A study of lattice reformulations for integer programming

Aardal, Karen; Scavuzzo, Lara; Wolsey, Laurence A.

DOI
10.1016/j.orl.2023.05.001

Publication date
2023

Document Version
Final published version

Published in
Operations Research Letters

Operations Research Letters 51 (2023) 401–407

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

A study of lattice reformulations for integer programming

Karen Aardal [a,*], Lara Scavuzzo [a], Laurence A. Wolsey [b]

[a] Technische Universiteit Delft, Mekelweg 4, 2628 CD Delft, the Netherlands
[b] CORE, UCLouvain, 34 voie du Roman Pays, 1348 Louvain-la-Neuve, Belgium

ARTICLE INFO

Article history:
Available online 9 May 2023

Keywords:
Integer optimization
Lattice-based reformulation
Branching on general disjunctions

ABSTRACT

Branch-and-bound for integer optimization typically uses single-variable disjunctions. Enumerative methods for integer optimization with theoretical guarantees use a non-binary search tree with general disjunctions based on lattice structure. These disjunctions are expensive to compute and challenging to implement. Here we compare two lattice reformulations that can be used to heuristically obtain general disjunctions in the original space, we develop a new lattice-based variant, and compare the derived disjunctions computationally with those produced by the algorithm of Lovász and Scarf.

1. Introduction

We consider the following linear integer set:

$$X = \{\boldsymbol{x} \in \mathbb{Z}^n \mid \boldsymbol{A}\boldsymbol{x} \leqq \boldsymbol{a}_0\}, \tag{1}$$

where $\leqq$ symbolizes $\leq$ or $=$ constraints.

The standard way of optimizing over X is by branch-and-bound, which constructs a binary search tree. Each node of the tree is either pruned or *branched* on to create two children using single-variable disjunctions. Determining which variable to branch on is commonly done using sophisticated heuristic decision rules [3] and has recently received a lot of attention from the machine learning community [8,18]. There is no upper bound on the size of the branch-and-bound search tree for (1) that depends only on n.

The enumerative algorithms by Lenstra [12] and by Lovász and Scarf [15], or improvements thereof, have such an upper bound, but use a different concept of branching: "branching on hyperplanes". Both algorithms build a search tree where branches are created by imposing a new constraint $\boldsymbol{d}\boldsymbol{x} = \beta$, for all β in a range that is bounded, in the worst case, by a constant that depends only on n. This generates subproblems of lower dimension, leading to a search tree that has depth at most n. The direction $\boldsymbol{d}$ is determined in each subproblem by a lattice basis reduction algorithm. Even though the theoretical results for the Lenstra, and Lovász-Scarf algorithms are very appealing, little is known of their actual computational power. The only computational study we are aware of is by Cook et al. [4], where encouraging observations were made. The use of general disjunctions is also studied in, e.g., [2,6,9,10,16,22].

In [2] and [10], reformulations of the feasible set X are suggested. In both cases, a variable in the new coordinate system can be expressed as a function of the original variables, so branching on it can be viewed as a general disjunction in the original space. The approach of Aardal et al. [2] combines a reformulation of the problem as a full-dimensional problem with a shape changing step in the new coordinate system, whereas Krishnamoorthy and Pataki [10] only use a shape changing procedure.

In the present work, we compare the two approaches, and in particular investigate the computational effect of the dimension reduction and the shape change. In addition, we suggest a new norm that, for single-row problems, mimics a step of Lenstra's algorithm and yields a variant of the reformulation in [2]. The goal here is to make the resulting polytope regular so that it can be inscribed in a box of relatively small volume. An explicit upper bound on this volume is derived. We also look at the lattice bases produced by the Aardal et al. and the Krishnamoorthy-Pataki reformulations and compare them to the Lovász-Scarf reduced bases, which in a sense can be viewed as optimal.

Some of the test instances we consider are 0-1 integer programs. For such instances the worst-case branch-and-bound tree size is already polynomial for fixed n. There are aspects of branching on general disjunctions that may still be interesting for such instances. If, for instance, the solution space is embedded in an affine subspace and/or if the constraints are not combinatorial, then general disjunctions may result in smaller search trees in practice. This is addressed in our computational study.

The outline is as follows. In Section 2 we give some basic definitions and sketch the principles of the hyperplane branching algo-

* Corresponding author.
E-mail addresses: k.i.aardal@tudelft.nl (K. Aardal), L.V.ScavuzzoMontana@tudelft.nl (L. Scavuzzo), laurence.wolsey@uclouvain.be (L.A. Wolsey).

https://doi.org/10.1016/j.orl.2023.05.001

rithms by Lenstra and by Lovász and Scarf. In Section 3 we briefly review the reformulations by Aardal et al. and by Krishnamoorthy and Pataki, and introduce the new norm and volume bound for the single-row problems. Computations are presented in Section 4.

This paper is dedicated to the memory of our most valued colleague and friend Gerhard J. Woeginger.

2. Mathematical background

2.1. Lattices and basis reduction

Let $\boldsymbol{b}^1, \ldots, \boldsymbol{b}^l$ be linearly independent vectors. The set $L = \{\boldsymbol{x} \mid \boldsymbol{x} = \sum_{j=1}^{l} \lambda_j \boldsymbol{b}^j,\ \lambda_j \in \mathbb{Z},\ j = 1, \ldots, l\}$ is called a *lattice*. Setting $\boldsymbol{b}^j = \boldsymbol{e}^j$, where $\boldsymbol{e}^j$ is the jth unit vector, we obtain the so-called *standard lattice* $\mathbb{Z}^l$. Notice that a basis for a lattice is not unique, so even though $\boldsymbol{e}^j$ are obvious basis vectors for $\mathbb{Z}^l$, these basis vectors may not be the best choice for representing the lattice.

The direction $\boldsymbol{d}$ that is used in both the Lenstra and Lovász-Scarf algorithms to branch on hyperplanes $\boldsymbol{d}\boldsymbol{x} = \beta$, is determined by lattice basis reduction. Lenstra uses the polynomial-time LLL-reduction [11], and Lovász and Scarf use the generalized basis reduction algorithm [15], which runs in polynomial time in fixed dimension.

