Thinner is not Always Better: Cascade Knapsack Problems

Bala Krishnamoorthy

Washington State University, Vancouver, WA, USA

bkrishna@math.wsu.edu

Abstract

In the context of branch-and-bound (B&B) for integer programming (IP) problems, a direction along which the polyhedron of the IP has minimum width is termed a *thin* direction. We demonstrate that a thin direction need not always be a good direction to branch on for solving the problem efficiently. Further, the *integer width*, which is the number of B&B nodes created when branching on the direction, may also not be an accurate indicator of good branching directions.

Keywords: Branch-and-bound; hyperplane branching; thin direction; column basis reduction.

1 Introduction

Deciding which variables, or combinations of variables, to branch on is perhaps the most critical step that determines the performance of any branch-and-bound (B&B) algorithm to solve integer programs (IPs); see [5], for instance. Most state of the art IP and mixed integer programming (MIP) solvers use branching on individual variables (rather than their combinations). Variable selection done by the default B&B branching on individual variables can be viewed as a special case of selecting hyperplanes in the more general B&B that branches on linear combinations of variables. Branching on hyperplanes is one of the key steps in Lenstra's seminal algorithm for solving integer programs (IPs) in fixed dimensions [15]. This theoretical algorithm finds a sequence of "thin" directions, i.e., directions along which the width of the polyhedron of the IP is "small". Branching on these thin directions in sequential order will solve the IP relatively quickly. The thin directions are identified using lattice basis reduction.

When individual binary variables are present, the default choice to branch on them (as opposed to finding good combinations of the variables) is in line with the idea of selecting thin directions. Recall that a binary variable x_j satisfies $0 \leq x_j \leq 1$, and hence the width of the polyhedron along the direction of x_j is not larger than 1. Motivated in part by the use of thin directions in Lenstra's algorithm, the author and Pataki previously created and analyzed [13] a general class of inequality-constrained knapsack feasibility problems called decomposable knapsack problems (DKPs), whose coefficients have the form $\mathbf{a} = \mathbf{p}M + \mathbf{r}$ for a suitably large positive number M. We showed that DKPs are difficult for ordinary branch-and-bound, i.e., B&B branching on individual variables (an independent analysis of the difficulty of B&B on integer knapsacks was presented separately [12]). At the same time, $\mathbf{p}$ indicates a thin direction for the DKPs, and hence the problems are easy when one branches on the *backbone* hyperplane defined by $\mathbf{px}$. The DKPs subsume several known families of hard IPs, including those proposed by Jeroslow [9], Todd as well as Avis (as attributed to by Chvátal) [6], and by Aardal and Lenstra [3]. All these instances suggest that branching on thin directions—as represented by individual binary variables in typical instances, or by the backbone constraint given by $\mathbf{px}$ in the case of DKPs—is typically a good choice for B&B algorithms.

Even when such thin directions are present in an IP instance, identifying them might not be straightforward. We had previously proposed [13] a simple preconditioning method termed column basis reduction (CBR) that provides reformulations of general IPs. For the DKPs, we proved that the thin direction $\mathbf{px}$ is identified as the last variable in the preconditioned IP given by CBR. Branching on the last variable solves the problem in one step.

Our Contribution: We demonstrate that branching on thin directions need not *always* lead to B&B algorithms running efficiently. On the other hand, branching on certain directions that are *not* thin might lead to quick convergence of B&B algorithms on certain instances. To this end, we create a generalization of the DKPs called *cascade knapsack problems* (CKPs), which are inequality-constrained binary knapsack feasibility problems whose coefficients have the form $\mathbf{a} = \mathbf{p}_1 M_1 + \mathbf{p}_2 M_2 + \mathbf{r}$. We show that the width of CKP polytope along $\mathbf{p}_1$ is larger than 1, while the width along unit directions is equal to 1. As such, $\mathbf{p}_1$ is not a thin direction. Similarly, $\mathbf{p}_2$ is also not a thin direction. Nonetheless, branching on the hyperplane defined by $\mathbf{p}_1\mathbf{x}$ followed by branching on the hyperplane defined by $\mathbf{p}_2\mathbf{x}$ solves the problem quickly, while the original CKP is hard for ordinary B&B. A similar behavior is observed even when we consider *integer width*, which is the number of nodes created when branching on a hyperplane, in place of width. We also show that this behavior extends to higher order CKPs, e.g., with $\mathbf{a} = \mathbf{p}_1 M_1 + \mathbf{p}_2 M_2 + \mathbf{p}_3 M_3 + \mathbf{r}$.

We demonstrate that (CBR is effective in solving the CKPs quickly in practice. Extending the previous analysis for DKPs [13], we argue that branching on the collection of good directions defined by $\mathbf{p}_1\mathbf{x}, \mathbf{p}_2\mathbf{x}, \mathbf{p}_3\mathbf{x}, \ldots$ is captured by branching on the last few variables in the preconditioned IP give by CBR.

Related Work: Following Lenstra's algorithm [15], Kannan [10], and Lovász and Scarf [17] developed similar algorithms for solving IPs in fixed dimension. Cook et al. [7] reported a practical implementation

of the Lovász and Scarf algorithm [17], and obtained reductions in the number of B&B nodes for solving certain network design problems. But finding the branching direction(s) at each node was quite expensive.

Mahajan and Ralphs [18] showed that it is NP-hard to find branching directions that are optimal with respect to width. On the other hand, Aardal and co-workers studied a basis reduction-based reformulation technique for equality constrained IPs, which uses the idea of branching on *good* hyperplanes [1, 2, 3]. Specifically, Aardal and Lenstra [3] studied a class of equality-constrained integer knapsack problems whose reformulations have a specific thin direction, which is also identified by a reformulation technique. Our preconditioning method CBR [13] applies to more general, i.e., not necessarily equality-constrained, IPs. The reformulation technique of Aardal et al. is subsumed by the more general CBR.

The use of branching on good hyperplanes on more general IPs was demonstrated by Aardal et al. [1], who used their basis reduction-based reformulation technique to solve otherwise hard-to-solve marketshare problems, which are multiple equality-constrained binary IPs [8, 11]. Louveaux and Wolsey [16] extended these results to a more general class of IPs with some special structure. In related work, Mehrotra and Li [19] proposed a general framework for identifying branching hyperplanes for mixed IPs, which also benefits from basis reduction.

Pataki, Tural, and Wong [20] studied studied the efficacy of branch-and-bound on CBR-type reformulations of general IPs. In particular, they proved an upper bound on the width of the polyhedron of the reformulated IP along the last unit vector. This bound implies that as the size of coefficients in the constraint matrix of the original IP increases, the reformulation of almost all instances generated from a standard distribution are solved at the root node. This result follows from the fact that the last unit vector in the reformulation is equivalent to a direction along which the width of the original polyhedron is small, and hence branching on the last unit vector solves the problems easily.

