Recovering Dantzig-Wolfe Bounds by Cutting Planes

Rui Chen[1], Oktay Günlük[2], Andrea Lodi[1]
[1] Cornell Tech, Cornell University ({rui.chen,andrea.lodi}@cornell.edu)
[2] School of ORIE, Cornell University (oktay.gunluk@cornell.edu)

October 6, 2023

Abstract

Dantzig-Wolfe (DW) decomposition is a well-known technique in mixed-integer programming (MIP) for decomposing and convexifying constraints to obtain potentially strong dual bounds. We investigate cutting planes that can be derived using the DW decomposition algorithm and show that these cuts can provide the same dual bounds as DW decomposition. More precisely, we generate one cut for each DW block, and when combined with the constraints in the original formulation, these cuts imply the objective function cut one can simply write using the DW bound. This approach typically leads to a formulation with lower dual degeneracy that consequently has a better computational performance when solved by standard MIP solvers in the original space. We also discuss how to strengthen these cuts to improve the computational performance further. We test our approach on the Multiple Knapsack Assignment Problem and the Temporal Knapsack Problem, and show that the proposed cuts are helpful in accelerating the solution time without the need to implement branch and price.

1 Introduction

In this paper, we present a computationally effective approach for generating cutting planes from Dantzig-Wolfe (DW) decomposition [1] to enhance branch and cut in the space of original variables. We focus on mixed-integer (linear) programs (MIPs) with the following structure:

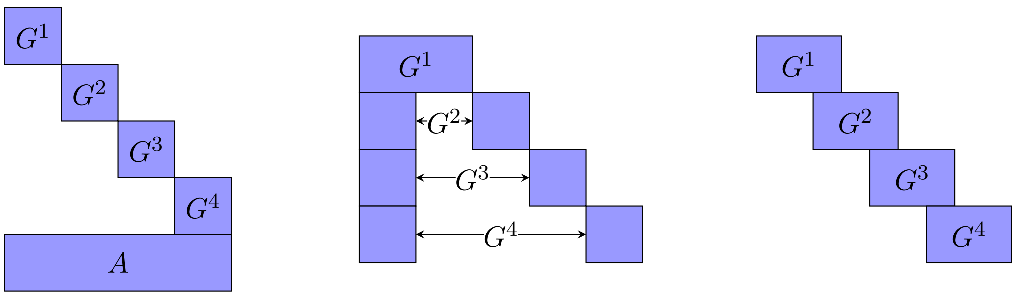
$$\begin{aligned} z^* := \min\ & c^\top x \\ \text{s.t. } & x_{I(j)} \in P^j, \quad j \in J := \{1, \dots, q\}, \\ & Ax \geq b,\ x \in X, \end{aligned} \tag{1}$$

where the index set $I(j) \subseteq \{1, \dots, n\}$ contains the indices of the variables in "block" $j \in J$, and we use the notation x_I to denote the subvector of x with indices in I. The set $P^j = \{y \in \mathbb{R}^{|I(j)|} : G^j y \geq g^j\}$ is a polyhedral set for $j \in J$; and $X \subseteq \mathbb{R}^n$ represents integrality constraints on some of the variables. We do not assume the index sets $I(j)$ for $j \in J$ to be disjoint, see Figure 1.

DW decomposition was originally developed for solving large-scale linear programs with loosely coupled blocks and later extended to MIPs with similar block structures to obtain strong dual bounds. Typically, these so-called DW bounds are stronger than the linear programming (LP) relaxation bounds as they exploit the block structure to partially convexify the solution space using integrality information. DW decomposition has been found to be effective in various applications, such as transportation [2], traffic scheduling [3], energy [4], and multi-stage stochastic planning [5].

Computing the DW bound requires reformulating the MIP using the extreme points and extreme rays of the mixed-integer hulls of the block problems. Consequently, while the DW reformulation approach often leads to good dual bounds, using this to solve the MIP exactly requires

Figure 1: Constraint Matrices of MIPs With Different Types of Block Structures (Left: Loosely Coupled, Middle: Two-Stage, Right: Overlapping)



specialized techniques that are not readily present in off-the-shelf solvers. In other words, to exploit DW decomposition one needs to solve the continuous relaxation of the reformulated MIP by column generation followed by an *ad-hoc* branching step [6], after which one has to again resort to column generation, leading to an algorithmic framework called branch and price [7]. While this approach has been successfully implemented in some special cases, most notably for vehicle routing problems [8], generic branch-and-price solvers, including ABACUS [9], G12 [10], GCG [11], DIP [12], BaPCod [13], to name a few, are still in their infancy with respect to solving general MIPs with block structure. However, the most up-to-date (and always improving) MIP solvers are based on the branch-and-cut (or cut-and-branch) scheme [14].

In this paper, we aim to make a step towards bridging this gap by developing a new scheme to incorporate the dual bounds produced by DW reformulations into the standard branch-and-cut framework. More precisely, we first compute the DW reformulation bound z_D of the MIP (1) at the root node and then generate cutting planes to incorporate this bound into the original formulation to solve the problem to optimality in the space of the original variables using standard MIP technology. Our approach, therefore, requires column generation only at the root node and not throughout the enumeration tree. Apart from our proposed approach, one trivial method to enforce the DW bound z_D in the original MIP is to augment the formulation by adding the *objective function cut*, $c^\top x \geq z_D$, which is known (folklore) to be not only computationally ineffective but also numerically unstable. To the best of our knowledge, however, no detailed theoretical or computational investigation has been conducted on the objective function cut. Earlier cutting plane approaches include [15, 16, 17], where valid inequalities are added iteratively to separate candidate solutions from the DW relaxation polytope. Since we only add one round of valid inequalities, our approach tends to be less computational intensive and leads to a less complex MIP formulation in the end.

Paper Contributions. Our first contribution is to confirm the instability of using the objective cut by mostly attributing it to dual degeneracy. We then show how to overcome this issue by using a family of cutting planes that essentially decompose the objective function cut. This leads to a formulation with lower dual degeneracy, and consequently a better computational performance when solved by standard MIP solvers in the original space. Moreover, we propose two distinct ways to strengthen these cuts to improve their computational performance further. As a case study, we test our approach on the Multiple Knapsack Assignment Problem and the Temporal Knapsack Problem to show that the proposed cuts are helpful in accelerating the solution time without the need to implement branch and price. Finally, we consider a standard multi-thread computational environment and present a simple machine learning approach to identify problem instances when our cutting plane approach has the potential to improve computation time. This framework is simple and general, holding the promise of being implementable in a relatively easy way in general-purpose MIP solvers.

Paper Organization. The remainder of the paper is organized as follows. In Section 2, we provide some preliminaries on DW decomposition and introduce what we call DW Block cuts. In Section 3, we propose our new approach for recovering the DW bound in the original space using these cuts and discuss how they relate to the objective function cut. In Section 4, we discuss Lagrangian relaxation as an alternative approach for computing the DW bound and generating cutting planes. Some techniques for strengthening the proposed cuts are presented in Section 5. Section 6 reports our computational investigation while Section 7 proposes and tests the multi-thread hybrid algorithm. Finally, some short conclusions are drawn in Section 8.

2 Preliminaries

We assume MIP (1) is feasible. Throughout the paper, we call formulation (1) the original formulation of the MIP and call constraints $Ax \geq b$ (potentially empty) linking constraints. In the context of Dantzig-Wolfe decomposition, (1) is sometimes called the compact formulation. MIPs of this form with disjoint index sets $I(j)$ are called loosely coupled MIPs [18]. Note that we do not assume the original MIP (1) is loosely coupled and as such the supports of different blocks can have overlaps (e.g., in MIPs with two-stage [19] or overlapping [20] block structures). We call the LP relaxation of (1), obtained by dropping the integrality constraints $x \in X$, the natural LP relaxation, and denote its optimal objective value by z_L. Throughout, we assume that all data is rational.

2.1 Dantzig-Wolfe Decomposition

We next consider replacing the constraints $x_{I(j)} \in P^j$ in (1) by $x_{I(j)} \in \operatorname{conv}(Q^j)$, where

$$Q^j = \{y \in \mathbb{R}^{|I(j)|} : G^j y \geq g^j, y \in X^j\},$$

and X^j has the integrality constraints inherited from X for the variables $x_{I(j)}$. Then, one obtains the DW reformulation of problem (1). Relaxing the integrality constraints $x \in X$ in this formulation leads to the DW relaxation of (1):

$$\begin{aligned} z_D := \min\ & c^\top x \\ \text{s.t. } & x_{I(j)} \in \operatorname{conv}(Q^j), \quad j \in J, \\ & Ax \geq b. \end{aligned} \tag{2}$$

The DW bound z_D is potentially stronger than the natural LP relaxation bound z_L as $\operatorname{conv}(Q^j) \subseteq P^j$ for all $j \in \{1, \ldots, q\}$. Consequently, we have $z^* \geq z_D \geq z_L$. In practice, computing z_D is not a straightforward task as polyhedra $\big(\operatorname{conv}(Q^j)\big)_{j=1}^q$ are not given explicitly. For $j \in J$, let V^j and R^j denote the set of extreme points and the set of extreme rays of $\operatorname{conv}(Q^j)$, respectively. The DW relaxation (2) can be reformulated using $(V^j)_{j \in J}$ and $(R^j)_{j \in J}$, leading to the following extended formulation:

$$
\begin{aligned}
z_D = \min\ & c^\top x && && \text{(3a)}\\
\text{s.t. } & x_{I(j)} = \sum_{v\in V^j} \lambda_v v + \sum_{r\in R^j} \mu_r r, && j \in J, && \text{(3b)}\\
& \sum_{v\in V^j} \lambda_v = 1, && j \in J, && \text{(3c)}\\
& \lambda \geq 0,\ \mu \geq 0, && && \text{(3d)}\\
& Ax \geq b. && && \text{(3e)}
\end{aligned}
$$

This formulation can now be solved iteratively via column generation. Specifically, at each iteration, a *restricted* LP is solved by replacing V^j and R^j with a small collection of extreme points $\hat{V}^j \subseteq V^j$ and extreme rays $\hat{R}^j \subseteq R^j$. In each iteration, a new extreme point or a new extreme ray is generated by solving a *pricing* subproblem

$$D_j(\pi^j) := \min\left\{(\pi^j)^\top v : v \in \operatorname{conv}(Q^j)\right\} = \min\left\{(\pi^j)^\top v : v \in Q^j\right\} \tag{4}$$

for each block $j \in J$, where π^j above is the dual solution associated with constraints (3b) in the restricted LP relaxation of (3). By convention, we define $D_j(\pi^j) = -\infty$ if the pricing subproblem (4) is unbounded, and when this happens one can obtain an extreme ray $r \in R^j$ by finding an unbounded ray of $\min\{(\pi^j)^\top v : v \in P^j\}$, which is added to $\hat{R}^j$. If, on the other hand, $D_j(\pi^j)$ is finite, one obtains a solution of (4) as an extreme point $v \in V^j$. This point is added to $\hat{V}^j$ provided that it has a negative reduced cost that is computed by subtracting the dual variable associated with the j-th constraint of (3c) from $D_j(\pi^j)$. The restricted LP, with augmented vertices and rays, is then solved again and this process is repeated until no such points or rays are generated.

The restricted LP at each iteration gives an *upper bound* for z_D, and these upper bounds converge to z_D in a finite number of iterations as $|V^j|$ and $|R^j|$ are finite for all $j \in J$. When the algorithm terminates, in addition to the lower bound z_D, an optimal solution to (2) is obtained. If this solution does not satisfy the integrality constraints $x \in X$, branching is necessary to obtain an optimal solution of the original problem. The subproblems in the branch-and-bound tree can again be solved using column generation, leading to a branch-and-price procedure.

2.2 Dantzig-Wolfe Block Cuts

Note that while solving the DW relaxation, the pricing subproblems (4) can also be used to derive valid inequalities for (2). Specifically, for each $j \in \{1, \ldots, q\}$ any inequality of the form $\pi^\top y \geq D_j(\pi)$ is valid for $\operatorname{conv}(Q^j)$ for $\pi \in \mathbb{R}^{I(j)}$. In this paper, we call these inequalities DW Block cuts.

Definition 1. *An inequality is called a* Dantzig-Wolfe Block (DWB) cut *for* (1) *if it is of the form*

$$\pi^\top x_{I(j)} \geq D_j(\pi) \tag{5}$$

for some $j \in \{1, \ldots, q\}$ *and* $\pi \in \mathbb{R}^{I(j)}$*, where* $D_j(\cdot)$ *is defined in* (4).

As DWB cuts are essentially valid inequalities for the block polyhedra $\big(\operatorname{conv}(Q^j)\big)_{j=1}^q$, they can be viewed as special cases of a broader class of cutting planes called Fenchel cuts [21].

Adding some of these DWB cuts to the natural LP relaxation of (1) can lead to a stronger formulation and therefore a better dual bound than z_L. Let S denote the feasible region of the original problem (1). We next present a relationship between the dimension of a DWB cut in $\operatorname{conv}(S)$ and its restriction in $\operatorname{conv}(Q^j)$.

Definition 2. *We say that MIP* (1) *has* block relative feasibility *if, for each $j \in \{1, \ldots, q\}$ and $y \in Q^j$, there exists $x \in S$ such that $x_{I(j)} = y$, i.e., $proj_{x_{I(j)}}(S) = Q^j$ for $j \in J$.*

The block relative feasibility assumption holds for a broad class of MIP problems with decomposable structures, including two-stage stochastic integer programs with relatively complete recourse [22].