Definition 1. (LLL-reduced basis, [11]). Given is a basis $\boldsymbol{b}^1, \boldsymbol{b}^2, \ldots, \boldsymbol{b}^l$ of the lattice L, and the associated Gram-Schmidt vectors $\boldsymbol{b}^{1*}, \boldsymbol{b}^{2*}, \ldots, \boldsymbol{b}^{l*}$. Let $\mu_{ij} = \langle \boldsymbol{b}^i, \boldsymbol{b}^{j*} \rangle / \langle \boldsymbol{b}^{j*}, \boldsymbol{b}^{j*} \rangle$, for $1 \le j < i \le l$. The basis is *reduced* if

$$|\mu_{ij}| \le \frac{1}{2}, \text{ for } 1 \le j < i \le l \tag{2}$$

and, for $y \in (\frac{1}{4}, 1)$ and $1 < i \le l$,

$$\|\boldsymbol{b}^{i*} + \mu_{i,i-1}\boldsymbol{b}^{(i-1)*}\|^2 \ge y\, \|\boldsymbol{b}^{(i-1)*}\|^2 . \tag{3}$$

Notice that condition (3) is satisfied if $\|\boldsymbol{b}^{(i-1)*}\|^2 \le c\|\boldsymbol{b}^{i*}\|^2$, where $1/c = (y - \mu_{i,i-1}^2)$ with $y \in (1/4,\ 1)$ and $\mu_{i,i-1}^2 \le 1/4$. A *c-reduced basis* is an LLL-reduced basis for a given value of the constant c. This will be used in Schoof's Lemma in Section 3.1. For a description of the algorithm we refer to [11].

Lovász and Scarf [15] introduced the *generalized basis reduction algorithm*. In this algorithm a different norm, or distance function, from the Euclidean norm is used. In the case of integer programming, the distance function is related to the polyhedron of the underlying linear relaxation.

Let C be a compact convex body in $\mathbb{R}^n$ of positive volume and symmetric about the origin. The distance function $F(\boldsymbol{x})$ is defined as

$$F(\boldsymbol{x}) = \inf\{\lambda \ge 0 \mid \frac{\boldsymbol{x}}{\lambda} \in C\} . \tag{4}$$

Associated with the body C is its polar body C^* defined as $C^* = \{\boldsymbol{y} \in \mathbb{R}^n \mid \boldsymbol{y}^\top \boldsymbol{x} \le 1,\ \text{for all } \boldsymbol{x} \in C\}$. We can use C^* to define the polar formulation of the distance function: $F(\boldsymbol{y}) = \max_{\boldsymbol{x} \in C^*} \boldsymbol{y}^\top \boldsymbol{x}$.

We can now use the distance function (4) when defining a reduced basis in the sense of Lovász and Scarf. Let $\boldsymbol{b}^1, \ldots, \boldsymbol{b}^n$ be a basis for the lattice L. Instead of using the Gram-Schmidt vectors, Lovász and Scarf use projections of the basis vectors. Let C_i be the projection of C along the vectors $\boldsymbol{b}^1, \ldots, \boldsymbol{b}^{i-1}$ into the affine space $E_i = \sum_{j=i}^{n} \mathbb{R}\boldsymbol{b}^j$. The lattice L_i is obtained by projecting L along $\boldsymbol{b}^1, \ldots, \boldsymbol{b}^{i-1}$ into $\sum_{j=i}^{n} \mathbb{Z}\boldsymbol{b}^j$.

Definition 2. The distance function $F_i(\boldsymbol{x})$ associated with C_i is defined for $\boldsymbol{x} \in E_i$ by $F_i(\boldsymbol{x}) = \min_{\alpha_1, \ldots, \alpha_{i-1}} F(\boldsymbol{x} + \alpha_1 \boldsymbol{b}^1 + \cdots + \alpha_{i-1}\boldsymbol{b}^{i-1})$.

Using the polar formulation of F one can show that the value $F_i(\boldsymbol{x})$ is equal to

$$\max_{\boldsymbol{z} \in C^*} \{\boldsymbol{x}^\top \boldsymbol{z} \mid\ (\boldsymbol{b}^1)^\top \boldsymbol{z} = 0, \ldots, (\boldsymbol{b}^{i-1})^\top \boldsymbol{z} = 0\} . \tag{5}$$

Definition 3. (LS-reduced basis, [15]). Given a lattice L with basis $\boldsymbol{b}^1, \ldots, \boldsymbol{b}^n$, fix $0 < y < 1/2$. The basis is reduced for $i = 1, \ldots, n-1$, if

$$F_i(\boldsymbol{b}^{i+1}) \ge (1 - y) F_i(\boldsymbol{b}^i),$$
$$F_i(\boldsymbol{b}^{i+1} + \mu \boldsymbol{b}^i) \ge F_i(\boldsymbol{b}^{i+1}),\ \text{for all } \mu \in \mathbb{Z} .$$

We refer to Cook et al. [4] for a detailed description of an implementation of the Lovász-Scarf algorithm.

2.2. Determining the direction $\boldsymbol{d}$ when branching on hyperplanes

Since the norms used by Lenstra and by Lovász and Scarf are different, they also relate differently to the LP-relaxation polytope $P = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{a}_0\}$. Lenstra transforms the polytope and the lattice such that the transformed polytope τP can be approximated well by concentric balls, one inscribed in τP and one containing τP. LLL-reduction is now applied to a basis of the transformed lattice $\tau \mathbb{Z}^n$, and the direction $\boldsymbol{d}$ is chosen as the longest of the reduced basis vectors.

To understand how the Lovász-Scarf norm relates to the polytope we first consider a well-known fact from lattice theory. Let $L \subset E$ be a lattice of full rank in a Euclidean vector space E. Then the set $L^* = \{\boldsymbol{x} \in E \mid \boldsymbol{x}^\top \boldsymbol{y} \in \mathbb{Z},\ \text{for all } \boldsymbol{y} \in L\}$ is also a lattice in E of full rank, called the polar lattice. For $\boldsymbol{x} \in E - \{\boldsymbol{0}\}$ we have, by definition, that $\boldsymbol{x} \in L^*$ if and only if the lattice L is contained in $\{\boldsymbol{y} \in E \mid \boldsymbol{x}^\top \boldsymbol{y} \in \mathbb{Z}\}$, which can be written as

$$\{\boldsymbol{y} \in E \mid \boldsymbol{x}^\top \boldsymbol{y} \in \mathbb{Z}\} = (\mathbb{R}\boldsymbol{x})^\perp + k\boldsymbol{x}' , \tag{6}$$

where $(\mathbb{R}\boldsymbol{x})^\perp$ is the orthogonal complement of the space spanned by $\boldsymbol{x}$, $k \in \mathbb{Z}$, and $\boldsymbol{x}' = \boldsymbol{x}/\|\boldsymbol{x}\|^2$. So (6) implies that $\boldsymbol{x} \in L^*$ if and only if L is contained in the translates $(\mathbb{R}\boldsymbol{x})^\perp + k\boldsymbol{x}'$. If the vector $\boldsymbol{x}$ is short, the vector $\boldsymbol{x}'$ is long, so finding a short vector in the polar lattice L^* is equivalent to finding widely spaced parallel "lattice hyperplanes".

