illustrates the relevant sets of the example program, where black lines indicate the follower's feasible set $\mathbb{Y}(x)$ and orange dashed lines the leader's feasible set. Black dots correspond to follower feasible responses, where the green ones are optimal. Point (3, 3) is not bilevel-feasible as it violates leader's constraint $2x + 2y \leq 11$. Therefore, the remaining three points (0, 1), (1, 1), and (2, 0) form the bilevel-feasible set $\mathcal{F}$. An optimal solution is obtained by optimizing the leader's objective function over $\mathcal{F}$, which corresponds to (1, 1) with an objective value -5 . The HPR, illustrated in Figure 1c, achieves optimal solution (2, 3) with objective value -14 .

The associated binary expansion introduces new variables $x_1, x_2, y_1, y_2 \in \{0, 1\}$ which satisfy $x = x_1 + 2x_2$ and $y = y_1 + 2y_2$, yielding the binary bilevel program

$$\begin{aligned} \min \quad & -x_1 - 2x_2 - 4y_1 - 8y_2 \\ \text{s. t.} \quad & 2x_1 + 4x_2 - 2y_1 - 4y_2 \leq 3 \\ & 2x_1 + 4x_2 + 2y_1 + 4y_2 \leq 11 \\ & x_1, x_2 \in \{0, 1\} \\ & y_1 + 2y_2 \in \mathcal{R}(x_1 + 2x_2). \end{aligned}$$

$\begin{aligned} \min_{x,y} \quad & -x - 4y \\ \text{s. t.} \quad & 2x - 2y \leq 3 \\ & 2x + 2y \leq 11 \\ & x \in \mathbb{Z} \\ & y \in \mathcal{R}(x) \end{aligned}$	$\begin{aligned} \arg \min_y \quad & y \\ \text{s. t.} \quad & 3 - 2x \leq 4y \leq 18 - 2x \\ & -9 + 4x \leq y \leq 3 + 4x \\ & 4y \leq 12 + x \\ & y \in \mathbb{Z}_+ \end{aligned}$	$\begin{aligned} \min_{x,y} \quad & -x - 4y \\ \text{s. t.} \quad & 2x - 2y \leq 3 \\ & 2x + 2y \leq 11 \\ & x \in \mathbb{Z} \\ & y \in \mathbb{Y}(x) \end{aligned}$
(a) Bilevel model	(b) Follower's optimal response set $\mathcal{R}(x)$	(c) High-point relaxation

Figure 1: Example integer bilevel programming model (a), the follower's optimal response set (b), and the associated high-point relaxation (c).

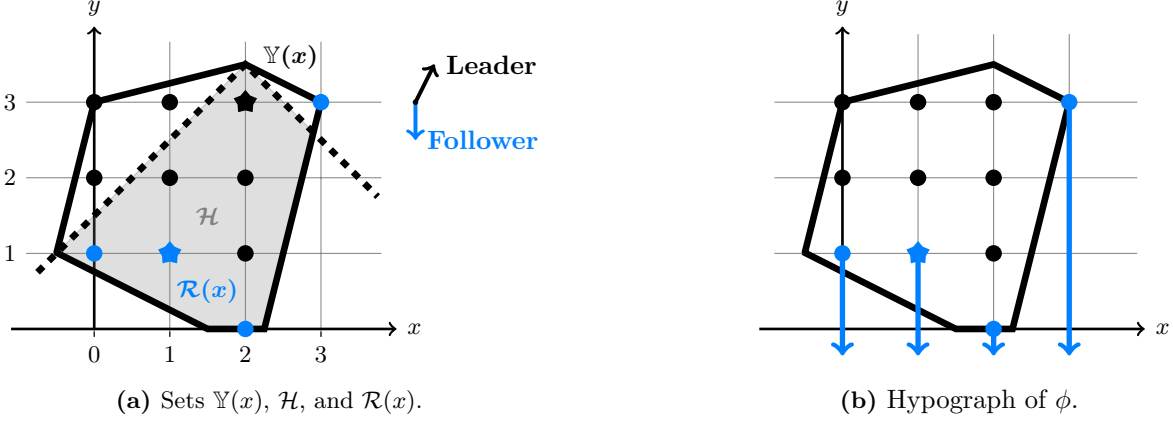


Figure 2: Sets for the example integer bilevel program. The leader's feasible set is depicted by the region enclosed by the dashed lines. The bilevel-feasible set $\mathcal{F}$ consists of pairs (x, y) in this region satisfying $y \in \mathcal{R}(x)$. In figure (b), the blue downward rays, including their upper endpoints, represent the hypograph of ϕ , containing all points $(x, d^\top y)$ with $d^\top y \leq \phi(x)$; here, $d^\top y = y$.

Its optimal solution is $(x_1, x_2, y_1, y_2) = (1, 0, 1, 0)$, corresponding to $x = 1$, $y = 1$, with objective value -5 . We keep this binary representation throughout this work for illustrative purposes. $\square$

The graph of function ϕ corresponds to the set $\text{gph } \phi = \{(x, z) \in \text{dom } \mathbb{Y} \times \mathbb{R} \cup \{+\infty\} : z = \phi(x)\}$, and the hypograph of ϕ corresponds to the set $\text{hyp } \phi = \{(x, z) \in \text{dom } \mathbb{Y} \times \mathbb{R} \cup \{+\infty\} : z \leq \phi(x)\}$. We can now write program (1) as

$$\min \left\{ c_L^\top x + c_F^\top y : (x, y) \in \mathcal{H}, (x, d^\top y) \in \text{hyp } \phi \right\}. \quad (3)$$

Figure 2b illustrates $\text{hyp } \phi$ for the follower's value function of the program in Example 1.

Computing the value function via network flows. To compute the follower's value function via linear programming, Lozano et al. [2026] propose a network flow-based reformulation. The authors first show how to compile a layered directed acyclic network $\mathcal{D}_{\text{gph } \phi} = (\mathcal{N}, \mathcal{A}, l, \nu)$, with root node r and sink node t , that encodes all points (x, z) in $\text{gph } \phi$ through r - t paths. Here, $\mathcal{N}$ is a set of nodes, $\mathcal{A}$ is a set of arcs, l is an arc-length vector, and ν is an arc-value vector. Furthermore, the arc set $\mathcal{A}$ is partitioned as $\mathcal{A} = \mathcal{A}_L^0 \cup \mathcal{A}_L^1 \cup \mathcal{A}_F$, where $\mathcal{A}_L^0$ and $\mathcal{A}_L^1$ are the set of arcs with value 0 and 1, respectively, encoding x values, and $\mathcal{A}_F$ is the set of arcs encoding z values. Moreover,

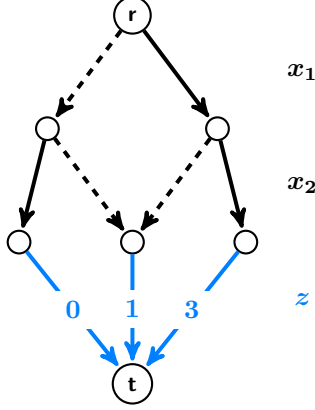


Figure 3: Network $\mathcal{D}_{\text{gph } \phi}$ used to compute the follower's value function of the running example, using the binary representation $x = x_1 + 2x_2$. Black arcs are associated with x values and blue arcs with z values. Dashed arcs represent 0-arcs and solid arcs represent 1-arcs. Arc labels represent lengths.

arc lengths in l are set to 0 for arcs in $\mathcal{A}_L = \mathcal{A}_L^1 \cup \mathcal{A}_L^0$, and to the evaluation of ϕ for arcs in $\mathcal{A}_F$. For any arc $a \in \mathcal{A}_L$, we denote as $\mathbf{i}(a)$ the component of x associated with a . This way, for any point $(\hat{x}, \hat{z}) \in \text{gph } \phi$, there is exactly one r - t path $\mathcal{P} = \{a_1, \dots, a_{n_L+1}\}$ such that $\hat{x}_{\mathbf{i}(a)} = \nu_a$ for all $a \in \mathcal{P} \setminus \{a_{n_L+1}\}$ and $\hat{z} = \sum_{a \in \mathcal{P}} l_a = l_{a_{n_L+1}}$. Figure 3 depicts the network that encodes the graph of the follower's value function associated with the binary expansion of the problem in the running example 1. Given $\mathcal{D}_{\text{gph } \phi}$, we have that

$$\min c_L^\top x + c_F^\top y \quad (4a)$$

$$\text{s. t. } (x, y) \in \mathcal{H} \quad (4b)$$

$$\sum_{a \in \delta^+(u)} w_a - \sum_{a \in \delta^-(u)} w_a = \mathbf{1}(u = r) - \mathbf{1}(u = t) \quad u \in \mathcal{N} \quad (4c)$$

$$x_j = \sum_{a \in \mathcal{A}_L: \mathbf{i}(a)=j} \nu_a w_a \quad j \in [n_L] \quad (4d)$$

$$d^\top y \leq l^\top w \quad (4e)$$

$$w \geq 0. \quad (4f)$$

is an exact reformulation of program (3), where $\mathbf{1}(\cdot)$ denotes the indicator function. Note that directly solving program (4) with an off-the-shelf MIP solver may quickly become intractable as $\mathcal{D}_{\text{gph } \phi}$ grows. To address this issue, Lozano et al. [2026] propose relaxations obtained from replacing $\mathcal{D}_{\text{gph } \phi}$ in (4) with an approximate value network. This corresponds to another network $\mathcal{D}_{\text{gph } \hat{\phi}}$ that encodes points in $\text{gph } \hat{\phi}(x)$, for some upper bounding function $\hat{\phi} \geq \phi$. This approach leads to a trade-off between network size and bound quality: in general, larger networks provide stronger bounds, as they encode the graph of more accurate approximations of ϕ . Thus, as long as compilation remains tractable, it is desirable to work with larger networks.

3 Network flow-based reformulations

We introduce two alternative network flow-based reformulations of (5) based on $\mathcal{D}_{\text{gph } \phi}$. Both reformulations can be systematically outer approximated via linear inequalities obtained from $\mathcal{D}_{\text{gph } \phi}$, which allows to use the network as an oracle and, thus, address the tractability issues of directly

solving program (4). First, we state a simple yet important result that serves as the cornerstone of our reformulations. We refer to $\text{conv } \mathcal{X}$ as the convex hull of set $\mathcal{X}$.

