An algorithm for solving a diophantine equation with lower and upper bounds on the variables

Karen Aardal* Arjen K. Lenstra† Cor Hurkens‡

Abstract

We develop an algorithm for solving a diophantine equation with lower and upper bounds on the variables. The algorithm is based on lattice basis reduction, and first finds short vectors satisfying the diophantine equation. The next step is to branch on linear combinations of these vectors, which either yields a vector that satisfies the bound constraints or provides a proof that no such vector exists. The research was motivated by the need for solving constrained diophantine equations as subproblems when designing integrated circuits for video signal processing. Our algorithm is tested with good result on real-life data.

Subject classification: Primary: 90C10. Secondary: 45F05, 11Y50.

1 Introduction and problem description

We develop an algorithm for solving the following integer feasibility problem:

$$\exists \text{ a vector } \mathbf{x} \in \mathbb{Z}^n \text{ such that } \mathbf{a}\mathbf{x} = a_0, \quad \mathbf{0} \leq \mathbf{x} \leq \mathbf{u}? \tag{1}$$

We assume that $\mathbf{a}$ is an n-dimensional row vector, $\mathbf{u}$ is an n-dimensional column vector, and that a_0 is an integer scalar. This is an NP-complete problem; in the absence of bound constraints, it can be solved in polynomial time. The research was motivated by a need for solving such problems when designing integrated circuits (ICs) for video signal processing, but several other problems can be viewed as problem (1), or generalizations of (1). One such example is the *Frobenius* problem that was recently considered by Cornuéjols, Urbaniak, Weismantel and Wolsey [3]. The instances related to video signal processing were difficult to tackle by linear programming (LP) based branch-and-bound due to the characteristics of the input. In order to explain the structure of these

*aardal@cs.ruu.nl. Department of Computer Science, Utrecht University. Research partially supported by ESPRIT Long Term Research Project No. 20244 (Project ALCOM-IT: *Algorithms and Complexity in Information Technology*), and by NSF grant CCR-9307391 through David B. Shmoys, Cornell University.

†arjen.lenstra@citicorp.com. Emerging Technology, Citibank N.A. Research partially supported by ESPRIT Long Term Research Project No. 20244 (Project ALCOM-IT: *Algorithms and Complexity in Information Technology*).

‡wscor@win.tue.nl. Department of Mathematics and Computing Science, Eindhoven University of Technology.

instances we briefly explain the orgin of the problem below. We also tested our algorithm with good results on the Frobenius instances of Cornuéjols et al.

In one of the steps of the design of ICs for video signal processing one needs to assign so-called *data streams* to processors. A data stream is a repetitive set of arithmetic operations. The attributes of a data stream are the starting time of the first execution, the number of repetitions, and the period of repetition. One can view a data stream as a set of nested loops. The outer loop has an iterator $i_0 : 0 \leq i_0 \leq I_0$. The following loop has iterator $i_1 : 0 \leq i_1 \leq I_1$, and so forth. The periodicity corresponding to a loop is the time interval between two consecutive iterations. When constructing an assignment of the streams to the processors the following *conflict detection problem* occurs: check whether there is any point in time at which operations of two different streams are carried out. If such a point in time exists, then the streams should not be assigned to the same processor. Consider an arbitrary data stream f. Let $\mathbf{i}_f = (i_{f0}, i_{f1}, \ldots, i_{fm})^T$ be the *iterator vector* of the stream. The iterator vector satisfies upper and lower bounds, $\mathbf{0} \leq \mathbf{i}_f \leq \mathbf{I}_f$. Let $\mathbf{p}_f$ denote the *period vector* and s_f the *starting time* of the stream. The point in time at which execution $\mathbf{i}_f$ of data stream f takes place is expressed as $t(\mathbf{i}_f) = s_f + \mathbf{p}_f^T \mathbf{i}_f$. The conflict detection problem can be formulated mathematically as the following integer feasibility problem: Given data streams f and g, $\exists$ iterator vectors $\mathbf{i}_f$ and $\mathbf{i}_g$ such that

$$s_f + \mathbf{p}_f^T \mathbf{i}_f = s_g + \mathbf{p}_g^T \mathbf{i}_g, \text{ and such that } \mathbf{0} \leq \mathbf{i}_f \leq \mathbf{I}_f, \quad \mathbf{0} \leq \mathbf{i}_g \leq \mathbf{I}_g?$$

The Frobenius problem is defined as follows: given nonnegative integers $(a_1, \ldots, a_n)$ with $\gcd(a_1, \ldots, a_n) = 1$, find the largest integer a_0 that cannot be expressed as a nonnegative integer combination of $a_1, \ldots, a_n$. The number a_0 is called the Frobenius number. The instances considered by Cornuéjols et al. [3] were also hard to solve using LP-based branch-and-bound. They developed a test set approach that was successful on their instances.

When solving a feasibility problem such as (1) by LP-based branch-and-bound, two difficulties may arise. First, the search tree may become large depending on the magnitude of the upper bounds on the variables, and second, round-off errors may occur. The size of the branch-and-bound tree may also be sensitive to the objective function that is used. For our problem (1) an objective function does not have any meaning since it is a feasibility problem; as long as we either find a feasible vector, or are able to verify that no feasible vector exists, the objective function as such does not matter. The problem is that one objective function may give an answer faster than another, but which one is best is hard to predict. An objective function also introduces an aspect to the problem that is not natural. Round-off errors occur quite frequently for the instances related to the conflict detection problem, since the coefficients of some of the variables are very large ($\approx 10^7$). The special characteristics of these instances – some very large and some relatively small coefficients and a very large right-hand side value a_0 – are due to the difference in periodicity of the nested loops. This difference is explained by the composition of a television screen image. Such an image consist of 625 lines, and each line is composed of 720 pixels. Every second 25 pictures are shown on the screen, so the time

between two pictures is 40 ms. The time between two lines and between two pixels are 64 μs and 74 ns respectively. Since the output rate of the signals has to be equal to the input rate, we get large differences in periodicity when the data stream corresponds to operations that have to be repeated for all screens, lines and pixels. Due to the large difference in the magnitude of the coefficients we often observe that the LP-based branch-and-bound algorithm terminates with a solution in which for instance variable x_j takes value 4.999999, simply because the hardware does not allow for greater precision. If one would round x_j to $x_j = 5.0$, then one would obtain a vector $\mathbf{x}$ sucht that $\mathbf{ax} \neq d$. It is obviously a serious drawback that the algorithm terminates with an infeasible solution.