To determine whether the full-dimensional polytope $P = \{\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{a}_0\}$ contains an integer vector, we observe that the sets $(P - P)$ and $(P - P)^*$ are compact, convex sets that are symmetric about the origin, and have positive volume. The algorithm finds an approximation of the shortest vector given the distance function (4) for the set $C = (P - P)^*$, which is then the same as finding a vector $\boldsymbol{d}$ such that the lattice is contained in widely spaced translates of the orthogonal complement of $\boldsymbol{d}$, or equivalently, a direction $\boldsymbol{d}$ in which the width of P is thin. In this polyhedral case the distance function $F_i(\boldsymbol{d})$ using its dual representation (5) is now written as $F_i(\boldsymbol{d}) = \max\{\boldsymbol{d}(\boldsymbol{x} - \boldsymbol{y}) \mid \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{a}_0,\ \boldsymbol{A}\boldsymbol{y} \le \boldsymbol{a}_0,\ (\boldsymbol{b}^1)^\top(\boldsymbol{x} - \boldsymbol{y}) = 0, \ldots, (\boldsymbol{b}^{i-1})^\top(\boldsymbol{x} - \boldsymbol{y}) = 0\}$.

The question is whether the approximation by balls combined with polynomial basis reduction is computationally more demanding than computing the generalized reduced basis. To our knowledge, no implementation of Lenstra's algorithm has been reported on, whereas the Lovász-Scarf algorithm has been implemented and successfully tested by Cook et al. [4]. This study dates back to 1993. An implementation advantage of the Lovász-Scarf algorithm is that the main "engine" of the reduction algorithm in the polyhedral case is linear programming.

3. Lattice-based reformulations

Some of the lattice reformulations that we describe in this section reformulate using the lattice of the null-space of the constraint matrix $\boldsymbol{A}$. The set (1) in this case is assumed to have the following form:

$$X = \{\boldsymbol{x} \in \mathbb{Z}^n \mid \boldsymbol{A}\boldsymbol{x} = \boldsymbol{a}_0,\ \boldsymbol{x} \geq 0\}. \tag{7}$$

We assume that $\boldsymbol{A} \in \mathbb{Z}^{m\times n}$ has full row rank. The linear relaxation of X is denoted by X_{LP}.

It is well-known that if the system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{a}_0$ is integer feasible, then there exist integer vectors $\bar{\boldsymbol{x}},\ \boldsymbol{x}^j,\ j = 1,\dots,n-m$, such that $\{\boldsymbol{x} \in \mathbb{Z}^n \mid \boldsymbol{A}\boldsymbol{x} = \boldsymbol{a}_0\} = \bar{\boldsymbol{x}} + \sum_{j=1}^{n-m} \lambda_j \boldsymbol{x}^j$, where $\boldsymbol{x}^j,\ j = 1,\dots,n-m$, are linearly independent. In particular, we can take $\boldsymbol{x}^j,\ j = 1,\dots,n-m$, to be basis vectors of the lattice $\ker_{\mathbb{Z}} \boldsymbol{A} = \{\boldsymbol{x} \in \mathbb{Z}^n \mid \boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}\}$, that is, we write $\boldsymbol{x}^j = \boldsymbol{b}^j,\ j = 1,\dots,n-m$, where $\boldsymbol{B} = [\boldsymbol{b}^j]_{j=1}^{n-m}$ can be any basis for $\ker_{\mathbb{Z}} \boldsymbol{A}$. The vectors $\bar{\boldsymbol{x}},\ \boldsymbol{b}^j,\ j = 1,\dots,n-m$, can be derived in polynomial time by for instance a Hermite Normal Form computation, see Schrijver [19].

Once we have $\bar{\boldsymbol{x}},\ \boldsymbol{b}^j,\ j = 1,\dots,n-m$, we can reformulate (7) as

$$X^{\boldsymbol{\lambda}} = \{\boldsymbol{\lambda} \in \mathbb{Z}^{n-m} \mid \boldsymbol{B}\boldsymbol{\lambda} \geq -\bar{\boldsymbol{x}}\}. \tag{8}$$

We denote the linear relaxation of $X^{\boldsymbol{\lambda}}$ by $X^{\boldsymbol{\lambda}}_{LP}$. Notice that the sets $X^{\boldsymbol{\lambda}}$ and $X^{\boldsymbol{\lambda}}_{LP}$ are full-dimensional. It is clear that upper bounds $\boldsymbol{u}$ on the variables $\boldsymbol{x}$ can be handled by adding the constraints $\boldsymbol{B}\boldsymbol{\lambda} \leq \boldsymbol{u} - \bar{\boldsymbol{x}}$.

For all reformulations in this section a variable disjunction in the reformulated space corresponds to a general disjunction in the original space.

3.1. Single-row reformulations

Here we consider the integer knapsack case, i.e., the matrix $\boldsymbol{A}$ in formulation (7) of X has one row $\boldsymbol{a}$ ($m = 1$). Furthermore, we assume that $a_0,\ a_1,\dots,a_n > 0$, and that $\gcd(a_1,\dots,a_n) = 1$.

We will study the volume of a rectangular box, oriented in the coordinate directions, that contains the linear relaxation of the problem at hand. This volume gives us an upper bound on the number of branch-and-bound nodes needed to solve the instance when branching along the coordinate directions. It is straightforward to determine the volume of the smallest such box containing X_{LP}, which is

$$V(X_{LP}) = \frac{a_0^n}{\Pi_{i=1}^n a_i}. \tag{9}$$

In this section we derive an expression for an upper bound on the volume of a box containing $X^{\boldsymbol{\lambda}}_{LP}$, denoted $V(X^{\boldsymbol{\lambda}}_{LP})$ by choosing an appropriate basis $\boldsymbol{B}$ of the lattice $\ker_{\mathbb{Z}} \boldsymbol{a}$. The idea comes from the approximation step in Lenstra's algorithm [12], where an appropriate polytope transformation is derived by mapping a high-volume simplex, which is contained in the polytope, to a regular simplex.

For our proof we will use a linear map $\boldsymbol{D}$ of $\mathbb{R}^n$ that maps X_{LP} to a regular simplex, $\boldsymbol{D} = \boldsymbol{a}\boldsymbol{I}^n$. Applying this map to $X_{LP} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \sum_{i=1}^n a_i x_i = a_0\}$ yields a simplex that intersects each of the coordinate axes at the point $a_0\boldsymbol{e}^i,\ i = 1,\dots,n$: $X^D_{LP} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \sum_{i=1}^n x_i = a_0\}$.

Applying $\boldsymbol{D}$ to $\mathbb{Z}^n$ yields the lattice $\Lambda = \boldsymbol{D}\mathbb{Z}^n$ with basis $\boldsymbol{D}$. Under this map, the lattice $\ker_{\mathbb{Z}} \boldsymbol{a}$ becomes $\boldsymbol{D}\ker_{\mathbb{Z}} \boldsymbol{a} = \{\boldsymbol{x} \in \Lambda \mid \sum_{i=1}^n x_i = 0\} = \ker_{\Lambda} \boldsymbol{1}$.