Proposition 1. *Assume problem* (1) *has block relative feasibility. If $(\pi^j)^\top y \geq D_j(\pi^j)$ defines a d-dimensional face of conv(Q^j) for some $j \in \{1, \ldots, q\}$, then* (5) *defines a face of conv(S) of dimension at least d.*

Proof. Since $(\pi^j)^\top y \geq D_j(\pi^j)$ defines a d-dimensional face of $\text{conv}(Q^j)$, there exists $d+1$ affinely independent points $\{y^k\}_{k=1}^{d+1} \subseteq Q^j$. By the block relative feasibility assumption, there exist points $\{x^k\}_{k=1}^{d+1} \subseteq S$ such that $x^k_{I(j)} = y^k$. Points $\{x^k\}_{k=1}^{d+1}$ are affinely independent from each other by affine independence of $\{y^k\}_{k=1}^{d+1}$. The conclusion then follows from the fact that $\{x^k\}_{k=1}^{d+1}$ are all on the face associated with (5) in $\text{conv}(S)$. □

Proposition 1 indicates that DWB cuts whose restrictions in the space of $x_{I(j)}$ correspond to high-dimensional faces of $\text{conv}(Q^j)$ are likely to define high-dimensional faces in $\text{conv}(S)$. This motivates the idea of strengthening some DWB cuts to obtain higher-dimensional DWB cuts, which will be discussed in Section 5. We next investigate how to generate critical DWB cuts for obtaining strong dual bounds.

3 Incorporating the DW Bound Into the Formulation

In a number of applications, it has been shown that the DW bound can be significantly stronger than the natural LP relaxation bound of (1) [23, 24, 25]. However, even if this bound can be effectively computed, enforcing the integrality constraints $x \in X$ requires a specialized branch-and-price algorithm. A straightforward approach to recover the DW bound z_D in the original space is to add a single cut $c^\top x \geq z_D$, to the LP relaxation of (1) which we call the *objective function cut*. However, it is well known that adding such an objective function cut often slows down MIP solvers in practice.

We next observe a basic property of the optimal face of the LP relaxation after adding the objective function cut.

Proposition 2. *Let P be a polyhedron in $\mathbb{R}^n$. If neither $c^\top x \leq v$ nor $c^\top x \geq v$ is valid for P, then dim$(P \cap \{x : c^\top x = v\}) =$ dim$(P) - 1$.*

Proof. Define $P^+ := P \cap \{x : c^\top x \leq v\}$. Note that $c^\top x \leq v$ is an irredundant inequality for P^+ because $c^\top x \leq v$ is not valid for P. By [26, Lemma 3.26], $\dim(P \cap \{x : c^\top x = v\}) = \dim(P^+) - 1$. We next show that the affine hull of P^+ is equal to the affine hull of P, and thus $\dim(P^+) = \dim(P)$. By [26, Theorem 3.17], we only need to show the following two statements:

1. Equality $c^\top x = v$ is not valid for P^+;

2. If $a^\top x \leq a_0$ is valid for P but $a^\top x = a_0$ is not valid for P, then $a^\top x = a_0$ is not valid for P^+.

Note that $c^\top x \geq v$ is not valid for P. Therefore, there exists $\hat{x} \in P$ such that $c^\top \hat{x} < v$. The first statement then follows from the fact that $\hat{x} \in P^+$. Similarly, there exists $\bar{x} \in P$ such that $a^\top \bar{x} < a_0$. For $\lambda \in (0, 1)$, define $x^\lambda := (1 - \lambda)\hat{x} + \lambda \bar{x}$. Because $a^\top \bar{x} \leq a_0$, we have $x^\lambda \in P^+$ but $a^\top x^\lambda < a_0$ for a small enough λ. The second statement then follows. □

Therefore, if $z_D > z_L$, adding the objective function cut to the original formulation would often create an optimal face almost as high-dimensional as the original LP relaxation polyhedron. This, in turn, can cause not only performance variability [27], but also serious computational issues especially in early stages of the branch and cut in terms of branching [28], as well as cutting plane generation.

3.1 An alternative approach

In Section 2.2, we observed that valid inequalities for the DW relaxation can be generated while solving the pricing subproblems. We next show that using a small number of such cuts can readily recover z_D. Assume that at iteration t of DW decomposition, the restricted LP is of the form

$$\begin{aligned} z_D^t = \min\ & c^\top x \\ \text{s.t. }\ & x_{I(j)} = \sum_{v\in\hat{V}^j} v\lambda_v^j + \sum_{r\in\hat{R}^j} r\mu_r^j, && j\in J, && (\pi^{j,t}) \\ & Ax \geq b, && && (\beta^t) \\ & \sum_{v\in\hat{V}^j} \lambda_v^j = 1, && j\in J, && (\theta_j^t) \\ & \lambda^j \geq 0,\ \mu^j \geq 0, && j\in J, \end{aligned} \tag{6}$$

and let $(\pi^{1,t},\ldots,\pi^{q,t},\beta^t,\theta_1^t,\ldots,\theta_q^t)$ be the optimal dual solution associated with this restricted LP. We then have the following result for DWB cuts derived from the optimal dual solution of the last iteration of DW decomposition.

Theorem 3. *Assume DW decomposition terminates in $\bar{t}$ iterations. Then,*

$$z_D = \min\ c^\top x \tag{7a}$$
$$s.t.\ (\pi^{j,\bar{t}})^\top x_{I(j)} \geq D_j(\pi^{j,\bar{t}}), \quad j\in J, \tag{7b}$$
$$Ax \geq b. \tag{7c}$$

Proof. The "$\geq$" direction of (7a) is implied by the definition of z_D in (2) as inequality (7b) is valid for $\operatorname{conv}(Q^j)$ for $j\in J$. We next show the "$\leq$" direction. Based on LP duality of (6) at iteration $\bar{t}$ and the termination condition of DW decomposition, the following equalities hold:

1. $z_D = b^\top\beta^{\bar{t}} + \sum_{j=1}^q \theta_j^{\bar{t}}$;
2. $c_i = A_i^\top\beta^{\bar{t}} + \sum_{j:i\in I(j)} \pi_i^{j,\bar{t}}$, $i=1,\ldots,n$.

Note that at the last iteration $\bar{t}$, the DW pricing subproblems are bounded. Let $(v^{j,\bar{t}})_{j=1}^q$ denote the solutions of the DW pricing subproblems at iteration $\bar{t}$. Note that the reduced costs associated with points $(v^{j,\bar{t}})_{j=1}^q$ are nonnegative at iteration $\bar{t}$ of DW decomposition, i.e., $(\pi^{j,\bar{t}})^\top v^{j,\bar{t}} - \theta_j^{\bar{t}} = D_j(\pi^{j,\bar{t}}) - \theta_j^{\bar{t}} \geq 0$ for $j\in J$. Therefore, for each solution x satisfying (7b) and (7c), we have the following inequality:

$$\begin{aligned} c^\top x = \sum_{i=1}^n c_i x_i = \sum_{i=1}^n \Big[x_i A_i^\top \beta^{\bar{t}} + \sum_{j:i\in I(j)} x_i\pi_i^{j,\bar{t}}\Big] &= \underbrace{(\beta^{\bar{t}})}_{\geq 0}{}^\top \underbrace{Ax}_{\geq b} + \sum_{j=1}^q \underbrace{(\pi^{j,\bar{t}})^\top x_{I(j)}}_{\geq D_j(\pi^{j,\bar{t}})} \\ &\geq b^\top\beta^{\bar{t}} + \sum_{j=1}^q D_j(\pi^{j,\bar{t}}) \geq b^\top\beta^{\bar{t}} + \sum_{j=1}^q \theta_j^{\bar{t}} = z_D. \end{aligned} \tag{8}$$

□

We call inequalities (7b) *last-iteration DWB cuts*. Theorem 3 shows that q last-iteration DWB cuts together with linking constraints recover the DW bound z_D. We remark that the last-iteration DWB cuts are not necessarily all nontrivial. It is possible that $\pi^{j,\bar{t}} = 0$ for some $j \in J$, which implies that the convexification of the j-th block has no impact on improving the dual bound.

It is also worth emphasizing that (7) is not a valid formulation for the MIP (1) even if we add integrality constraints $x \in X$ to it. One should use last-iteration DWB cuts as cutting planes and add them to the original formulation (1) to obtain a valid formulation whose LP relaxation bound is precisely z_D. Also note that Theorem 3 does not imply that last-iteration DWB cuts dominate other DWB cuts that can be generated at intermediate iterations $\tau < \bar{t}$, in the sense that intermediate-iteration DW cuts may still cut off fractional points that do not violate any of the last-iteration DWB cuts.

3.2 Dual Degeneracy and LP Optimal Face

When comparing the strength of different collections of cutting planes or different formulations, very often the LP relaxation bound is used as the sole criterion. However, the effectiveness of two formulations in branch and cut may differ significantly even when they have very similar (or, the same) LP relaxation bounds. An additional property that should also be taken into account is the dual degeneracy of the LP relaxation of the formulation [28]. A dual basic solution of an LP is called *dual degenerate* if at least one of the dual basic variables is set to 0 in that solution. Next, we formally define the degeneracy level of a dual basic solution of an LP (given in inequality form).

Definition 3. *Consider an LP with n variables and m inequality constraints, and let w be a basic feasible* dual *solution. We define the* degeneracy level *of w to be $n - \|w\|_0$.*

A highly dual degenerate LP relaxation is associated with many alternative LP basic primal optimal solutions, which usually corresponds to a large optimal face. The following result shows how the size of the optimal face (more precisely, its dimension) is related to the degeneracy level of a dual basic optimal solution.

Proposition 4. *Assume $w^* \in \mathbb{R}^m_+$ is a dual basic optimal solution of an LP with n variables and m inequality constraints. Then, the optimal face of the LP has dimension at most $n - \|w^*\|_0$. Furthermore, if w^* is the unique dual optimal solution, then the optimal face of the LP has dimension exactly $n - \|w^*\|_0$.*

Proof. Assume the LP is of the form $\min\{c^\top x : Gx \geq h\}$. Let F denote the optimal face of the LP, i.e.,

$$F = \{x : Gx \geq h,\ c^\top x \leq h^\top w^*\}. \tag{9}$$

Let $(g^k)^\top$ denote the k-th row of G. By complementary slackness of LP, $(g^k)^\top x = h_k$ for all $x \in F$ for all k with $w^*_k > 0$. Since w^* is a dual basic optimal solution, $\{(g^k, h_k)\}_{k:w^*_k>0}$ are linearly independent. Otherwise, there exists $\beta \in \mathbb{R}^m \setminus \{\mathbf{0}\}$ such that $\beta_k = 0$ for all k with $w^*_k = 0$ and $\sum_{k:w^*_k>0} \beta_k(g^k, h_k) = \mathbf{0}$. Then, note that $w^* + \epsilon\beta$ and $w^* - \epsilon\beta$ are both dual optimal solutions of the LP for small enough positive ϵ, which contradicts the fact that w^* is a dual basic optimal solution. Therefore, $\dim(F) \leq n - \operatorname{rank}(\{g^k\}_{k:w^*_k>0}) = n - \operatorname{rank}(\{(g^k, h_k)\}_{k:w^*_k>0}) = n - \|w^*\|_0$. Here, the first inequality follows from [26, Theorem 3.17], the second equality follows from the consistency of the linear system $\{(g^k)^\top x = h_k\}_{k:w^*_k>0}$, and the third equality follows from linear independence of $\{(g^k, h_k)\}_{k:w^*_k>0}$.

If w^* is the unique dual optimal solution, by strict complementary slackness of LP [29], there exists an optimal solution $x^* \in F$ of the LP, such that $(g^k)^\top x^* > h_k$ for all k with $w_k^* = 0$. It implies that $\{(g^k)^\top x = h_k\}_{k:w_k^*>0}$ and $c^\top x = h^\top w^*$ are exactly all the implicit equalities that hold in the inequality description (9) of F. By LP duality, $c^\top x = h^\top w^*$ is implied by $\{(g^k)^\top x = h_k\}_{k:w_k^*>0}$. Therefore, by [26, Theorem 3.17], $\dim(F) = n - \operatorname{rank}(\{g^k\}_{k:w_k^*>0}) = n - \|w^*\|_0$. □

We remark that Proposition 4 does not extend to dual nonbasic optimal solutions, moreover, the ℓ_0-norm of dual nonbasic optimal solutions can be greater than n. Note that the dual optimal solution is unique when the primal solution is nondegenerate. For the primal degenerate case, even if there exists a unique dual *basic* optimal solution, it is possible that the dimension of the optimal face of the LP is strictly less than the degeneracy level of that dual basic optimal solution. See Example 2 in Appendix C for an example where the unique dual *basic* optimal solution has a strictly positive dual degeneracy level but the primal optimal solution is still unique.

Under some mild assumptions, Proposition 4 implies the following.

Proposition 5. *Assume the LP* (7) *has a unique dual optimal solution. Then, the optimal face of* (7) *has dimension* $n - q - \|\beta^{\bar{t}}\|_0$.

Proof. Note that the proof of Theorem 3 implies that $(1, \ldots, 1, \beta^{\bar{t}})$ is a dual optimal solution of (7). The result then follows from Proposition 4. □

Proposition 5 also implies that if the dual optimal solution is unique, then none of the last-iteration DWB cuts can be redundant. If this is not the case, the dimension of the optimal face after adding the last iteration cuts depends on the number of cuts that are active. Note that when applying the last-iteration DWB cuts in practice, we would add them to the original formulation (1), resulting in an LP optimal face whose size can be even smaller due to constraints $x_{I(j)} \in P^j$, $j \in J$.

4 Bound Computation and Cut Generation via Lagrangian Relaxation

We next discuss an alternative approach to generate cutting planes to recover the DW bound that uses Lagrangian relaxation [30]. As we discuss later, this approach has better computational performance in practice due to stabilization.