To overcome the mentioned deficiencies we have developed an algorithm based on the L^3 basis reduction algorithm as developed by Lenstra, Lenstra and Lovász [9]. The motivation behind choosing basis reduction as a core of our algorithm is twofold. First, basis reduction allows us to work directly with integers, which avoids the round-off problems. Second, basis reduction finds short, nearly orthogonal vectors belonging to the lattice described by the basis. Given the lower and upper bounds on the variables, we can interpret problem (1) as checking whether there exists a *short vector* satisfying a given diophantine equation. It is easy to find an initial basis that describes the lattice containing all vectors of interest to our problem. This initial basis is not "good" in the sense that it contains very long vectors, but it is useful as we can prove structural properties of the reduced basis obtained by applying the L^3 algorithm to it. It is important to note that basis reduction does not change the lattice, it only derives an alternative way of spanning it. Furthermore, our algorithm is designed for feasibility problems. Once we have obtained the vectors given by the reduced basis, we use them as input to a heuristic that tries to find a feasible vector fast or, in case the heuristic fails, we call an algorithm that branches on linear combinations of vectors and yields either a vector satisfying the bound constraints, or a proof that no such vector exists.

In Section 2 we give a short description of the L^3 basis reduction algorithm and a brief review of the use of basis reduction in integer programming. In Section 3 we introduce a lattice that contains all interesting vectors for our problem (1), and provide an initial basis spanning that lattice. We also derive structural properties of the reduced basis. Our algorithm is outlined in Section 4. Some of our test problems are of the type $\exists$ a vector $\mathbf{x} \in \mathbb{Z}^n$ such that:

$$\mathbf{Ax} = \mathbf{b},\ \mathbf{0} \leq \mathbf{x} \leq \mathbf{u}?$$

Here, $\mathbf{A}$ is a $k \times n$ matrix, $k < n$, and $\mathbf{b}$ a k-vector. In Section 5 we discuss how our algorithm generalizes to this case. Our computational experience is presented in Section 6.

2 Basis reduction and its use in integer programming

We begin by giving the definition of a lattice and a reduced basis.

Definition 1 *A subset $L \subset \mathbb{R}^n$ is called a* lattice *if there exists a basis $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_k$ of $\mathbb{R}^n$ such that*

$$L = \{\sum_{j=1}^{k} \alpha_j \mathbf{b}_j : \ \alpha_j \in \mathbb{Z},\ 1 \leq j \leq k\}. \tag{2}$$

Gram-Schmidt orthogonalization is an algorithm for deriving orthogonal vectors $\mathbf{b}_j^*,\ 1 \leq j \leq n$ from independent vectors $\mathbf{b}_j,\ 1 \leq j \leq n$. The vectors $\mathbf{b}_j^*,\ 1 \leq j \leq n$ and the real numbers $\mu_{jk},\ 1 \leq k < j \leq n$ are defined inductively by:

$$\mathbf{b}_j^* = \mathbf{b}_j - \sum_{k=1}^{j-1} \mu_{jk} \mathbf{b}_k^* \tag{3}$$

$$\mu_{jk} = (\mathbf{b}_j)^T \mathbf{b}_k^* / (\mathbf{b}_k^*)^T \mathbf{b}_k^* \tag{4}$$

Lenstra, Lenstra and Lovász [9] used the following definition of a reduced basis:

Definition 2 *A basis $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ is* reduced *if*

$$|\mu_{jk}| \leq \frac{1}{2} \ \textit{for}\ 1 \leq k < j \leq n \tag{5}$$

and

$$||\mathbf{b}_j^* + \mu_{j,j-1}\mathbf{b}_{j-1}^*||^2 \geq \frac{3}{4}||\mathbf{b}_{j-1}^*||^2 \ \textit{for}\ 1 < j \leq n. \tag{6}$$

The vector $\mathbf{b}_j^*$ is the projection of $\mathbf{b}_j$ on the orthogonal complement of $\sum_{k=1}^{j-1} \mathbb{R}\mathbf{b}_k$, and the vectors $\mathbf{b}_j^* + \mu_{j,j-1}\mathbf{b}_{j-1}^*$ and $\mathbf{b}_{j-1}^*$ are the projections of $\mathbf{b}_j$ and $\mathbf{b}_{j-1}$ on the orthogonal complement of $\sum_{k=1}^{j-2} \mathbb{R}\mathbf{b}_k$. The constant $\frac{3}{4}$ in inequality (6) is arbitrarily chosen and can be replaced by any fixed real number $\frac{1}{4} < y < 1$. Lenstra et al. [9] developed a polynomial time algorithm for obtaining a reduced basis for a lattice given an initial basis. The algorithm consists of a sequence of size reductions and interchanges as described below. For the precise algorithm we refer to [9].

Size reduction: If for any pair of indices $i, j :\ 1 \leq k < j \leq n$ condition (5) is violated, then replace $\mathbf{b}_j$ by $\mathbf{b}_j - \lceil \mu_{jk} \rfloor \mathbf{b}_k$, where $\lceil \mu_{jk} \rfloor$ is the integer nearest to μ_{jk}.

Interchange: If condition (6) is violated for an index $j,\ 1 < j \leq n$, then interchange vectors $\mathbf{b}_{j-1}$ and $\mathbf{b}_j$.

Basis reduction was introduced in integer programming by H.W. Lenstra, Jr. [10], who showed that the problem of determining if there exists a vector $\mathbf{x} \in \mathbb{Z}^n$ such that $\mathbf{Ax} \leq \mathbf{d}$ can be solved in polynomial time when n is fixed. Before this result was published, only the cases $n = 1, 2$ were known to be polynomially solvable. The idea behind Lenstra's algorithm can be explained considering a two-dimensional convex body. Suppose that this body is "thin" as illustrated

in Figure 1. If it extends arbitrarily far in both directions, as indicated in the figure, then an LP-based branch-and-bound tree will become arbitrarily deep before concluding that no feasible solution exists. It is easy to construct a similar example in which a feasible vector does exist. So, even if $n = 2$, an LP-based branch-and-bound algorithm may require exponentially many iterations in terms of the dimension. What Lenstra observed was the following. Assume

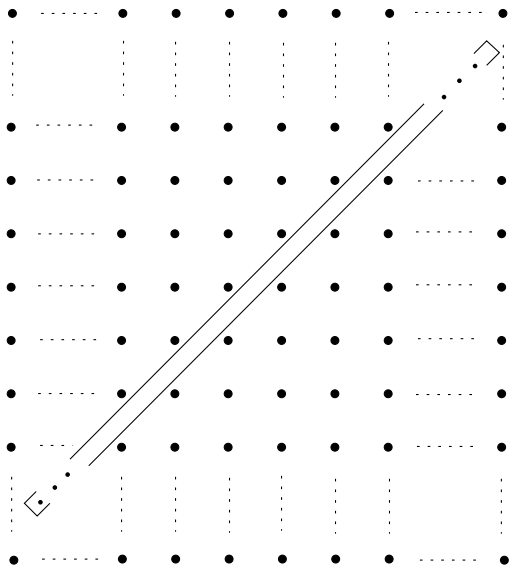

Figure 1: A THIN CONVEX BODY IN $\mathbb{Z}^2$.