Lemma 1. $\text{hyp } \phi = \text{conv}(\text{hyp } \phi) \cap (\{0, 1\}^{n_L} \times \mathbb{R})$

Proof. Let $\mathcal{S} := \text{hyp } \phi$ and $\overline{\mathcal{S}} := \text{conv}(\text{hyp } \phi) \cap (\{0, 1\}^{n_L} \times \mathbb{R})$. We want to prove $\mathcal{S} = \overline{\mathcal{S}}$. We know $\mathcal{S} \subseteq \overline{\mathcal{S}}$, so we show $\mathcal{S} \supseteq \overline{\mathcal{S}}$. Let $(\bar{x}, \bar{z}) \in \overline{\mathcal{S}}$ and consider a finite set $\mathcal{X} \subseteq \text{hyp } \phi$ such that $(\bar{x}, \bar{z}) = \sum_{(x,z) \in \mathcal{X}} \lambda_{(x,z)}(x, z)$ for some vector $\lambda \in \mathbb{R}_+^{|\mathcal{X}|}$. As $\bar{x} \in \{0, 1\}^{n_L}$, it is an extreme point of the box $[0, 1]^{n_L}$ and, hence, $x = \bar{x}$ for all $(x, z) \in \mathcal{X}$. This way, we obtain $\bar{z} = \sum_{(x,z) \in \mathcal{X}} \lambda_{(x,z)} z \leq \sum_{(x,z) \in \mathcal{X}} \lambda_{(x,z)} \phi(x) = \phi(\bar{x})$. Thus, $(\bar{x}, \bar{z}) \in \text{hyp } \phi$, concluding the proof. $\square$

As $x \in \{0, 1\}^{n_L}$ is implied by $(x, y) \in \mathcal{H}$, Lemma 1 provides a relaxation of program (3) which is tight at binary x values.

Corollary 1. *Program*

$$\min \quad c_L^\top x + c_F^\top y \tag{5a}$$

$$\text{s. t. } (x, y) \in \mathcal{H} \tag{5b}$$

$$(x, d^\top y) \in \text{conv}(\text{hyp } \phi). \tag{5c}$$

is an exact reformulation of program (3).

Observe that, contrary to $\text{hyp } \phi$, the set $\text{conv}(\text{hyp } \phi)$ is polyhedral, as x can take finitely many values. This allows to build outer approximations via linear inequalities for program (5). In particular, constraints (4c)-(4f) from the network flow-based reformulation proposed by Lozano et al. [2026] model $(x, d^\top y) \in \text{conv}(\text{hyp } \phi)$ on an extended variable space, where flow variables w act as convex multipliers. Moreover, if $\mathcal{D}_{\text{gph } \phi}$ is replaced with an approximate value network $\mathcal{D}_{\text{gph } \hat{\phi}}$ in program (4), then via Corollary 1, we can conclude that the resulting relaxation represents the following program

$$\begin{aligned} \min \quad & c_L^\top x + c_F^\top y \\ \text{s. t. } \quad & (x, y) \in \mathcal{H} \\ & (x, d^\top y) \in \text{conv}(\text{hyp } \hat{\phi}) \supseteq \text{conv}(\text{hyp } \phi) \end{aligned}$$

but in an extended variable space.

As mentioned in Section 2, larger networks are useful since they provide better bounds on the optimal value of program (3) by yielding tighter outer approximations of $\text{conv}(\text{hyp } \phi)$. In what follows, we propose two alternative network flow-based reformulations of program (5) that, contrary to (4), do not require embedding $\mathcal{D}_{\text{gph } \phi}$. These reformulations remain in the original variable space and dynamically refine the outer approximation of $\text{conv}(\text{hyp } \phi)$ via cuts, allowing for improved scalability.

Remark 1. Lemma 1 requires x to be binary; Figure 2b serves as a counterexample for the general integer case, since $(x, z) = (2, 2)$ belongs to $\text{conv}(\text{hyp } \phi) \cap (\mathbb{Z}^{n_L} \times \mathbb{R})$, but not to $\text{hyp } \phi$.

3.1 Optimality-based reformulation

For any vector $\hat{x} \in \text{dom } \mathbb{Y} \cap \{0, 1\}^{n_L}$, we have $\phi(\hat{x}) = \max\{z : (\hat{x}, z) \in \text{conv}(\text{hyp } \phi)\}$. Therefore, given $\mathcal{D}_{\text{gph } \phi} = (\mathcal{N}, \mathcal{A}, l, \nu)$, we obtain

$$\phi(\hat{x}) = \max l^\top w \quad (6a)$$

$$\text{s. t. } \sum_{a \in \delta^+(u)} w_a - \sum_{a \in \delta^-(u)} w_a = \mathbf{1}(u = r) - \mathbf{1}(u = t) \quad u \in \mathcal{N} \quad (6b)$$

$$\sum_{a \in \mathcal{A}_L : i(a)=j} \nu_a w_a = \hat{x}_j \quad j \in [n_L] \quad (6c)$$

$$w \geq 0. \quad (6d)$$

By strong duality, we can reformulate (2) as

$$\min_{x, y} \left\{ c_L^\top x + c_F^\top y : \begin{array}{l} (x, y) \in \mathcal{H}, \\ d^\top y \leq \min_{\pi, \lambda} \left\{ \pi_r + x^\top \lambda : (\pi, \lambda) \in \mathcal{P}(\mathcal{D}_{\text{gph } \phi}) \right\} \end{array} \right\},$$

where

$$\mathcal{P}(\mathcal{D}_{\text{gph } \phi}) := \left\{ (\pi, \lambda) \in \mathbb{R}^{|\mathcal{N}|} \times \mathbb{R}^{n_L} : \begin{array}{ll} \pi_u - \pi_v + \nu_{uv} \lambda_{i(u,v)} \geq 0 & (u, v) \in \mathcal{A}_L, \\ \pi_u - \pi_v \geq l_{uv} & (u, v) \in \mathcal{A}_F, \\ \pi_t = 0 \end{array} \right\}$$

corresponds to the feasible solution set of the dual of program (6). This way, we obtain the exact reformulation

$$\min c_L^\top x + c_F^\top y \quad (7a)$$

$$\text{s. t. } (x, y) \in \mathcal{H} \quad (7b)$$

$$d^\top y \leq \pi_r + \lambda^\top x \quad (\pi, \lambda) \in \mathcal{P}(\mathcal{D}_{\text{gph } \phi}). \quad (7c)$$

Proposition 1. *Program (7) is an exact reformulation of (5).*

Proof. As both programs consider $(x, y) \in \mathcal{H}$, we focus on comparing $(x, d^\top y) \in \text{conv}(\text{hyp } \phi)$ with constraints (7c). We first prove that any solution $(\hat{x}, \hat{y})$ feasible to program (7) is also feasible to program (5). Equivalently, we show that if $(\hat{x}, d^\top \hat{y}) \notin \text{conv}(\text{hyp } \phi)$, then there exists a dual vector (π, λ) for which $(\hat{x}, \hat{y})$ violates (7c). Since $(\hat{x}, \hat{y}) \in \mathcal{H}$, then we have $d^\top \hat{y} > \phi(\hat{x})$. Due to strong duality, there exists a vector $(\hat{\pi}, \hat{\lambda}) \in \arg \min\{\pi_r + \hat{x}^\top \lambda : (\pi, \lambda) \in \mathcal{P}(\mathcal{D}_{\text{gph } \phi})\}$ such that $\phi(\hat{x}) = \hat{\pi}_r + \hat{\lambda}^\top \hat{x}$. Hence, we get $d^\top \hat{y} > \phi(\hat{x}) = \hat{\pi}_r + \hat{\lambda}^\top \hat{x}$, so $(\hat{x}, \hat{y})$ is not feasible to program (7).

We now prove that any solution $(\hat{x}, \hat{y})$ feasible to program (5) is also feasible to program (7). By weak duality, we have $d^\top \hat{y} \leq \phi(\hat{x}) \leq \min\{\pi_r + \hat{x}^\top \lambda : (\pi, \lambda) \in \mathcal{P}(\mathcal{D}_{\text{gph } \phi})\}$. Hence, $(\hat{x}, \hat{y})$ is feasible to program (7). $\square$

In this work, we refer to inequalities (7c) as *optimality cuts*. Observe that optimality cuts can be dynamically separated by solving the linear program $\min\{\pi_r + \lambda^\top x : (\pi, \lambda) \in \mathcal{P}(\mathcal{D}_{\text{gph } \phi})\}$. However, we can achieve a more efficient algorithm by exploiting $\mathcal{D}_{\text{gph } \phi}$. By adding (redundant) constraints

$$\sum_{a \in \mathcal{A}_L : i(a)=j} (1 - \nu_a) w_a = 1 - \hat{x}_j \quad \forall j \in [n_L]$$

to program (6) and then taking its dual, it becomes clear that the separation problem $\min\{\pi_r + \hat{x}^\top \lambda : (\pi, \lambda) \in \mathcal{P}(\mathcal{D}_{\text{gph } \phi})\}$ is equivalent to

$$\min_{\pi, \lambda^1, \lambda^0} \pi_r + \sum_{j \in [n_L]} \hat{x}_j \lambda_j^1 + (1 - \hat{x}_j) \lambda_j^0 \quad (8a)$$

$$\text{s. t. } \pi_u - \pi_v + \nu_{uv} \lambda_{\mathbf{i}(u,v)}^1 + (1 - \nu_{uv}) \lambda_{\mathbf{i}(u,v)}^0 \geq 0 \quad (u, v) \in \mathcal{A}_L \quad (8b)$$

$$\pi_u - \pi_v \geq l_{uv} \quad (u, v) \in \mathcal{A}_F \quad (8c)$$

$$\pi_t = 0. \quad (8d)$$

This equivalent program allows to fix the value of π_r without loss of optimality and, in turn, use a modification of the approach proposed in Lozano and Smith [2018] to derive a dual optimal solution.