In Lagrangian relaxation, separate auxiliary variables are created for each block and these auxiliary variables are related to the original variables using additional (copying) constraints. The copying constraints together with the linking constraints $Ax \geq b$ are then dualized into the objective to obtain a Lagrangian relaxation of (2). More precisely, one writes

$$\begin{aligned} z_D = \min\ & c^\top x && && \text{(10a)}\\ \text{s.t. } & y^j \in \operatorname{conv}(Q^j), && j \in J, && \text{(10b)}\\ & y^j = x_{I(j)}, && j \in J, \qquad (\pi^j) && \text{(10c)}\\ & Ax \geq b. && \qquad\qquad (\beta) && \text{(10d)} \end{aligned}$$

After dualizing constraints (10c) and (10d) using multipliers π and $\beta \geq 0$, one obtains the following

Lagrangian relaxation:

$$\begin{aligned} z(\pi, \beta) = \min\ & c^\top x + \sum_{j=1}^{q} (\pi^j)^\top (y^j - x_{I(j)}) + \beta^\top (b - Ax), \\ \text{s.t. } & y^j \in Q^j, \quad j \in J. \end{aligned} \tag{11}$$

Note that when the index sets $\{I(j)\}_{j=1}^q$ are nonoverlapping, (10) and (11) can be simplified by properly removing the copying constraints and the associated dual variables π.

In general, it follows from Lagrangian duality [31] that the largest Lagrangian relaxation bound matches z_D, i.e.,

$$z_D = \max_{\beta \geq 0, \pi} z(\pi, \beta). \tag{12}$$

Note that x is unconstrained in (11) and therefore $z(\pi, \beta) = -\infty$ unless the coefficients of the x variables in the objective function are zero, i.e.,

$$c_i - \sum_{j: i \in I(j)} \pi_i^j - \beta^\top A_i = 0 \qquad \text{for all } i \in \{1, \dots, n\},$$

where A_i denotes the i-th column of A. Consequently, the Lagrangian dual problem (12) can be equivalently written as the following Wolfe dual problem:

$$z_D = \max\ z(\pi, \beta) \tag{13a}$$

$$\text{s.t. } \sum_{j: i \in I(j)} \pi_i^j + \beta^\top A_i = c_i, \quad i = 1, \dots, n, \tag{13b}$$

$$\beta \geq 0. \tag{13c}$$

For (π, β) satisfying (13b) and (13c), it holds that

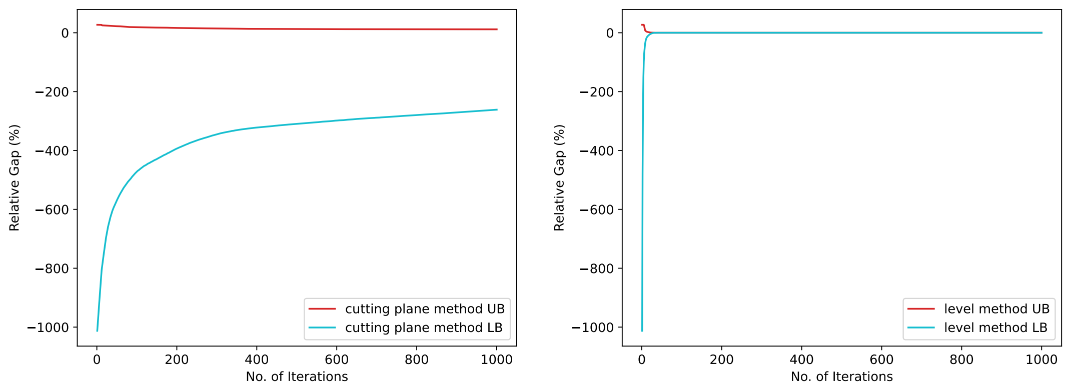
$$z(\pi, \beta) = \sum_{j=1}^{q} D_j(\pi^j) + b^\top \beta,$$

where $D_j : \mathbb{R}^{|I(j)|} \to \mathbb{R} \cup \{-\infty\}$ is a piecewise linear concave function of the form

$$D_j(\pi^j) = \min\{(\pi^j)^\top v : v \in Q^j\}. \tag{14}$$

Therefore, (13) is a nonsmooth convex optimization problem with a separable objective function. It is worth emphasizing that the pricing problem (4) in DW decomposition has exactly the same form as (14). The function values and supergradients of the concave function $D_j(\cdot)$ can be evaluated by solving (14) [30] (an optimal solution of (14) is a supergradient of $D_j(\cdot)$ at π^j). This alternative way of viewing DW bound z_D as the optimal value of (13) allows us to use various convex optimization methods for computing DW bound z_D. For example, DW decomposition is equivalent to applying the classical cutting plane method [32] to solve (13). Since the description of Q^j involves integer variables in general, functions $D_j(\cdot)$ are often piecewise linear concave with exponentially many pieces. In that case, convex optimization methods with some stabilization techniques (e.g., the level method [33]) often outperform the cutting plane method, and the difference can be significant. Figure 2 is a representative example of the difference in performance between the cutting plane method and the level method for computing the DW bound z_D (averaged over a set of multiple knapsack assignment problem instances that are used in Section 6). Details about our implementation of the level method are presented in Appendix B.

Figure 2: Comparison of the Cutting Plane Method (Left) and the Level Method (Right)



4.1 Cut Generation From the Dual

As discussed earlier, solving the dual problem can be computationally more efficient than the standard DW decomposition. During the solution of the dual problem (13), DWB cuts similar to (5) can also be generated every time we evaluate the function values of $D_j(\cdot)$. The following result demonstrates the strength of DWB cuts generated from the evaluation of the Lagrangian dual function at any point (π, β) with $z(\pi, \beta) > -\infty$.

Proposition 6. *Let* (π, β) *be dual multipliers for* (11) *satisfying constraints* (13b) *and* (13c). *Then,*

$$z(\pi,\beta) \le \min\ c^\top x \qquad \tag{15}$$
$$\text{s.t. } (\pi^j)^\top x_{I(j)} \ge D_j(\pi^j), \quad j \in J,$$
$$Ax \ge b.$$

Proof. Consider any $x \in \mathbb{R}^n$ feasible to the right-hand side LP of (15). Since $c_i = \sum_{j:i\in I(j)} \pi_i^j + \beta^\top A_i$ for $i = 1, \dots, n$ by (13b), we have

$$c^\top x = \sum_{i=1}^{n} \sum_{j:i\in I(j)} \pi_i^j x_i + \beta^\top Ax = \sum_{j=1}^{q} \underbrace{(\pi^j)^\top x_{I(j)}}_{\ge D_j(\pi^j)} + \underbrace{\beta}_{\ge 0}{}^\top \underbrace{Ax}_{\ge b} \ge \sum_{j=1}^{p} D_j(\pi^j) + b^\top \beta(\tau) = z(\pi,\beta).$$

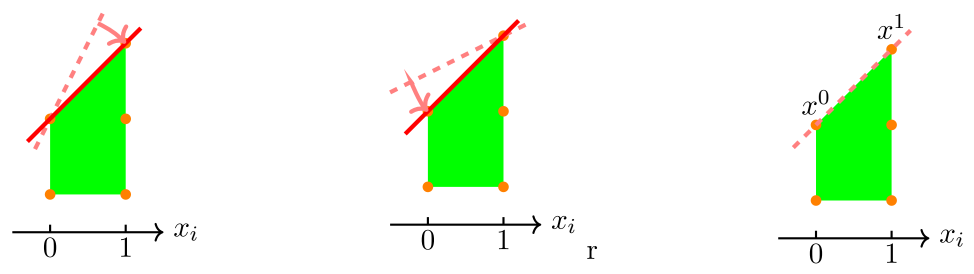
□

Note that, unlike Theorem 3, Proposition 6 does not depend on how the dual multiplier is obtained. If one can solve the dual problem (13) to optimality, then Proposition 6 implies that one can recover DW bound using DWB cuts associated with that optimal solution of (13). The single block case ($q = 1$) of Proposition 6 simplifies to the idea of Lagrangian cuts [34].

Corollary 7. *Let* $(\bar\pi, \bar\beta)$ *be an optimal solution of* (13). *Then,*

$$z_D = \min\ c^\top x \tag{16a}$$
$$\text{s.t. } (\bar\pi^j)^\top x_{I(j)} \ge D_j(\bar\pi^j), \quad j \in J, \tag{16b}$$
$$Ax \ge b. \tag{16c}$$

Figure 3: Disjunctive Coefficient Strengthening for Three Cases (Dashed: Original Cut, Solid: Strengthened Cut)



Results similiar to Propostion 5 can be derived for the optimal face of the LP (16) that utilizes DWB cuts associated with an optimal Lagrangian dual solution. We also call inequalities (16b) *last-iteration DWB cuts* as they recover the same dual bound z_D and are often generated in the last iteration of the Lagrangian dual algorithm. Even if the Lagrangian dual problem is not solved to optimality, Proposition 6 still guarantees that a dual bound that is at least as strong as the best Lagrangian dual bound can be obtained by generating DWB cuts associated with the dual solution that provides the strongest bound so far. Besides, there may be values for adding DWB cuts obtained at different dual multipliers. See Example 1 in Appendix C showing that DWB cuts can potentially provide stronger dual bounds than the best Lagrangian dual bound.

5 Generating a Stronger Relaxation

In this section we describe how to strengthen DWB cuts to obtain a stronger relaxation. The strengthened cutting planes are still DWB cuts and therefore do not lead to bounds stronger than z_D. However, these strengthened cutting planes may potentially make more constraints active in the LP relaxation, and therefore can help reduce the dual degeneracy level of the formulation and improve MIP solver performance. Moreover, the strengthened cutting planes are more likely to define high-dimensional faces of the original problem, as they define higher-dimensional faces of the block polyhedra $\text{conv}(Q^j)$.

5.1 Disjunctive Coefficient Strengthening

We first describe a disjunctive coefficient strengthening technique for binary variables [35] that strengthens the coefficients of a valid inequality one at a time. Given a valid inequality $a^\top x \geq f$ for a mixed integer linear set Q, let $Q^= := \{x \in Q \ : \ a^\top x = f\}$. For a binary variable x_i, its coefficient a_i in the cut can be strengthened if one of the following two cases hold: (i) $x_i = 0$ for all $x \in Q^=$, or, (ii) $x_i = 1$ for all $x \in Q^=$. If there are points $x^0, x^1 \in Q^=$ such that $x_i^0 = 0$ and $x_i^1 = 1$, then this approach does not improve (i.e., decrease) the coefficient of the variable. Figure 3 shows two-dimensional examples of all three cases.

Note that if we solve

$$\bar{f} = \min\{a^\top x : x \in Q, x_i = 1\} \tag{17}$$

and observe that $\bar{f} > f$, then we can strengthen the original inequality $a^\top x \geq f$ to be $a^\top x \geq f + (\bar{f} - f)x_i$ using the disjunction

$$Q = \{x \in Q : x_i = 0\} \cup \{x \in Q : x_i = 1\}.$$

Similarly, if we solve

$$\bar{f} = \min\{a^\top x : x \in Q, x_i = 0\}, \tag{18}$$

and observe that $\bar{f} > f$, then we can strengthen the original inequality $a^\top x \geq f$ to be $a^\top x \geq f + (\bar{f} - f)(1 - x_i)$. If either problem (17) or (18) is infeasible, then one can simply fix variable x_i to 0 or 1, respectively. It is easy to verify that the original inequality is implied by the strengthened inequality together with the bound constraint $x_i \geq 0$ or $x_i \leq 1$. Therefore, if the original inequality is active in the LP relaxation, then one round of coefficient strengthening (if applicable) would potentially make a bound constraint active in the LP relaxation and reduce the dual degeneracy level. Note that this approach does not increase the size of the formulation.

For strengthening a DWB cut obtained from a block j, we set $Q = Q^j$ and apply coefficient strengthening sequentially to all coefficients of binary variables. Note that using a different ordering of the binary variables may lead to different strengthened cutting planes in the end. For simplicity, we use the ordering of the variables in the original formulation to strengthen DWB cuts in our numerical experiments. We also keep a set L of points that are known to be elements of $Q^=$, generated from previous solutions of (14), (17) and (18). If there are $x^0, x^1 \in L$ such that $x^0_i = 0$ and $x^1_i = 1$, then without solving (17) or (18) we conclude that disjunctive strengthening cannot be applied to the i-th coefficient.

5.2 Strengthening via Tilting

We next describe a tilting technique introduced by [36, 37] that starts with a valid inequality and iteratively tilts it to obtain a facet-defining inequality. Let $a^\top x \geq f$ be a valid inequality for Q and assume that it is not facet-defining. Also assume that $\mathrm{conv}(Q)$ is full dimensional and there is a set $Q' \subseteq Q$ such that all points $x \in Q'$ satisfy $a^\top x = f$. The algorithm first generates a point $\bar{x} \in Q \setminus Q'$ that satisfies $a^\top \bar{x} > f$, and a vector (v, w) such that $v^\top x = w$ for all $x \in Q' \cup \{\bar{x}\}$. The algorithm then does the following:

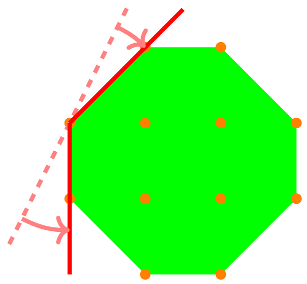
1. If $v^\top x \geq w$ is valid for Q, then the algorithm outputs $\bar{x}$. Otherwise the algorithm computes the largest $\lambda^+ \in \mathbb{R}_+$ such that $(a + \lambda^+ v)^\top x \geq f + \lambda^+ w$ is valid for Q and outputs this inequality together with a point $\bar{x}^+ \in Q \setminus Q'$ satisfying $(a + \lambda^+ v)^\top \bar{x}^+ = f + \lambda^+ w$;

2. If $v^\top x \leq w$ is valid for Q, then the algorithm outputs $\bar{x}$. Otherwise the algorithm computes the largest $\lambda^- \in \mathbb{R}_+$ such that $(a - \lambda^- v)^\top x \geq f - \lambda^- w$ is valid for Q and outputs this inequality together with a point $\bar{x}^- \in Q \setminus Q'$ satisfying $(a - \lambda^- v)^\top \bar{x}^- \geq f - \lambda^- w$.