that we start with the full-dimensional bounded convex body $X \in \mathbb{R}^n$ and that we consider the lattice $\mathbb{Z}^n$. The problem is to determine whether there exists a vector $\mathbf{x} \in (X \cap \mathbb{Z}^n)$. We refer to this problem as problem P. We use $\mathbf{b}_j = \mathbf{e}_j,\ 1 \leq j \leq n$ as a basis for the lattice $\mathbb{Z}^n$, where $\mathbf{e}_j$ is the vector where all elements of the vector are equal to zero, except element j that is equal to one. To avoid having a convex body that is thin we apply a linear transformation τ to X to make it appear "regular". Problem P is equivalent to the problem of determining whether there exists a vector $\mathbf{x} \in (\tau X \cap \tau\mathbb{Z}^n)$. The new convex body τX has a regular shape but the basis vectors $\tau\mathbf{e}_j$ are not necessarily orthogonal any longer, so from the point of view of branching the difficulty is still present. We can view this as having shifted the problem we had from the convex body to the lattice. This is where basis reduction proves useful. By applying the L^3 algorithm to the basis vectors $\tau\mathbf{e}_j$, we obtain a new basis $\hat{\mathbf{b}}_1, \ldots, \hat{\mathbf{b}}_n$ spanning the same lattice, $\tau\mathbb{Z}^n$, but having short, nearly-orthogonal vectors. In particular it is possible to show that the distance d between any two consecutive hyperplanes $H + k\hat{\mathbf{b}}_n,\ H + (k+1)\hat{\mathbf{b}}_n$, where $H = \sum_{j=1}^{n-1} \mathbb{R}\mathbf{b}_j$ and $k \in \mathbb{Z}$, is not too short, which means that if we branch on these hyperplanes, then there cannot be too many of them. Each branch at a certain level of the search tree corresponds to a subproblem with dimension one less than the dimension of its predecessor. In Figure 2 we show how the distance between

hyperplanes $H + k\hat{\mathbf{b}}_n$ increases if we use a basis with orthogonal vectors instead of a basis with long non-orthogonal ones.

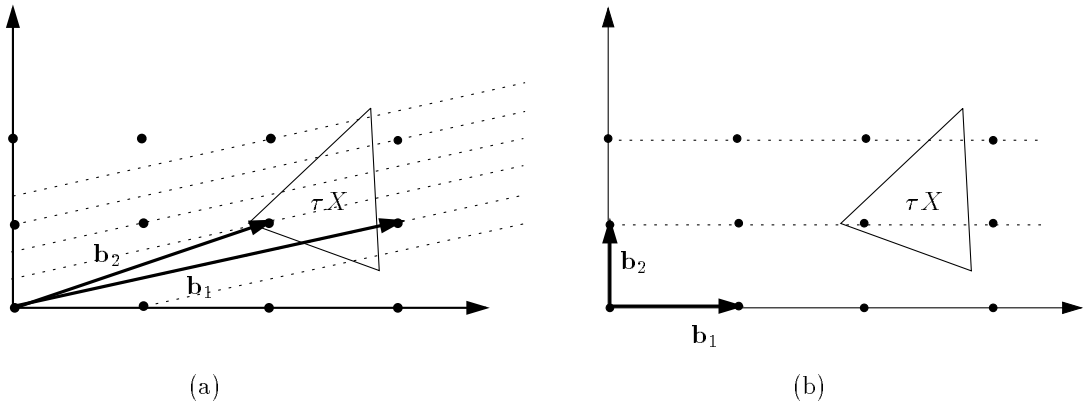

Figure 2: (a) NON-ORTHOGONAL BASIS. (b) ORTHOGONAL BASIS.

Due to the special structure of our instances we do not use a transformation τ in our algorithm. We can simply write down an initial basis for a lattice that contains all vectors of interest for our problem, and then apply the L^3 algorithm directly to this basis.

For the integer programming problem P, Lovász and Scarf [12] developed an algorithm that, as Lenstra's algorithm, uses branching on hyperplanes. Instead of using a transformation τ to transform the convex body and the initial basis vectors, and then applying a basis reduction algorithm, their algorithm produces a "Lovász-Scarf-reduced" basis by measuring the width of the considered convex body in different independent directions. Lovász and Scarf's definition of a reduced basis is a generalization of the definition given by Lenstra et al [9]. Cook, Rutherford, Scarf and Shallcross [1] report on a successful implementation of the Lovász-Scarf algorithm. Cook et al. were able to solve some integer programming problems arising in network design that could not be solved by traditional LP-based branch-and-bound.

Integer programming is not the only application of lattice basis reduction. A prominent application is factoring polynomials with rational coefficients. Lenstra et al. [9] developed a polynomial-time algorithm based on basis reduction for finding a decomposition into irreducible factors of a non-zero polynomial in one variable with rational coefficients. In cryptography, basis reduction has been used to solve subset sum problems arising in connection with certain cryptosystems, see for instance [4], [8], [14], [15]. A recent application in cryptography is due to Coppersmith [2] who uses basis reduction to find small integer solutions to a polynomial in a single variable modulo N, and to a polynomial in two variables over the integers. This has applications to some RSA-based cryptographic schemes. In extended g.c.d.-computations, basis reduction is used by for instance Havas, Majewski and Matthews [7]. Here, the aim is to find a short multiplier vector $\mathbf{x}$ such that $\mathbf{ax} = a_0$, where $a_0 = \gcd(a_1, a_2, \ldots, a_n)$.

3 Structure of initial and reduced basis

Here we consider a lattice that contains all vectors of interest to our problem (1):

$$\exists \text{ a vector } \mathbf{x} \in \mathbb{Z}^n \text{ such that } \mathbf{ax} = a_0, \quad \mathbf{0} \leq \mathbf{x} \leq \mathbf{u}?$$

Without loss of generality we assume that $\gcd(a_1, a_2, ..., a_n) = 1$. We formulate an initial basis $\mathbf{B}$ that generates this lattice and derive structural properties of the reduced basis $\mathbf{RB}$ obtained after applying the L^3 algorithm to $\mathbf{B}$.