Proposition 2 ([Vásquez et al., 2025, Proposition 3]). *Program (8) admits an optimal solution $(\pi^*, \lambda^{1*}, \lambda^{0*})$ satisfying $\pi_r^* = \phi(\hat{x})$ and $\hat{x}_j \lambda_j^{1*} = (1 - \hat{x}_j) \lambda_j^{0*} = 0$ for all $j \in [n_L]$.*

A natural optimal solution $(\hat{\pi}, \hat{\lambda}^1, \hat{\lambda}^0)$ to program (8) can be obtained by setting $\hat{\pi}_r = \phi(\hat{x})$, $\hat{\pi}_t = 0$, and $\hat{\lambda}_j^1 = M(1 - \hat{x}_j)$, $\hat{\lambda}_j^0 = M\hat{x}_j$ for all $j \in [n_L]$, where M is a large enough constant. The resulting inequality corresponds to the traditional “no-good” cut

$$-M \left[\sum_{j \in [n_L]} (1 - \hat{x}_j) x_j + \hat{x}_j (1 - x_j) \right] + d^\top y \leq \phi(\hat{x})$$

which is tight at $x = \hat{x}$, but may be relatively weak for different values. To obtain a tighter inequality, we now describe a simple procedure that leverages $\mathcal{D}_{\text{gph } \phi}$ to compute an alternative dual optimal solution satisfying the conditions from Proposition 2. We start by setting $\hat{\pi}_r = \phi(\hat{x})$ and $\hat{\pi}_t = 0$. Then, for each arc $(u, t) \in \mathcal{A}_F$, we set $\hat{\pi}_u = l_{ut}$, and for each arc $(u, v) \in \mathcal{A}_L$ such that $u \neq r$, we set $\hat{\pi}_u = \mathbf{1}(\nu_{uv} = \hat{x}_{\mathbf{i}(uv)}) \hat{\pi}_v$. Note that, as each node has at most two outgoing arcs with different value in ν , the value assignment is consistent. Furthermore, it guarantees that if there exists a (unique) u - t path encoding a subvector of $\hat{x}$, then $\hat{\pi}_u$ is equal to its length. Finally, we build $\hat{\lambda}^1$ and $\hat{\lambda}^0$ by first setting $\hat{\lambda}_j^1 = 0$ for all $j \in [n_L]$ such that $\hat{x}_j = 1$, and $\hat{\lambda}_j^0 = \hat{x}_j$ for all $j \in [n_L]$ such that $\hat{x}_j = 0$. Then, for the remaining components, we set

$$\begin{aligned} \hat{\lambda}_j^1 &= \max_{(u,v) \in \mathcal{A}_L: \mathbf{i}(u,v)=j} \{\pi_v - \pi_u\} \quad \forall j \in [n_L] : \hat{x}_j = 0, \\ \hat{\lambda}_j^0 &= \max_{(u,v) \in \mathcal{A}_L: \mathbf{i}(u,v)=j} \{\pi_v - \pi_u\} \quad \forall j \in [n_L] : \hat{x}_j = 1 \end{aligned}$$

This way, we obtain the valid inequality $(\hat{\lambda}^0 - \hat{\lambda}^1)^\top x + d^\top y \leq \hat{\pi}_r + \mathbf{1}^\top \hat{\lambda}^0$, which is also tight at $x = \hat{x}$ due to optimality of $(\hat{\pi}, \hat{\lambda}^1, \hat{\lambda}^0)$.

3.2 Feasibility-based reformulation

Consider a point $(x, z) \in \{0, 1\}^{n_L} \times \mathbb{R}$. Given a network $\mathcal{D}_{\text{gph } \phi} = (\mathcal{N}, \mathcal{A}, l, \nu)$, the program

$$\min_w 0 \tag{9a}$$

$$\text{s. t. } \sum_{a \in \delta^+(u)} w_a - \sum_{a \in \delta^-(u)} w_a = \mathbf{1}(u = r) - \mathbf{1}(u = t) \quad u \in \mathcal{N} \tag{9b}$$

$$\sum_{a \in \mathcal{A}_L: \mathbf{i}(a)=j} \nu_a w_a = x_j \quad j \in [n_L] \tag{9c}$$

$$z \leq \sum_{a \in \mathcal{A}_F} l_a w_a \tag{9d}$$

$$w \geq 0 \tag{9e}$$

is feasible if and only if $(x, z) \in \text{conv}(\text{hyp } \phi)$. Hence, for any point $(\hat{x}, \hat{z}) \notin \text{conv}(\text{hyp } \phi)$, the dual of program (9), corresponding to

$$\max_{\pi, \lambda, \gamma} \pi_r + \hat{x}^\top \lambda + \hat{z} \gamma \tag{10a}$$

$$\text{s. t. } \pi_u - \pi_v + \nu_{uv} \lambda_{\mathbf{i}(u,v)} \leq 0 \quad (u, v) \in \mathcal{A}_L \tag{10b}$$

$$\pi_u - \pi_v + l_{uv} \gamma \leq 0 \quad (u, v) \in \mathcal{A}_F \tag{10c}$$

$$\pi_t = 0 \tag{10d}$$

$$\gamma \geq 0 \tag{10e}$$

is unbounded (the feasible region contains the origin). This way, given a feasible dual solution $(\hat{\pi}, \hat{\lambda}, \hat{\gamma})$, the inequality $\hat{\pi}_r + \hat{\lambda}^\top x + \hat{\gamma} z \leq 0$ is valid for $\text{conv}(\text{hyp } \phi)$ and violated by $(\hat{x}, \hat{z})$. This leads to the exact reformulation of program (2)

$$\min c_L^\top x + c_F^\top y \tag{11a}$$

$$\text{s. t. } (x, y) \in \mathcal{H} \tag{11b}$$

$$\pi_r + \lambda^\top x + \gamma d^\top y \leq 0 \quad (\pi, \lambda, \gamma) \in \mathcal{Q}(\mathcal{D}_{\text{gph } \phi}). \tag{11c}$$

where

$$\mathcal{Q}(\mathcal{D}_{\text{gph } \phi}) = \left\{ (\pi, \lambda, \gamma) \in \mathbb{R}^{|\mathcal{N}|} \times \mathbb{R}^{n_L} \times \mathbb{R}_+ : \begin{array}{ll} \pi_u - \pi_v + \nu_{uv} \lambda_{\mathbf{i}(u,v)} \leq 0 & (u, v) \in \mathcal{A}_L, \\ \pi_u - \pi_v + l_{uv} \gamma \leq 0 & (u, v) \in \mathcal{A}_F, \\ \pi_t = 0 \end{array} \right\}$$

is the cone defining the feasible region of the dual of program (9).

Proposition 3. *Program (11) is an exact reformulation of (5).*

Proof. As both programs consider $(x, y) \in \mathcal{H}$, we focus on comparing $(x, d^\top y) \in \text{conv}(\text{hyp } \phi)$ with constraints (11c). We first prove that any solution $(\hat{x}, \hat{y})$ feasible to program (11) is also feasible to program (5). Equivalently, if $(\hat{x}, d^\top \hat{y}) \notin \text{conv}(\text{hyp } \phi)$, then there exists a ray $(\hat{\pi}, \hat{\lambda}, \hat{\gamma}) \in \mathcal{Q}(\mathcal{D}_{\text{gph } \phi})$ such that $\hat{\pi}_r + \hat{\lambda}^\top \hat{x} + \hat{\gamma} d^\top \hat{y} > 0$. Hence, $(\hat{x}, \hat{y})$ is not feasible to program (11).

We now prove that any solution $(\hat{x}, \hat{y})$ feasible to program (5) is also feasible to program (11). First, observe that

$$\text{proj}_{\pi, \lambda}(\mathcal{Q}(\mathcal{D}_{\text{gph } \phi}) \cap \{\gamma = t\}) = \begin{cases} -t \mathcal{P}(\mathcal{D}_{\text{gph } \phi}) & t > 0, \\ -\text{rec}(\mathcal{P}(\mathcal{D}_{\text{gph } \phi})) & t = 0, \end{cases}$$

where $\text{rec}(\mathcal{P}(\mathcal{D}_{\text{gph } \phi}))$ denotes the recession cone of $\mathcal{P}(\mathcal{D}_{\text{gph } \phi})$. For $t > 0$, we have that any ray $(\hat{\pi}, \hat{\lambda}, \hat{\gamma}) \in \mathcal{Q}(\mathcal{D}_{\text{gph } \phi})$ yields $(-\hat{\pi}, -\hat{\lambda}) \in \hat{\gamma}\mathcal{P}(\mathcal{D}_{\text{gph } \phi})$. By weak duality, we have $\min\{\pi_r + \lambda^\top x : (\pi, \lambda) \in \hat{\gamma}\mathcal{P}(\mathcal{D}_{\text{gph } \phi})\} \geq \hat{\gamma}\phi(\hat{x})$. Hence, we obtain that $-\hat{\pi}_r - \hat{\lambda}^\top \hat{x} \geq \hat{\gamma}\phi(\hat{x}) \geq \hat{\gamma}d^\top \hat{y}$ holds for any ray $(\hat{\pi}, \hat{\lambda}, \hat{\gamma}) \in \mathcal{Q}(\mathcal{D}_{\text{gph } \phi})$ such that $\hat{\gamma} > 0$. For $t = 0$, since $\phi(\hat{x})$ is finite, every recession direction (π, λ) satisfies $\pi_r + \lambda^\top \hat{x} \geq 0$. Therefore, every ray $(\hat{\pi}, \hat{\lambda}, 0) \in \mathcal{Q}(\mathcal{D}_{\text{gph } \phi})$ satisfies $\hat{\pi}_r + \hat{\lambda}^\top \hat{x} \leq 0$. Thus, $(\hat{x}, \hat{y})$ is feasible to program (11). $\square$