2 Width, Integer Width, and Branching Directions: Examples

Definition 2.1. *Given a polyhedron Q, and an integer vector $\mathbf{c}$, the* width *and the* integer width *of Q in the direction of $\mathbf{c}$ are*

$$\text{width}(\mathbf{c}, Q) = \ \max\{\mathbf{c}\mathbf{x} \,|\, \mathbf{x} \in Q\} - \min\{\mathbf{c}\mathbf{x} \,|\, \mathbf{x} \in Q\,\},$$
$$\text{iwidth}(\mathbf{c}, Q) = \ \lfloor \max\{\mathbf{c}\mathbf{x} \,|\, \mathbf{x} \in Q\} \rfloor - \lceil \min\{\mathbf{c}\mathbf{x} \,|\, \mathbf{x} \in Q\} \rceil + 1.$$

Further, an integer direction $\mathbf{c}^*$ is termed a thin *direction of Q if $\mathbf{c}^* \in \ \text{argmin}_{\mathbf{c}} \ \text{width}(\mathbf{c}, Q)$. The quantity* $\text{iwidth}(\mathbf{c}, Q)$ *is the number of nodes generated by branch-and-bound when branching on the constraint given by $\mathbf{c}\mathbf{x}$.*

We point out that integer width is *not* given by requiring the optima in the definition of width be attained by integer vectors. Consider the simple 2D example where $Q = \{\mathbf{x} \in \mathbb{R}^2 \,|\, 0 \leq x_1, x_2 \leq 1\}$. For $\mathbf{c} = \begin{bmatrix} 1 & 0 \end{bmatrix}$, we get that $\text{width}(\mathbf{c}, Q) = 1$, with the maximum and minimum being attained by $\mathbf{x} = \begin{bmatrix} 1 & 0 \end{bmatrix}$ and the zero vector, respectively. But $\text{iwidth}(\mathbf{c}, Q) = 2$, since we would consider two branches when branching on the direction defined by x_1, corresponding to $x_1 = 1$ and $x_1 = 0$.

We present several instances of integer programming which demonstrate that thin directions might not always be the best choices for branching. On the contrary, certain specific directions along which the width, or even the integer width, of the polytope is *larger* than the minimum (integer) width could help solve the problem quickly using B&B.

We consider classes of inequality-constrained binary knapsack feasibility problems of the form

$$\{\mathbf{x} \in \mathbb{Z}^n \,|\, \beta' \leq \mathbf{a}\mathbf{x} \leq \beta, \ \mathbf{0} \leq \mathbf{x} \leq \mathbf{1}\},$$

that have *no* integer feasible solutions, and our goal is to *prove* its integer infeasibility using B&B. For an integer program labeled (IP) and an integer vector $\mathbf{c}$, we denote by $\text{width}(\mathbf{c}, \text{IP})$ the width of the LP-relaxation of (IP) in the direction $\mathbf{c}$, and similarly denote $\text{iwidth}(\mathbf{c}, \text{IP})$.

Our use of the term knapsack problem is a generalization of how it is referred to in most literature (see, e.g., [22, Section 16.6]), where the single constraint has the form $\mathbf{ax} \leq \beta$ (or $\mathbf{ax} = \beta$ in the equality version). It happens to be the case that we have $\beta' = \beta$ in all the following examples (KP1,KP2,KP3,KP4). But our construction (in Section 3) is more general, allowing $\beta' < \beta$. Indeed, several of the larger instances we generated (see Section 4) do have $\beta' < \beta$.

Example 1. Consider the following knapsack feasibility problem with $n = 12$ binary variables:

$$75\,x_1 + 86\,x_2 + 97\,x_3 + 105\,x_4 + 142\,x_5 + 153\,x_6 +$$
$$161\,x_7 + 172\,x_8 + 209\,x_9 + 217\,x_{10} + 228\,x_{11} + 239\,x_{12} = 1023 \qquad \text{(KP1)}$$
$$0 \leq x_j \leq 1, \ x_j \in \mathbb{Z}, \ j = 1, \ldots, 12.$$

There are no integer feasible solutions, and CPLEX 12.6.3.0 applying branch-and-cut (B&C) on original variables with no objective function takes 484 B&B nodes to prove integer infeasibility of this instance (*all* computations presented in this document were done on a Intel PC with 8 cores and a 2.33 GHz CPU using CPLEX 12.6.3.0 as the MIP solver). The knapsack coefficient vector $\mathbf{a} = (75, 86, \ldots, 239)$ is a thin direction (trivially), as $\mathrm{width}(\mathbf{a}, \mathrm{KP1}) = 0$. It can also be checked that $\mathrm{width}(\mathbf{e}_j, \mathrm{KP1}) = 1$ for all unit vectors $\mathbf{e}_j$, as the maximum and minimum of x_j over (KP1) are 1 and 0, respectively, for all $j \in \{1, \ldots, 12\}$.

The knapsack coefficients have the form $\mathbf{a} = \mathbf{p}_1 M_1 + \mathbf{p}_2 M_2 + \mathbf{r}$, where

$$
\begin{aligned}
\mathbf{p}_1 &= (\ 1, \ 1, \ 1, \ 1, \ 2, \ 2, \ 2, \ 2, \ 3, \ 3, \ 3, \ 3), \\
\mathbf{p}_2 &= (\ 1, \ 2, \ 3, \ 4, \ 1, \ 2, \ 3, \ 4, \ 1, \ 2, \ 3, \ 4), \quad \text{and} \\
\mathbf{r} &= (-1, \ 0, \ 1, \ -1, \ 0, \ 1, \ -1, \ 0, \ 1, \ -1, \ 0, \ 1).
\end{aligned}
\qquad (1)
$$

with $M_1 = 66$ and $M_2 = 10$. We observe that $\mathrm{width}(\mathbf{p}_1, \mathrm{KP1}) = 13.99 - 12.13 = 1.86$, which is larger than the width along any $\mathbf{e}_j$. Still, we branch on the direction of $\mathbf{p}_1$ by adding the constraint $\mathbf{p}_1\mathbf{x} = 13$. We now get the maximum and minimum of $\mathbf{p}_2\mathbf{x}$ for this branch as 16.90 and 16.10. Thus, branching on the hyperplanes defined by $\mathbf{p}_1\mathbf{x}$ and $\mathbf{p}_2\mathbf{x}$ in that order proves the integer infeasibility of the instance easily, in only two B&B nodes. We also point out that $\mathrm{width}(\mathbf{p}_2, \mathrm{KP1}) = 22.16 - 9.99 = 12.17$, and hence does not define a helpful direction to branch on.

One could argue that while $\mathrm{width}(\mathbf{p}_1, \mathrm{KP1})$ is indeed larger than the width along any individual variable, $\mathrm{iwidth}(\mathbf{p}_1, \mathrm{KP1}) = 1$ is in fact smaller than $\mathrm{iwidth}(\mathbf{e}_j, \mathrm{KP1}) = 1 - 0 + 1 = 2$. But the next two examples illustrate that small integer widths might also not indicate good branching directions.

Example 2. Consider the following knapsack feasibility problem with $n = 12$ binary variables:

$$71\,x_1 + 82\,x_2 + 93\,x_3 + 101\,x_4 + 134\,x_5 + 145\,x_6 +$$
$$153\,x_7 + 164\,x_8 + 197\,x_9 + 205\,x_{10} + 216\,x_{11} + 227\,x_{12} = 981 \qquad \text{(KP2)}$$
$$0 \leq x_j \leq 1, \ x_j \in \mathbb{Z}, \ j = 1, \ldots, 12.$$

There are no integer feasible solutions, and CPLEX 12.6.3.0 applying B&C on original variables with no objective function takes 462 B&B nodes to prove integer infeasibility. Similar to (KP1), we get $\mathrm{width}(\mathbf{a}, \mathrm{KP2}) = 0$ and $\mathrm{width}(\mathbf{e}_j, \mathrm{KP2}) = 1$.