Let $\mathbf{x}_0$ denote the vector $(x_0, x_1, \ldots, x_n)^T$. We refer to $\mathbf{x}_0$ as the *extended* $\mathbf{x}$-vector. Here, x_0 is a variable that we associate with the right-hand side coefficient a_0. Given the bounds on the variables, we tackle problem (1) by trying to find a short vector $\mathbf{x}$ satisfying $\mathbf{ax} = a_0$. This can be done by finding short vectors in the lattice L_s containing the vectors

$$(x_0, x_1, ..., x_n, \ (-a_0 x_0 + a_1 x_1 + \cdots + a_n x_n))^T, \tag{7}$$

where $x_0, x_1, \ldots, x_n \in \mathbb{Z}$. In particular, we want to find integral extended $\mathbf{x}$-vectors lying in the null-space of $\mathbf{a}_0 = (-a_0, a_1, ..., a_n)$, denoted by $\mathrm{N}(\mathbf{a}_0)$, i.e., vectors $\mathbf{x}_0 \in \mathbb{Z}^{n+1}$ that satisfy $-a_0 x_0 + \mathbf{ax} = 0$. Moreover, we want $\mathbf{x}$ to satisfy the upper and lower bound constraints, and x_0 to be equal to one. Below we will show how we can use basis reduction to find such a vector.

The lattice L spanned by the basis $\mathbf{B}$ given by

$$\mathbf{B} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & 0 \\ 0 & 0 & \cdots & \cdots & 1 \\ -Na_0 & Na_1 & \cdots & \cdots & Na_n \end{pmatrix}$$

contains the vectors

$$(\mathbf{x}_0, h)^T = (x_0, x_1, ..., x_n, \ N(-a_0 x_0 + a_1 x_1 + \cdots + a_n x_n))^T, \tag{8}$$

where N is a large integral number. Note that the basis vectors are given columnwise, and that the basis consists of $n+1$ vectors $\mathbf{b}_j = (b_{0j}, b_{1j}, \ldots, b_{n+1,j})^T$, $0 \leq j \leq n$. The vectors (7) in the lattice L_s that belong to $\mathrm{N}(\mathbf{a}_0)$ also belong to the lattice L. Since basis reduction is used to derive a basis that describes the given lattice using short basis vectors, we use the multiplier N to "force" the basis reduction algorithm to find as many vectors $\mathbf{x}_0 \in \mathrm{N}(\mathbf{a}_0)$ as possible. This will become clear in the proof of Theorem 2.

Lemma 1 (Lenstra, Lenstra, Lovász [9]) *Let $\Lambda \subset R^n$ be a lattice with reduced basis $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n \in \mathbb{R}^n$. Let $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_t \in \Lambda$ be linearly independent. Then we have*

$$||\mathbf{b}_j||^2 \leq 2^{n-1} \max\{||\mathbf{y}_1||^2, ||\mathbf{y}_2||^2, ..., ||\mathbf{y}_t||^2\} \text{ for } 1 \leq j \leq t. \tag{9}$$

Let $\hat{\mathbf{b}}_j = (\hat{b}_{0j}, \hat{b}_{1j}, \ldots, \hat{b}_{n+1,j})^T$, $0 \leq j \leq n$, denote the vectors of the reduced basis, **RB**, obtained by applying L^3 to **B**.

Theorem 2 *There exists a number N_0 such that if $N > N_0$, then the vectors $\hat{\mathbf{b}}_j \in \mathbb{Z}^{n+2}$, $0 \leq j \leq n-1$, of the reduced basis* **RB** *have the following properties:*

1. $\hat{\mathbf{b}}_j \in \mathrm{N}(\mathbf{a}_0, 0)$ *for* $0 \leq j \leq n-1$, *i.e.,* $\hat{b}_{n+1,j} = 0$ *for* $0 \leq j \leq n-1$.
2. $\gcd(\hat{b}_{00}, \hat{b}_{01}, ..., \hat{b}_{0,n-1}) = 1$.

Proof: We first determine N_0. Apply Gram-Schmidt orthogonalization to the vectors $\mathbf{a}_0^T, \mathbf{e}_1, \ldots, \mathbf{e}_{n+1}$. This gives rational vectors $\mathbf{a}_0^T, \mathbf{v}_1, \ldots, \mathbf{v}_{n+1}$, where one of the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_{n+1}$ consists of zeros only. Permute the $\mathbf{v}$-vectors such that $\mathbf{v}_{n+1}$ is the zero-vector. The vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are orthogonal to $\mathbf{a}_0^T$, and hence $\mathbf{v}_j \in \mathrm{N}(\mathbf{a}_0)$, $1 \leq j \leq n$. Next, multiply the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ by a large enough integral number such that we obtain integral vectors $\mathbf{z}_1, \ldots, \mathbf{z}_n$. We have $\mathbf{z}_j \in \mathrm{N}(\mathbf{a}_0)$, $1 \leq j \leq n$. Select N_0 such that $N_0^2 > 2^{n+1} \max\{||\mathbf{z}_1||^2, \ldots, ||\mathbf{z}_n||^2\}$.

Property 1. Let $\mathbf{y}_j = (\mathbf{z}_j, 0)^T$ for $1 \leq j \leq n$. The vectors $\mathbf{y}_j$, $1 \leq j \leq n$ are in the lattice L spanned by B, and $||\mathbf{y}_j|| = ||\mathbf{z}_j||$, $1 \leq j \leq n$. According to Lemma 1 we have $||\hat{\mathbf{b}}_j||^2 \leq 2^{n+1} \max\{||\mathbf{y}_1||^2, \ldots, ||\mathbf{y}_n||^2\} < N_0^2$ for $0 \leq j \leq n-1$. Suppose that $\hat{b}_{n+1,j} \neq 0$ for some $0 \leq j \leq n-1$. Then $||\hat{\mathbf{b}}_j||^2 \geq \hat{b}_{n+1,j} \geq N^2$ as N divides $\hat{b}_{n+1,j}$. As a consequence, $||\hat{\mathbf{b}}_j||^2 > N_0$, which contradicts the outcome of Lemma 1. We therefore have that $\hat{b}_{n+1,j} = 0$ for $0 \leq j \leq n-1$.