We refer to inequalities (11c) as *feasibility cuts*. The proof of Proposition 3 also shows how feasibility cuts generalize optimality cuts. In particular, every feasibility cut with $\gamma > 0$ can be normalized to obtain an optimality cut, whereas feasibility cuts with $\gamma = 0$ are induced by negatives of recession directions of $\mathcal{P}(\mathcal{D}_{\text{gph } \phi})$. Observe that feasibility cuts can be separated by directly solving the dual of program (9). Alternatively, we can use a projected subgradient approach, as proposed in Davarnia and van Hoeve [2021], to solve a Lagrangian dual of program (9). Given $(\hat{x}, \hat{z}) \notin \text{conv}(\text{hyp } \phi)$, $\mu^L \in \mathbb{R}^{n_L}$, and $\mu^F \geq 0$, define the Lagrangian function

$$\begin{aligned} \mathcal{L}(\mu^L, \mu^F) = \min_{w \geq 0} \quad & \sum_{j \in [n_L]} \mu_j^L \left(\hat{x}_j - \sum_{a \in \mathcal{A}_L^1 : i(a)=j} w_a \right) + \mu^F \left(\hat{z} - \sum_{a \in \mathcal{A}_F} l_a w_a \right) \\ \text{s. t.} \quad & \sum_{a \in \delta^+(u)} w_a - \sum_{a \in \delta^-(u)} w_a = \mathbf{1}(u = r) - \mathbf{1}(u = t) \quad u \in \mathcal{N} \end{aligned}$$

obtained by relaxing (9c)-(9d) from program (9). A supergradient of $\mathcal{L}$ is $\partial \mathcal{L}(\mu^L, \mu^F) = (\hat{x}, \hat{z}) - p$, where p is a longest path on $\mathcal{D}_{\text{gph } \phi}$ after redefining the arc length vector l using the function

$$l_a(\mu^L, \mu^F) = \begin{cases} \nu_a \mu_{i(a)}^L & \text{if } a \in \mathcal{A}_L, \\ l_a \mu^F & \text{if } a \in \mathcal{A}_F, \end{cases}$$

for all arc $a \in \mathcal{A}$. Then, we can solve the Lagrangian dual $\max\{\mathcal{L}(\mu^L, \mu^F) : \mu^F \geq 0\}$ using a projected subgradient method. Alternatively, due to the definition of the hypograph, we can set $\mu^F = 1$ and use a subgradient method. Note that, due to weak duality, any dual vector (μ^L, μ^F) yielding $\mathcal{L}(\mu^L, \mu^F) > 0$ is sufficient to build a feasibility cut violated by $(\hat{x}, \hat{y})$.

Before finishing this Section, we make a brief comment about the dimension of the faces induced by valid inequalities.

Remark 2. Let $\tilde{\mathcal{F}} := \{(x, y) : x \in \text{dom } \mathbb{Y}, y \in R(x)\}$ denote the set obtained from $\mathcal{F}$ after relaxing the leader constraints. If an inequality $\alpha^\top x + \gamma z \leq \beta$, with $\gamma > 0$, induces a face $F \subseteq \text{conv}(\text{hyp } \phi)$ such that $\dim F = k$, then $\alpha^\top x + \gamma d^\top y \leq \beta$ induces a face $\tilde{F} \subseteq \text{conv } \tilde{\mathcal{F}}$ such that $\dim \tilde{F} \geq k$. Indeed, since $\gamma > 0$, the points in F belong to $\text{conv}(\text{gph } \phi)$. Therefore, each of $k + 1$ affinely independent points in F can be lifted to a point in $\tilde{F}$. These lifted points remain affinely independent because their images under the map $(x, y) \mapsto (x, d^\top y)$ are affinely independent. Note that this dimension bound does not necessarily extend to $\text{conv } \mathcal{F}$, since the leader constraints may exclude some of the lifted points.

In what follows, we show how to strengthen structured valid inequalities via tilting by leveraging an arc length-parametrized version of $\mathcal{D}_{\text{gph } \phi}$.

4 Strengthening cuts via tilting

We next introduce a tilting procedure for structured valid inequalities inspired by the approaches in Behle [2007] and Castro et al. [2022]. This procedure computes *tilting quantities*, corresponding to the maximum component-wise increase of the left-hand side that results in a strengthened valid inequality. We further explain how to exploit the combinatorial structure of $\mathcal{D}_{\text{gph } \phi}$ to efficiently compute them.

4.1 Coefficient Tilting

We focus on valid inequalities of the form

$$\alpha^\top x + \gamma z \leq \beta, \quad (12)$$

where $\alpha \in \mathbb{R}^{n_L}$, and $\gamma, \beta \in \mathbb{R}$. Moreover, let

$$F(\alpha, \gamma, \beta) = \left\{ (x, z) \in \text{conv}(\text{hyp } \phi) : \alpha^\top \bar{x} + \gamma \bar{z} = \beta \right\}$$

be the face of $\text{conv}(\text{hyp } \phi)$ induced by the cut defined by (α, γ, β) . We want to know how much we can increase the coefficients of α while keeping the resulting inequality valid for $\text{conv}(\text{hyp } \phi)$. To do so, for each $j \in [n_L]$ and $\theta \in \{0, 1\}$, we define the *tilting quantity function* $t_j^\theta : \mathbb{R}^{n_L} \times \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}_+ \cup \{+\infty\}$ as

$$t_j^\theta(\alpha, \gamma, \beta) = \beta - \theta \alpha_j - \max_{x, z} \left\{ \gamma z + \sum_{i \in [n_L] \setminus \{j\}} \alpha_i x_i : (x, z) \in \text{conv}(\text{hyp } \phi), x_j = \theta \right\}. \quad (13)$$

Functions t_j^θ compute the maximum increment α_j can achieve while keeping the tilted inequality

$$\left[\alpha + \left(t_j^1(\alpha, \gamma, \beta) - t_j^0(\alpha, \gamma, \beta) \right) e^j \right]^\top x + \gamma z \leq \beta - t_j^0(\alpha, \gamma, \beta) \quad (14)$$

valid for $\text{conv}(\text{hyp } \phi)$, where e^j is the j -th canonical vector. Note that if $t_j^\theta(\alpha, \gamma, \beta) = +\infty$, then $x_j = 1 - \theta$ throughout $\text{conv}(\text{hyp } \phi)$, and x_j can be fixed. Hence, in what follows, we consider only components $j \in [n_L]$ for which both tilting quantities t_j^0 and t_j^1 are finite.

Proposition 4. *Let (α, γ, β) define a valid inequality of the form (12). Then, the tilted inequality (14) is valid for $\text{conv}(\text{hyp } \phi)$.*

Proof. Consider any vector $(\hat{x}, \hat{z}) \in \text{conv}(\text{hyp } \phi)$. By definition of t_j^0 , we have

$$\begin{aligned} t_j^0(\alpha, \gamma, \beta) &= \beta - \max_{x, z} \left\{ \gamma z + \sum_{i \in [n_L] \setminus \{j\}} \alpha_i x_i : (x, z) \in \text{conv}(\text{hyp } \phi), x_j = 0 \right\} \\ &\leq \beta - \sum_{i \in [n] \setminus \{j\}} \alpha_i \hat{x}_i - \gamma \hat{z}. \end{aligned}$$

Therefore, if $\hat{x}_j = 0$, then $(\hat{x}, \hat{z})$ satisfies (14). Similarly, by definition of t_j^1 , we have

$$\begin{aligned} t_j^1(\alpha, \gamma, \beta) &= \beta - \alpha_j - \max_{x, z} \left\{ \gamma z + \sum_{i \in [n_L] \setminus \{j\}} \alpha_i x_i : (x, z) \in \text{conv}(\text{hyp } \phi), x_j = 1 \right\} \\ &\leq \beta - \alpha_j - \sum_{i \in [n] \setminus \{j\}} \alpha_i \hat{x}_i - \gamma \hat{z}. \end{aligned}$$

By subtracting $t_j^0(\alpha, \gamma, \beta)$ on both sides of the inequality, we obtain that $(\hat{x}, \hat{z})$ satisfies (14). We conclude that (14) is valid for $\text{conv}(\text{hyp } \phi)$. $\square$

We now use Theorem 2 in Castro et al. [2022] to show that tilted inequalities induce higher dimensional faces.

Proposition 5. Let (α, γ, β) define a valid inequality of the form (12). Assume there exists $j \in [n_L]$ such that $t_j^\theta(\alpha, \gamma, \beta) > 0$ for some $\theta \in \{0, 1\}$, and let $\hat{\alpha} = \alpha + (t_j^1(\alpha, \gamma, \beta) - t_j^0(\alpha, \gamma, \beta))e^j$, $\hat{\beta} = \beta - t_j^0(\alpha, \gamma, \beta)$. Then, $\dim F(\hat{\alpha}, \gamma, \hat{\beta}) \geq \dim F(\alpha, \gamma, \beta) + 1$.

Proof. We distinguish two cases: both tilting functions evaluate to a positive value, or only one of them does. If $t_j^0(\alpha, \gamma, \beta), t_j^1(\alpha, \gamma, \beta) > 0$, then there is no point $(x, z) \in F(\alpha, \gamma, \beta)$ such that $x_j = 0$ or $x_j = 1$, i.e., the inequality is not tight at any point in $\text{conv}(\text{hyp } \phi)$; otherwise, one of the two tilting functions would evaluate to zero. Therefore, after tilting, we obtain a new vector $(\hat{\alpha}, \gamma, \hat{\beta})$ such that $F(\hat{\alpha}, \gamma, \hat{\beta})$ contains a point (x, z) with $x_j = 1$ (and another with $x_j = 0$). Hence, we only focus on the second case and assume $|F(\alpha, \gamma, \beta)| \geq 1$.