Notice that if $\boldsymbol{B} = (\boldsymbol{b}^1,\dots,\boldsymbol{b}^{n-1})$ is a basis for $\ker_{\mathbb{Z}} \boldsymbol{a}$, then the vectors $\hat{\boldsymbol{b}}^i = \boldsymbol{D}\boldsymbol{b}^i,\ i = 1,\dots,n-1$, form a basis for $\ker_{\Lambda} \boldsymbol{1}$. Similarly, if $\bar{\boldsymbol{x}}$ is a vector satisfying $\boldsymbol{a}\bar{\boldsymbol{x}} = a_0$, then the vector $\hat{\boldsymbol{x}} = \boldsymbol{D}\bar{\boldsymbol{x}}$ satisfies $\sum_{i=1}^n \hat{x}_i = a_0$.

Lemma 1 *(Schoof, see Section 12 of [13]). Given is a lattice L of rank n generated by the basis $\boldsymbol{B} = (\boldsymbol{b}^1,\dots,\boldsymbol{b}^n)$, and a vector $\boldsymbol{x} \in \mathbb{R}^n$. Assume that $\boldsymbol{B}$ is a c-reduced basis for L, with $c \geq 1$, and let $\boldsymbol{b}^{1*},\dots,\boldsymbol{b}^{n*}$ be the Gram-Schmidt vectors corresponding to $\boldsymbol{B}$. Let $\lambda_1,\dots,\lambda_n$ be the coefficients in the unique expression of $\boldsymbol{x}$ in terms of the basis vectors $\boldsymbol{b}^j$, i.e., $\boldsymbol{x} = \sum_{j=1}^n \lambda_j \boldsymbol{b}^j$. Then,*

$$|\lambda_j| \leq \left(\frac{3\sqrt{c}}{2}\right)^{n-j} \cdot \frac{\|\boldsymbol{x}\|}{\|\boldsymbol{b}^{j*}\|} \quad \textit{for } j = 1,\dots,n.$$

We are now ready to state our result.

Theorem 2. *Given $a_0,\ \boldsymbol{a} = (a_1,\dots,a_n)$, a basis $\hat{\boldsymbol{B}}$ of $\ker_{\Lambda} \boldsymbol{1}$, and a vector $\bar{\boldsymbol{x}} \in \mathbb{Z}^n$ satisfying $\boldsymbol{a}\bar{\boldsymbol{x}} = a_0$, assume that*

- *$a_0,\ a_1,\dots,a_n > 0$,*
- *$gcd(a_1,\dots,a_n) = 1$,*
- *$\hat{\boldsymbol{B}}$ is c-reduced with $c \geq 1$.*

Then, $X^{\boldsymbol{\lambda}}_{LP} = \{\boldsymbol{\lambda} \in \mathbb{Z}^{n-1} \mid -\boldsymbol{B}\boldsymbol{\lambda} \leq \bar{\boldsymbol{x}}\}$, with $\boldsymbol{B} = \boldsymbol{D}^{-1}\hat{\boldsymbol{B}}$, is contained in a box with volume at most

$$\left(\frac{3\sqrt{c}}{2}\right)^{\frac{(n-1)(n-2)}{2}} \cdot \frac{2^{\frac{n-1}{2}}}{\sqrt{n}} \cdot \frac{a_0^{n-1}}{\prod_{i=1}^n a_i}. \tag{10}$$

Proof. Let $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ be two arbitrary vectors in X_{LP}, and consider their representation in terms of $\boldsymbol{B}$, namely $\boldsymbol{x}^{(1)} = \bar{\boldsymbol{x}} + \sum_{j=1}^{n-1} \lambda_j^{(1)} \boldsymbol{b}^j$, $\boldsymbol{x}^{(2)} = \bar{\boldsymbol{x}} + \sum_{j=1}^{n-1} \lambda_j^{(2)} \boldsymbol{b}^j$. We can apply Schoof's lemma to the vector

$$\boldsymbol{D}(\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(2)}) = \sum_{j=1}^{n-1} (\lambda_j^{(1)} - \lambda_j^{(2)}) \boldsymbol{D}\boldsymbol{b}^j = \sum_{j=1}^{n-1} (\lambda_j^{(1)} - \lambda_j^{(2)}) \hat{\boldsymbol{b}}^j.$$

This yields, for $j = 1,\dots,n-1$,

$$|\lambda_j^{(1)} - \lambda_j^{(2)}| \leq \left(\frac{3\sqrt{c}}{2}\right)^{n-1-j} \cdot \frac{\|\boldsymbol{D}(\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(2)})\|}{\|\hat{\boldsymbol{b}}^{j*}\|}.$$

Here $\hat{\boldsymbol{b}}^{j*},\ j = 1,\dots,n-1$, are the Gram-Schmidt vectors associated with the basis $\hat{\boldsymbol{B}}$. Notice that the vectors $\boldsymbol{D}\boldsymbol{x}^{(1)}$ and $\boldsymbol{D}\boldsymbol{x}^{(2)}$ are in the regular simplex X^D_{LP} and therefore $||\boldsymbol{D}\boldsymbol{x}^{(1)} - \boldsymbol{D}\boldsymbol{x}^{(2)}|| \leq \sqrt{2} \cdot a_0$. Using this and the fact that $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ were chosen arbitrarily, we can obtain an upper bound for the volume of $X^{\boldsymbol{\lambda}}_{LP}$

$$V(X^{\boldsymbol{\lambda}}_{LP}) \leq \prod_{j=1}^{n-1} \left(\frac{3\sqrt{c}}{2}\right)^{n-1-j} \cdot \frac{\sqrt{2} \cdot a_0}{\|\hat{\boldsymbol{b}}^*_j\|} = \left(\frac{3\sqrt{c}}{2}\right)^{\frac{(n-1)(n-2)}{2}} \cdot 2^{\frac{n-1}{2}} \cdot \frac{a_0^{n-1}}{\prod_{j=1}^{n-1} \|\hat{\boldsymbol{b}}^*_j\|}. \tag{11}$$

It is well-known that $\prod_{j=1}^{n-1} \|\hat{\boldsymbol{b}}^*_j\|$ is an expression for $d(\ker_{\Lambda} \boldsymbol{1})$, so we only need to obtain an expression for $d(\ker_{\Lambda} \boldsymbol{1})$ in terms of the input.

The lattice $\ker_{\Lambda} \boldsymbol{1}$ is a pure sublattice of Λ. From known lattice formulae (see (14), (15), and (16) in the appendix) we obtain

$$d(\ker_{\Lambda} \boldsymbol{1}) = \frac{d(\Lambda)}{d(\Lambda/\ker_{\Lambda} \boldsymbol{1})} = d(\Lambda) \cdot d((\Lambda/\ker_{\Lambda} \boldsymbol{1})^*) = d(\Lambda) \cdot d((\ker_{\Lambda} \boldsymbol{1})^{\perp}) \tag{12}$$

with $(\ker_\Lambda \mathbf{1})^\perp = \{\boldsymbol{x} \in \Lambda^* \mid \langle \boldsymbol{x}, \ker_\Lambda \mathbf{1}\rangle = 0\}$. The determinant of the lattice Λ is equal to $d(\Lambda) = \det(\boldsymbol{D}) = \prod_{j=1}^n a_j$.