Note that when $\mathrm{conv}(Q)$ is full dimensional, both $v^\top x \geq w$ and $v^\top x \leq w$ cannot be valid and consequently we obtain two (possibly identical) valid inequalities whose conic combination implies the original inequality $a^\top x \geq f$ (depicted in Figure 4). Moreover, each one of the these inequalities has one more *known* feasible point on its associated face than $a^\top x \geq f$. [37] show that recursively tilting one of the two obtained valid inequalities leads to a facet-defining inequality for $\mathrm{conv}(Q)$.

In our context, we apply the tilting idea with the following modifications. For a DWB cut associated with block j, we set Q to be Q^j, and instead of picking one of the two tilted inequalities for the subsequent tilting iteration, we apply tilting to both. By doing so, we create a binary tree where the root node corresponds to the original DWB cut and the remaining nodes correspond to inequalities obtained by tilting the inequalities associated with their parent nodes. For any such tree, cuts associated with the leaf nodes imply the original DWB cut associated with the root node. We call the collection of inequalities associated with the leaf nodes of a depth-d tree *depth-d tilted DWB cuts*. Using a depth-d tree means replacing one DWB cut with up to 2^d DWB cuts, which can be computationally expensive if d is large. Therefore, it is often beneficial to choose a relatively

Figure 4: Tilting One Valid Inequality Into Two Stronger Valid Inequalities



small value for d. Note that this process does not improve the DW bound but can potentially help reduce the dimension of the LP optimal face as more constraints are likely to become active at the optimal face.

To improve computational performance, we collect the feasible points for block $j \in J$ that we encounter during the overall algorithm and store them in a set $\hat{Q}^j \subseteq Q^j$. Using this set of points can reduce the number of oracle calls needed as follows. First, we can check if any point in $\hat{Q}^j$ satisfies the condition for $\bar{x}$ and use it for choosing (v, w), thus avoiding an extra call to the optimization oracle. In addition, when computing λ^+, we use $\hat{Q}^j$ to obtain the following upper bound on it:

$$\lambda^+ := \min_{x \in Q^j : v^\top x < w} \frac{a^\top x - f}{w - v^\top x} \quad \leq \quad \min_{x \in \hat{Q}^j : v^\top x < w} \frac{a^\top x - f}{w - v^\top x}.$$

This upper bound can be very close to the final λ^+ value and, in practice, we have observed a reduction in the number of oracle calls needed to compute λ^+. This in turn reduces the time spent on tilting significantly. We use the same idea when computing $\lambda^- \in \mathbb{R}_+$. Finally, we would like to remark that the overall procedure can be sensitive to the optimality gap tolerances in modern MIP solvers. One has to be cautious about the validity of the generated cuts in practice.

6 Computational Experiments

We test our approaches on the Multiple Knapsack Assignment Problem (MKAP) introduced by [38] and the Temporal Knapsack Problem (TKP) as coined by [20]. Detailed descriptions of MKAP and TKP, their DW reformulations as well as the test instances are presented in Appendices D and E, respectively.

We compare the performance of the following formulations:

1. *MIP*: The original formulation (1);

2. *OBJ*: Objective function cut $c^\top x \geq z_D$ added into the original formulation;

3. *DWB*: Last-iteration DWB cuts (16b) added into the original formulation;

4. *STR*: Last-iteration DWB cuts (16b) with disjunctive coefficient strengthening added into the original formulation;

5. *DdT*: Last-iteration DWB cuts (16b) with disjunctive coefficient strengthening and depth-d tilting added into the original formulation.

All experiments are run on a computer with a Intel(R) Xeon(R) Gold 6258R processor running at 2.7GHz, with up to 4 threads used. All optimization problems are solved using the optimization solver Gurobi with version 9.5.0 for MKAP and version 10.0.2 for TKP.

In MKAP, the parameters $|K|$, $|M|$, and $|N|$ define the size of the instances. In all the tables presented in this section, statistics presented in a row correspond to averages over 30 instances and 10 instances for MKAP and TKP, respectively. To avoid repetition, detailed statistics are reported only on MKAP test instances in Sections 6.1-6.3. For TKP, we concentrate on a smaller set of hard instances in Section 6.4.

6.1 Comparing Relaxations for MKAP

We first present some results regarding lower bounds for MKAP. We compare the DW bound z_D with the LP relaxation bound z_L and the Gurobi root node bound (denoted by z_R) obtained by adding solver cuts. With some abuse of notation, we use z_D to denote the best dual bound obtained by the level method. In theory, z_D should be equal to the DW bound. However, numerical issues happen occasionally when solving the quadratic program in the level method, especially when the DW bound is close to the LP relaxation bound, in which case z_D can be strictly less than z_L.

In Table 1, we present for MKAP the relative gaps $r_L := (z_D - z_L)/|z_D|$ and $r_R := (z_D - z_R)/|z_D|$ between z_D and natural LP relaxation bound z_L as well as the Gurobi root bound z_R, respectively. The running time t_R spent at the root node and running time t_D spent on solving the Lagrangian dual problem by the level method are also reported.

Depending on the instance size, the relative difference between the DW bound and the natural LP relaxation bound varies. When the gap is small, Gurobi can often generate a root node bound that is almost as strong as or even slightly stronger than the DW bound. When the gap is large, Gurobi can close some gap at the root node but the bound is often significantly weaker than the DW bound.

In our preliminary experiments, we observe that DWB cuts are more likely to be effective when DW bound is significantly stronger than the natural LP relaxation bound. Therefore, we focus on instance classes (defined by the size parameters $|K|$, $|M|$ and $|N|$) whose average natural LP relaxation bound is more than 1% weaker than average DW bound, especially the ones with Gurobi root bound more than 1% worse than DW bound. We denote these two sets of MKAP instances by I_1 and $I_2 (\subseteq I_1)$, respectively. In Table 1, I_1 instances correspond to rows with bold r_L and I_2 instances correspond to rows with bold r_R. Set I_1 consists of 870 instances and set I_2 consists of 270 instances. Over the I_1 instances, we observe that z_D is very close to the DW bound with relative gaps always less than 0.01% except on 3 instances (whose relative gaps are 0.03%, 0.16% and 0.34%, respectively).

We also observe that formulations with stronger strengthening tend to have less degenerate dual solutions. We report results regarding the dual degeneracy of optimal dual solutions associated with LP relaxations of different formulations in Appendix F. Time spent on generating these different formulations for MKAP is presented in Appendix G.

6.2 Objective Function Cut for MKAP

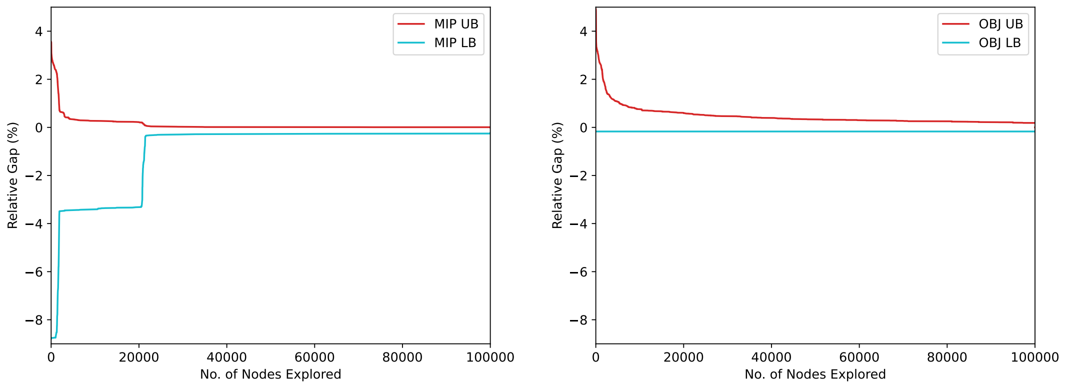
We next emprically investigate how the objective function cut (formulation *OBJ*) may increase the computing time on solving MKAP compared with simply using the original formulation *MIP*.

We consider MKAP instances with $(|K|, |M|, |N|) = (25, 20, 300)$, for which DW bound is much stronger than the natural LP relaxation bound (with average gap 10.18%). Out of these 30 instances, Gurobi can solve 22 using *MIP* and 2 using *OBJ*. We plot in Figure 5 the average

Table 1: Comparison of Different Bounds and Their Computing Time for MKAP

$\|K\|$	$\|M\|$	$\|N\|$	r_L (%)	r_R (%)	t_R (s)	t_D (s)
2	10	50	0.16	-0.04	0.1	0.8
		100	0.00	-0.03	0.3	2.0
		200	0.00	-0.01	0.5	9.9
		300	0.00	-0.01	0.5	22.0
	20	50	**1.98**	0.08	0.4	0.7
		100	0.02	0.00	0.3	2.8
		200	0.00	0.00	0.6	12.7
		300	-0.01	-0.01	0.8	37.0
	30	50	**12.05**	-0.06	0.1	0.6
		100	0.18	0.10	0.4	2.8
		200	0.00	0.00	0.8	15.4
		300	-0.05	-0.05	1.1	49.7
	40	50	**39.11**	0.00	0.0	0.7
		100	0.68	0.18	1.9	2.7
		200	0.00	0.00	0.9	17.3
		300	-0.34	-0.34	1.4	55.3
5	10	50	**1.27**	0.44	0.3	0.6
		100	0.11	0.07	0.2	2.4
		200	0.01	0.00	0.4	12.6
		300	0.00	0.00	0.6	43.0
	20	50	**2.97**	0.00	0.3	0.8
		100	0.10	0.06	0.3	3.0
		200	-0.44	-0.44	0.5	12.9
		300	-0.02	-0.02	0.8	50.5
	30	50	**12.27**	0.08	0.1	1.0
		100	0.41	0.15	1.5	3.3
		200	0.01	0.00	0.8	17.7
		300	0.00	0.00	1.2	55.8
	40	50	**39.11**	0.00	0.0	1.2
		100	1.00	0.14	1.5	3.4
		200	0.01	0.00	1.1	23.1
		300	-0.05	-0.05	1.5	47.0
10	10	50	**7.03**	0.86	0.2	0.5
		100	**5.65**	**4.13**	0.2	1.2
		200	**3.53**	**2.64**	0.3	4.5
		300	**3.64**	**2.79**	0.5	14.0
	20	50	**4.76**	-0.08	0.2	1.1
		100	0.74	0.37	1.4	2.7
		200	0.02	0.02	0.6	17.5
		300	0.00	0.00	0.9	45.5
	30	50	**12.82**	0.01	0.1	1.4
		100	0.88	0.15	1.2	3.5
		200	0.02	0.01	0.9	22.8
		300	0.00	0.00	1.3	26.1
	40	50	**39.11**	0.00	0.0	1.8
		100	**1.51**	0.12	1.0	4.0
		200	0.04	0.02	1.2	26.6
		300	0.00	0.00	1.8	25.8
25	10	50	**43.05**	0.21	0.1	0.9
		100	**54.17**	**1.76**	0.3	1.2
		200	**58.14**	**13.64**	0.8	1.6
		300	**66.02**	**16.63**	1.5	2.1
	20	50	**11.08**	-0.01	0.1	2.0
		100	**9.40**	0.45	0.7	2.7
		200	**8.74**	**6.83**	0.9	5.1
		300	**10.18**	**8.78**	0.9	9.4
	30	50	**14.35**	0.00	0.0	2.8
		100	**3.80**	0.13	1.0	4.2
		200	**1.79**	0.85	7.0	12.6
		300	**1.30**	**1.29**	1.4	28.3
	40	50	**39.21**	0.01	0.0	3.5
		100	**3.13**	0.00	1.1	5.5
		200	0.75	0.25	11.5	22.7
		300	0.25	0.24	2.0	24.8

Figure 5: Relative Gaps Between Bounds and Optimal Value by *MIP* (Left) and *OBJ* (Right) for MKAP



relative upper and lower bound gaps (with respect to the optimal value) obtained by Gurobi using formulations *MIP* and *OBJ* on $(25, 20, 300)$ instances as the number of branching nodes explored increases. We observe that although DW bound is almost equal to the optimal value for these instances (with a 0.01% average relative gap), Gurobi can hardly produce good feasible solutions to improve the primal bound when using the *OBJ* formulation. Instead, Gurobi can find better feasible solutions much more efficiently when using *MIP*, where all optimal solutions are found after exploring a bit more than 20,000 nodes, leaving a few instances unsolved because Gurobi cannot prove optimality within the timelimit. This shows that a single objective function cut can already significantly influence the performance of Gurobi's primal heuristics, which is plausibly due to worse branching decisions and cut generation for *OBJ*. A table reporting the average number of different cutting planes generated when using different formulations is presented in Appendix H.