If $t_j^\theta(\alpha, \gamma, \beta) = 0$ for $\theta = 0$ or $\theta = 1$, then any point $(x, z) \in F(\alpha, \gamma, \beta)$ satisfies $x_j = \theta$. By definition of t_j^θ , after tilting, we obtain a new vector $(\hat{\alpha}, \gamma, \hat{\beta})$ such that $F(\hat{\alpha}, \gamma, \hat{\beta})$ contains a point (x, z) satisfying $x_j = 1 - \theta$ which is affinely independent from all points in $F(\alpha, \gamma, \beta)$. Hence, $\dim F(\hat{\alpha}, \gamma, \hat{\beta}) \geq \dim F(\alpha, \gamma, \beta) + 1$. $\square$

As noted in Castro et al. [2022], if $\text{conv}(\text{hyp } \phi)$ is full-dimensional, Proposition 5 indicates that whenever we tilt $n_L - \dim F(\alpha, \gamma, \beta)$ components of a valid inequality defined by (α, γ, β) , the resulting inequality is facet-defining for $\text{conv}(\text{hyp } \phi)$. Nonetheless, it is possible to achieve facet-defining inequalities even if we tilt fewer components, e.g., if $\text{conv}(\text{hyp } \phi)$ is not full-dimensional or if tilting one component leads to an inequality inducing a face F' satisfying $\dim F' > \dim F(\alpha, \gamma, \beta) + 1$.

4.2 Computing tilting quantities with a network

In general, computing tilting quantity functions for binary linear programs poses a significant challenge [Castro et al., 2022]. In what follows, inspired by the approach in Castro et al. [2022], we explain how to leverage $\mathcal{D}_{\text{gph } \phi}$ to compute tilting quantities. Assume we have a network $\mathcal{D}_{\text{gph } \phi} = (\mathcal{N}, \mathcal{A}, l, \nu)$ and a vector (α, γ, β) defining an inequality of the form (12). Let $\ell(\alpha, \gamma) : \mathcal{A} \rightarrow \mathbb{R}^{|\mathcal{A}|}$ be a function that maps each arc $a \in \mathcal{A}$ to a length in the following way:

$$\ell_a(\alpha, \gamma) = \begin{cases} \nu_a \alpha_{\mathbf{i}(a)} & \text{if } a \in \mathcal{A}_L, \\ \nu_a \gamma & \text{if } a \in \mathcal{A}_F. \end{cases}$$

Let $\mathcal{D}_{\text{gph } \phi}(\alpha, \gamma) = (\mathcal{N}, \mathcal{A}, \ell(\alpha, \gamma), \nu)$ be an arc length-parametrized version of $\mathcal{D}_{\text{gph } \phi}$, i.e., it inherits the node and arc sets, as well as the arc values, but redefines the arc lengths. Observe that each r - t path in $\mathcal{D}_{\text{gph } \phi}(\alpha, \gamma)$ still uniquely maps to a vector $(x, z) \in \text{gph } \phi$, and its length equals $\alpha^\top x + \gamma z$. Now, for any node $u \in \mathcal{N} \setminus \{r, t\}$, let $\sigma_u^\downarrow$ denote the longest path length from the root node r to u , and $\sigma_u^\uparrow$ the longest path length from u to the sink node t , both in $\mathcal{D}_{\text{gph } \phi}(\alpha, \gamma)$. Moreover, for any arc $a \in \mathcal{A}_L$, let

$$\mu_a := \max_{w \geq 0} \left\{ \sum_{a' \in \mathcal{A}} \ell_{a'}(\alpha, \gamma) w_{a'} : w \text{ represents an } r\text{-}t \text{ path in } \mathcal{D}_{\text{gph } \phi}(\alpha, \gamma), w_a = 1 \right\}$$

be the maximum length of any r - t path in $\mathcal{D}_{\text{gph } \phi}(\alpha, \gamma)$ traversing arc a . For any $j \in [n_L]$ and $\theta \in \{0, 1\}$, we have

$$t_j^\theta(\alpha, \gamma, \beta) = \beta - \max\{\mu_a : a \in \mathcal{A}_L, \mathbf{i}(a) = j, \nu_a = \theta\}.$$

By definition, we have $\mu_{uv} = \sigma_u^\downarrow + \sigma_v^\uparrow + \ell_{uv}(\alpha, \gamma)$ for any arc $(u, v) \in \mathcal{A}$. Hence, we obtain

$$\begin{aligned}
t_j^\theta(\alpha, \gamma, \beta) &= \beta - \max\{\sigma_u^\downarrow + \sigma_v^\uparrow + \ell_{uv}(\alpha, \gamma) : (u, v) \in \mathcal{A}_L, \mathbf{i}(u, v) = j, \nu_{uv} = \theta\} \\
&= \beta - \theta\alpha_j - \max\{\sigma_u^\downarrow + \sigma_v^\uparrow : (u, v) \in \mathcal{A}_L, \mathbf{i}(u, v) = j, \nu_{uv} = \theta\}
\end{aligned}$$

for $\theta \in \{0, 1\}$. Observe that computing $\sigma_u^\downarrow$ or $\sigma_v^\uparrow$ for a fixed vector (α, γ) corresponds to solving a longest path problem over $\mathcal{D}_{\text{gph } \phi}$. Therefore, the time complexity of computing μ_a for any arc $a \in \mathcal{A}_L$ is linear in the size of $\mathcal{D}_{\text{gph } \phi}$. This, in turn, implies that for fixed n_L , the time complexity of computing all tilting quantity functions is also linear in the size of $\mathcal{D}_{\text{gph } \phi}$.

Example 2. We continue the running example by indicating how to compute tilting quantity functions using the network in Figure 3. Consider an initial valid inequality $-x + 3y \leq 6$, which is tight at $x = 3$ ($(x_1, x_2) = (1, 1)$ for the binary expansion). This inequality is depicted as a black line in Figure 4a. Vectors $\sigma^\downarrow$ and $\sigma^\uparrow$ are computed by solving longest path problems over $\mathcal{D}_{\text{gph } \phi}(\alpha, \gamma)$, where $\alpha = (-1, -2)$ and $\gamma = 3$ are illustrated as black arc labels in Figure 4b. We have

$$\begin{aligned}
(\text{Layer 1}) \quad & \sigma_r^\downarrow = 0, \\
(\text{Layer 2}) \quad & \sigma_1^\downarrow = 0, \quad \sigma_1^\uparrow = -2, \quad \sigma_2^\downarrow = -1, \quad \sigma_2^\uparrow = 7, \\
(\text{Layer 3}) \quad & \sigma_3^\downarrow = -2, \quad \sigma_3^\uparrow = -0, \quad \sigma_4^\downarrow = 0, \quad \sigma_4^\uparrow = 3, \quad \sigma_5^\downarrow = -3, \quad \sigma_5^\uparrow = 9, \\
(\text{Layer 4}) \quad & \sigma_t^\uparrow = 0.
\end{aligned}$$

We can now compute all tilting quantity functions:

$$\begin{aligned}
t_1^0(\alpha, \gamma, \beta) &= 6 - \max\{-2, 3\} = 3, \\
t_1^1(\alpha, \gamma, \beta) &= 6 - (-1) - \max\{3, 7\} = 0, \\
t_2^0(\alpha, \gamma, \beta) &= 6 - \max\{3, 2\} = 3, \\
t_2^1(\alpha, \gamma, \beta) &= 6 - (-2) - \max\{0, 8\} = 0.
\end{aligned}$$

As $t_1^1, t_2^1 = 0$ and $t_1^0, t_2^0 = 3$, the inequality can only be tilted with respect to $(1 - x_1)$ or $(1 - x_2)$ by at most 3 units. Tilting with respect to $(1 - x_1)$ by 1 unit yields the tilted inequality induced by $(\hat{\alpha}, \gamma, \hat{\beta})$, where $\hat{\alpha} = (\alpha_1 - 1, \alpha_2)$ and $\hat{\beta} = \beta - 1$. By repeating this procedure over $\mathcal{D}_{\text{gph } \phi}(\hat{\alpha}, \gamma)$, we can tilt the resulting inequality with respect to $(1 - x_2)$ by 2 units, achieving the final tilted inequality $-2x_1 - 4x_2 + 3y \leq 3$, illustrated as an orange line in Figure 4a. $\square$

5 Experimental Results

We next present a set of three computational experiments to evaluate the efficiency and strength of the proposed cuts. In the first one, we tackle instances of a multidimensional knapsack bilevel problem, aiming to assess the impact of network size on the available methods by increasing the number of follower constraints. In the second one, we address instances of a binary bilevel program in which the follower's constraint matrix exhibits a block-diagonal structure. As each block can be used to compile a separate network, we assess the impact of the number of networks on the available methods. Finally, we test the effectiveness of including the proposed cuts into state-of-the-art solvers when solving a subset of unsolved problems from the BOBILib benchmark instance library [Thürauf et al., 2026].

We refer to the state-of-the-art solvers of Fischetti et al. [2017] and Tahernejad et al. [2020] as FLMS and MibS, respectively. Both solvers are compiled using their latest version (MibS's is 1.2.1 and FLMS has a unique version) and are executed using their default settings. We compile networks using Algorithm 2 in Lozano et al. [2026], and sequentially strengthen valid inequalities by choosing the coefficient with the smallest tilting quantity, as proposed in Castro et al. [2022]. For the first

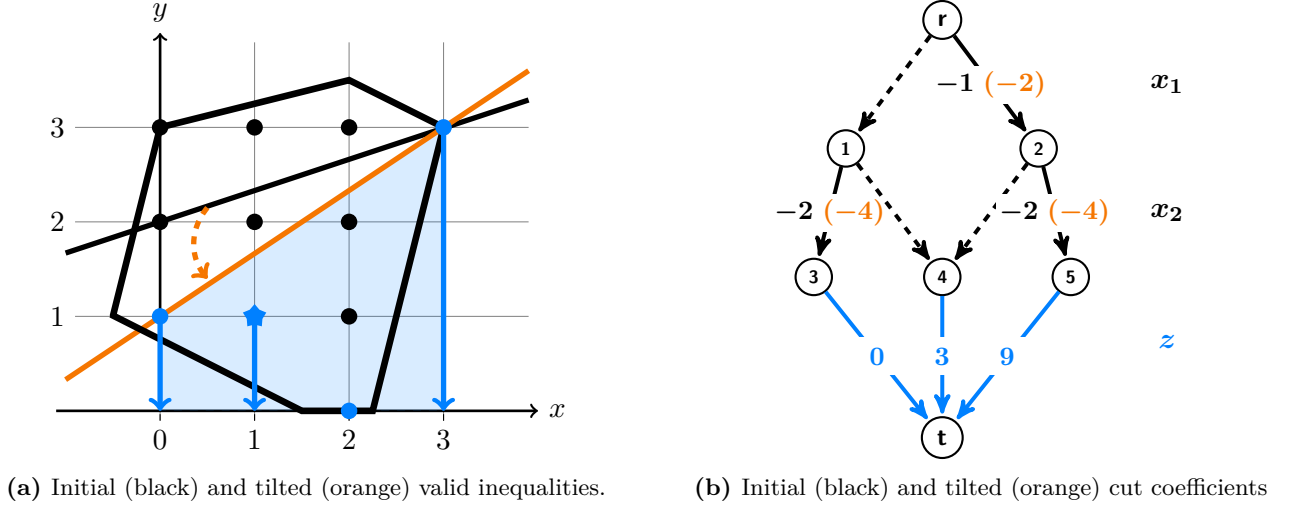


Figure 4: Example of tilting valid inequality $-x + 3y \leq 6$ into $-2x + 3y \leq 3$

two sets of experiments, we compile exact networks by setting infinite budget. For the last one, we compile relaxed networks (approximate value networks in Lozano et al. [2026]) according to the procedure proposed in Section 5.2 of Lozano et al. [2026]. We only include results for our cutting plane approach using tilted cuts, as they significantly outperform the non-tilted version.