The knapsack coefficient vector $\mathbf{a} = (71, 82, \ldots, 227)$ has the same structure as in (KP1), using the same vectors $\mathbf{p}_1, \mathbf{p}_2, \mathbf{r}$ in Equation (1), except with $M_1 = 62$. We observe that $\mathrm{width}(\mathbf{p}_1, \mathrm{KP2}) = 14.17 - 12.22 = 1.95$ and $\mathrm{iwidth}(\mathbf{p}_1, \mathrm{KP2}) = 2$, which is equal to the integer width along each $\mathbf{e}_j$. Nonetheless, branching on the hyperplane defined by $\mathbf{p}_1\mathbf{x}$ followed by that defined by $\mathbf{p}_2\mathbf{x}$ solves the problem in only three

Figure 1: Branching tree for (KP2). Crossed out nodes are pruned due to integer infeasibility.

B&B nodes, as shown in Figure 1. Similar to (KP1), $\mathrm{width}(\mathbf{p}_2, \mathrm{KP2}) = 22.25 - 10.23 = 12.02$, and hence $\mathbf{p}_2\mathbf{x}$ does not define a good direction to branch on.
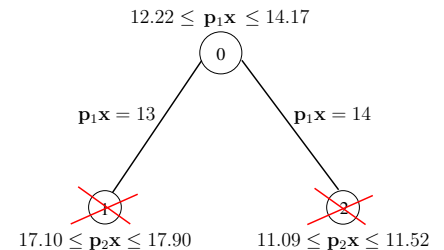
3

Example 3. Consider the following knapsack feasibility problem with $n = 12$ binary variables:

$$57\,x_1 + 68\,x_2 + 79\,x_3 + 87\,x_4 + 106\,x_5 + 117\,x_6 +$$
$$125\,x_7 + 136\,x_8 + 155\,x_9 + 163\,x_{10} + 174\,x_{11} + 185\,x_{12} = 847 \qquad \text{(KP3)}$$
$$0 \le x_j \le 1,\ x_j \in \mathbb{Z},\ j = 1, \ldots, 12.$$

There are no integer feasible solutions, and CPLEX 12.6.3.0 applying B&C on original variables with no objective function takes 451 B&B nodes to prove integer infeasibility of this instance. Similar to (KP1) and (KP2), we get $\mathrm{width}(\mathbf{a}, \mathrm{KP3}) = 0$ and $\mathrm{width}(\mathbf{e}_j, \mathrm{KP3}) = 1$.

The knapsack vector $\mathbf{a} = (57,\ 68, \ldots, 185)$ has the same structure as in (KP1), using the same vectors $\mathbf{p}_1, \mathbf{p}_2, \mathbf{r}$ in Equation (1), except with $M_1 = 48$. We observe that $\mathrm{width}(\mathbf{p}_1, \mathrm{KP3}) = 15.22 - 12.86 = 2.36$ and $\mathrm{iwidth}(\mathbf{p}_1, \mathrm{KP3}) = 3$,

Figure 2: Branching tree for (KP3). Crossed out nodes are pruned due to integer infeasibility.

which is strictly greater than the integer width along each $\mathbf{e}_j$. Nonetheless, branching on the hyperplane defined by $\mathbf{p}_1\mathbf{x}$ followed by that defined by $\mathbf{p}_2\mathbf{x}$ solves the problem in only four B&B nodes, as shown in Figure 2. Similar to (KP1) and (KP2), $\mathrm{width}(\mathbf{p}_2, \mathrm{KP3}) = 22.90 - 11.62 = 11.28$, and $\mathbf{p}_2\mathbf{x}$ does not define a good direction to branch on.
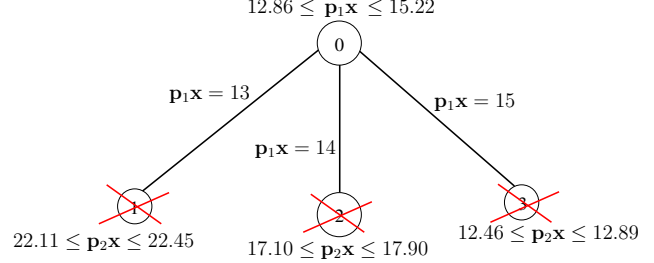
These examples have a common property: the effect of branching on the hyperplane defined by $\mathbf{p}_1\mathbf{x}$ *cascades* down to the subproblems thus created, which are all pruned by branching subsequently on the hyperplane defined by $\mathbf{p}_2\mathbf{x}$. Hence we call them *cascade knapsack problems* (CKPs). The next example illustrates the cascading effect observed over multiple $\mathbf{p}_i\mathbf{x}$ directions, i.e., branching on the hyperplane defined by $\mathbf{p}_1\mathbf{x}$ cascades down, and then the effect of branching on the hyperplane defined by $\mathbf{p}_2\mathbf{x}$ cascades down to the next levels of nodes, followed by branching on the hyperplane defined by $\mathbf{p}_3\mathbf{x}$, and so on.

Example 4. Consider the following knapsack feasibility problem with $n = 12$ binary variables:

$$723\,x_1 + 799\,x_2 + 875\,x_3 + 981\,x_4 + 1285\,x_5 + 1361\,x_6 +$$
$$1467\,x_7 + 1587\,x_8 + 1847\,x_9 + 1953\,x_{10} + 2029\,x_{11} + 2116\,x_{12} = 9312 \qquad \text{(KP4)}$$
$$0 \le x_j \le 1,\ x_j \in \mathbb{Z},\ j = 1, \ldots, 12.$$

The knapsack coefficients have the form $\mathbf{a} = \mathbf{p}_1 M_1 + \mathbf{p}_2 M_2 + \mathbf{p}_3 M_3 + \mathbf{r}$ for the same set of vectors $\mathbf{p}_1, \mathbf{p}_2$, and $\mathbf{r}$ given in Equation (1), with $\mathbf{p}_3 = (5, 3, 1, 2, 4, 2, 3, 5, 3, 4, 2, 1)$, $M_1 = 572$, $M_2 = 97$, and $M_3 = 11$. The vector $\mathbf{p}_3$ is chosen to be linearly independent of $\mathbf{p}_1$ and $\mathbf{p}_2$ here. This knapsack problem is also integer infeasible, and CPLEX 12.6.3.0 branching on original variables takes 491 B&B nodes to prove integer infeasibility. Further, $\mathrm{width}(\mathbf{a}, \mathrm{KP4}) = 0$ and $\mathrm{width}(\mathbf{e}_j, \mathrm{KP4}) = 1$ for all j, as in the previous instances. At the same time,

Figure 3: Branching tree for (KP4). Crossed out nodes are pruned due to integer infeasibility.

we can solve this problem easily if we branch on the hyperplanes defined by $\mathbf{p}_1\mathbf{x}, \mathbf{p}_2\mathbf{x}$, and $\mathbf{p}_3\mathbf{x}$, in that order. The branching tree is shown in Figure 3, and the problem is solved in seven B&B nodes.
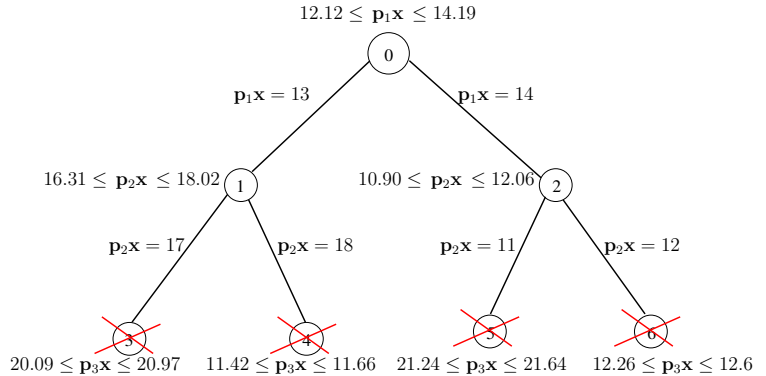
We could present a similar instance with $\mathbf{a} = \mathbf{p}_1 M_1 + \mathbf{p}_2 M_2 + \mathbf{p}_3 M_3 + \mathbf{r}$ where the integer width of its LP relaxation along $\mathbf{p}_1$ is 3, similar to (KP3). Just as in (KP4), we solve this problem quickly if we branch on the hyperplanes defined by $\mathbf{p}_1\mathbf{x}$, $\mathbf{p}_2\mathbf{x}$, and $\mathbf{p}_3\mathbf{x}$, in that order.