Since $\gcd(a_1, \ldots, a_n) = 1$, the lattice $(\ker_\Lambda \mathbf{1})^\perp$ is the lattice generated by the vector of all ones, i.e., $(\ker_\Lambda \mathbf{1})^\perp = \mathbb{Z}\mathbf{1}$ having determinant equal to $d((\ker_\mathbb{Z} \mathbf{1})^\perp) = \langle \mathbf{1}, \mathbf{1}\rangle^{1/2} = \sqrt{n}$. Expression (12) has now become:

$$d(\ker_\Lambda \mathbf{1}) = d(\Lambda) \cdot d((\ker_\Lambda \mathbf{1})^\perp) = \prod_{j=1}^n a_j \cdot \sqrt{n}\,.$$

We now substitute $\prod_{j=1}^{n-1} \|\hat{\boldsymbol{b}}_j^*\|$ in Expression (11) for $\prod_{j=1}^n a_j \cdot \sqrt{n}$ and obtain the desired result. □

If we compare expressions (9) and (10) we can make the following remarks. The first factor in the bound on $V(X_{LP}^{\boldsymbol{\lambda}})$ is due to the estimate that one gets from the LLL-reduction (see the proof given in [13]). In practice, the LLL-reduction estimates typically turn out much better than the theoretical bounds. In the last factor of the bound we have lost one power of the right-hand side a_0. It is not so clear how this gain compares with the first two factors. We investigate this in the computational study presented in Section 4. Another question is how the basis used in the proof of Theorem 2 compares computationally to the lattice reformulations of Aardal et al. [2] and of Krishnamoorthy and Pataki [10], which we describe next.

3.2. The Aardal-Hurkens-Lenstra and the Krishnamoorthy-Pataki reformulations

Aardal et al. [2] use LLL reduction to derive the reformulation (8) of the set $\{\boldsymbol{x} \in \mathbb{Z}^n \mid \boldsymbol{A}\boldsymbol{x} = \boldsymbol{a}_0\}$. They do so by considering a higher-dimensional lattice in which the vectors $\bar{\boldsymbol{x}},\ \boldsymbol{x}^j,\ j = 1, \ldots n-m$, as described in the introduction to Section 3, are short and such that finding an initial basis for that lattice is trivial. The LLL algorithm then outputs vectors $\bar{\boldsymbol{x}},\ \boldsymbol{x}^j,\ j = 1, \ldots n-m$ or gives a certificate for integer infeasibility in polynomial time. We refer to this reformulation as the AHL-reformulation.

The starting point of Krishnamoorthy and Pataki (KP) [10] is the set $X_\leq = \{\boldsymbol{x} \in \mathbb{Z}^n \mid \boldsymbol{l} \leq \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{u}\}$.

They reformulate the problem by reducing $\boldsymbol{A}$, i.e. multiplying $\boldsymbol{A}$ by a unimodular matrix $\boldsymbol{U}$ to obtain

$$X_\leq^{\boldsymbol{y}} = \{\boldsymbol{y} \in \mathbb{Z}^n \mid \boldsymbol{l} \leq \boldsymbol{A}\boldsymbol{U}\boldsymbol{y} \leq \boldsymbol{u}\}\,. \tag{13}$$

The KP-reformulation can of course also be applied to an equality system, but it does not result in a dimension reduction as in the AHL-reformulation.

It is worth noticing that if the original set is full-dimensional, then the resulting KP-reformulation is equivalent to the AHL-reformulation, see the appendix. In practice there are differences, as the AHL-reformulation involves the extended matrix $(\boldsymbol{A}, \boldsymbol{a}_0)$ instead of only $\boldsymbol{A}$. When using the KP reformulation for equations, we use $\boldsymbol{A}\boldsymbol{U}\boldsymbol{y} = \boldsymbol{a}_0$.

4. Computational study

4.1. Instances and setup

We perform the comparison using 7 sets of instances from the literature. A summary of their properties and a description of the models can be found in the appendix. Four of the collections are composed of single-row feasibility problems and three are multi-row binary problems. The single-row instances struct_s and struct_b have been generated such that the lattice $\ker_\mathbb{Z} \boldsymbol{a}$ has an $(n-2)$-dimensional sublattice with small determinant d, whereas $d(\ker_\mathbb{Z} \boldsymbol{a})$ is large. This implies that LLL-reduction will yield a basis with $n-2$ relatively short vectors and one long vector. In contrast, nostruct_s and nostruct_b are composed of instances with coefficients of the same order of magnitude as struct_s and struct_b, yielding a lattice determinant of the same order of magnitude, but with no special encoded structure. For all single-row instances, the right-hand side coefficient is the Frobenius number [17], which makes them infeasible. Market split (MS) instances have been shown to benefit from the AHL reformulation in past work, see [2]. Generalized assignment problems (GAP) and combinatorial auctions (CA) have not been tested in this context before.

Table 1
Geometric mean of the number of nodes over 30 instances and 5 randomization seeds (lower is better). We compare the original formulation with the four proposed reformulations.

Instance	Original	AHL	AHL$^{\boldsymbol{D}}$	AHL^{low}	KP
struct_s	$> 10^7$	12.70	10.81	12.63	19.19
struct_b	$> 10^7$	1.28	1.27	1.31	1.21
nostruct_s	121,456	97.10	68.64	90.43	128.98
nostruct_b	59,901	701.15	572.03	1779.06	566.99
MS	398,742	685.03	-	4111.48	1545.95
GAP	893.37	52.15	-	72.52	122.92
CA	11.26	21.03	-	95.08	16.14

Table 2
Geometric mean of the solving time (in seconds) over 30 instances and 5 randomization seeds (lower is better). Reduction time is not included, but we report it in Table 3.

Instance	Original	AHL	AHL$^{\boldsymbol{D}}$	AHL^{low}	KP
struct_s	> 3600	0.04	0.03	0.04	0.04
struct_b	> 3600	0.06	0.05	0.06	0.07
nostruct_s	21.74	0.12	0.10	0.11	0.10
nostruct_b	6.62	0.61	0.45	1.02	0.53
MS	37.98	0.81	-	2.40	1.64
GAP	3.77	1.05	-	1.11	1.39
CA	3.41	4.68	-	28.23	3.99

Table 3
Average reduction times (in seconds) of the multi-row instances. The reduction time of single-row instances is negligible.