Our algorithms are implemented in C++ and all computations are executed on a 12-core Intel(R) Xeon(R) Gold 6248R CPU 3.00GHz running Linux Ubuntu. All single-level optimization programs required by our algorithms are solved using Gurobi 10, while FLMS and MibS use CPLEX 12.

5.1 Multidimensional knapsack bilevel problem

We first study a class of multidimensional knapsack bilevel problems in which the leader problem has no constraints, *i.e.*, $m_L = 0$, and the follower problem has a number of knapsack-type constraints $m_F \in \{1, 2, 3, 4\}$. The purpose of this experiment is to evaluate and compare the performance of available network flow-based approaches for different network sizes. Due to the nature of the network construction algorithm in Lozano et al. [2026], a higher value of m_F leads to larger networks.

Instances. We consider $n_L = n_F = 30$ and denote as $\text{DU}[l, u]$ the discrete uniform distribution on the integers $\{l, \dots, u\}$. Objective function and follower constraint matrices entries are sampled as follows:

$$c_L \sim \text{DU}[-10, 10], \quad c_F \sim \text{DU}[-100, 100], \quad d \sim \text{DU}[-10, -1], \\ C \sim \text{DU}[-2, 2], \quad D \sim \text{DU}[0, 10].$$

The right-hand side vector of the follower's problem is computed as $b = \alpha \cdot (C + D)\mathbf{1}$, where $\alpha \in \{0.25, 0.5, 0.75\}$ is a scaling factor. These values are chosen to create instances in which the leader's influence over the follower's feasible region is not too strong, and the size of the network is well controlled by the tighter bounds on the entries of C . We generate 10 instances for each value of (m_F, α) , for a total of 120 instances.

Comparison. We compare directly solving program (4) using an off-the-shelf MIP solver, and a cutting-plane approach using the proposed cut families introduced in Section 3 after applying the

tilting procedure described in Section 4.2. We differentiate between “no-good cuts” (NGC) and enhanced optimality cuts; see Section 3.1. We also compare target cuts [Buchheim et al., 2008, Tjandraatmadja and van Hoeve, 2019], which can be separated by solving the dual of a network flow program over the network. If $\text{conv}(\text{dom } \mathbb{Y})$ is full-dimensional, then target cuts are guaranteed to be facet-defining [Tjandraatmadja and van Hoeve, 2019].

Observe that, as we compile exact networks, all these solution methods are exact. However, in general, we may not be capable of solving program (4) within the time limit, or we may use a relaxed network. In both cases, the result yields a valid lower bound. Hence, we are interested in evaluating what portion of this bound is recovered within the time limit. Let $\mathcal{D}$ be the compiled network, and $v^*(\mathcal{D})$ the optimal value of program (4) using $\mathcal{D}$ instead of $\mathcal{D}_{\text{gph } \phi}$. Moreover, consider a time limit $T > 0$ and a set S of solution methods. We define the bound recovery ratio for method $s \in S$ at time $t > 0$ as

$$r_t^s(\mathcal{D}) = \frac{v_{\text{LB},t}^s(\mathcal{D}) - \text{LB}}{\text{UB} - \text{LB}},$$

where $v_{\text{LB},t}^s(\mathcal{D})$ and $v_{\text{UB},t}^s(\mathcal{D})$ are the lower and upper bounds achieved by method s at time t , respectively, $\text{LB} = \min_{s \in S} \{v_{\text{LB},0}^s(\mathcal{D})\}$ is the lowest initial lower bound achieved by any method (*e.g.*, the LP relaxation), and $\text{UB} = \min_{s \in S} \{v_{\text{UB},T}^s(\mathcal{D})\}$ is the best value (upper bound) achieved by any solution method. We use the convention $r_t^s(\mathcal{D}) = 1$ when $\text{UB} = \text{LB}$, in which case optimality is already certified. This way, we have $r_t^s(\mathcal{D}) \in [0, 1]$ and $v^*(\mathcal{D}) \in [v_{\text{LB},t}^s(\mathcal{D}), \text{UB}]$ for any $t \in [0, T]$ and $s \in S$. Moreover, $r_t^s(\mathcal{D}) = 1$ if and only if $v_{\text{LB},t}^s(\mathcal{D}) = \text{UB} = v^*(\mathcal{D})$. Thus, $r_t^s(\mathcal{D}) = 1$ indicates that method s achieves the optimal value of program (4) using $\mathcal{D}$ by time t .

Results. Figure 5 illustrates average values of the bound recovery measure r for different m_F values with a time limit $T = 3,600$. This time limit does not consider the time used to compile the network. We observe that, for small size networks ($m_F \in \{1, 2\}$), directly solving the reformulation (4) corresponds to the fastest approach. For larger networks ($m_F \in \{3, 4\}$), the cutting-plane approaches show a better performance than the reformulation, especially those that do not require solving a linear program. Furthermore, among the proposed cuts, feasibility cuts exhibit an overall faster bound recovery, indicating that the higher cut generation time used to solve the Lagrangian relaxation gets paid off by providing a stronger seed inequality.

5.2 Block-diagonal follower bilevel problem

We now study a class of binary bilevel problems in which the follower’s problem has a block-diagonal constraint matrix. We can alternatively interpret these problems as binary bilevel programs with multiple independent followers [Calvete and Galé, 2007]. The objective of this experiment is to assess the effect of decomposable structures in the follower’s problem. Let $k > 0$ be the number of follower blocks. In these problems, the follower’s problem constraint matrix D has a block-diagonal structure with k blocks, each with a knapsack constraint, *i.e.*, $m_F = k$. Accordingly, the follower’s objective function vector d and right-hand side vector b , as well as the leader’s objective function vector c_F and constraint matrix B , are split into k smaller vectors (matrices) $\{d^p\}_{p \in [k]}$, $\{b^p\}_{p \in [k]}$, $\{c_F^p\}_{p \in [k]}$, and $\{B^p\}_{p \in [k]}$, respectively, of appropriate dimension. Matrix C is defined for each block as $\{C^p\}_{p \in [k]}$. This way, we can rewrite the follower’s value function as $\phi(x) = \sum_{p \in [k]} \phi^p(x)$, where

$$\phi^p(x) = \min \{d^{p\top} y^p : D^p y^p \leq b^p - C^p x\}$$

is the optimal value of the follower’s problem associated with the p -th block. By extending Theorem 3.1 in Calvete and Galé [2007] to the binary setting, we obtain that $(x, z) \in \text{hyp } \phi$ if and only if there

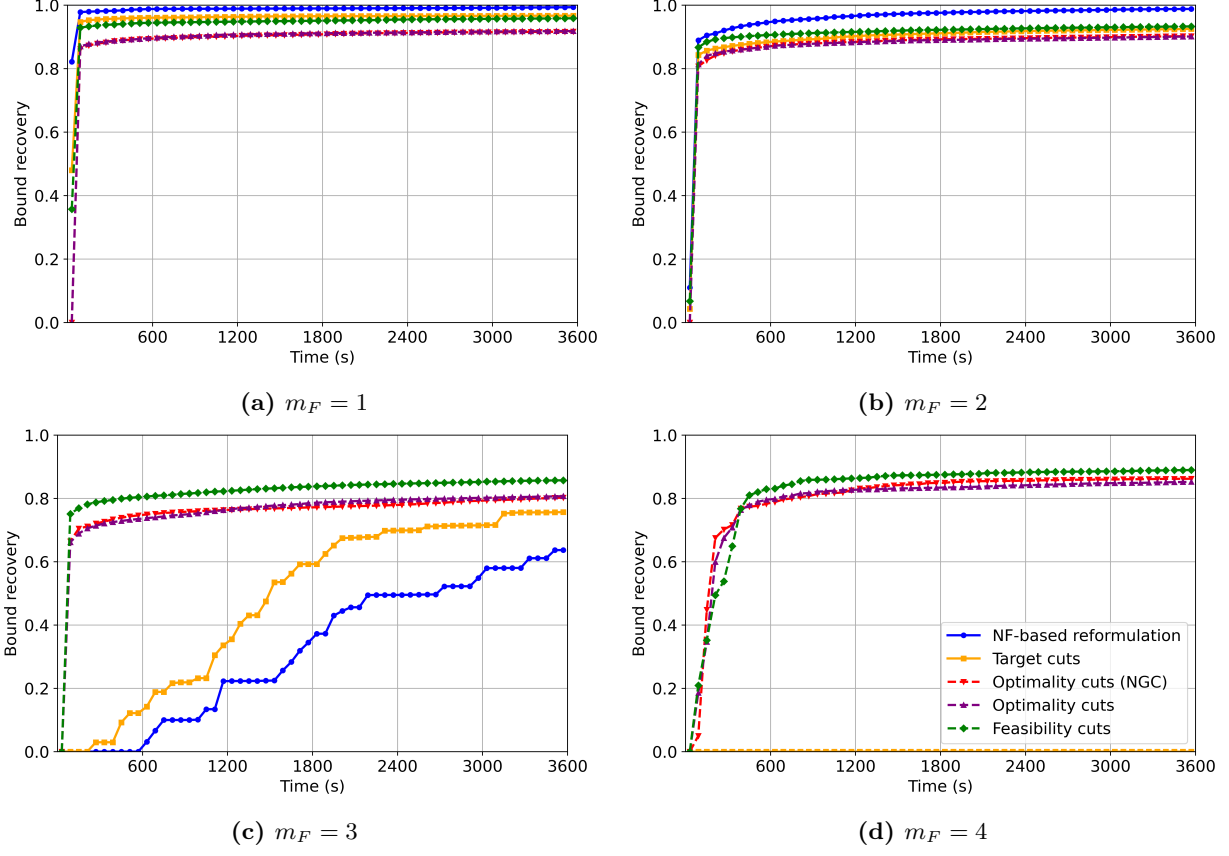


Figure 5: Results for the multidimensional knapsack instances

exist z^p such that $(x, z^p) \in \text{hyp } \phi^p$ for every $p \in [k]$ and $z = \sum_{p \in [k]} z^p$. Hence, we can equivalently write program (5) as