3 Procedure for creating a CKP

We present a procedure (Figure 4) to create CKP instances with the structure illustrated in Example 2 (KP1), where $\mathbf{a} = \mathbf{p}_1 M_1 + \mathbf{p}_2 M_2 + \mathbf{r}$ and branching on the hyperplane defined by $\mathbf{p}_1\mathbf{x}$ followed by branching on the hyperplane defined by $\mathbf{p}_2\mathbf{x}$ solves the problem easily. The remaining examples (KP2, KP3, KP4) (and instances in the computational tests—see 4) were generated by appropriate modifications of this procedure. We use the following notation in the description.

Definition 3.1. *For $x \in \mathbb{R}$, we define*

$$\lfloor x \rfloor = \begin{cases} \lfloor x \rfloor & \text{if } x \notin \mathbb{Z}, \\ x - 1 & \text{if } x \in \mathbb{Z}; \end{cases} \quad \text{and} \quad \lceil x \rceil = \begin{cases} \lceil x \rceil & \text{if } x \notin \mathbb{Z}, \\ x + 1 & \text{if } x \in \mathbb{Z}. \end{cases}$$

Other notation: We denote the vector of ones by $\mathbf{1}$, and the box with upper bound $\mathbf{1}$ by $B_{\mathbf{1}} = \{ \mathbf{x} \in \mathbb{R}^n \mid \mathbf{0} \leq \mathbf{x} \leq \mathbf{1} \}$. We denote by $\mathbb{Z}_+^n$ the set of all n-vectors with positive integer entries.

PROCEDURE CKP

Input: Vectors $\mathbf{p}_1, \mathbf{p}_2, \mathbf{r}$ with $\mathbf{p}_1, \mathbf{p}_2 \in \mathbb{Z}_+^n$, $\mathbf{r} \in \mathbb{Z}^n$, and $\mathbf{p}_1, \mathbf{p}_2$ linearly independent.

Output: M_1, M_2, β', β forming the CKP instance with $\mathbf{a} = \mathbf{p}_1 M_1 + \mathbf{p}_2 M_2 + \mathbf{r}$.

1. Choice of k_1, k_2:

   Find $\gamma_1 = \max\{\mathbf{p}_1\mathbf{x} \mid \mathbf{x} \in B_{\mathbf{1}}\}$, $\delta_1 = \min\{\mathbf{p}_1\mathbf{x} \mid \mathbf{x} \in B_{\mathbf{1}}\}$; set $k_1 = \lfloor (\lfloor \gamma_1 \rfloor + \lceil \delta_1 \rceil)/2 \rceil$.

$$\begin{aligned} \text{Find } \gamma_2 &= \max\{\mathbf{p}_2\mathbf{x} \mid \mathbf{p}_1\mathbf{x} = k_1 + 1, \mathbf{x} \in B_{\mathbf{1}}\}, \\ \text{and } \delta_2 &= \min\{\mathbf{p}_2\mathbf{x} \mid \mathbf{p}_1\mathbf{x} = k_1 + 1, \mathbf{x} \in B_{\mathbf{1}}\}; \\ \text{set } k_2 &= \lfloor (\lfloor \gamma_2 \rfloor + \lceil \delta_2 \rceil)/2 \rceil. \end{aligned} \tag{2}$$

2. Choice of M_2, β_2', β_2:

$$\begin{aligned} \text{Find } \eta_2 &= \max\{\mathbf{r}\mathbf{x} \mid \mathbf{p}_1\mathbf{x} = k_1 + 1, \mathbf{p}_2\mathbf{x} \leq k_2, \mathbf{x} \in B_{\mathbf{1}}\}, \\ \text{and } \mu_2 &= \min\{\mathbf{r}\mathbf{x} \mid \mathbf{p}_1\mathbf{x} = k_1 + 1, \mathbf{p}_2\mathbf{x} \geq k_2 + 1, \mathbf{x} \in B_{\mathbf{1}}\}. \end{aligned} \tag{3}$$

$$\text{If } \lceil \eta_2 \rceil - \lfloor \mu_2 \rfloor \geq 1, \text{ set } M_2 = \lceil \eta_2 \rceil - \lfloor \mu_2 \rfloor, \text{ and set } \mathbf{a}_2 = \mathbf{p}_2 M_2 + \mathbf{r}. \tag{4}$$

$$\begin{aligned} \text{Find } \eta_2' &= \max\{\mathbf{a}_2\mathbf{x} \mid \mathbf{p}_1\mathbf{x} = k_1 + 1, \mathbf{p}_2\mathbf{x} \leq k_2, \mathbf{x} \in B_{\mathbf{1}}\}, \\ \text{and } \mu_2' &= \min\{\mathbf{a}_2\mathbf{x} \mid \mathbf{p}_1\mathbf{x} = k_1 + 1, \mathbf{p}_2\mathbf{x} \geq k_2 + 1, \mathbf{x} \in B_{\mathbf{1}}\}. \end{aligned} \tag{5}$$

$$\text{If } \lceil \eta_2' \rceil \leq \lfloor \mu_2' \rfloor \text{ set } \beta_2' = \lceil \eta_2' \rceil \text{ and } \beta_2 = \lfloor \mu_2' \rfloor. \tag{6}$$

3. Choice of $M_1, \beta_1', \beta_1,$:

$$\begin{aligned} \text{Find } \eta_1 &= \max\{\mathbf{a}_2\mathbf{x} \mid \mathbf{p}_1\mathbf{x} \leq k_1, \mathbf{x} \in B_{\mathbf{1}}\}, \quad \text{and} \\ \mu_1 &= \min\{\mathbf{a}_2\mathbf{x} \mid \mathbf{p}_1\mathbf{x} \geq k_1 + 2, \mathbf{x} \in B_{\mathbf{1}}\}. \end{aligned} \tag{7}$$

$$\text{Set } M_1 = \lceil \max(\eta_1 - \beta_2', \beta_2 - \mu_1, M_2) \rceil. \tag{8}$$

$$\text{Set } \mathbf{a}_1 = \mathbf{p}_1 M_1 + \mathbf{a}_2, \quad \beta_1' = \beta_2' + (k_1 + 1)M_1, \quad \text{and} \quad \beta_1 = \beta_2 + (k_1 + 1)M_1. \tag{9}$$

4. Output instance: Set $\beta' = \beta_1'$, $\beta = \beta_1$, $\mathbf{a} = \mathbf{a}_1$, and return $(M_1, M_2, \beta', \beta, \mathbf{a})$.

Figure 4: Procedure to generate CKPs.

We now present two lemmas describing the Procedure (Figure 4). Notice that we refer to both the final knapsack problem and its LP relaxation as (CKP), with the exact choice evident from the context.

Lemma 3.2. *If the quantities defined in Equations (4), (6), and (8) exist, then* (CKP)$\land (\mathbf{p_1 x} = k_1 + 1) \neq \emptyset$.