Instance	AHL	AHL^{low}	KP
Market split	$2 \cdot 10^{-3}$	$3 \cdot 10^{-3}$	$2 \cdot 10^{-3}$
Generalized assignment problem	21.4	21.2	2.5
Combinatorial auctions	172.6	168.6	3.0

In our computations, LLL-reductions were performed using the NTL library [21] using a reduction coefficient $y = 0.99$ in all cases except AHL^{low} where we used $y = 0.3$. Each instance is solved 5 times with different randomization seeds. We use the solver SCIP [20] version 8.0.1 with default parameters. This means that we do not provide the solver with information about which variable to branch on in the reformulations, even though we have reason to believe that the last coordinate directions should be preferred. We set a time limit of one hour. Timeouts are indicated with the symbol '>'. We refer to the online supplement for extended results.

4.2. Experiments with single-row instances

We compare the original formulation with four reformulations: the AHL reformulation (AHL), AHL-reformulation with low quality reduction (AHL^{low}), the AHL reformulation with $\boldsymbol{B} = \boldsymbol{D}^{-1}\hat{\boldsymbol{B}}$ and $\hat{\boldsymbol{B}}$ a reduced basis of $\ker_\Lambda \mathbf{1}$ (AHL$^{\boldsymbol{D}}$), and the Krishnamoorthy-Pataki reformulation (KP).

We report the averaged number of branch-and-bound nodes and solving time in the first four rows of Table 1 and Table 2,

Table 4
Average logarithm of the volume of the smallest box, oriented in the coordinate directions, that contains the LP-relaxation of the respective reformulation. We calculate this by maximizing and minimizing the value of each variable over the LP relaxation. The last column shows the logarithm of the bound in (10), split into two terms: (i) the logarithm of the first factor and (ii) the logarithm of the remaining factors.

Instance	Original	AHL	AHL$^{\boldsymbol{D}}$	AHL^{low}	KP	Theorem 2
struct_s	62.35	50.84	46.37	50.21	53.57	20.0+50.0
struct_b	412.91	449.40	401.72	447.19	449.79	2697.2+464.0
nostruct_s	25.39	17.57	14.48	17.17	18.51	20.0+16.8
nostruct_b	56.25	85.65	77.05	76.35	86.39	2697.2+110.9

respectively. The results per instance can be found in Tables 8–11 in the online supplement. Each of the reformulations yields a remarkably easier problem. This is particularly apparent for struct_s and struct_b, which go from being unsolvable within the time limit to being remarkably easy. Notice that with higher dimension, the Frobenius number decreases quite substantially resulting in relatively easier instances.

For the structured instances, AHL$^{\boldsymbol{D}}$ gives the best results, both in terms of nodes and time. All reductions, however, work well, even AHL^{low}, so it does not pay off significantly to spend time on a high-quality reduction. Finally, we observe that the large structured instances become trivial for all reformulations, with most of them being solved without branching, see Table 10.

For the non-structured instances the results change depending on the instance size. For the smaller instances, AHL^{D} yields the smallest trees, followed by AHL^{low} and AHL. For the larger instances the results indicate that KP gives the lowest average number of nodes, but AHL$^{\boldsymbol{D}}$ is faster. We also observe that, in particular for the smaller instances, AHL^{low} yields sparser matrices than AHL, due to a smaller number of column operations on the basis matrix. Yet, they are both able to find the overall best direction (see Section 4.4). The computational results indicate that the solver is not always able to make use of the higher-quality basis provided by AHL and in fact a sparser constraint matrix can be beneficial.

Tables 1 and 2 suggest that the effect of the dimensionality reduction provided by all AHL variants is more pronounced for lower-dimensional problems, whereas in higher dimension the shape transformation is the main driver of improvements. This is also supported by the volume results reported in Table 4. Here we show the average logarithm of the volume of the smallest box, oriented in the coordinate directions, that contains the LP-relaxation of the respective reformulation. AHL^{D} yields the smallest volume for all but the larger non-structured instances. For these instances it is in fact the original formulation that has the smallest relaxation volume. This means that the improvements in Table 1 can only be a consequence of the transformed shape. Table 4 also shows that the bound in Theorem 2 is far from being tight, due to the constant arising from the worst-case performance of LLL.

4.3. Multi-row instances

We test the same formulations as in Section 4.2, except for AHL$^{\boldsymbol{D}}$, which is only valid for single-row problems. Results are shown in Tables 1 and 2, last three rows, see Tables 12–14 in the online supplement for results per instance.

The market split instances are non full-dimensional instances with a dense constraint matrix with integer entries between 0 and 100. All reformulations perform significantly better than the original formulation, with AHL being the best. This can be an indication that dimension-reduction plays a role. We can also conclude that for these instances the quality of the reduction matters.

The GAP instances, in contrast, are 0-1, full-dimensional, and part of the constraint matrix contains numbers that are different from 0 and 1. This may result in an LP-relaxation having a "non-regular" shape. Again we see that all reformulations perform better than the original formulation. This supports the idea that constraint branching can be beneficial even for (some types of) 0-1 instances. It is surprising that both AHL-reformulations perform much better than KP, since one would expect that AHL and KP would perform more or less equally well. The only difference between the two in practice is that AHL reduces $(\boldsymbol{A}, \boldsymbol{a}_0)$, whereas KP reduces only $\boldsymbol{A}$, see also Section 3.2 and the appendix.

Table 5
Is the last coordinate direction a short lattice vector in the Lovász-Scarf sense? (✓) yes, (✗) no and it is a combination of many variables. If the answer is no, but the short vector is a simple combination of variables, we indicate that combination. Notice that the last λ-index is $n-m$.

Instance	AHL	AHL$^{\boldsymbol{D}}$	AHL^{low}	KP
struct_s_1	✓	✓	✓	$19\lambda_n - 30\lambda_{n-1}$
struct_b_1	✓	✓	✓	$-9\lambda_n + 10\lambda_{n-1}$
nostruct_s_1	✓	✓	✓	✗
nostruct_b_1	✓	✓	✓	✗
MS_1	$\lambda_{n-m} - \lambda_{n-m-5}$	-	✓	✗
GAP_1	✓	-	✓	✓
CA_1	✓	-	✓	✓

The CA instances are full-dimensional and have 0-1 entries in the constraint matrix. Here we observe that the original formulation performs the best, even though both KP and AHL perform reasonably well in comparison. So, with no dimension reduction and a combinatorial problem structure there seems to be no gain in reformulating.

In Fig. 1 (online supplement) we present, for all instance types, an overview of the performance per instance over randomized seeds.

4.4. Comparison with Lovász-Scarf

The generalized basis reduction algorithm of Lovász and Scarf gives us an approximation of a direction in which the LP relaxation is thin, see Section 2.2. The reformulations we study can be regarded as a heuristic way of obtaining thin directions. In particular, by construction, the last coordinate direction should indicate such a direction. We run the generalized basis reduction algorithm on the polytope X^{λ}, where the polytope is generated by the reformulations AHL, AHL^{D}, AHL^{low} and KP, to investigate if indeed this last coordinate direction coincides with the direction given by the Lovász-Scarf (LS) algorithm. Results are shown in Table 5. A check mark indicates that the last coordinate direction coincides with the first LS-direction. A cross indicates that the two directions are far from coinciding, and the expressions involving λ_js indicate that the directions are close to coinciding and give the precise relation between the LS-direction and the output of the reformulations. These results show that AHL, in all its variants, is successful in finding such direction in many cases, and when it does not, it finds a close direction. In contrast, KP is in most cases unable to find the Lovász-Scarf direction, especially when the problem at hand is not full dimensional.