6.3 MIP Experiments on MKAP

Table 2: Comparison of Gurobi Performance on MKAP I_2 Instances

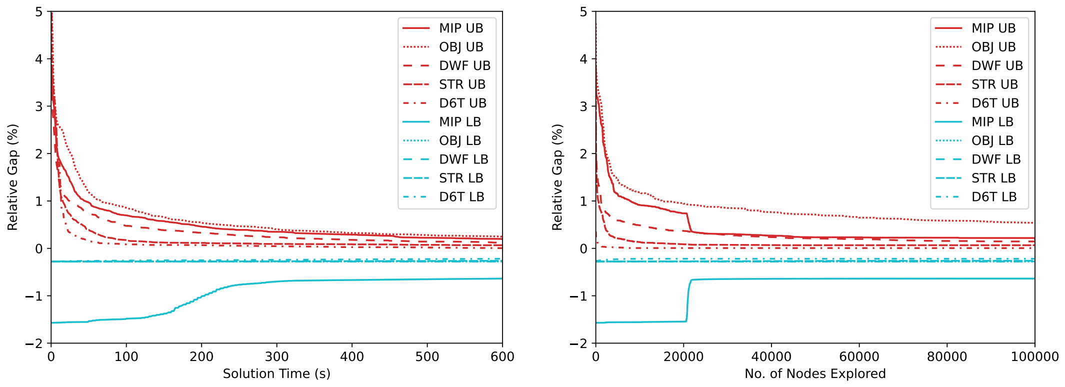
$(\lvert K\rvert, \lvert M\rvert, \lvert N\rvert)$	Number of Instances Solved						Average Solution Time (s)					
	MIP	*OBJ*	*DWB*	*STR*	*D3T*	*D6T*	*MIP*	*OBJ*	*DWB*	*STR*	*D3T*	*D6T*
(10,10,100)	28/30	18/30	**30**/30	**30**/30	**30**/30	**30**/30	$\geq$ 65	$\geq$298	2	**0**	**0**	**0**
(10,10,200)	11/30	10/30	**30**/30	**30**/30	**30**/30	**30**/30	$\geq$424	$\geq$457	6	**1**	2	2
(10,10,300)	7/30	11/30	**30**/30	**30**/30	**30**/30	**30**/30	$\geq$489	$\geq$454	41	**2**	12	7
(25,10,100)	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**0**	1	**0**	**0**	**0**	**0**
(25,10,200)	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	2	25	**0**	**0**	**0**	**0**
(25,10,300)	**30**/30	29/30	**30**/30	**30**/30	**30**/30	**30**/30	6	$\geq$ 40	**0**	**0**	1	3
(25,20,200)	29/30	9/30	**30**/30	**30**/30	**30**/30	**30**/30	$\geq$ 44	$\geq$499	3	**2**	1	1
(25,20,300)	22/30	2/30	29/30	**30**/30	**30**/30	**30**/30	$\geq$224	$\geq$590	$\geq$ 48	5	**3**	7
(25,30,300)	1/30	0/30	0/30	2/30	1/30	**3**/30	$\geq$595	$\geq$600	$\geq$600	$\geq$590	$\geq$591	$\geq$575

Table 3: Comparison of Gurobi Performance on MKAP I_2 Instances (Cont'd)

$(\lvert K\rvert, \lvert M\rvert, \lvert N\rvert)$	Average Optimality Gap (%)						Average Number of Nodes					
	MIP	*OBJ*	*DWB*	*STR*	*D3T*	*D6T*	*MIP*	*OBJ*	*DWB*	*STR*	*D3T*	*D6T*
(10,10,100)	0.04	0.03	**0.00**	**0.00**	**0.00**	**0.00**	$\geq$ 22761	$\geq$1698156	4481	220	2	**1**
(10,10,200)	0.27	0.09	**0.00**	**0.00**	**0.00**	**0.00**	$\geq$2778880	$\geq$3898948	6828	407	33	**3**
(10,10,300)	0.21	0.09	**0.00**	**0.00**	**0.00**	**0.00**	$\geq$2616971	$\geq$2781174	131139	1208	631	**6**
(25,10,100)	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	93	1123	**1**	**1**	**1**	**1**
(25,10,200)	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	3853	20272	**1**	**1**	**1**	**1**
(25,10,300)	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	8488	$\geq$ 25864	**1**	**1**	**1**	**1**
(25,20,200)	0.02	0.08	**0.00**	**0.00**	**0.00**	**0.00**	$\geq$ 27987	$\geq$ 438310	2255	426	3	**2**
(25,20,300)	0.24	0.17	**0.00**	**0.00**	**0.00**	**0.00**	$\geq$ 113353	$\geq$ 410728	$\geq$ 13861	929	**1**	**1**
(25,30,300)	0.84	0.53	0.40	0.33	0.29	**0.22**	$\geq$ 81545	$\geq$ 379999	$\geq$105115	$\geq$35197	$\geq$20036	$\geq$6568

We then conduct experiments on MKAP instances to compare Gurobi's performance on formulations *MIP*, *OBJ*, *STR*, *D3T* and *D6T*. We set a 10-minute timelimit for each formulation. For I_2 instances, we summarize in Tables 2 and 3 some statistics regarding the number of instances solved by different formulations, average solution time (excluding time for the Lagrangian dual problem and cut generation/strengthening), average ending optimality gap and average number of branching nodes needed to solve the instances. We observe that fewer instances are solved to optimality when using formulation *OBJ* than formulation *MIP*, which is consistent with the folklore that simply adding an objective function cut to improve dual bound is mostly ineffective. The ending optimality gap of *OBJ* might be better than that of *MIP* because *OBJ* has a much stronger lower bound, whereas finding good feasible solutions can be hard for *OBJ*. Without any strengthening, formulation *DWB* already significantly outperforms both *MIP* and *OBJ*. For most instances, *D6T* requires the smallest number of branching nodes to solve the problem, while *STR* has the best solution time. This can be explained by the fact that *D6T* has the strongest LP relaxation while *STR* has a more compact formulation. Compared to *D3T* or *D6T*, the processing of each branching node takes less time for STR. For the hardest $(25, 30, 300)$ instances, a stronger polyhedral relaxation becomes more important for closing the optimality gap, where $D6T$ has the best performance. In Figure 6 we plot gap closed as time or number of nodes explored increases for different methods on the hardest $(25, 30, 300)$ instances where the relative gap is computed by comparing with the best feasible solution found by all methods. We observe that although all methods except MIP start from the same root bound (DW bound z_D), with coefficient strengthening and tilting the solver can find good primal solutions more efficiently. We also note that each branching node takes longer time to explore when coefficient strengthening and tilting are applied. Additional tables comparing MIP solver performance on $I_1 \setminus I_2$ instances are presented in Appendix I.

Figure 6: Gap Closed as Time (Left) and Number of Nodes Explored (Right) Increases on a set of MKAP instances



6.4 MIP Experiments on TKP

Finally, in this section we report some experiments conducted on hard TKP instances. In preliminary experiments, we observe that the tilting techniques introduced in Section 5.2 are not very effective for TKP. This is mainly due to the fact that the DWB cuts after disjunctive coefficient strengthening are already defining high-dimensional faces (sometimes facets) of the block polyhedra for TKP. Therefore, we only present results regarding the performance of formulations *MIP*, *OBJ-B*, *DWB-B* and *STR-B* (defined at the beginning of Section 6) for $B \in \{32, 64\}$, where B indicates the number constraints used in each block to define the particular DW reformulation. Note that, unlike MKAP that has a loosely coupled block structure, the blocks in TKP's DW reformulations are overlapping, and in principle the DW relaxation associated with $B = 64$ is stronger than the DW relaxation associated with $B = 32$. It is also worth mentioning that we solve subproblems (14), (17) and (18) to exact optimality rather than the default 0.01% optimality gap tolerance to avoid numerical issues.

Table 4: Comparison of Gurobi Performance on TKP Instances

Subclass	Number of Instances Solved							Average Solution Time (s)						
	MIP	*OBJ* -32	*DWB* -32	*STR* -32	*OBJ* -64	*DWB* -64	*STR* -64	*MIP*	*OBJ* -32	*DWB* -32	*STR* -32	*OBJ* -64	*DWB* -64	*STR* -64
XVIII	5/10	0/10	2/10	5/10	0/10	1/10	**8**/10	≥ 360	≥ 600	≥ 520	≥ 381	≥ 600	≥ 567	≥ 264
XIX	**8**/10	0/10	4/10	7/10	0/10	5/10	**8**/10	≥ 232	≥ 600	≥ 412	≥ 266	≥ 600	≥ 457	≥ 160
XX	6/10	0/10	2/10	**8**/10	0/10	2/10	7/10	≥ 296	≥ 600	≥ 516	≥ 287	≥ 600	≥ 557	≥ 212

Table 5: Comparison of Gurobi Performance on TKP Instances (Cont'd)

Subclass	Average Optimality Gap (%)							Average Number of Nodes						
	MIP	*OBJ* -32	*DWB* -32	*STR* -32	*OBJ* -64	*DWB* -64	*STR* -64	*MIP*	*OBJ*-32	*DWB*-32	*STR*-32	*OBJ*-64	*DWB*-64	*STR*-64
XVIII	0.04	0.17	0.07	0.05	0.06	0.06	**0.02**	$\geq$425495	$\geq$2922669	$\geq$1282107	$\geq$ 913574	$\geq$2649095	$\geq$1865134	$\geq$949243
XIX	**0.02**	0.17	0.06	0.04	0.08	0.03	**0.02**	$\geq$273020	$\geq$2739441	$\geq$ 927412	$\geq$1036383	$\geq$2677712	$\geq$1223015	$\geq$574392
XX	0.04	0.17	0.08	0.04	0.10	0.07	**0.03**	$\geq$315943	$\geq$2449477	$\geq$1054214	$\geq$ 849443	$\geq$2469405	$\geq$1839669	$\geq$807173

In Tables 4 and 5, we compare Gurobi performance of different formulations on TKP test instances, from the hard subclasses XVIII-XX in [25], with a 10-minute timelimit for running each formulation (excluding the time spent on generating the formulation). We want to emphasize here that, unlike for MKAP, time spent on generating some formulations for TKP is not negligible. Time spent on generating these different formulations is presented in Appendix J. We observe that, although formulation *DWB-B* is not as effective as *MIP* for $B \in \{32, 64\}$, *STR*-32 is comparable with *MIP* and *STR*-64 outperforms *MIP* in terms of number of instances solved and average final optimality gap. We also note that the performance of formulations *OBJ-B*, *DWB-B* and *STR-B* relatively increases as B increases, illustrating the benefit of using DWB cuts generated from a tighter DW relaxation. Comparing *DWB-B* and *STR-B*, we observe a significant improvement in performance when coefficient strengthening is applied. This can be attributed to the fact that the TKP instances have relatively few blocks (32 blocks for $B = 32$ and 16 blocks for $B = 64$), in which case the last-iteration DWB cuts without strengthening would likely lead to a formulation with relatively high dual degeneracy.

7 Hybrid Implementation in a Multi-Thread Context

Computationally, using the last iteration DWB cuts (and their strengthened versions) does not always lead to a performance improvement as it (*i*) requires additional computational effort to generate the cuts and (*ii*) increases the size of the formulation. Similar to other MIP techniques, one needs to decide whether or not to use this approach for a given problem instance. Machine learning (ML) techniques have been widely investigated to make algorithmic decisions inside MIP solvers [39, 40].

In the remainder of this section, we present some preliminary experimental results with a simple ML model to predict whether or not one may benefit from DWB cuts on MKAP I_1 instances. We consider a computational environment with 4 threads where one starts with running the MIP solver on *MIP* (using 3 threads) and solves the Lagrangian dual problem (using 1 thread) in parallel until the Lagrangian dual problem is solved. After this first phase, one decides to either allocate all 4 threads to the original formulation *MIP* or to the strengthened formulation *STR*.

To collect labels for the training phase, we use 20% of MKAP I_1 instances as the training set. We first solve the original formulation (using 3 threads) and Lagrangian dual (using 1 thread) in parallel until the Lagrangian dual computation is completed, and collect some simple features for each instance. We then separately try allocating all 4 threads to (i) solving the original formulation *MIP*, and (ii) solving the strengthened formulation *STR*. We label an instance as promising if one of the following holds when both methods are run for 10 minutes:

1. The instance can be solved with *STR* but not with *MIP*, or,

2. Neither method can solve the instance, and *STR* has a smaller optimality gap (at least 0.01% smaller), or,

Table 6: Comparing *MIP*, *STR* and *HYB* on MKAP Test Instances

	MIP	*STR*	*HYB*
Number of Instances Solved	167/242	206/242	208/242
Average Optimality Gap (%)	0.18%	0.05%	0.05%
Average Solution Time (s)	≥ 223	≥ 117	≥ 110

3. Both methods can solve the instance and *STR* reduces the solution time by 10% or more.

Using this labeled data, we train a simple decision stump (depth-1 decision tree) to determine whether or not we should restart the MIP using the formulation *STR*. The features we use are composites of z_L, z_D, z_R, z_{LB} and z_{UB}, where z_{LB} and z_{UB} denote the lower and upper bounds obtained by the solver from solving the original formulation until the Lagrangian dual problem is solved. In our experiments, z_{UB} is always finite as the solver can easily find out that the all-zero solution is feasible for MKAP. The resulting decision stump obtained after training is as follows:

"If $(z_D - z_{\text{LB}})/z_{\text{UB}} > 0.05\%$, then switch to the strengthened formulation *STR*."

Instances that can be solved using the original MIP formulation before we finish computing z_D are dropped from the training set and the test set as there is no algorithmic decision of interest to make in that situation. This leaves us 62 training instances and 242 test instances.

Next, we compare a hybrid implementation with default Gurobi. To further simplify the implementation, we replace the parallelization of the solution of the original formulation and of the Lagrangian dual by a simulated parallelization. Such a simulated hybrid implementation is described below:

Step 1. (a) With 1 thread, solve problem (13) with level method. Store solution time t_D.

(b) With 3 threads, run a MIP solver using original MIP formulation with timelimit t_D.

Step 2. If the MIP is solved to optimality in Step 1, then stop.

Else if $(z_D - z_{\text{LB}})/z_{\text{UB}} > 0.05\%$, then restart the MIP solver with all 4 threads solving formulation *STR* and using the best primal solution (bound) obtained in 1(b).

Else, allocate all 4 threads to the MIP solver and continue with the original formulation.

In Table 6 we summarize results obtained on MKAP by directly solving the original formulation ('*MIP*'), directly generating and solving the strengthened formulation ('*STR*'), and applying our simulated hybrid implementation ('*HYB*'). For each method, we report the number of instances solved within the timelimit of 10 minutes, the ending optimality gap and average time spent on solving the problem including cut generating time for *STR* and *HYB* in seconds. Note that *STR* is likely to outperform *MIP* on the test instances because the DW bound is already significantly stronger than the LP relaxation bound on those instances. We observe that both *STR* and *HYB* outperform *MIP* on the test instances while *HYB* slightly outperforms *MIP* in terms of solution time. This is because *HYB* avoids solving the more expensive *STR* formulation in some cases when the bound provided by *STR* is not significantly stronger than *MIP*. Another interesting observation is that *HYB* solves exactly all instances that can be solved by either *MIP* or *STR*, including two instances that cannot be solved by *STR* but can be solved by *MIP*, demonstrating the value of having the flexibility of switching between different formulations. We believe this hybrid version would be relatively easy to implement by modern MIP solvers.