Proof. Let $\mathbf{x}_1$ and $\mathbf{x}_2$ attain the optima for the LPs in Equation (7) (the maximum and the minimum, respectively), based on which M_1 is defined in Equation (8). We get

$$
\begin{aligned}
\max\{\mathbf{a_1 x} \mid \mathbf{p_1 x} \le k_1, \mathbf{x} \in B_1\} \;\; &\le \;\; k_1 M_1 + \max\{\mathbf{a_2 x} \mid \mathbf{p_1 x} \le k_1, \mathbf{x} \in B_1\} = k_1 M_1 + \eta_1 \\
&= \;\; [(k_1 + 1)M_1 + \beta_2'] - (M_1 - (\eta_1 - \beta_2')) \\
&= \;\; \beta_1' - (M_1 - (\eta_1 - \beta_2')) \;\; < \;\; \beta_1'.
\end{aligned}
$$

The third equality follows from the definition of β_1' in Equation (9), and the last strict inequality follows from the definition of M_1 in Equation (8). Similarly, we get

$$
\begin{aligned}
\min\{\mathbf{a_1 x} \mid \mathbf{p_1 x} \ge k_1 + 2, \mathbf{x} \in B_1\} \;\; &\ge \;\; (k_1 + 2)M_1 + \min\{\mathbf{a_2 x} \mid \mathbf{p_1 x} \ge k_1 + 2, \mathbf{x} \in B_1\} \\
&= \;\; (k_1 + 2)M_1 + \mu_1 \;\; = \;\; [(k_1 + 1)M_1 + \beta_2] + (M_1 - (\beta_2 - \mu_1)) \\
&= \;\; \beta_1 + (M_1 - (\beta_2 - \mu_1)) \;\; > \;\; \beta_1.
\end{aligned}
$$

Hence $\mathbf{x}' \in \{\mathbf{x} \mid \beta_1' \le \mathbf{a_1 x} \le \beta_1, \mathbf{x} \in B_1\}$ implies that $k_1 < \mathbf{p_1 x}' < k_1 + 2$. Further, since $\mathbf{a}_2$ and $\mathbf{p}_1$ have positive entries, the structure of the LPs in Equation (7) implies that we will have $\mathbf{p_1 x}_1 = k_1$ and $\mathbf{p_1 x}_2 = k_1 + 2$. Thus we can find a constant $\lambda \in (0,1)$ such that $\mathbf{x}' = \lambda \mathbf{x}_1 + (1 - \lambda)\mathbf{x}_2$ satisfies $\beta_1' \le \mathbf{a_1 x}' \le \beta_1$ with $\mathbf{p_1 x}' = k_1 + 1$. Thus when we branch on the hyperplane defined by $\mathbf{p_1 x}$ in the original (CKP) problem, the branch created by setting $\mathbf{p_1 x} = k_1 + 1$ cannot be pruned due to LP infeasibility. $\qquad\square$

Lemma 3.3. *If the quantities defined in Equations (4), (6), and (8) exist, then the integer infeasibility of* (CKP) *is proven by* $(\mathbf{p_1 x} = k_1 + 1) \bigwedge (\mathbf{p_2 x} \le k_2 \lor \mathbf{p_2 x} \ge k_2 + 1)$.

Proof. Let $\mathbf{x}_3$ and $\mathbf{x}_4$ attain the optima for the LPs in Equation (5) (the maximum and the minimum, respectively). Since $\eta_2' < \beta_2' \le \beta_2 < \mu_2'$, and due to the way $\mathbf{a}_1, \beta_1'$, and β_1 are defined (in Equation (9)), there is some $\lambda' \in (0,1)$ such that $\mathbf{x}'' = \lambda' \mathbf{x}_3 + (1 - \lambda')\mathbf{x}_4$ and $\mathbf{x}'' \in \{\mathbf{x} \mid \beta_1' \le \mathbf{a_1 x} \le \beta_1, \mathbf{p_1 x} = k_1 + 1, \mathbf{x} \in B_1\}$. Since $\mathbf{a}_2$ and $\mathbf{p}_2$ both have positive entries, the structure of LPs in Equation (5) implies we have $\mathbf{p_2 x}_3 = k_2$ and $\mathbf{p_2 x}_4 = k_2 + 1$. Further,

$$\mathbf{x}'' \in \{\mathbf{x} \mid \beta_1' \le \mathbf{a_1 x} \le \beta_1, \mathbf{p_1 x} = k_1 + 1, \mathbf{x} \in B_1\} \Rightarrow \mathbf{x}'' \in \{\mathbf{x} \mid \beta_2' \le \mathbf{a_2 x} \le \beta_2, \mathbf{p_1 x} = k_1 + 1, \mathbf{x} \in B_1\}$$
$$\Rightarrow k_2 < \mathbf{p_2 x}'' < k_2 + 1,$$

as $\mathbf{x}'' = \lambda' \mathbf{x}_3 + (1 - \lambda')\mathbf{x}_4$ must hold for some $\lambda' \in (0,1)$. Thus (CKP)$\land (\mathbf{p_1 x} = k_1 + 1) \neq \emptyset$ as already shown by Lemma 3.2, and (CKP)$\land (\mathbf{p_1 x} = k_1 + 1) \bigwedge (\mathbf{p_2 x} \le k_2 \lor \mathbf{p_2 x} \ge k_2 + 1) = \emptyset$. $\qquad\square$

The preceding two lemmas show that branching on the hyperplane defined by $\mathbf{p_1 x}$ followed by that defined by $\mathbf{p_2 x}$ proves the integer infeasibility of (CKP). While Lemma 3.2 shows the existence of some $\mathbf{x}' \in$ (CKP) satisfying $\mathbf{p_1 x}' = k_1 + 1$, we have not explicitly shown that $\text{width}(\mathbf{p}_1, \text{CKP}) > 1$. In fact, the Procedure (in Figure 4) is not guaranteed to work on *every* choice of the input vectors. For instance, we might not get a valid choice for M_2 in Equation (4), or for β' and β in Equation (6). Further, while the Procedure assumes only that $\mathbf{p}_1$ and $\mathbf{p}_2$ are linearly independent, we might not get the required structure for the CKP if they are "too close to each other", e.g., the two vectors differ in only a few entries for large n.

To create instances with $\text{iwidth}(\mathbf{p}_1, \text{CKP}) = 2$ as in (KP2), we could replace the constraint $\mathbf{p_1 x} = k_1 + 1$ in Equations (2), (3), and (5) with $k_1 + 1 \le \mathbf{p_1 x} \le k_1 + 2$, and the constraint $\mathbf{p_1 x} \ge k_1 + 2$ in Equation (7) with $\mathbf{p_1 x} \ge k_1 + 3$. Alternatively, we could make only the latter change (to $\mathbf{p_1 x} \ge k_1 + 3$), while sticking with $\mathbf{p_1 x} = k_1 + 1$ in the former Equations, as we did in computational tests.

Example 2 continued: We created the instance (KP1) using the procedure described in Figure 4. We obtained $\gamma_1 = 24$ and $\delta_1 = 0$, setting $k_1 = 12$, and then $\gamma_2 = 23$, $\delta_2 = 9$, setting $k_2 = 16$. The remaining parameter values are obtained as follows. $\eta_2 = 4$, $\mu_2 = -4$, giving $M_2 = 10$, and $\eta_2' = 164$, $\mu_2' = 166$, and hence $\beta_2' = \beta_2 = 165$. We finally get $\eta_1 = 221$, $\mu_1 = 100$, giving $M_1 = 66$, and $\beta_1' = \beta_1 = 1023$.