$$\begin{aligned}
 \min \quad & c_L^\top x + \sum_{p \in [k]} c_F^{p^\top} y^p \\
 \text{s. t.} \quad & Ax + \sum_{p \in [k]} B^p y^p \leq a \\
 & C^p x + D^p y^p \leq b^p \quad p \in [k] \\
 & (x, d^{p^\top} y^p) \in \text{conv}(\text{hyp } \phi^p) \quad p \in [k].
 \end{aligned}$$

As it is done for $(x, d^\top y) \in \text{conv}(\text{hyp } \phi)$, we can enforce each constraint $(x, d^{p^\top} y^p) \in \text{conv}(\text{hyp } \phi^p)$ with a network $\mathcal{D}_{\text{gph } \phi^p} = (\mathcal{N}^p, \mathcal{A}^p, l^p, \nu^p)$ compiled only considering the p -th block problem param-

ters. Hence, we can rewrite the network flow-based reformulation (4), for this setting, as

$$\begin{aligned}
\min \quad & c_L^\top x + \sum_{p \in [k]} c_F^p{}^\top y^p \\
\text{s. t.} \quad & Ax + \sum_{p \in [k]} B^p y^p \leq a \\
& C^p x + D^p y^p \leq b^p & p \in [k] \\
& \sum_{a \in \delta^+(u)} w_a^p - \sum_{a \in \delta^-(u)} w_a^p = \mathbf{1}(u = r^p) - \mathbf{1}(u = t^p) & u \in \mathcal{N}^p \\
& x_j = \sum_{a \in \mathcal{A}_L^p : \mathbf{i}(a)=j} \nu_a^p w_a^p & j \in [n_L], p \in [k] \\
& d^p{}^\top y^p \leq l^p{}^\top w^p & p \in [k] \\
& w^p \geq 0 & p \in [k].
\end{aligned}$$

Similarly, we can separate optimality and feasibility cuts using each network $\mathcal{D}_{\text{gph } \phi^p}$, as Propositions 1 and 3 hold.

Instances. We consider $n_L = n_F \in \{30, 45, 60, 75\}$. Objective function and follower constraint matrices entries are sampled as follows:

$$c_L, c_F \sim \text{DU}[-100, 100], \quad d \sim \text{DU}[-100, -1], \quad C \sim \text{DU}[0, 10], \quad D \sim \text{DU}[0, 100].$$

As in the previous problem class, the right-hand side vector of the follower's problem is computed as $b = \alpha \cdot (C + D)\mathbf{1}$, where $\alpha \in \{0.25, 0.5, 0.75\}$ is a scaling factor. These values are chosen to achieve a relatively generic set of instances, where the leader problem has no constraints. We generate 10 instances for each value of (n_F, k, α) , for a total of 360 instances.

Comparison. We compare directly solving program (4), the cutting plane approach using feasibility cuts, and state-of-the-art bilevel solvers FLMS [Fischetti et al., 2017] and MibS [Tahernejad et al., 2020]. Furthermore, we include two extra variants of each bilevel solver: (i) we replace the HPR with reformulation (4), and (ii) we add feasibility cuts to the HPR. Networks compilation and cuts generation (if applicable) is done within a time limit of 300 seconds, after which the corresponding bilevel solver is executed for 3,300 seconds. Both variants use the same networks. The variant of MibS with reformulation (4) is excluded as it fails to produce finite gaps for several instances within the time limit.

Results. Figures 6 and 7 present two-panel performance plots. The left panes show the cumulative number of solved instances, and the right panes the optimality gap of the unsolved ones. We limit the maximum gap to 100% for illustrative purposes. Overall, when $k = 1$, directly solving reformulation (4) yields the best results, especially for $n = 75$. For any other configuration, the cutting plane approach outperforms the other methods. In particular, the results for $k > 1$ align with the fact that the cutting plane approach fully decomposes the convexification of $\text{hyp } \phi$ and the information retrieval process into different networks, whereas the reformulation (4) manages all networks within the same program, yielding, on average, a higher number of variables as k increases. Regarding state-of-the-art solvers, we observe that including feasibility cuts into the initial reformulation significantly improves its performance, whereas replacing the HPR with (4), in

general, hurts it. Even though reformulation (4) provides an exact representation of $\text{conv}(\text{hyp } \phi)$, the extended variable space and the larger formulation greatly impact the effectiveness of these solvers. This also highlights the benefits of retrieving the networks' information through cuts that are projected into the original variable space.

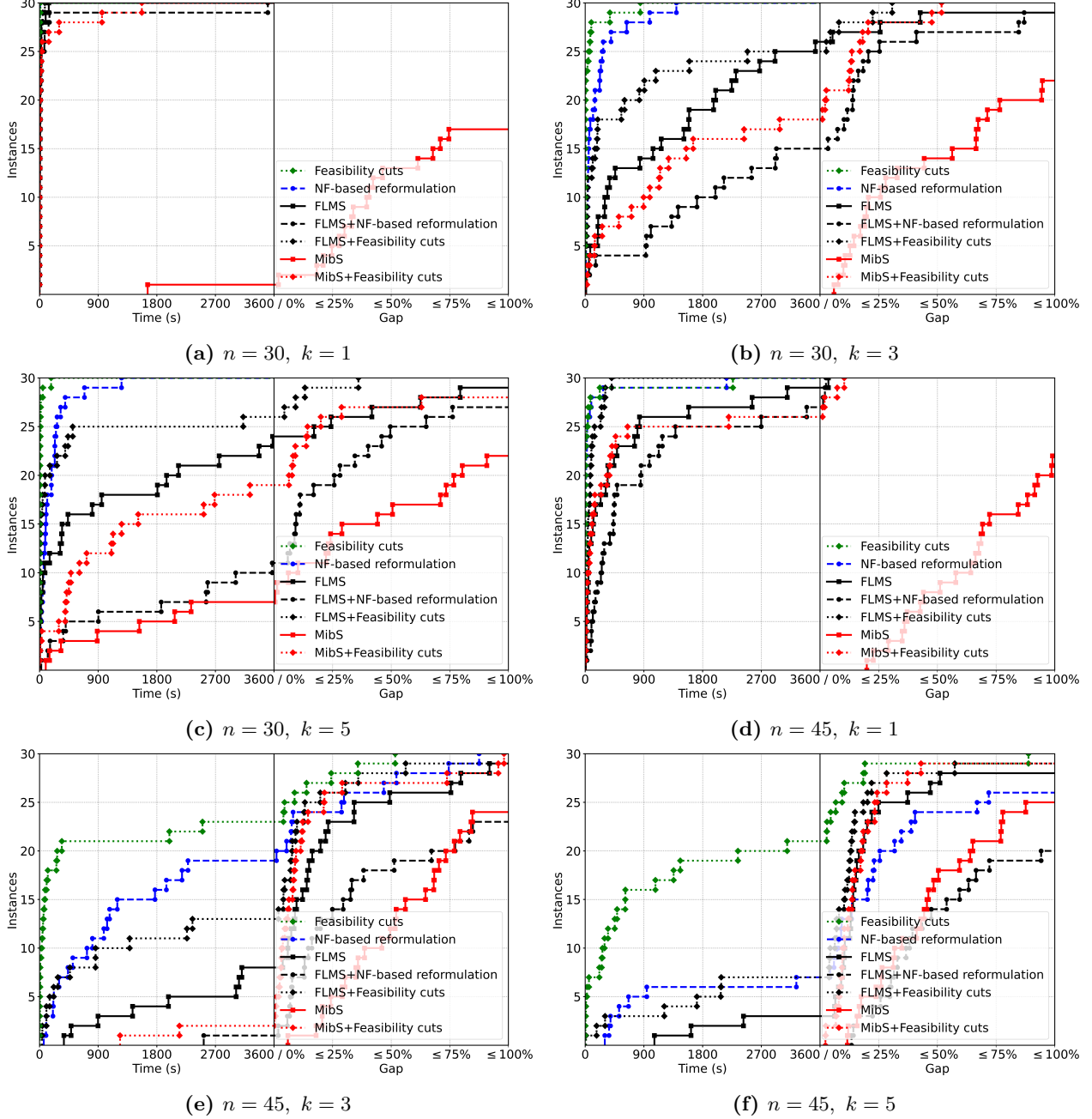


Figure 6: Results for the block-diagonal follower instances

5.3 BOBILib instances

Finally, we assess the impact of adding feasibility cuts into state-of-the-art bilevel solvers to tackle benchmark instances from BOBILib [Thürauf et al., 2026]. We consider the `cclw` and `inter-kp`