8 Conclusions

In this paper, we develop practical methods to generate and strengthen cutting planes that one can derive from DW relaxation of MIPs with block structures. Numerical experiments show that adding these cutting planes is effective in cases when the DW bound is strong. We also describe how to incorporate our methodology into a MIP solver when multiple threads are available.

Empirically, we observe that adding too many cuts to the formulation can slow down LPs significantly. A potential fix is to consider cut selection within our cut generation approach. One may also consider adaptive tilting schemes rather than tilting each inequality to the same depth. We also observe that the Lagrangian dual problem often has nonunique optimal solutions. It is therefore interesting to investigate whether using alternative dual solutions can lead to stronger cuts.

Acknowledgements

We are indebted to two anonymous referees whose detailed comments helped us to significantly improve the paper.

References

[1] G. B. Dantzig and P. Wolfe, "Decomposition principle for linear programs," *Operations Research*, vol. 8, no. 1, pp. 101–111, 1960.

[2] M. E. Lübbecke and U. T. Zimmermann, "Engine routing and scheduling at industrial in-plant railroads," *Transportation Science*, vol. 37, no. 2, pp. 183–197, 2003.

[3] J. Rios and K. Ross, "Massively parallel Dantzig-Wolfe decomposition applied to traffic flow scheduling," *Journal of Aerospace Computing, Information, and Communication*, vol. 7, no. 1, pp. 32–45, 2010.

[4] M. F. Anjos, A. Lodi, and M. Tanneau, "A decentralized framework for the optimal coordination of distributed energy resources," *IEEE Transactions on Power Systems*, vol. 34, no. 1, pp. 349–359, 2018.

[5] K. J. Singh, A. B. Philpott, and R. K. Wood, "Dantzig-Wolfe decomposition for solving multistage stochastic capacity-planning problems," *Operations Research*, vol. 57, no. 5, pp. 1271–1286, 2009.

[6] F. Vanderbeck, "On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm," *Operations Research*, vol. 48, no. 1, pp. 111–128, 2000.

[7] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance, "Branch-and-price: Column generation for solving huge integer programs," *Operations Research*, vol. 46, no. 3, pp. 316–329, 1998.

[8] A. Pessoa, R. Sadykov, E. Uchoa, and F. Vanderbeck, "A generic exact solver for vehicle routing and related problems," *Mathematical Programming*, vol. 183, pp. 483–523, Sep 2020.

[9] M. Jünger and S. Thienel, "The ABACUS system for branch-and-cut-and-price algorithms in integer programming and combinatorial optimization," *Software: Practice and Experience*, vol. 30, no. 11, pp. 1325–1352, 2000.

[10] J. Puchinger, P. J. Stuckey, M. Wallace, and S. Brand, "From high-level model to branch-and-price solution in G12," in *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*, pp. 218–232, Springer, 2008.

[11] G. Gamrath and M. E. Lübbecke, "Experiments with a generic Dantzig-Wolfe decomposition for integer programs," in *International Symposium on Experimental Algorithms*, pp. 239–252, Springer, 2010.

[12] M. V. Galati, T. K. Ralphs, and J. Wang, "Computational experience with generic decomposition using the DIP framework," *Proceedings of RAMP*, vol. 2012, 2012.

[13] R. Sadykov and F. Vanderbeck, *BaPCod-a generic branch-and-price code.* PhD thesis, Inria Bordeaux Sud-Ouest, 2021.

[14] A. Lodi, "Mixed integer programming computation," in *50 Years of Integer Programming 1958-2008*, pp. 619–645, Springer, Berlin, Heidelberg, 2010.

[15] T. K. Ralphs, L. Kopman, W. R. Pulleyblank, and L. E. Trotter, "On the capacitated vehicle routing problem," *Mathematical programming*, vol. 94, pp. 343–359, 2003.

[16] T. K. Ralphs and M. V. Galati, "Decomposition in integer linear programming," in *Integer Programming*, pp. 73–126, CRC Press, 2005.

[17] P. Avella, M. Boccia, and I. Vasilyev, "A computational study of exact knapsack separation for the generalized assignment problem," *Computational Optimization and Applications*, vol. 45, pp. 543–555, 2010.

[18] M. Bodur, S. Ahmed, N. Boland, and G. L. Nemhauser, "Decomposition of loosely coupled integer programs: A multiobjective perspective," *Mathematical Programming*, pp. 1–51, 2022.

[19] S. Ahmed, "Two-stage stochastic integer programming: A brief introduction," *Wiley Encyclopedia of Operations Research and Management Science*, pp. 1–10, 2010.

[20] M. Bartlett, A. M. Frisch, Y. Hamadi, I. Miguel, S. A. Tarim, and C. Unsworth, "The temporal knapsack problem and its solution," in *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*, pp. 34–48, Springer, 2005.

[21] E. A. Boyd, "Fenchel cutting planes for integer programs," *Operations Research*, vol. 42, no. 1, pp. 53–64, 1994.

[22] A. Shapiro, D. Dentcheva, and A. Ruszczynski, *Lectures on Stochastic Programming: Modeling and Theory.* SIAM, 2021.

[23] M. Savelsbergh, "A branch-and-price algorithm for the generalized assignment problem," *Operations Research*, vol. 45, no. 6, pp. 831–841, 1997.

[24] A. Ceselli and G. Righini, "A branch-and-price algorithm for the capacitated p-median problem," *Networks: An International Journal*, vol. 45, no. 3, pp. 125–142, 2005.

[25] A. Caprara, F. Furini, and E. Malaguti, "Uncommon Dantzig-Wolfe reformulation for the temporal knapsack problem," *INFORMS Journal on Computing*, vol. 25, no. 3, pp. 560–571, 2013.

[26] M. Conforti, G. Cornuéjols, and G. Zambelli, *Integer Programming*, vol. 271. Springer, 2014.

[27] A. Lodi and A. Tramontani, "Performance variability in mixed-integer programming," in *Theory Driven by Influential Applications*, pp. 1–12, INFORMS, 2013.

[28] G. Gamrath, T. Berthold, and D. Salvagnin, "An exploratory computational analysis of dual degeneracy in mixed-integer programming," *EURO Journal on Computational Optimization*, vol. 8, no. 3-4, pp. 241–261, 2020.

[29] A. J. Goldman and A. W. Tucker, "Theory of linear programming," in *Linear Inequalities and Related Systems*, pp. 53–98, Princeton University Press, 1956.

[30] M. L. Fisher, "The Lagrangian relaxation method for solving integer programming problems," *Management Science*, vol. 27, no. 1, pp. 1–18, 1981.

[31] L. A. Wolsey and G. L. Nemhauser, *Integer and Combinatorial Optimization*, vol. 55. John Wiley & Sons, 1999.

[32] J. E. Kelley, Jr, "The cutting-plane method for solving convex programs," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 4, pp. 703–712, 1960.

[33] C. Lemaréchal, A. Nemirovskii, and Y. Nesterov, "New variants of bundle methods," *Mathematical Programming*, vol. 69, no. 1, pp. 111–147, 1995.

[34] J. F. Shapiro, "Generalized Lagrange multipliers in integer programming," *Operations Research*, vol. 19, no. 1, pp. 68–76, 1971.

[35] K. Andersen and Y. Pochet, "Coefficient strengthening: a tool for reformulating mixed-integer programs," *Mathematical Programming*, vol. 122, no. 1, pp. 121–154, 2010.

[36] D. Espinoza, R. Fukasawa, and M. Goycoolea, "Lifting, tilting and fractional programming revisited," *Operations Research Letters*, vol. 38, no. 6, pp. 559–563, 2010.

[37] V. Chvátal, W. Cook, and D. Espinoza, "Local cuts for mixed-integer programming," *Mathematical Programming Computation*, vol. 5, no. 2, pp. 171–200, 2013.

[38] S. Kataoka and T. Yamada, "Upper and lower bounding procedures for the multiple knapsack assignment problem," *European Journal of Operational Research*, vol. 237, no. 2, pp. 440–447, 2014.

[39] T. Berthold, M. Francobaldi, and G. Hendel, "Learning to use local cuts," *arXiv e-prints*, pp. arXiv–2206, 2022.

[40] P. Bonami, A. Lodi, and G. Zarpellon, "A classifier to decide on the linearization of mixed-integer quadratic problems in CPLEX," *Operations Research*, 2022.

[41] F. Di Pasquale, "A new Lagrangian approach to the multiple knapsack assignment problem," Master's thesis, University of Pisa, 2021.

A Dantzig-Wolfe Decomposition

Algorithm 1 Standard Dantzig-Wolfe Decomposition

```
1: Initialize:
       $\hat{V}^j \leftarrow$ a subset of extreme points of conv$(Q^j)$, $j = 1, 2, \ldots, q$
       $\hat{R}^j \leftarrow$ a subset of extreme rays of conv$(Q^j)$, $j = 1, 2, \ldots, q$
       $t \leftarrow 0$
2: $t \leftarrow t + 1$, solve
```

$$\begin{aligned} z_D^t = \min\ & c^\top x \\ \text{s.t. } & x_{I(j)} = \sum_{v \in \hat{V}^j} v\lambda_v^j + \sum_{r \in \hat{R}^j} r\mu_r^j, && j \in J, && (\pi^j) \\ & Ax \geq b, && && (\beta) \\ & \sum_{v \in \hat{V}^j} \lambda_v^j = 1, && j \in J, && (\theta_j) \\ & \lambda^j \geq 0,\ \mu^j \geq 0, && j \in J \end{aligned} \tag{19}$$

```
   (assume $\{\hat{V}^j\}_{j=1}^q$ and $\{\hat{R}^j\}_{j=1}^q$ are initialized such that (19) is always feasible)
3: let $(\pi^1, \ldots, \pi^q, \theta_1, \ldots, \theta_q)$ denote the values of optimal dual variables for (19)
4: for $j = 1, 2, \ldots, q$ do
5:     solve the following pricing problem:
```

$$D_j(\pi^j) := \min\{(\pi^j)^\top v : v \in \text{conv}(Q^j)\} = \min\{(\pi^j)^\top v : v \in Q^j\}. \tag{20}$$

```
6:     if the pricing problem (20) is bounded then
7:         let $v^j$ denote an optimal solution
8:         $\zeta^j \leftarrow (\pi^j)^\top v^j - \theta_j$
9:         if $\zeta^j < 0$ then
10:            $\hat{V}^j \leftarrow \hat{V}^j \cup \{v^j\}$
11:        end if
12:    else
13:        $\zeta^j \leftarrow -\infty$
14:        let $r^j$ denote an extreme ray of conv$(Q^j)$ with $(\pi^j)^\top r^j < 0$
15:        $\hat{R}^j \leftarrow \hat{R}^j \cup \{r^j\}$
16:    end if
17:    if $\zeta^j \geq 0$ for all $j \in J$ then
18:        return $z_D = z_D^t$
19:    else
20:        go to step 2
21:    end if
22: end for
```

B The Level Method for the Dual Problem

The level method adds on top of the cutting plane method a regularization step that requires solving a convex quadratic program to find a promising candidate multiplier $(\pi, \beta) = (\bar{\pi}, \bar{\beta})$ to explore while staying close to the previous multiplier that is already explored. A pseudocode of (the multi-cut version of) the level method is given in Algorithm 2. For generating the upper bound $\bar{z}$, we give the original formulation to the solver. We pick the second feasible solution $\bar{x}$ found by the solver (often much better than the first feasible solution) and set $\bar{z} = c^\top \bar{x}$. Constraint (21d) is important in early iterations of the algorithm to ensure that problem (13) is bounded, but becomes redundant when UB< $\bar{z}$ in later iterations of the algorithm. Within the level method, LB and UB store the best lower and upper bounds of z_D found by the algorithm. If we set the termination condition to be UB-LB=0, then the algorithm returns the exact DW bound z_D. To avoid some numerical issues, when implementing the level method we terminate the algorithm if difference between LB and UB is within 10^{-6} relative tolerance, or if the solver fails to solve the quadratic

Algorithm 2 The Level Method for Solving (13)

```
1: Initialize:
       $\hat{V}^j \leftarrow \emptyset$, $\hat{R}^j \leftarrow \emptyset$, $j = 1, 2, \ldots, p$
       $\bar{z} \leftarrow$ an upper bound of $z_D$
       LB $\leftarrow -\infty$, UB $\leftarrow \infty$, $t \leftarrow 0$
2: Main Loop: $t \leftarrow t + 1$, solve:
```

$$
\begin{aligned}
\text{UB} \leftarrow \max \ & \sum_{j=1}^{q} \theta_j + b^\top \beta && && \text{(21a)}\\
\text{s.t. } & \theta_j \le v^\top \pi^j, && v \in \hat{V}^j,\ j \in J, && \text{(21b)}\\
& r^\top \pi^j \ge 0, && r \in \hat{R}^j,\ j \in J, && \text{(21c)}\\
& \sum_{j=1}^{q} \theta_j + b^\top \beta \le \bar{z}, && && \text{(21d)}\\
& \sum_{j: i \in I(j)} \pi_i^j + \beta^\top A_i = c_i, && i = 1, \ldots, n, && \text{(21e)}\\
& \beta \ge 0. && && \text{(21f)}
\end{aligned}
$$

```
3: if $LB = -\infty$ then
4:     $(\bar{\pi}, \bar{\beta}) \leftarrow$ optimal value of $(\pi, \beta)$ in (21)
5: else
6:     solve:
```

$$
\begin{aligned}
\min \ & \left\| \left(\pi - \bar{\pi}, \beta - \bar{\beta}\right) \right\|_2^2 \\
\text{s.t. } & \sum_{j=1}^{q} \theta_j + b^\top \beta \ge 0.7 \cdot \text{UB} + 0.3 \cdot \text{LB} \qquad \text{(22)}\\
& \text{(21b)} - \text{(21f)}
\end{aligned}
$$

```
7:     $(\bar{\pi}, \bar{\beta}) \leftarrow$ optimal value of $(\pi, \beta)$ in (22)
8: end if
9: for $j = 1, 2, \ldots, q$ do
10:    solve (14) for $\pi^j = \bar{\pi}^j$
11:    if (14) is bounded then
12:        let $v^j$ denote an optimal solution
13:        $\hat{V}^j \leftarrow \hat{V}^j \cup \{v^j\}$
14:    else
15:        let $r^j$ denote an extreme ray of conv$(Q^j)$ with $(\pi^j)^\top r^j < 0$
16:        $\hat{R}^j \leftarrow \hat{R}^j \cup \{r^j\}$
17:    end if
18: end for
19: LB $\leftarrow \max\left\{\text{LB}, \sum_{j=1}^{q} D_j(\bar{\pi}^j) + b^\top \bar{\beta}\right\}$
20: if UB-LB is small enough then
21:     return LB
22: else
23:     go to step 2
24: end if
```

master problem (22). We observe that the second case rarely happens but can sometimes lead to a Lagrangian dual bound weaker than the natural LP relaxation bound since we do not solve the Lagrangian dual problem to optimality.