Property 2. Recall that $\gcd(a_1, a_2, ..., a_n) = 1$. The equation $\mathbf{ax} = a_0$ has an integral solution $\hat{\mathbf{x}}$. This is true since a_0 is an integral multiple of $\gcd(a_1, a_2, ..., a_n)$. The vector $(\mathbf{x}_0, h)^T = (1, \hat{x}_1, \hat{x}_2, ..., \hat{x}_n, 0)^T$ belongs to the lattice L spanned by **B**. By applying L^3 to **B** we do not change the lattice L, we only change the way of representing L. Suppose that $\gcd(\hat{b}_{00}, \hat{b}_{01}, ..., \hat{b}_{0,n-1}) > 1$. This would imply that the vector $(1, \hat{x}_1, \hat{x}_2, ..., \hat{x}_n, 0)^T$ cannot be obtained by taking a linear integer combination of the basis vectors $\hat{\mathbf{b}}_1, \hat{\mathbf{b}}_2, ..., \hat{\mathbf{b}}_{n-1}$ of **RB**, which contradicts that we do not change the lattice. ■

Example 1 Consider the following instance of problem (1). Does there exist a vector $\mathbf{x} \in \mathbb{Z}^5$ such that:

$$3,000,000x_1 + 2,999,870x_2 + 6,722x_3 + 6,720x_4 + 15x_5 = 103,329,757; \quad (10)$$

$$0 \leq x_1 \leq 34;\ 0 \leq x_2 \leq 34;\ 0 \leq x_3 \leq 349;\ 0 \leq x_4 \leq 199;\ 0 \leq x_5 \leq 440?$$

Let $N = 10,000$. The initial basis **B** looks as follows:

$$\mathbf{B} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -1,033,297,570,000 & 30,000,000,000 & 29,998,700,000 & 67,220,000 & 67,200,000 & 150,000 \end{pmatrix}$$

After applying L^3 to $\mathbf{B}$ we obtain:

$$\mathbf{RB} = \begin{pmatrix} 0 & 0 & 2 & 0 & 3 & 0 \\ 1 & -1 & 34 & -51 & 61 & -1 \\ -1 & 1 & 35 & 51 & 42 & 1 \\ -5 & -10 & -28 & 0 & 78 & -2 \\ 5 & 10 & -22 & 1 & 70 & 2 \\ -8 & 10 & 8 & -6 & 1 & 9 \\ 0 & 0 & 0 & 0 & 0 & 10,000 \end{pmatrix}$$

Notice that the last elements of the first five basis vectors of $\mathbf{RB}$ are all equal to zero, and that the g.c.d. of the first elements of the first five vectors is equal to one. We also note that the first five vectors of $\mathbf{RB}$ are short relative to the vectors of $\mathbf{B}$. ■

4 The algorithm

Here we discuss how we can use the properties stated in Theorem 2 to design an algorithm for solving the feasibility problem (1). Since we are only interested in the columns of the reduced basis that lie in the null-space $\mathrm{N}(\mathbf{a}_0, 0)$, we will consider only the first $n+1$ elements of the first n vectors of $\mathbf{RB}$. The set of vectors $\bar{\mathbf{b}}_j = (\hat{b}_{0j}, \ldots, \hat{b}_{nj})^T$, $0 \leq j \leq n-1$, forms a basis $\mathbf{B}_{\text{NULL}}$ for $\mathbb{Z}^{n+1} \cap \mathrm{N}(\mathbf{a}_0)$.

If the first element of a basis vector $\bar{\mathbf{b}}_j$ is equal to one (corresponding to $x_0 = 1$), then $x_i = \bar{b}_{ij}$, $1 \leq i \leq n$, constitutes a solution to the equation $-a_0 1 + \mathbf{ax} = 0$ or, equivalently, to $\mathbf{ax} = a_0$. Due to Property 2 of Theorem 2, we know that $\gcd(\bar{b}_{00}, \bar{b}_{01}, ..., \bar{b}_{0,n-1}) = 1$. Hence, we can apply the operations corresponding to repeated calls to Euclid's algorithm for g.c.d. computations, see Algorithm 1, to the full columns of $\mathbf{B}_{\text{NULL}}$ in order to obtain $\gcd(\bar{b}_{00}, \bar{b}_{01}, ..., \bar{b}_{0,n-1}) = 1$, and thereby a basis vector $\bar{\mathbf{b}}_j$ as described above. The basis obtained after this step is called $\mathbf{B}'_{\text{NULL}}$ and consists of vectors $\bar{\mathbf{b}}'_j = (\bar{b}'_{0j}, \bar{b}'_{1j}, \ldots, \bar{b}'_{nj})^T$, $0 \leq j \leq n-1$. We now have a vector j such that $\bar{b}'_{0j} = 1$. All other vectors $k \neq j$ of $\mathbf{B}'_{\text{NULL}}$ have $\bar{b}'_{0k} = 0$. The vectors $(\bar{b}'_{1k}, \ldots, \bar{b}'_{nk})^T$ all lie in the null-space of $\mathbf{a} = (a_1, \ldots, a_n)$, denoted by $\mathrm{N}(\mathbf{a})$.

To simplify the search for a feasible solution we re-index the basis vectors of $\mathbf{B}'_{\text{NULL}}$ such that the vector having $\bar{b}'_{0j} = 1$ becomes the leftmost column i.e., $j \leftarrow 0$, and such that the remaining columns of $\mathbf{B}'_{\text{NULL}}$ are indexed in order of increasing length. If the vector $\mathbf{x}_{\text{D}} = (\bar{b}'_{10}, \ldots, \bar{b}'_{n0})^T$ does not satisfy the upper and lower bound constraints, we can add an integer linear combination of the vectors $\mathbf{x}_k = (\bar{b}'_{1k}, \ldots, \bar{b}'_{nk})^T$, $1 \leq k \leq n-1$, to $\mathbf{x}_{\text{D}}$, to try to obtain a feasible vector. The resulting vector still satisfies the diophantine equation $\mathbf{ax} = a_0$ since the vectors $\mathbf{x}_k$, $1 \leq k \leq n-1$, all belong to $\mathrm{N}(\mathbf{a})$. Hence, in order to solve our feasibility problem (1) we can branch on linear integer combinations of $\mathbf{x}_k$, $1 \leq k \leq n-1$.