Table 6
Vanilla SCIP. We report the geometric mean of the number of nodes over 30 instances and 5 randomization seeds when deactivating presolve, cutting, conflict analysis and primal heuristics. We compare the original formulation with the four proposed reformulations.

Instance	Original	AHL	AHL$^{\boldsymbol{D}}$	AHL^{low}	KP
struct_s	$> 10^7$	37.70	22.83	32.67	82.96
struct_b	$> 10^7$	2.67	2.10	2.84	43.18
nostruct_s	141,377	197.3	129.9	193.5	293.1
nostruct_b	76,081	1,400	915.2	1,238	1,517
MS	1,041,742	989.5	-	6,493	2,101
GAP	$> 378,502$	308.0	-	354.6	$> 3,541$
CA	21.08	26.39	-	100.2	27.19

4.5. Computational experiments with vanilla SCIP

We repeat the experiment presented in Table 1 with a vanilla version of SCIP: presolve, cutting, conflict analysis and primal heuristics are deactivated. This allows us to isolate the effect of branching. These additional results, shown in Table 6, confirm the findings discussed in Section 4. Additionally, we see that the benefits of AHL$^{\boldsymbol{D}}$ and the dimensionality reduction in single-row instances are more apparent under these settings. Here, of all the reformulations, KP is the most affected by the lack of additional solver components.

5. Discussion

While the theoretical results of Lenstra [12] and Lovász and Scarf [15] tell us to branch on general disjunctions, most IP solvers implement single-variable branching schemes. Some reasons are that (i) theoretically strong disjunctions are computationally costly to find and difficult to implement into standard branch-and-bound schemes, and (ii) not all instance types benefit from using them. In this paper, we have studied problem reformulations for which the resulting variables can be viewed as hyperplanes in the original formulation. These methods can be seen as heuristic ways to obtain good branching disjunctions. Through our computational study, we observed that the reformulations studied can be effective in finding strong disjunctions, with substantial improvements in the case of non full-dimensional instances (even 0-1). The reformulations also yield better performance for 0-1 instances in which the constraint matrix is not combinatorial. This shows that general disjunctions can be useful in practice even in the 0-1 case, where single-variable disjunctions already provide theoretically "thin directions".

In our study, we solve the reformulated instances by standard variable branching, which is mathematically equivalent to branching on general disjunctions in the original space. However, in practice, other solver components have an impact on the performance. As an additional experiment to isolate the effect of branching we turned off presolve, cutting, heuristics and conflict analysis. These experiments support once again our findings that the reformulated variables provide good branching directions.

Overall, our results point to the great potential of the AHL and KP reformulations for tackling problems where single-variable branching fails. Still, applying basis reduction can be computationally expensive as the number of variables grows (see Table 3). Future directions of research can be to identify a subset of constraints to reformulate, and to test reformulations on a collection of instances with more diverse structures.

Data availability

Data will be made available on request.

Acknowledgements

This research is financed in part by The Netherlands Organisation for Scientific Research, NWO, Grant OCENW.GROOT.2019.015.

Appendix A

A.1. Additional background on lattices

The *determinant* of a full-dimensional lattice L, $d(L)$, is equal to $\det(\boldsymbol{B})$, where $\det(.)$ denotes the determinant, and where $\boldsymbol{B}$ is any basis for L. If L is not full-dimensional we can compute $d(L)$ as $d(L) = \sqrt{\det(\boldsymbol{B}^T\boldsymbol{B})}$, where $\boldsymbol{B}$ again is any basis for L.

Let L be a lattice in a Euclidean vector space E with $\dim E = \operatorname{rk} L$. Then the *polar lattice* L^* of L is defined as follows: $L^* = \{\boldsymbol{x} \in E : \langle \boldsymbol{x}, L\rangle \subset \mathbb{Z}\}$. For a lattice L and its polar L^* we have $\operatorname{rk} L = \operatorname{rk} L^*$, $L^{**} = L$, and

$$d(L) = \frac{1}{d(L^*)}\,. \tag{14}$$

Let L be a lattice in a Euclidean vector space E, and let K be a subgroup of L. If there exists a subspace D of E such that $K = L \cap D$, then K is called a *pure* sublattice. Suppose that K is a pure sublattice of the lattice L. Then, the following holds:

$$d(L) = d(K) \cdot d(L/K)\,. \tag{15}$$

Let L be a lattice with polar L^*, and let K be a pure sublattice of L. Then $K^\perp = \{\boldsymbol{x} \in L^* \mid \langle \boldsymbol{x}, K\rangle = 0\}$, and we can write

$$K^\perp = (L/K)^*\,. \tag{16}$$

A thorough treatment of this topic can be found in [13].

A.2. Equivalence between the AHL- and KP-reformulations in the full-dimensional case

We demonstrate this equivalence on a set with only upper bounds, but adding lower bounds follows easily. Consider the system $\boldsymbol{A}\boldsymbol{x} \le \boldsymbol{a}_0$. We add slack variables and obtain $\boldsymbol{A}\boldsymbol{x} + \boldsymbol{I}\boldsymbol{s} = \boldsymbol{a}_0$. The AHL-reformulation now needs, as a starting point, a vector $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{s}})^\mathsf{T}$ satisfying $\boldsymbol{A}\bar{\boldsymbol{x}} + \boldsymbol{I}\bar{\boldsymbol{s}} = \boldsymbol{a}_0$ and a basis for the lattice $\ker_{\mathbb{Z}}(\boldsymbol{A}\ \ \boldsymbol{I})$. We may choose $(\bar{\boldsymbol{x}},\ \bar{\boldsymbol{s}})^\mathsf{T} = (\boldsymbol{0},\ \boldsymbol{a}_0)$ and the lattice basis

$$\begin{pmatrix} \boldsymbol{I} \\ -\boldsymbol{A} \end{pmatrix}. \tag{17}$$

Reducing the basis (17) yields the AHL-reformulation, which is also precisely what Krishnamoorthy and Pataki do.

A.3. Instance models

This section presents a more in-depth description of the instances used in the computational study. For a summary of their characteristics see Table 7.

Table 7
Instance collections used in our computations.