4 Column Basis Reduction and CKPs

We present computations illustrating the structure of CKPs on larger instances at http://www.math.wsu.edu/faculty/bkrishna/CKP/. We applied the reformulation technique termed column basis reduction (CBR) [13] on these instances. CBR is a simple preconditioning method for IP feasibility that replaces the problem

$$\{\mathbf{x} \in \mathbb{Z}^n \mid \mathbf{b}' \le A\mathbf{x} \le \mathbf{b}\} \quad \text{with} \quad \{\mathbf{y} \in \mathbb{Z}^n \mid \mathbf{b}' \le AU\mathbf{y} \le \mathbf{b}\}, \tag{10}$$

where U is a unimodular matrix computed using basis reduction (BR) applied on A, which makes the columns of A short, i.e., have small euclidean norms, and nearly orthogonal. The variables $\mathbf{y}$ in the reformulation and the original variables $\mathbf{x}$ are related as $\mathbf{y} = U^{-1}\mathbf{x}$. Standard methods of BR include the Lenstra-Lenstra-Lovász (LLL) reduction [14], which runs in time polynomial time, and versions of Korkine-Zolotarev (KZ) reduction including block-KZ or BKZ reduction [21], which results in a higher quality of reduction but runs in polynomial time only when the dimension is fixed.

While the CKPs remain intractable, their CBR reformulations are each solved in a few B&B nodes. We previously analyzed the efficacy of CBR on DKPs [13], where the quantities in the problems in the Inequality (10) are specified as follows.

$$A = \begin{bmatrix} \mathbf{a} \\ I \end{bmatrix} = \begin{bmatrix} \mathbf{p}M + \mathbf{r} \\ I \end{bmatrix}, \quad \text{where } I \text{ is the identity matrix, } \mathbf{b}' = \begin{bmatrix} \beta' \\ \mathbf{0} \end{bmatrix}, \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} \beta \\ \mathbf{u} \end{bmatrix} \text{ for } \mathbf{u} \in \mathbb{Z}_+^n.$$

Under some assumptions on the size of M compared to the norms of $\mathbf{p}$ and $\mathbf{r}$, we showed that branching on the last few variables in the CBR reformulation is equivalent to branching on the hyperplane defined by $\mathbf{p}\mathbf{x}$ in the original DKP. We now present a more general analysis which suggests a similar behavior for the CBR reformulation of CKPs, i.e., branching on the important directions defined by $\mathbf{p}_1\mathbf{x}, \mathbf{p}_2\mathbf{x}, \mathbf{p}_3\mathbf{x}, \dots$ is captured by branching on the last few individual variables in the reformulation.

We first introduce some definitions and notation related to BR. Given a matrix $B \in \mathbb{R}^{m \times n}$ with $m \ge n$ and linearly independent columns, the *lattice* generated by the columns of B is $\mathcal{L}(B) = \{B\mathbf{x} \mid \mathbf{x} \in \mathbb{Z}^n\}$, i.e., the set of all integer combinations of columns of B. The k-th successive minimum of $\mathcal{L}(B)$ is

$$\Lambda_k(\mathcal{L}(B)) = \min\{t \mid \exists k \text{ linearly independent vectors in } \mathcal{L}(B) \text{ with norm at most } t\}.$$

Suppose there is a constant c_n that depends only on n with the following property: if $\tilde{\mathbf{b}}_j$ denote the columns of $\tilde{B} = BU$ computed by BR, then

$$\max\{\|\tilde{\mathbf{b}}_1\|, \dots, \|\tilde{\mathbf{b}}_\ell\|\} \le c_n \Lambda_\ell(\mathcal{L}(B)) \text{ for } \ell = 1, \dots, n.$$

Then c_n is termed the *strength* of BR; the smaller the value of c_n, the more reduced are the columns of BU. LLL reduction has strength $c_n = 2^{(n-1)/2}$ while KZ reduction has strength $c_n = \sqrt{n}$ (see [21], for instance). Finally, the *kernel lattice* or *null lattice* of the columns of B is $\mathcal{N}(B) = \{\mathbf{x} \in \mathbb{Z}^n \mid B\mathbf{x} = \mathbf{0}\}$.

We consider a general CKP of the form $\{\mathbf{x} \in \mathbb{Z}^n \mid \beta' \le \mathbf{a}\mathbf{x} \le \beta, \ \mathbf{0} \le \mathbf{x} \le \mathbf{1}\}$ where the coefficient vector has the structure $\mathbf{a} = \mathbf{p}_1 M_1 + \mathbf{p}_2 M_2 + \cdots + \mathbf{p}_t M_t + \mathbf{r}$ for positive linearly independent vectors $\mathbf{p}_i \in \mathbb{Z}_+^n, i = 1, \dots, t$ and $\mathbf{r} \in \mathbb{Z}^n$ (we assume $t \ge 2$). The multipliers M_i satisfy $M_1 > M_2 > \cdots > M_t$. The CBR reformulation of this CKP is of the form $\{\mathbf{y} \in \mathbb{Z}^n \mid \beta' \le (\mathbf{a}U)\mathbf{y} \le \beta, \ \mathbf{0} \le U\mathbf{y} \le \mathbf{1}\}$, where U is the unimodular matrix obtained by applying BR on the matrix

$$A = \begin{bmatrix} \mathbf{a} \\ I \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1 M_1 + \mathbf{p}_2 M_2 + \cdots + \mathbf{p}_t M_t + \mathbf{r} \\ I \end{bmatrix}. \tag{11}$$

We denote by $\tilde{A} = AU$ the matrix resulting from applying BR on A in Equation (11). With $P \in \mathbb{Z}_+^{t \times n}$ denoting the matrix obtained by stacking the rows $\mathbf{p}_1, \dots, \mathbf{p}_t$ vertically in that order, we similarly denote $\tilde{P} = PU$, $\tilde{\mathbf{a}} = \mathbf{a}U$, as well as $\tilde{\mathbf{p}}_i = \mathbf{p}_i U$ for $i = 1, \dots, t$. For $1 \le r \le s \le n$, we denote the subset of r-th to s-th entries of the i-th row of $\tilde{P}$ (equivalently of $\tilde{\mathbf{p}}_i$) by $\tilde{P}_{i,r:s}$ (or $\tilde{\mathbf{p}}_{i,r:s}$). The following theorem describes why CBR might be effective in solving CKPs. We assume the strength of BR c_n is fixed.

Theorem 4.1. *There exist functions $f_1, \ldots, f_t$ such that given $\mathbf{s} \in \mathbb{Z}^t$ with entries satisfying*

$$1 \leq s_t \leq \cdots \leq s_1 \leq n - t, \tag{12}$$

if

$$M_i > f_i(M_{i+1}, \ldots, M_t, s_i, P, \mathbf{r}, c_n), \ i = 1, \ldots, t, \tag{13}$$

then

$$\tilde{P}_{i,1:s_i} = 0, \ i = 1, \ldots, t. \tag{14}$$

Further, there exist M_i with size polynomial in size(P), size(r), and size(c_n) satisfying the Inequality (13).

Before presenting the somewhat technical proof of Theorem 4.1, we give some intuition for its implication. As an example, consider the result in Equation (14) for $n = 12, t = 3$ (as in Example KP4), and let $s_1 = 9, s_2 = s_3 = 8$. Then the matrix $\tilde{P}$ has the form

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxtimes & \boxtimes & \boxtimes \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxtimes & \boxtimes & \boxtimes & \boxtimes \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxtimes & \boxtimes & \boxtimes & \boxtimes \end{bmatrix},$$

where $\boxtimes$ represent possible nonzero entries.