Given two non-negative integers a and b, compute $\gcd(a, b)$.

```
procedure gcd(a, b)
begin
    while (b ≠ 0) do
        begin
            r ← a mod b;
            a ← b;
            b ← r;
        end
    return a;
end
```

Algorithm 1: EUCLID'S ALGORITHM

Example 1 (*continued*). After applying Euclid's algorithm as described above to $\mathbf{B}_{\text{NULL}}$, and after re-indexing the columns of $\mathbf{B}'_{\text{NULL}}$ we obtain:

$$\mathbf{B}'_{\text{NULL}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 27 & 1 & -1 & -20 & -51 \\ 7 & -1 & 1 & 21 & 51 \\ 106 & -5 & -10 & -240 & 0 \\ 92 & 5 & 10 & -206 & 1 \\ -7 & -8 & 10 & 22 & -6 \end{pmatrix}$$

The vector

$$\mathbf{x}_{\text{D}} = (\bar{b}'_{10}, \ldots, \bar{b}'_{50})^T = (27, 7, 106, 92, -7)^T \tag{11}$$

satisfies equation (10), but the last element violates the lower bound constraint. The vectors $\mathbf{x}_k = (\bar{b}'_{1k}, \ldots, \bar{b}'_{nk})^T$, $1 \leq k \leq 4$, all lie in $\text{N}(\mathbf{a})$, i.e., they satisfy $3,000,000x_1 + 2,999,870x_2 + 6,722x_3 + 6,720x_4 + 15x_5 = 0$. Subtracting the vector $(\bar{b}'_{11}, \ldots, \bar{b}'_{51})^T = (1, -1, -5, 5, -8)^T$ from $\mathbf{x}_{\text{D}}$ yields the vector $\mathbf{x} = (26, 8, 111, 87, 1)^T$ that satisfies both (10) and the lower and upper bounds. ■

For the feasible instances that we considered in our computational study, either the vector $\mathbf{x}_{\text{D}} = (\hat{b}_{10}, \ldots, \hat{b}_{n0})^T$ satisfies the bound constraints, or it is enough to add or subtract one or two vectors in the null-space $\text{N}(\mathbf{a})$ in order to obtain a vector satisfying the bound constraints. In order to speed up our algorithm we first call a heuristic that performs a simple search on linear combinations of the columns $\bar{\mathbf{b}}'_j, 1 \leq j \leq n$. The heuristic works as follows. Suppose that we are at iteration t of the heuristic and that an integer linear combination of $t_0 \leq t$ vectors of $\text{N}(\mathbf{a})$ has been added to vector $\mathbf{x}_{\text{D}}$. The vector obtained in this way is called the "current vector". For simplicity we assume that only variable x_k of the current vector violates one of its bound constraints. At iteration t we add or subtract an integer multiple λ_t of $(b'_{1t}, \ldots, b'_{nt})^T$ if the violation of variable x_k's bound constraint is reduced and if no other bound constraints becomes violated. As soon as the value of x_k satisfies its bounds, we do not consider any larger values of λ_t. If the heuristic does not find any

```
procedure main(a, a_0, u)
begin
    store initial basis B;
    RB = L^3(B);
    extract B_NULL from RB;
    compute the basis B'_NULL from B_NULL using repeated calls to Euclid's algorithm;
    re-index B'_NULL;
    x_D = (b'_10, ..., b'_n0)^T;
    if 0 ≤ x_D ≤ u then return x_D;
    heuristic(B'_NULL);
    if heuristic fails then
        branch on linear combinations of vectors j = 1, ..., n − 1 of B'_NULL;
    return feasible vector x, or a proof that no such vector exists;
end
```

Algorithm 2: A SUMMARY OF THE COMPLETE ALGORITHM

feasible solution, we call an exact branching algorithm that branches on linear combinations of vectors in $\mathrm{N}(\mathbf{a})$. A summary of the complete algorithm is given as Algorithm 2.

5 The case of k diophantine equations

Some of the test problems that we received from Philips Research Labs are of the type:

$$\exists \text{ a vector } \mathbf{x} \in \mathbb{Z}^n \text{ such that } \mathbf{A}\mathbf{x} = \mathbf{d},\ \mathbf{0} \leq \mathbf{x} \leq \mathbf{u}? \tag{12}$$

Here $\mathbf{A}$ is a $k \times n$ integral matrix, where $k < n$, and $\mathbf{d}$ an integral k-vector. The ith row of the matrix $\mathbf{A}$ is denoted by $\mathbf{a}_i$. In this section we show how our algorithm can be generalized to deal with this case. We assume that $\gcd(a_{i1}, \ldots, a_{in}) = 1$ for $1 \leq i \leq k$, and that $\mathbf{A}$ has full row rank.

The lattice L that we consider in this case contains the vectors

$$(\mathbf{x}_0, h_1, \ldots, h_k)^T = (\mathbf{y}, \mathbf{x}, h_1, \ldots, h_k)^T = \tag{13}$$

$$(y_1, \ldots, y_k, x_1, \ldots, x_n,\ N(-d_1 y_1 + \mathbf{a}_1\mathbf{x}), \ldots, N(-d_k y_k + \mathbf{a}_k\mathbf{x}))^T.$$

Here we introduce one variable y_i for each right-hand side coefficient d_i, which gives an extended $\mathbf{x}$-vector $\mathbf{x}_0 = (y_1, \ldots, y_k, x_1, \ldots, x_n)^T$. The lattice L is spanned by the following basis $\mathbf{B}$:

$$\mathbf{B} = \begin{pmatrix} \mathbf{I}^{(k)} & \mathbf{0}^{(k\times n)} \\ \mathbf{0}^{(n\times k)} & \mathbf{I}^{(n)} \\ -N\,\mathrm{diag}(\mathbf{d}) & N\mathbf{A} \end{pmatrix} \tag{14}$$

where $\mathbf{I}^{(j)}$ denotes the j-dimensional identity matrix, $\mathbf{0}^{(p\times q)}$ denotes the $p \times q$-matrix consisting of zeros, and $\mathrm{diag}(\mathbf{d})$ denotes the matrix where the right-hand side coefficients d_i occur along the main diagonal and where all other elements

are equal to zero. Again, N is a sufficiently large integral number, cf. Section 3. The basis $\mathbf{B}$ consists of $k+n$ vectors $\mathbf{b}_j = (b_{1j}, \ldots, b_{n+2k,j})^T$. The null-space $\mathrm{N}(\mathbf{A}_0)$ is defined as the set of vectors $\{\mathbf{x}_0 : -\mathrm{diag}(\mathbf{d})\mathbf{y} + \mathbf{A}\mathbf{x} = \mathbf{0}\}$. After applying the L^3 basis reduction algorithm to $\mathbf{B}$ we obtain a reduced basis $\mathbf{RB}$, where

$$\mathbf{RB} = \begin{pmatrix} \mathbf{C}^{(k\times n)} & \mathbf{D}^{(k\times k)} \\ \mathbf{E}^{(n\times(n+k))} & \\ \mathbf{F}^{(k\times n)} & \mathbf{G}^{(k\times k)} \end{pmatrix} \tag{15}$$

In the following theorem we present a result analogous to Theorem 2.

Theorem 3 *There exists a number N_0 such that if $N > N_0$, then the submatrix $\mathbf{F}$ of the reduced basis $\mathbf{RB}$ (15) consists of zeros only.*

We skip the proof since it is similar to the proof of Property 1 of Theorem 2.