C Examples

Example 1. *Consider the following MIP:*

$$\begin{aligned} z^* = \min\ & x_1 + x_2 + 2x_3 + 2x_4 \\ s.t.\ & x_2 + x_4 \geq 3, \\ & 3x_1 + x_2 + 3x_3 + x_4 \geq -12, \\ & 0.5 \leq x_i \leq 2.5,\ x_i \in \mathbb{Z},\ i = 1,2,3,4. \end{aligned}$$

Define $I_1 = \{1,2\}$, $I_2 = \{3,4\}$, $Q^j = \{y \in \mathbb{Z}^2 : 0.5 \leq y_1 \leq 2.5, 0.5 \leq y_2 \leq 2.5\}$ *for* $j \in \{1,2\}$ *and the linking constraints* $Ax \geq b$ *to be*

$$\begin{bmatrix} 0 & 1 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} \geq \begin{bmatrix} 3 \\ 12 \end{bmatrix}.$$

In this example, the MIP has symmetric blocks but asymmetric objective coefficients. The LP relaxation of the problem gives the lower bound 7. Let $\pi(1) = (1, 1/4, 2, 5/4)^\top$, $\beta(1) = (3/4, 0)^\top$, $\pi(2) = (2/11, 8/11, 13/11, 19/11)^\top$, $\beta(2) = (0, 3/11)^\top$. *Note that the dual multipliers* $\big(\pi(\tau), \beta(\tau)\big)_{\tau \in \{1,2\}}$ *satisfy the assumptions in Proposition 6. The best dual bound is given by*

$$\max_{\tau \in \{1,2\}} z\big(\pi(\tau), \beta(\tau)\big) = \max\{27/4, 78/11\} = 78/11.$$

Adding DWB cuts associated with $\pi(1)$ *and adding DWB cuts associated with* $\pi(2)$ *into the LP relaxation of the original formulation yield bounds* 125/16 *and* 149/19*, respectively. These bounds are stronger than the corresponding Lagrangian relaxation bounds* 27/4 *and* 78/11*. On the other hand, the LP*

$$\begin{aligned} \min\ & c^\top x \\ s.t.\ & \big(\pi^j(\tau)\big)^\top x_{I(j)} \geq D_j\big(\pi^j(\tau)\big), \quad j \in J,\ \tau = 1,2, \\ & Ax \geq b \end{aligned}$$

has the optimal objective value equal to 8, which happens to be z^* *in this example.*

Example 2. *Consider the following primal and dual LP pair:*

$$(\textit{Primal}):\ \begin{aligned} \min\ & x_1 - x_2 \\ s.t.\ & x_1 + x_2 \geq 1, \\ & x_2 \geq 1, \\ & -x_2 \geq -1; \end{aligned} \qquad (\textit{Dual}):\ \begin{aligned} \max\ & w_1 + w_2 - w_3 \\ s.t.\ & w_1 = 1, \\ & w_1 + w_2 - w_3 = -1, \\ & w \geq 0. \end{aligned}$$

In this example, the primal LP has a unique optimal solution $x = (0,1)^\top$. *Therefore, the dimension of the optimal face is 0. Also, the problem has a unique dual* basic *solution* $w = (1,0,2)^\top$, *whose 0-norm is 2, which is strictly less than n minus the dimension of the optimal face. However, note that all dual solutions of the form* $w = (1, \lambda, 2+\lambda)^\top$ *with* $\lambda \geq 0$ *are optimal for the dual LP, i.e., the problem has nonunique dual optimal solutions.*

D MKAP and Test Instances

In MKAP, there are a finite number of items, knapsacks and item classes. Each item belongs to exactly one item class. The decision maker has to pack a subset of items into knapsacks, subject to the constraints that only items belonging to the same item class with a total weight no larger than the capacity of the knapsack can be packed into the same knapsack, with the aim of maximizing the total profit of the packed items. Specifically, let N, M and K denote the index sets of items, knapsacks and item classes, respectively. For $j \in N$, let p_j denote the profit of item j and w_j denote the weight of item j. For $i \in M$, let C_i denote the capacity of knapsack i. For $k \in K$, let S_k denote the set of items that belong to item class k, i.e., $\{S_k\}_{k \in K}$ is a partition of N. We use variables $x \in \{0,1\}^{M \times N}$ to represent the packing decisions, where $x_{ij} = 1$ if and only if item j is packed into knapsack i. Variables $y \in \{0,1\}^{M \times K}$ represent the assignment decisions, where $y_{ik} = 1$ if and only if knapsack i is used to pack items from item class k. MKAP can then be formulated as the following integer program (we flip the sign of the objective function to formulate it as a minimization problem):

$$\min \quad -\sum_{i \in M} \sum_{j \in N} p_j x_{ij} \tag{23a}$$

$$\text{s.t.} \quad \sum_{j \in S_k} w_j x_{ij} \le C_i y_{ik}, \qquad i \in M,\ k \in K, \tag{23b}$$

$$\sum_{i \in M} x_{ij} \le 1, \qquad j \in N, \tag{23c}$$

$$\sum_{k \in K} y_{ik} \le 1, \qquad i \in M, \tag{23d}$$

$$x \in \{0,1\}^{M \times N},\ y \in \{0,1\}^{M \times K}. \tag{23e}$$

It has been observed in [41] that applying Lagrangian relaxation to (23) with constraints (23c) and (23d) dualized can lead to a dual bound potentially much stronger than the LP relaxation bound. By equivalence between DW decomposition and Lagrangian relaxation, this observation also applies to DW decomposition. Specifically, we define $|M| \times |K|$ blocks in our DW decomposition, where for each $i \in M$ and $k \in K$ block $Q^{i,k}$ is defined by a knapsack constraint $\sum_{j \in S_k} w_j x_{ij} \le C_i y_{ik}$ together with constraints forcing $(x_{ij})_{j \in S_k}$ and y_{ik} to be binary.

We generate the instances following the scheme of [38] but using different instance sizes. We consider instances with $|K| \in \{2, 5, 10, 25\}$, $|M| \in \{10, 20, 30, 40\}$ and $|N| \in \{50, 100, 200, 300\}$. For each combination $(|K|, |M|, |N|)$ of the parameters, we generate three types (uncorrelated, weakly correlated, strongly correlated) of MKAP instances with 10 instances generated using 10 different random seeds for each type. We refer readers to [38] for a more detailed description of the instance generation procedure. It is worth mentioning that the item classes $\{S_k\}_{k \in K}$ all have equal sizes $|N|/|K|$ for our test instances.

E TKP and Test Instances

TKP is defined by a set of items $N := \{1, 2, \ldots, n\}$. Each item $i \in N$ is associated with a profit p_i, a weight w_i, and a time period $[s_i, t_i)$ during which the item would be active. The decision maker decides to pack a subset of the items into the knapsack, so that at any time point, the total weight of the active items selected is at most C. We use variables $x \in \{0,1\}^N$ to represent the packing decision, where $x_i = 1$ if and only if item $i \in N$ is packed into the knapsack. Let

$S_j := \{i : s_i \le s_j,\ t_i > s_j\}$ be the set of items that are active at time s_j for $j \in N$. Then TKP can be formulated as the following integer program (we flip the sign of the objective function to formulate it as a minimization problem):

$$\min\ -\sum_{i\in N} p_i x_i \tag{24a}$$

$$\text{s.t.}\ \sum_{i\in S_j} w_i x_i \le C, \quad j \in N, \tag{24b}$$

$$x \in \{0,1\}^N. \tag{24c}$$

[25] suggest nonstandard DW reformulations of (24) by partitioning the constraints (24b) into different blocks, leading to the following DW reformulation:

$$\min\ -\sum_{i\in N} p_i x_i \tag{25a}$$

$$\text{s.t.}\ x_{I(k)} \in \operatorname{conv}(Q^k), \quad k \in K, \tag{25b}$$

$$x \in \{0,1\}^N. \tag{25c}$$

where Q^k is defined by B consecutive constraints in the original problem and $I(k)$ contains the indices of the variables that appear in these constraints. Specifically, for $k \in K := \{1, \dots, \lceil n/B \rceil\}$, we have $I(k) = \bigcup_{j\in N_k} S_j$ where $N_k = \big\{(k-1)B+1, (k-1)B+2, \dots, \min\{kB, n\}\big\}$, and $Q^k = \{x_{I(k)} \in \{0,1\}^{I(k)} : \sum_{i\in S_j} w_i x_i \le C, j \in N_k\}$.

As anticipated, according to empirical results in [25], the strength of the associated DW bound tends to increase as B grows. In our numerical experiments, we consider two DW reformulations with group sizes $B = 32$ and $B = 64$, respectively. Since current commercial solvers (Gurobi in particular) can easily solve the original formulation for most test instances in [25], we consider only the 30 relatively hard instances from groups XVIII-XX in [25].

F Dual Degeneracy of Different Formulations for MKAP

In Table 7, we report the relative normalized degeneracy level of the optimal dual solution computed by Gurobi for LP relaxations of different formulations for MKAP. The relative degeneracy level is calculated as the degeneracy level $n - \|w^*\|_0$ divided by the dimension n, i.e., $1 - \|w^*\|_0/n$. Bold rows correspond to I_2 instances. Due to numerical tolerances, the optimal values w_k^* of some dual variables can sometimes be very close to but not equal to zero, especially the ones associated with bound constraints. We treat w_k^* as 0 if $|w_k^*| \le 10^{-8}$ for all entries w_k^* of w^* when computing $\|w^*\|_0$. Although the strengthening techniques in Section 5 do not guarantee a monotonic decrease in degeneracy level, it is clear from Table 7 that formulations with stronger strengthening tend to have less degenerate optimal dual solutions. We observe that the degeneracy level of *DWB* is already comparable to *MIP* on I_2 instances but is higher than *MIP* on $I_1 \setminus I_2$ instances. With coefficient strengthening, *STR* has lower average dual degeneracy than *MIP* in almost all cases except on the instance class (5,40,50). The dual degeneracy continues to decrease with further tilting. Finally, it is worth noting that, as expected, the relative degeneracy level of *OBJ* is extremely high, close to 100%.

Table 7: Relative Degeneracy Levels of LP Optimal Dual Solutions for Different MKAP Formulations