Intuitively, if M_1 is sufficiently larger than $M_2, \ldots, M_t$, then $\mathbf{p}_1 M_1$ contributes the most to the length of $\mathbf{a}$. Subsequently, if M_2 is sufficiently larger than $M_3, \ldots, M_t$, then the next biggest contribution to the norm of $\mathbf{a}$ comes from $\mathbf{p}_2 M_2$, and so on. Hence, to shorten the columns of A in Equation (11), the best option is to zero out "many" components of $\mathbf{p}_1$, followed by possibly fewer components of $\mathbf{p}_2$, and so on. Since $P\mathbf{x} = PU\mathbf{y} = \tilde{P}\mathbf{y}$, exploring all possible branches for $y_n, \ldots, y_{s_1+1}$ in the CBR reformulation is equivalent to branching on the hyperplane defined by $\mathbf{p}_1\mathbf{x}$ in the original CKP, exploring all possible branches for $y_n, \ldots, y_{s_2+1}$ is equivalent to branching on the hyperplane defined by $\mathbf{p}_2\mathbf{x}$, and so on.

We first present a lemma, which we use in the proof of Theorem 4.1. For brevity, we let

$$\alpha_k = \Lambda_k(\mathcal{N}(P)), \ k = 1, \ldots, n - t. \tag{15}$$

In words, α_k is the smallest number such that there are k linearly independent vectors in $\mathcal{N}(P)$ with length bounded by α_k.

Lemma 4.2. *For the matrix A given in Equation (11),*

$$\Lambda_k(\mathcal{L}(A)) \leq (\|\mathbf{r}\| + 1)\alpha_k, \ for \ k = 1, \ldots, n - t. \tag{16}$$

Proof. Let $k \leq n - t$, and let $\mathbf{v}_1, \ldots, \mathbf{v}_k \in \mathcal{N}(P)$ be linearly independent vectors. Thus $\mathbf{p}_i\mathbf{v}_j = 0$ for all $i = 1, \ldots, t$ and $j = 1, \ldots, k$. Then $A\mathbf{v}_1, \ldots, A\mathbf{v}_k \in \mathcal{L}(A)$ are linearly independent, and

$$A\mathbf{v}_j = \begin{bmatrix} \mathbf{a} \\ I \end{bmatrix} \mathbf{v}_j = \begin{bmatrix} \mathbf{p}_1 M_1 + \mathbf{p}_2 M_2 + \cdots + \mathbf{p}_t M_t + \mathbf{r} \\ I \end{bmatrix} \mathbf{v}_j = \begin{bmatrix} \mathbf{r}\mathbf{v}_j \\ \mathbf{v}_j \end{bmatrix} \ \forall j.$$

Thus we get $\| A\mathbf{v}_j \| \leq (\|\mathbf{r}\| + 1) \|\mathbf{v}_j\| \ \forall j$, and the bound in the Inequality (16) follows. $\square$

Proof of Theorem 4.1: For brevity, we denote $\mathbf{r}_i = \sum_{j=i}^{t} \mathbf{p}_j M_j + \mathbf{r}$ and $\tilde{\mathbf{r}}_i = \mathbf{r}_i U$ for $i = 1, \ldots, t$. We also let $\rho = \max\{\|\mathbf{p}_1\|, \ldots, \|\mathbf{p}_t\|, \|\mathbf{r}\| + 1\}$. We show that

$$f_i(M_{i+1}, \ldots, M_t, s_i, P, \mathbf{r}, c_n) = c_n \alpha_{s_i} \rho^2 (M_{i+1} + \cdots + M_t + 1)$$

are suitable functions. Given that α_{s_i} have size polynomial in the size of P [22], one could use these functions f_i to choose a set of M_i that have size polynomial in the sizes of $P, \mathbf{r}, c_n$.

Given $\mathbf{s} \in \mathbb{Z}^t$ satisfying the Inequalities (12), and assuming Inequality (13) holds, we show that $\tilde{P}_{i,1:s_i} = 0$ holds for $i = 1, \ldots, \ell$ with $\ell \leq t$ by induction, with the base case of $\ell = 0$ holding trivially by extending

8

the definitions to the case of $t = 0$. Let $\ell \geq 1$, and suppose this result holds for all $i < \ell$. We prove the result for $i = \ell$. Fix $j \leq s_\ell$. We are done if we manage to show

$$\tilde{P}_{\ell,j} = 0. \tag{17}$$

If $i < \ell$ and $j \leq s_\ell$, then $j \leq s_\ell \leq s_i$. Hence the induction hypothesis implies $\tilde{P}_{i,j} = 0$. Recalling that $\tilde{a}_j$ denotes the jth entry of $\tilde{\mathbf{a}}$, we get

$$\tilde{a}_j = \sum_{i=1}^{\ell-1} \tilde{\mathbf{p}}_{i,j} M_i + \tilde{\mathbf{p}}_{\ell,j} M_\ell + \tilde{\mathbf{r}}_{\ell+1,j} \ = \ \tilde{\mathbf{p}}_{\ell,j} M_\ell + \tilde{\mathbf{r}}_{\ell+1,j}.$$

To get a contradiction, assume Equation (17) does not hold. Then we get

$$\| \tilde{A}_{:,j} \| \geq |\tilde{a}_j| = |\tilde{\mathbf{p}}_{\ell,j} M_\ell + \tilde{\mathbf{r}}_{\ell+1,j}|$$
$$\geq |\tilde{\mathbf{p}}_{\ell,j} M_\ell| - |\tilde{\mathbf{r}}_{\ell+1,j}| \ \geq \ M_\ell - |\tilde{\mathbf{r}}_{\ell+1,j}|,$$

with the last inequality following from the fact that $\tilde{\mathbf{p}}_{\ell,j} \in \mathbb{Z}$. Hence we get

$$\begin{aligned} M_\ell &\leq \ |\tilde{\mathbf{r}}_{\ell+1,j}| + \| \tilde{A}_{:,j} \| &&= \ |\mathbf{r}_{\ell+1} U_{:,j}| + \| \tilde{A}_{:,j} \| \\ &\leq \ \| \mathbf{r}_{\ell+1} \| \| U_{:,j} \| + \| \tilde{A}_{:,j} \| &&\leq \ \| \mathbf{r}_{\ell+1} \| \| \tilde{A}_{:,j} \| + \| \tilde{A}_{:,j} \| \\ &= \ (\| \mathbf{r}_{\ell+1} \| + 1) \| \tilde{A}_{:,j} \| . \end{aligned} \tag{18}$$

The fourth inequality above, which replaced $\| U_{:,j} \|$ with $\| \tilde{A}_{:,j} \|$, follows from the definition of $\tilde{A} = AU$, which has U as its submatrix (see Equation (11)). Since $s_\ell \leq n - t$, there are s_ℓ linearly independent vectors in $\mathcal{N}(P)$ with norm bounded by α_{s_ℓ}, and hence by Lemma 4.2 there are the same number of linearly independent vectors in $\mathcal{L}(A)$ with norm bounded by $(\| \mathbf{r} \| + 1)\alpha_{s_\ell}$. Also, since $\tilde{A}$ was computed by BR with strength c_n and since $j \leq s_\ell$, we get that

$$\| \tilde{A}_{:,j} \| \leq c_n(\| \mathbf{r} \| + 1)\alpha_{s_\ell}. \tag{19}$$