Example 2 Consider the following instance of problem (12). Determine whether there exists a vector $\mathbf{x} \in \mathbb{Z}^6$ such that

$$\begin{array}{rcrcrcrcrcl} 6x_1 & + & x_2 & + & 3x_3 & + & 3x_4 & & & = & 17 \\ & & & & & & & 2x_5 & + \quad x_6 & = & 11 \\ & & & & 4x_3 & + & 1x_4 & & + \quad 2x_6 & = & 27 \end{array}$$

$$0 \le x_1 \le 2;\ 0 \le x_2 \le 3;\ 0 \le x_3 \le 5;\ 0 \le x_4 \le 2;\ 0 \le x_5 \le 5;\ 0 \le x_6 \le 14$$

Let $N = 10^3$. The initial basis $\mathbf{B}$ for the lattice L for this instance is:

$$\mathbf{B} = \left(\begin{array}{rrr|rrrrrr} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline -17000 & 0 & 0 & 6000 & 1000 & 3000 & 3000 & 0 & 0 \\ 0 & -11000 & 0 & 0 & 0 & 0 & 0 & 2000 & 1000 \\ 0 & 0 & -27000 & 0 & 0 & 4000 & 1000 & 0 & 2000 \end{array}\right)$$

After applying L^3 to this basis we obtain:

$$\mathbf{RB} = \left(\begin{array}{rrrrrr|rrr} -1 & 1 & 0 & -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ \hline -3 & 2 & 0 & -1 & 1 & 1 & 0 & 0 & -1 \\ 1 & 2 & -3 & -2 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & -5 & 0 & -1 & 0 \\ 0 & 0 & 0 & -4 & -2 & -3 & 0 & 1 & 2 \\ 0 & 1 & 1 & 0 & 5 & 1 & 0 & -1 & 0 \\ 0 & -2 & -2 & 0 & 1 & -2 & 0 & 2 & -1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 10^3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -10^3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^3 & 0 \end{array}\right)$$

Note that we have partitioned $\mathbf{B}$ and $\mathbf{RB}$ in submatrices according to (14) and (15). We observe that the 3×6-submatrix $\mathbf{F}$ of $\mathbf{RB}$ consists of zeros only. ■

Let $\hat{\mathbf{b}}_j = (\hat{b}_{1j}, \ldots, \hat{b}_{n+2k,j})^T, 1 \leq j \leq n+k$ be the vectors of the reduced basis $\mathbf{RB}$. As in the case where $k = 1$, we only consider the basis vectors belonging to the null space $\mathrm{N}(\mathbf{A}_0, \mathbf{0}^{(k\times k)})$ for our further computations. The basis $\mathbf{B}_{\mathrm{NULL}}$ that spans the lattice $\mathbb{Z}^{k+n} \cap \mathrm{N}(\mathbf{A}_0)$ consists of vectors $\bar{\mathbf{b}}_j = (\hat{b}_{1j}, \ldots, \hat{b}_{k+n,j})^T$, $1 \leq j \leq n$, so $\mathbf{B}_{\mathrm{NULL}}$ is obtained from $\mathbf{RB}$ by deleting the last k rows, and the last k columns.

$$\mathbf{B}_{\mathrm{NULL}} = \begin{pmatrix} \mathbf{C}^{(k\times n)} \\ \mathbf{E}'^{(n\times n)} \end{pmatrix}$$

where $\mathbf{E}'^{(n\times n)}$ is the submatrix consisting of the first n columns of $\mathbf{E}^{(n\times(n+k))}$.

Next, we want to derive a vector $\mathbf{x}_{\mathrm{D}}$ satisfying $\mathbf{A}\mathbf{x}_{\mathrm{D}} = \mathbf{d}$, which means that we need to find a vector $(\mathbf{y}, \mathbf{x})^T = (1, \ldots, 1, \mathbf{x}_{\mathrm{D}})^T \in \mathbb{Z}^{k+n} \cap \mathrm{N}(\mathbf{A}_0)$. If $\mathbf{0} \leq \mathbf{x}_{\mathrm{D}} \leq \mathbf{u}$, then we are done. Otherwise we have to check whether there exists an integer linear combination of integral vectors belonging to the null-space $\mathrm{N}(\mathbf{A})$, $\mathbf{x}_\lambda$, such that $\mathbf{0} \leq \mathbf{x}_{\mathrm{D}} + \mathbf{x}_\lambda \leq \mathbf{u}$. To find the vector $\mathbf{x}_{\mathrm{D}}$ and integral vectors belonging to $\mathrm{N}(\mathbf{A})$, we perform the following algorithm, see Algorithm 3. Let $\bar{b}'_{ij}$ be the current values of the elements of the basis for the lattice $\mathbb{Z}^{k+n} \cap \mathrm{N}(\mathbf{A}_0)$, and let $\mathbf{B}'_{\mathrm{NULL}} = (\bar{b}'_{ij})$ for $1 \leq i \leq k+n,\ 1 \leq j \leq n$. At the beginning of the algorithm we have $\mathbf{B}'_{\mathrm{NULL}} = \mathbf{B}_{\mathrm{NULL}}$. For all $1 \leq i \leq k$ determine $\gcd(\bar{b}'_{ii}, \ldots, \bar{b}'_{in})$. The operations defined by Euclid's algorithm to perform the g.c.d.-computations are performed on all elements of the relevant columns $\bar{\mathbf{b}}'_i, \ldots, \bar{\mathbf{b}}'_n$. At the end of each iteration there exists an index $j : i \leq j \leq n$ such that $\bar{b}'_{ij} \neq 0$. All other elements $\bar{b}_{iq} = 0$ for $i \leq q \leq n,\ q \neq j$. This nonzero element is precisely equal to $\gcd(\bar{b}'_{ii}, \ldots, \bar{b}'_{in})$. Permute the columns of $\mathbf{B}'_{\mathrm{NULL}}$ such that column j is moved to position i, i.e., $j \leftarrow i$, and $l \leftarrow l+1$ for $i \leq l < j$. We have now obtained a basis $\mathbf{B}'_{\mathrm{NULL}}$ for $\mathbb{Z}^{k+n} \cap \mathrm{N}(\mathbf{A}_0)$ having the following structure:

$$\mathbf{B}'_{\mathrm{NULL}} = \begin{pmatrix} \mathbf{C}_1'^{(k\times k)} & \mathbf{C}_2'^{(k\times(n-k))} \\ \mathbf{E}_1'^{(n\times k)} & \mathbf{E}_2'^{(n\times(n-k))} \end{pmatrix} \tag{16}$$

where $\mathbf{C}'_1$ is a lower triangular matrix.

```
procedure triangulize(B_NULL)
begin
    B'_NULL = B_NULL;
    for i = 1, ..., k do
        begin
            Determine gcd(b̄'_ii, ..., b̄'_in) and apply the operations defined by
            Euclid's algorithm to all elements of the relevant columns b̄_i, ..., b̄_n;
            /*
             * For some j : i ≤ j ≤ n we have b̄'_ij ≠ 0. All other
             * elements b̄'_iq = 0 for i ≤ q ≤ n.
             */
            Move column j of the current basis B'_NULL to position i;
            for l = i, ..., j − 1 do
                begin
                    Move column l moved to position l + 1;
                end
        end
    return B'_NULL;
end
```