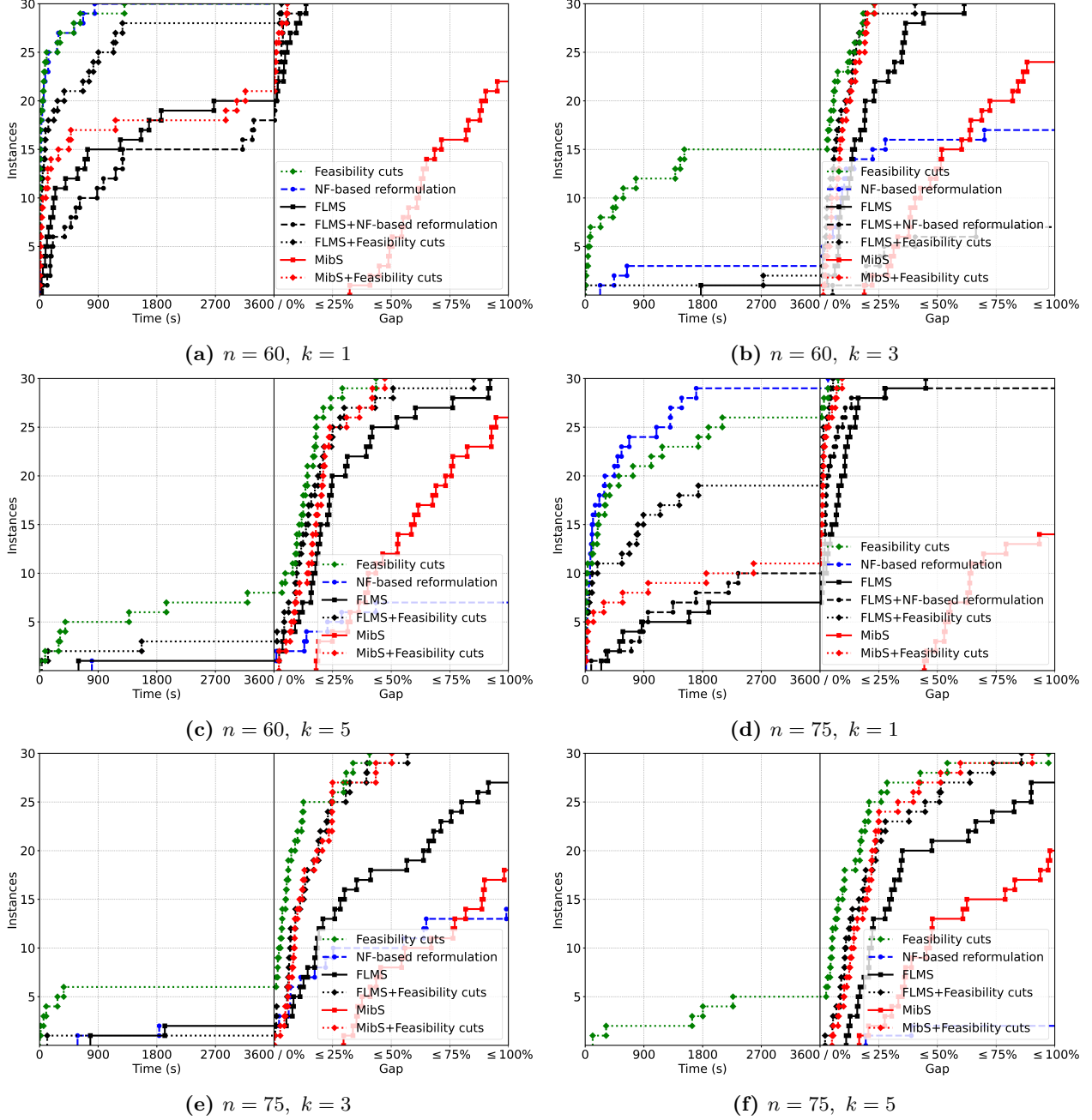


Figure 7: Results for the block-diagonal follower instances (contd.)

knapsack-interdiction families, the `imkp` multidimensional-knapsack-interdiction family, and the MIPLIB-derived family. Within these families, we select all instances for which every leader and follower variable is binary and whose BOBILib status is either feasible or open. We exclude the substantially larger or family to prevent a single family from dominating the test set. This selection yields 125 instances. Because compiling the full networks is intractable, we use approximate value networks capped at 50 nodes per layer. Initial experiments indicate that larger caps produce worse bounds under the same compilation and cut-generation time limit.

Table 1: Results for BOBILib instances with binary variables.

Class	Subclass	FLMS			FLMS+Feasibility cuts		
		Solved	Time (s)	Gap (%)	Solved	Time (s)	Gap (%)
Knapsack interdiction	cclw	2/15	3,370	36.27	5/15	854	20.60
	inter-kp	0/25	–	32.40	0/25	–	14.19
Multidimensional knapsack interdiction	imkp	0/5	–	14.94	2/5	1,635	8.48
	weish	1/47	2,829	48.43	1/47	66	33.33
MIPLIB		1/33	2,505	164.81	4/33	205	172.11

Comparison. We compare FLMS with its version with added feasibility cuts. In the latter, we compile and generate cuts within 300 seconds, and then execute FLMS for 3,300 seconds.

Results. Table 1 reports the number of solved instances, average solution times over solved instances, and average final gaps over unsolved instances. Feasibility cuts reduce times and gaps, allowing FLMS to close eight additional instances, and up to 13 using a variant of the approximate network compilation procedure in Lozano et al. [2026]. Appendix A lists the solved instances.

6 Conclusions

We develop a cutting-plane framework for binary bilevel programs based on the network encoding of the follower’s value function proposed by Lozano et al. [2026]. We introduce two exact network flow-based reformulations and separate their associated valid inequalities in the original variable space, strengthening them through tilting. Our computational results reveal a trade-off between embedding the networks in an extended formulation and using them as separation oracles: the former is effective for small networks, whereas the latter performs better as networks become larger or more numerous. On BOBILib instances classified as feasible or open, adding the cuts to FLMS reduces solution times and gaps, closing eight additional instances.

References

- Jonathan F Bard and James T Moore. A branch and bound algorithm for the bilevel programming problem. *SIAM Journal on Scientific and Statistical Computing*, 11(2):281–292, 1990.
- Markus Behle. *Binary decision diagrams and integer programming*. PhD thesis, 2007.
- Jerome Bracken and James T. McGill. Mathematical Programs with Optimization Problems in the Constraints. *Operations Research*, 21(1):37–44, 1973.
- Christoph Buchheim, Frauke Liers, and Marcus Oswald. Local cuts revisited. *Operations Research Letters*, 36(4):430–433, 2008.
- Herminia I Calvete and Carmen Galé. Linear bilevel multi-follower programming with independent followers. *Journal of Global Optimization*, 39(3):409–417, 2007.
- Margarita P Castro, Andre A Cire, and J Christopher Beck. A combinatorial cut-and-lift procedure with an application to 0–1 second-order conic programming. *Mathematical Programming*, 196(1):115–171, 2022.

- Rui Chen, Oktay Günlük, and Andrea Lodi. Recovering dantzig–wolfe bounds by cutting planes. *Operations Research*, 73(2):1128–1142, 2025.
- Danial Davarnia and Willem-Jan van Hoes. Outer approximation for integer nonlinear programs via decision diagrams. *Mathematical Programming*, 187:111–150, 2021.
- Matteo Fischetti, Ivana Ljubić, Michele Monaci, and Markus Sinnl. A new general-purpose algorithm for mixed-integer bilevel linear programs. *Operations Research*, 65(6):1615–1637, 2017.
- José Fortuny-Amat and Bruce McCarl. A representation and economic interpretation of a two-level programming problem. *Journal of the Operational Research Society*, 32(9):783–792, 1981.
- Henri Lefebvre and Martin Schmidt. A dantzig–wolfe single-level reformulation for mixed-integer linear bilevel optimization: Exact and heuristic approaches. *Optimization Online*, 2025.
- Leonardo Lozano and J Cole Smith. A binary decision diagram based algorithm for solving a class of binary two-stage stochastic programs. *Mathematical Programming*, pages 1–24, 2018.
- Leonardo Lozano, David Bergman, and Andre A. Cire. Constrained Shortest-Path Reformulations for Discrete Bilevel and Robust Optimization. *arXiv preprint arXiv:2206.12962*, 2022.
- Leonardo Lozano, David Bergman, and Andre A Cire. Network relaxations for combinatorial bilevel optimization under linear interactions. *Operations Research*, 2026.
- Sahar Tahernejad, Ted K Ralphs, and Scott T DeNegre. A branch-and-cut algorithm for mixed integer bilevel linear optimization problems and its implementation. *Mathematical Programming Computation*, 12(4):529–568, 2020.
- Johannes Thürauf, Thomas Kleinert, Ivana Ljubić, Ted Ralphs, and Martin Schmidt. Bobilib: Bilevel optimization (benchmark) instance library. *Mathematical Programming Computation*, pages 1–58, 2026.
- Christian Tjandraatmadja and Willem-Jan van Hoes. Target cuts from relaxed decision diagrams. *INFORMS Journal on Computing*, 31(2):285–301, 2019.
- Sebastián Vásquez, Leonardo Lozano, and Willem-Jan van Hoes. A single-level reformulation of binary bilevel programs using decision diagrams. *Mathematical Programming*, pages 1–54, 2025.
- M Hosein Zare, Juan S Borrero, Bo Zeng, and Oleg A Prokopyev. A note on linearized reformulations for a class of bilevel linear integer problems. *Annals of Operations Research*, 272:99–117, 2019.

A BOBILib instances solved

Table 2 shows the selected BOBILib instances that are closed by at least one method.

Table 2: BOBILib instances solved. An asterisk (*) indicates that an instance is closed using a slightly different procedure to generate an approximate value network.

Class	Subclass	Instance	FLMS	FLMS+Feasibility cuts
Knapsack interdiction	cclw	interdiction40-5	✓	
		interdiction40-6		✓
		interdiction45-6		✓
		interdiction50-7	✓	✓
		interdiction55-7		✓
		interdiction55-8		✓
	inter-kp	K5050W06.KNP		✓*
		K5050W18.KNP		✓*
Multidimensional knapsack interdiction	imkp	pet7.dat-0		✓*
		pet7.dat-50		✓
		pet7.dat-100		✓
		sento2.dat-100		✓*
	weish	weish14.dat-50	✓	
		weish17.dat-50		✓
		weish17.dat-100		✓*
MIPLIB		lseu-0.500000	✓	
		cap6000-0.500000		✓
		p0201-0.100000		✓
		p0282-0.900000		✓
		p0548-0.900000		✓