Name	Number	n	m	Ref
struct_s	30	10	1	[1]
nostruct_s	30	10	1	[1]
struct_b	30	100	1	[1]
nostruct_b	30	100	1	[1]
MS	30	30	4	[5]
GAP	30	600	606	[7]
CA	30	500	100	[14]

Single-row instances

For a given number of variables n, find a vector $\boldsymbol{x} \in \mathbb{Z}^n_{\geq 0}$ that satisfies $\boldsymbol{a}\boldsymbol{x} = a_0$. The number a_0 is chosen to be the Frobenius number corresponding to $\boldsymbol{a}$. This number is computed following the procedure described in [1]. For the structured instances, $\boldsymbol{a}$ is generated as $\boldsymbol{a} = M\boldsymbol{p} + N\boldsymbol{r}$, with $M \in [10000, 20000]$, $N \in [1000, 2000]$, $\boldsymbol{p} \in [1, 10]^n$ and $\boldsymbol{r} \in [-10, 10]^n$ sampled uniformly. In the case of instances with no structure, the vector $\boldsymbol{a}$ is sampled uniformly in $[10000, 15000]^n$.

Market split (MS)

For a given number m of constraints and $n = 10 \cdot (m - 1)$ variables, we solve the feasibility problem of finding a vector in $\{\boldsymbol{x} \in \{0, 1\}^n | \boldsymbol{A}\boldsymbol{x} = \boldsymbol{a}_0\}$. We generate the coefficients of $\boldsymbol{A}$ uniformly at random in the range $[0, 99] \cap \mathbb{Z}$ and we set $(\boldsymbol{a}_0)_i = \frac{1}{2}\sum_{j=1}^n \boldsymbol{A}_{ij}$.

Generalized assignment problem (GAP)

Given n items with respective prices $\{p_j\}_{j=1}^n$ and weights $\{w_j\}_{j=1}^n$, and m knapsacks with capacities $\{c_i\}_{i=1}^m$, the generalized assignment problem consists in placing a number of items in each of the knapsacks such that the price of the selected items is maximized, while the capacity of the knapsacks is not exceeded by the total weight of the items therein. Formally:

$$\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^m \sum_{j=1}^n p_j x_{ij} \\
\text{subject to} \quad & \sum_{j=1}^n w_j x_{ij} \leq c_i, \quad i = 1, \ldots, m \\
& \sum_{i=1}^m x_{ij} \leq 1, \quad j = 1, \ldots, n \\
& x_{ij} \in \{0, 1\} \ \forall i, j
\end{aligned}$$

where each variable x_{ij} represents the decision of placing item j in knapsack i.

Combinatorial auctions (CA)

For m items, we are given n bids $\{\mathcal{B}_j\}_{j=1}^n$. Each bid $\mathcal{B}_j$ is a subset of the items with an associated bidding price p_j. The associated combinatorial auction problem is of the following form:

$$\begin{aligned}
\text{maximize} \quad & \sum_{j=1}^n p_j x_j \\
\text{subject to} \quad & \sum_{j: i \in \mathcal{B}_j} x_j \leq 1, \quad i = 1, \ldots, m \\
& x_j \in \{0, 1\} \ j = 1, \ldots, n
\end{aligned}$$

where x_j represents the action of choosing bid $\mathcal{B}_j$.

Appendix B. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.orl.2023.05.001.

References

[1] K. Aardal, A.K. Lenstra, Hard equality constrained integer knapsacks, Math. Oper. Res. 29 (3) (2004) 724–738. Erratum: Math. Oper. Res. 31 (4) (2006) 846.
[2] K. Aardal, C.A.J. Hurkens, A.K. Lenstra, Solving a system of linear Diophantine equations with lower and upper bounds on the variables, Math. Oper. Res. 25 (3) (2000) 427–442.
[3] T. Achterberg, T. Koch, A. Martin, Branching rules revisited, Oper. Res. Lett. 33 (1) (2005) 42–54.
[4] W. Cook, T. Rutherford, H.E. Scarf, D. Shallcross, An implementation of the generalized basis reduction algorithm for integer programming, ORSA J. Comput. 5 (2) (1993) 206–212.
[5] G. Cornuéjols, M. Dawande, A class of hard small 0-1 programs, INFORMS J. Comput. 11 (2) (1999) 205–210.
[6] S. Elhedhli, J. Naoum-Sawaya, Improved branching disjunctions for branch-and-bound: an analytic center approach, Eur. J. Oper. Res. 247 (2015) 37–45.
[7] A.S. Fukunaga, A branch-and-bound algorithm for hard multiple knapsack problems, Ann. Oper. Res. 184 (1) (2011) 97–119.
[8] M. Gasse, D. Chételat, N. Ferroni, L. Charlin, A. Lodi, Exact combinatorial optimization with graph convolutional neural networks, NeurIPS 32 (2019).
[9] M. Karamanov, G. Cornuéjols, Branching on general disjunctions, Math. Program. 128 (2011) 403–436.
[10] B. Krishnamoorthy, G. Pataki, Column basis reduction and decomposable knapsack problems, Discrete Optim. 6 (3) (2009) 242–270.
[11] A.K. Lenstra, H.W. Lenstra Jr., L. Lovász, Factoring polynomials with rational coefficients, Math. Ann. 261 (4) (1982) 515–534.
[12] H.W. Lenstra Jr., Integer programming with a fixed number of variables, Math. Oper. Res. 8 (4) (1983) 538–548.
[13] H.W. Lenstra Jr., Lattices, in: Algorithmic Number Theory: Lattices, Number Fields, Curves and Cryptography, in: Math. Sci. Res. Inst. Publ., vol. 44, Cambridge Univ. Press, Cambridge, 2008, pp. 127–181.
[14] K. Leyton-Brown, M. Pearson, Y. Shoham, Towards a universal test suite for combinatorial auction algorithms, in: Proceedings of the 2nd ACM Conference on Electronic Commerce, 2000, pp. 66–76.
[15] L. Lovász, H.E. Scarf, The generalized basis reduction algorithm, Math. Oper. Res. 17 (3) (1992) 751–764.
[16] S. Mehrotra, Z. Li, Branching on hyperplane methods for mixed integer linear and convex programming using adjoint lattices, J. Glob. Optim. 49 (4) (2011) 623–649.
[17] J.L. Ramírez Alfonsín, The Diophantine Frobenius Problem, OUP, Oxford, 2005.
[18] L. Scavuzzo, F.Y. Chen, D. Chételat, M. Gasse, A. Lodi, N. Yorke-Smith, K. Aardal, Learning to branch with tree mdps, NeurIPS 35 (2022).
[19] A. Schrijver, Theory of Linear and Integer Programming, Wiley-Interscience Series in Discrete Mathematics and Optimization, Chichester, UK, 1986.
[20] SCIP, Solving constraint integer programs, https://www.scipopt.org/.
[21] V. Shoup, NTL – a library for doing number theory, https://libntl.org/.
[22] Y. Yang, N. Boland, M. Savelsbergh, Multi-variable branching: a 0-1 knapsack problem case study, INFORMS J. Comput. 33 (4) (2021) 1354–1367.