Combining the bounds in Inequalities (18) and (19) yields

$$\begin{aligned} M_\ell &\leq \ c_n \alpha_{s_\ell} (\| \mathbf{r} \| + 1)(\| \mathbf{r}_{\ell+1} \| + 1) &&= \ c_n \alpha_{s_\ell} (\| \mathbf{r} \| + 1)(\| M_{\ell+1} \mathbf{p}_{\ell+1} + \cdots + M_t \mathbf{p}_t + \mathbf{r} \| + 1) \\ &\leq c_n \alpha_{s_\ell} (\| \mathbf{r} \| + 1)(M_{\ell+1} \| \mathbf{p}_{\ell+1} \| + \cdots + M_t \| \mathbf{p}_t \| + \| \mathbf{r} \| + 1) \\ &\leq \ c_n \alpha_{s_\ell} \rho^2 (M_{\ell+1} + \cdots + M_t + 1), &&\text{which provides the contradiction.} \qquad \square \end{aligned}$$

5 Discussion

Restricting to directions defined by rational vectors, we could model the problem of finding the direction along which the width of the polyhedron of a given IP is the smallest as a mixed integer program— see the work of Mahajan and Ralphs [18] for one such model. But this MIP has more variables and constraints than the original IP, and is typically harder to solve as well. For instance, one could solve this MIP corresponding to the knapsack instance (KP2) to identify the knapsack coefficient vector $\mathbf{a} = (71, 82, \ldots, 227)$ as the obvious thin direction, along which the polyhedron has the minimal width of 0. CPLEX takes 810 B&B nodes to solve this MIP (as compared to 462 nodes to solve the original (CKP) itself—see Example 2). But we could modify this MIP to identify another *useful* direction. For $\mathbf{p}' = (26, 30, 34, 37, 49, 53, 56, 60, 72, 75, 79, 83)$ as identified by the modified MIP, we get $\mathrm{width}(\mathbf{p}', \mathrm{KP2}) = 358.98 - 358.66 = 0.32$ and hence $\mathrm{iwidth}(\mathbf{p}', \mathrm{KP2}) = 0$, even though $\mathbf{p}'\mathbf{x}$ is not a thin direction.

CPLEX takes $2,644$ B&B nodes to solve the modified MIP that identified $\mathbf{p}'$. If we *knew* $\mathbf{p}'$ beforehand, we could solve (KP2) at the root node by branching on the hyperplane defined by $\mathbf{p}'\mathbf{x}$. At the same time, finding such a good branching direction is typically harder than solving the original IP itself. Also notice

that $\mathbf{p}'$ has larger coefficients than $\mathbf{p}_1$ and $\mathbf{p}_2$. It would be interesting to identify the class of IPs for which such a good direction is guaranteed to exist. Also, would such a good direction have "small" coefficients?

Our previous results on the hardness of ordinary B&B on DKPs [13] (as well on more general integer-infeasible knapsacks [12]) apply to the case of CKPs as well. Indirectly, these results imply the hardness of B&B on CKPs when thin directions as specified by the individual variables are used for branching.

While we demonstrated the effectiveness of CBR in solving the CKP instances quickly, the main message we want to convey is the structure of these problems: branching on the hyperplanes defined by $\mathbf{p}_1\mathbf{x}, \mathbf{p}_2\mathbf{x}$, and $\mathbf{p}_3\mathbf{x}$ solves them quickly, even though they might not be thin directions. On the other hand, branching on thin directions (along the individual variables) might not always be a good idea for B&B. Indeed, if the structure of the problem is assumed to be known, i.e., one is given $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{r}$ along with M_1, M_2, and M_3, one could verify directly that branching on these hyperplanes solves the problem. But if one is given just the final knapsack coefficient vector $\mathbf{a}$, CBR appears to be an effective method to *discover* that structure. Alternatively, one could try to *guess* $\mathbf{p}_1$ from $\mathbf{a}$, e.g., using the ideas of diophantine approximation [22], add an extra variable that models $\mathbf{p}_1\mathbf{x}$, and force CPLEX to branch on this extra variable. The idea of adding extra variables in a similar setting was explored by Aardal and Wolsey [4] in the context of lattice-based extended formulations for integer inequality systems. Such an approach would also be much more effective than trying to solve the original instances using ordinary branch-and-cut.

References

[1] K.Aardal, R.Bixby, C.A.J.Hurkens, A.Lenstra, and J.Smeltink. Market split and basis reduction: Towards a solution of the Cornuéjols-Dawande instances. *INFORMS J.Comput.*, 12(3):192–202, 2000.

[2] K.Aardal, C.A.J.Hurkens, and A.Lenstra. Solving a system of linear Diophantine equations with lower and upper bounds on the variables. *Math.O.R.*, 25(3):427–442, 2000.

[3] K.Aardal and A.Lenstra. Hard equality-constrained integer knapsacks. *Math.O.R.*, 29(3):724-738,2004.

[4] K.Aardal and L.A.Wolsey. Lattice based extended formulations for integer linear equality systems. *Math.Prog.*, 121(2):337, 2010.

[5] T.Achterberg, T.Koch, and A.Martin. Branching rules revisited. *Oper.Res.Lett.*, 33(1):42–54, 2005.

[6] V.Chvátal. Hard knapsack problems. *Oper.Res.*, 28(6):1402–1411, 1980.

[7] W.Cook, T.Rutherford, H.E.Scarf, and D.F.Shallcross. An implementation of the generalized basis reduction algorithm for integer programming. *ORSA J.Comput.*, 5(2):206–212, 1993.

[8] G.Cornuéjols and M.Dawande. A class of hard small 0–1 programs. In *IPCO '98*, vol.1412 *LNCS*, 284–293, 1998.

[9] R.Jeroslow. Trivial integer programs unsolvable by branch-and-bound. *Math.Prog.*, 6:105–109, 1974.

[10] R.Kannan. Improved algorithms for integer programming and related lattice problems. In *STOC '83*, 193–206, 1983.

[11] T.Koch, T.Achterberg, E.Andersen, et al. MIPLIB 2010. *Math.Prog.Comput.*, 3(2):103–163, 2011.

[12] B.Krishnamoorthy. Bounds on the size of branch-and-bound proofs for integer knapsacks. *Oper.Res.Lett.*, 36(1):19–25, 2008.

[13] B.Krishnamoorthy and G.Pataki. Column basis reduction and decomposable knapsack problems. *Disc.Opt.*, 6(3):242–270, 2009.

[14] A.Lenstra, H.W.Lenstra, Jr., and L.Lovász. Factoring polynomials with rational coefficients. *Math.Ann.*, 261:515–534, 1982.

[15] H.W.Lenstra, Jr. Integer programming with a fixed number of variables. *Math.O.R.*, 8:538–548, 1983.

[16] Q.Louveaux and L.A.Wolsey. Combining problem structure with basis reduction to solve a class of hard integer programs. *Math.O.R.*, 27(3):470–484, 2002.

[17] L.Lovász and H.E.Scarf. The generalized basis reduction algorithm. *Math.O.R.*, 17:751–764, 1992.

[18] A.Mahajan and T.Ralphs. On the complexity of selecting disjunctions in integer programming. *SIAM J.Opt.*, 20(5):2181–2198, 2010.

[19] S.Mehrotra and Z.Li. On generalized branching methods for mixed integer programming. *Opt.Onl.*, 2005. optimization-online.org/DB_HTML/2005/01/1035.html.

[20] G.Pataki, M.Tural, and E.B.Wong. Basis reduction and the complexity of branch-and-bound. In *SODA '10*, 1254–1261, 2010.

[21] C-P.Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theo.Comp.Sci.*, 53:201–225, 1987.

[22] A.Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1986.