$(\lvert K\rvert, \lvert M\rvert, \lvert N\rvert)$	$(1 - \lVert w^*\rVert_0/n) \times 100\%$									
	MIP	*OBJ*	*DWB*	*STR*	*D1T*	*D2T*	*D3T*	*D4T*	*D5T*	*D6T*
(2,20, 50)	56.27%	99.90%	78.30%	41.85%	36.37%	30.94%	25.77%	23.87%	21.86%	21.88%
(2,30, 50)	57.75%	99.94%	70.21%	52.25%	46.32%	41.81%	39.24%	39.52%	39.89%	39.26%
(2,40, 50)	58.06%	99.95%	71.04%	55.17%	51.67%	52.29%	51.82%	50.46%	50.46%	50.46%
(5,10, 50)	50.84%	99.82%	77.44%	42.29%	32.33%	22.32%	14.84%	11.09%	11.25%	11.04%
(5,20, 50)	53.20%	99.91%	68.12%	36.42%	28.41%	22.50%	20.10%	19.95%	19.04%	19.42%
(5,30, 50)	54.60%	99.94%	65.54%	45.31%	34.81%	34.89%	35.45%	36.05%	36.11%	36.11%
(5,40, 50)	54.90%	99.95%	70.72%	55.61%	51.21%	50.62%	50.61%	50.61%	50.61%	50.61%
(10,10, 50)	46.60%	99.83%	57.63%	26.99%	11.24%	3.01%	3.35%	3.46%	3.46%	3.46%
(10,10,100)	**50.65%**	**99.91%**	**56.98%**	**35.05%**	**22.84%**	**7.77%**	**1.36%**	**1.14%**	**1.49%**	**1.54%**
(10,10,200)	**53.30%**	**99.95%**	**53.77%**	**38.02%**	**26.06%**	**17.45%**	**7.19%**	**1.05%**	**0.26%**	**0.28%**
(10,10,300)	**54.23%**	**99.97%**	**53.50%**	**39.77%**	**26.75%**	**20.75%**	**17.99%**	**7.47%**	**1.43%**	**0.61%**
(10,20, 50)	48.77%	99.92%	57.55%	24.16%	16.96%	11.23%	11.46%	11.36%	11.36%	11.36%
(10,30, 50)	50.05%	99.94%	57.02%	38.97%	33.06%	31.63%	31.42%	31.42%	31.42%	31.42%
(10,40, 50)	50.32%	99.96%	62.98%	45.03%	44.05%	43.36%	43.36%	43.36%	43.36%	43.36%
(10,40,100)	54.98%	99.98%	80.55%	45.61%	35.69%	27.61%	25.77%	24.92%	25.41%	25.50%
(25,10, 50)	37.28%	99.87%	26.68%	11.18%	5.86%	5.86%	5.86%	5.86%	5.86%	5.86%
(25,10,100)	**44.57%**	**99.92%**	**40.47%**	**30.85%**	**14.51%**	**4.40%**	**3.70%**	**3.70%**	**3.70%**	**3.70%**
(25,10,200)	**49.75%**	**99.96%**	**47.23%**	**37.99%**	**29.72%**	**13.19%**	**3.71%**	**1.59%**	**1.36%**	**1.68%**
(25,10,300)	**51.73%**	**99.97%**	**56.84%**	**48.89%**	**34.16%**	**24.02%**	**8.05%**	**1.77%**	**1.03%**	**1.17%**
(25,20, 50)	39.01%	99.93%	34.14%	11.26%	6.44%	6.44%	6.44%	6.44%	6.44%	6.44%
(25,20,100)	47.02%	99.96%	52.18%	26.21%	10.38%	3.34%	3.57%	3.57%	3.57%	3.57%
(25,20,200)	**52.44%**	**99.98%**	**62.38%**	**38.30%**	**24.36%**	**7.41%**	**0.93%**	**0.91%**	**1.06%**	**1.25%**
(25,20,300)	**54.69%**	**99.98%**	**62.72%**	**42.19%**	**30.72%**	**22.26%**	**4.78%**	**0.56%**	**0.53%**	**0.62%**
(25,30, 50)	40.04%	99.96%	41.02%	23.72%	22.94%	22.94%	22.94%	22.94%	22.94%	22.94%
(25,30,100)	47.84%	99.97%	64.22%	32.04%	16.22%	7.41%	7.57%	7.57%	7.57%	7.57%
(25,30,200)	53.39%	99.99%	73.83%	43.35%	30.70%	15.59%	6.42%	5.51%	5.67%	5.89%
(25,30,300)	**55.68%**	**99.99%**	**78.99%**	**48.56%**	**41.44%**	**26.72%**	**8.59%**	**4.23%**	**3.58%**	**3.27%**
(25,40, 50)	40.26%	99.97%	50.78%	39.69%	39.52%	39.52%	39.52%	39.52%	39.52%	39.52%
(25,40,100)	48.39%	99.98%	61.63%	29.76%	20.09%	17.55%	17.17%	17.17%	17.17%	17.17%

G Time Spent on Obtaining Different MKAP Formulations

Since both *OBJ* and *DWB* can be obtained directly after solving the Lagrangian dual problem, we only report in Table 8 the total time spent on applying strengthening and different levels of tilting for *STR* and *DdT* with $d \in \{1, \ldots, 6\}$. Bold rows correspond to I_2 instances and other rows correspond to $I_1 \setminus I_2$ MKAP instances. We note that, given a dual optimal solution, it is extremely efficient to generate the *STR* formulation for MKAP while the time spent on generating *DdT* grows exponentially as d increases.

H Number of Cutting Planes Generated Using Different Formulations

In Table 9, we report the number of cutting planes generated by Gurobi within 10 minutes for formulations *MIP*, *OBJ*, *DWB*, *STR*, *D3T* and *D6T* averaged over the (25, 20, 300) MKAP instances.

Table 8: Comparison of Time Spent on Obtaining Different Formulations for MKAP

$(\lvert K\rvert,\lvert M\rvert,\lvert N\rvert)$	Generation Time Excluding Lagr. Dual Time (s)						
	STR	*D1T*	*D2T*	*D3T*	*D4T*	*D5T*	*D6T*
(2,20, 50)	0.1	0.3	0.5	1.1	2.1	3.7	6.0
(2,30, 50)	0.1	0.2	0.3	0.6	0.9	1.0	1.2
(2,40, 50)	0.1	0.2	0.1	0.2	0.2	0.2	0.2
(5,10, 50)	0.0	0.1	0.4	0.8	1.6	2.6	3.6
(5,20, 50)	0.1	0.2	0.5	0.9	1.3	1.5	1.6
(5,30, 50)	0.1	0.2	0.3	0.5	0.5	0.5	0.5
(5,40, 50)	0.1	0.1	0.1	0.1	0.1	0.1	0.1
(10,10, 50)	0.0	0.2	0.4	0.8	1.1	1.2	1.2
(10,10,100)	**0.1**	**0.3**	**1.0**	**2.2**	**4.5**	**8.7**	**15.7**
(10,10,200)	**0.3**	**0.9**	**2.2**	**5.0**	**10.3**	**20.9**	**42.4**
(10,10,300)	**0.9**	**2.9**	**7.0**	**15.2**	**30.5**	**60.0**	**118.7**
(10,20, 50)	0.1	0.2	0.4	0.6	0.7	0.7	0.7
(10,30, 50)	0.1	0.2	0.3	0.3	0.3	0.3	0.3
(10,40, 50)	0.1	0.1	0.1	0.1	0.1	0.1	0.1
(10,40,100)	0.3	0.9	2.0	3.8	5.5	6.6	6.9
(25,10, 50)	0.0	0.2	0.3	0.3	0.3	0.3	0.3
(25,10,100)	**0.0**	**0.4**	**1.1**	**2.1**	**2.8**	**2.8**	**2.8**
(25,10,200)	**0.1**	**0.7**	**2.1**	**4.9**	**10.3**	**20.2**	**38.7**
(25,10,300)	**0.1**	**0.9**	**3.0**	**7.2**	**15.5**	**31.7**	**63.5**
(25,20, 50)	0.1	0.2	0.2	0.2	0.2	0.2	0.2
(25,20,100)	0.1	0.6	1.6	2.6	3.1	3.1	3.1
(25,20,200)	**0.2**	**1.3**	**4.0**	**9.0**	**17.7**	**32.1**	**53.9**
(25,20,300)	**0.4**	**2.0**	**5.8**	**13.5**	**28.1**	**56.1**	**109.5**
(25,30, 50)	0.1	0.2	0.2	0.2	0.2	0.2	0.2
(25,30,100)	0.2	0.8	1.7	2.4	2.5	2.5	2.5
(25,30,200)	0.4	1.9	5.4	11.8	21.9	36.1	51.1
(25,30,300)	**0.8**	**3.1**	**8.3**	**19.3**	**39.7**	**77.9**	**148.1**
(25,40, 50)	0.1	0.1	0.0	0.0	0.0	0.0	0.0
(25,40,100)	0.2	0.8	1.5	1.9	2.0	2.0	2.0

Table 9: Average Number of Cutting Planes Generated by Gurobi

	MIP	*OBJ*	*DWB*	*STR*	*D3T*	*D6T*
Gomory	132.1	41.5	10.4	17.9	0.4	0.0
Lift-and-project	5.3	1.7	0.6	0.6	0.0	0.0
Cover	616.1	176.5	121.5	173.3	16.3	0.0
Clique	362.3	248.5	148.3	266.4	108.3	0.0
MIR	90.2	24.6	9.7	13.2	3.1	0.0
StrongCG	94.3	49.8	12.7	21.9	2.4	0.0
Flow cover	228.8	61.6	18.5	3.1	0.0	0.0
Zero half	34.9	13.9	2.0	0.4	0.0	0.0
RLT	19.7	0.8	1.4	0.7	0.0	0.0
Relax-and-lift	10.0	0.0	0.1	0.0	0.0	0.0

I Results for $I_1 \setminus I_2$ MKAP instances

In Tables 10 and 11, we report computational results for $I_1 \setminus I_2$ MKAP instances. We observe that the original *MIP* formulation is very competitive on these instances, whose performance is often better than *DWB* and even *STR*. This can be explained by the fact that although the natural LP relaxation is weak (r_L is large), Gurobi can close much of the gap by presolve and cutting planes at the root node (r_R is small).

Table 10: Comparison of Gurobi Performance on $I_1 \setminus I_2$ MKAP Instances

$(\lvert K\rvert, \lvert M\rvert, \lvert N\rvert)$	Number of Solved						Average Solution Time (s)					
	MIP	*OBJ*	*DWB*	*STR*	*D3T*	*D6T*	*MIP*	*OBJ*	*DWB*	*STR*	*D3T*	*D6T*
(2,20, 50)	**30**/30	23/30	**30**/30	**30**/30	**30**/30	**30**/30	**2**	$\geq$202	7	5	4	4
(2,30, 50)	**30**/30	29/30	**30**/30	**30**/30	**30**/30	**30**/30	**0**	$\geq$ 25	**0**	**0**	**0**	**0**
(2,40, 50)	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**0**	**0**	**0**	**0**	**0**	**0**
(5,10, 50)	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	4	62	7	4	2	**1**
(5,20, 50)	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	1	42	4	2	1	**0**
(5,30, 50)	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**0**	19	**0**	**0**	**0**	**0**
(5,40, 50)	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**0**	**0**	**0**	**0**	**0**	**0**
(10,10, 50)	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**0**	1	**0**	**0**	**0**	**0**
(10,20, 50)	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**0**	16	**0**	**0**	**0**	**0**
(10,30, 50)	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**0**	1	**0**	**0**	**0**	**0**
(10,40, 50)	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**0**	**0**	**0**	**0**	**0**	**0**
(10,40,100)	**30**/30	1/30	29/30	**30**/30	**30**/30	**30**/30	40	$\geq$600	$\geq$103	72	15	**12**
(25,10, 50)	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**0**	**0**	**0**	**0**	**0**	**0**
(25,20, 50)	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**0**	**0**	**0**	**0**	**0**	**0**
(25,20,100)	**30**/30	20/30	**30**/30	**30**/30	**30**/30	**30**/30	1	$\geq$204	**0**	**0**	**0**	**0**
(25,30, 50)	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**0**	**0**	**0**	**0**	**0**	**0**
(25,30,100)	**30**/30	10/30	**30**/30	**30**/30	**30**/30	**30**/30	5	$\geq$432	1	1	**0**	**0**
(25,30,200)	16/30	1/30	10/30	15/30	23/30	**26**/30	$\geq$377	$\geq$600	$\geq$475	$\geq$402	$\geq$255	$\geq$143
(25,40, 50)	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**30**/30	**0**	**0**	**0**	**0**	**0**	**0**
(25,40,100)	**30**/30	9/30	**30**/30	**30**/30	**30**/30	**30**/30	2	$\geq$464	1	1	**0**	**0**

At each row, the averages of 30 instances are reported.

Table 11: Comparison of Gurobi Performance on $I_1 \setminus I_2$ MKAP Instances (Cont'd)

$(\lvert K\rvert, \lvert M\rvert, \lvert N\rvert)$	Average Optimality Gap (%)						Average Number of Nodes					
	MIP	*OBJ*	*DWB*	*STR*	*D3T*	*D6T*	*MIP*	*OBJ*	*DWB*	*STR*	*D3T*	*D6T*
(2,20, 50)	**0.00**	0.06	**0.00**	**0.00**	**0.00**	**0.00**	**3439**	$\geq$755510	28880	16808	16826	7963
(2,30, 50)	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**89**	$\geq$177782	341	796	188	131
(2,40, 50)	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**1**	**1**	**1**	**1**	**1**	**1**
(5,10, 50)	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	12677	669382	62160	23721	5498	**1509**
(5,20, 50)	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	1865	334795	20703	9992	929	**614**
(5,30, 50)	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	275	151202	161	61	**17**	18
(5,40, 50)	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**1**	**1**	**1**	**1**	**1**	**1**
(10,10, 50)	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	1077	3227	39	51	**1**	**1**
(10,20, 50)	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	344	46614	803	379	**14**	**14**
(10,30, 50)	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	131	3045	372	129	**1**	**1**
(10,40, 50)	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**1**	**1**	**1**	**1**	**1**	**1**
(10,40,100)	**0.00**	0.14	0.01	0.01	**0.00**	**0.00**	23261	$\geq$302450	$\geq$137206	82878	11574	**9120**
(25,10, 50)	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	2	229	**1**	**1**	**1**	**1**
(25,20, 50)	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**1**	212	**1**	**1**	**1**	**1**
(25,20,100)	**0.00**	0.03	**0.00**	**0.00**	**0.00**	**0.00**	279	$\geq$202331	**1**	**1**	**1**	**1**
(25,30, 50)	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**1**	2	**1**	**1**	**1**	**1**
(25,30,100)	**0.00**	0.15	**0.00**	**0.00**	**0.00**	**0.00**	9866	$\geq$185765	1903	1121	**1**	**1**
(25,30,200)	0.12	0.44	0.22	0.15	0.07	**0.04**	$\geq$45681	$\geq$341784	$\geq$ 67369	$\geq$48540	$\geq$20550	$\geq$14553
(25,40, 50)	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**1**	28	**1**	**1**	**1**	**1**
(25,40,100)	**0.00**	0.08	**0.00**	**0.00**	**0.00**	**0.00**	450	$\geq$257189	1520	2483	**47**	**47**

At each row, the averages of 30 instances are reported.

J Time Spent on Obtaining Different TKP Formulations

In Table 12, we report time spent on generating formulations *OBJ*/*DWB*-*B* and *STR*-*B* for $B \in \{32, 64\}$. Unlike Table 8 for MKAP, time reported in Table 12 includes time spent on solving the Lagrangian dual problem.

Table 12: Comparison of Time Spent on Obtaining Different Formulations for TKP

Subclass	Generation Time (s)			
	OBJ/DWB-32	*STR*-32	*OBJ/DWB*-64	*STR*-64
XVIII	78	204	83	772
XIX	80	218	82	643
XX	106	267	101	793