Algorithm 3: DETERMINE $\mathbf{B}'_{\text{NULL}}$

Theorem 4 *Suppose that there exists an integral vector* $\mathbf{x}$ *satisfying* $\mathbf{Ax} = \mathbf{d}$. *Let* $\mathbf{w}_1$ *be the solution to the system* $\mathbf{C}'_1\mathbf{w}_1 = \mathbf{1}$. *The vector* $\mathbf{x}_{\text{D}} = \mathbf{E}'_1\mathbf{w}_1$ *is integral and satisfies* $\mathbf{Ax}_{\text{D}} = \mathbf{d}$.

Proof: We first show that the solution $\mathbf{w}_1$ to $\mathbf{C}'_1\mathbf{w}_1 = \mathbf{1}$ is unique. A column of $\mathbf{B}'_{\text{NULL}}$ with k leading zeros belongs to $\text{N}(\mathbf{0}^{(k\times k)}, \mathbf{A})$. The columns of $\mathbf{B}'_{\text{NULL}}$ are linearly independent. Hence there are at most $n-k$ columns in $\mathbf{B}'_{\text{NULL}}$ with k leading zeros. From the g.c.d. computations of Algorithm 3 it follows that we have at least $n-k$ such columns. Hence we have precisely $n-k$ such columns, and these are the last $n-k$ columns of $\mathbf{B}'_{\text{NULL}}$.

$\mathbf{C}'_1$ has full rank due to the following argument. Suppose there exists a vector $\mathbf{q} \neq 0$ with $\mathbf{C}'_1\mathbf{q} = \mathbf{0}$. Then $\mathbf{e} = \mathbf{E}'_1\mathbf{q} \neq \mathbf{0}$ since the column rank of $\mathbf{B}'_{\text{NULL}}$ is equal to n. For the same reason, vector $\mathbf{e}$ is linearly independent of the columns of $\mathbf{E}'_2$. This implies that $\mathbf{e}$, together with the columns in $\mathbf{E}'_2$, belongs to $\text{N}(\mathbf{A})$, which contradicts that the dimension of $\text{N}(\mathbf{A})$ is equal to $n-k$. Hence, the solution $\mathbf{w}_1$ to $\mathbf{C}'_1\mathbf{w}_1 = \mathbf{1}$ is unique.

Next, we show that the for any vector $\mathbf{x}$ that satisfies $\mathbf{Ax} = \mathbf{d}$ there exists an integer vector $(\mathbf{w}_1, \mathbf{w}_2)^T$ such that $\mathbf{B}'_{\text{NULL}}(\mathbf{w}_1, \mathbf{w}_2)^T = (\mathbf{1}^{(k\times 1)}, \mathbf{x})^T$, or equivalently, that $\mathbf{C}'_1\mathbf{w}_1 = \mathbf{1}$.

The reduced basis is obtained by applying a series of elementary column operations to $\mathbf{B}$. The elementary column operations can be described by a $(n+k)\times(n+k)$ unimodular matrix $\mathbf{Q}$ with determinant $\det(\mathbf{Q}) = \pm 1$. Hence $\mathbf{RB} = \mathbf{B}\ \mathbf{Q}$. Take any vector $\mathbf{x}$ such that $\mathbf{Ax} = \mathbf{d}$. We then have,

$$(\mathbf{y}, \mathbf{x}, h_1, \ldots, h_k)^T = (1, \ldots, 1, \mathbf{x}, 0, \ldots, 0)^T = \mathbf{B}(1, \ldots, 1, \mathbf{x})^T = \tag{17}$$

$$\mathbf{B}\ \mathbf{Q}\ \mathbf{Q}^{-1}(1, \ldots, 1, \mathbf{x})^T \equiv \mathbf{RB}\ \mathbf{z}.$$

The vector $\mathbf{z}$ is integral since $\mathbf{Q}^{-1}$ is integral. Notice that the rank of the last k rows of $\mathbf{B}$ is equal to k since we assume that $\mathbf{A}$ has full row rank. The rank of the last k rows of $\mathbf{RB}$ is therefore equal to k as well. From Theorem 3 we know that the first n columns of the last k rows of $\mathbf{RB}$ (submatrix $\mathbf{F}$) consists of zeros only. Therefore, the lower right $k \times k$ submatrix (submatrix $\mathbf{G}$) has rank k. From (17) we can now conclude that the last k components of $\mathbf{z}$: $z_{n+1}, \ldots, z_{n+k}$ are all equal to zero.

The matrix $\mathbf{B}'_{\text{NULL}}$ is obtained by applying elementary column operations to $\mathbf{RB}$, i.e., we can write $\mathbf{B}'_{\text{NULL}}$ as

$$\mathbf{B}'_{\text{NULL}} = ([\mathbf{I}^{(n+k)}\ \mathbf{0}^{(n+k\times k)}]\mathbf{RB}[\mathbf{I}^{(n)}\ \mathbf{0}^{(n\times k)}]^T)\mathbf{S},$$

where $\mathbf{S}$ is the unimodular matrix describing the g.c.d. computations of Algorithm 3. We now have

$$\mathbf{B}'_{\text{NULL}}(w_1, \ldots, w_n)^T \equiv \mathbf{B}'_{\text{NULL}}\mathbf{S}^{-1}(z_1, \ldots, z_n)^T = \tag{18}$$

$$([\mathbf{I}^{(n+k)}\ \mathbf{0}^{(n+k\times k)}]\mathbf{RB}[\mathbf{I}^{(n)}\ \mathbf{0}^{(n\times k)}]^T)(z_1, \ldots, z_n)^T = [\mathbf{I}^{(n+k)}\ \mathbf{0}^{(n+k\times k)}]\mathbf{RB}\ \mathbf{z} =$$

$$[\mathbf{I}^{(n+k)}\ \mathbf{0}^{(n+k\times k)}](1, \ldots, 1, \mathbf{x}, 0, \ldots, 0)^T = (1, \ldots, 1, \mathbf{x})^T.$$

To summarize, we have

$$\mathbf{B}'_{\text{NULL}}(w_1, \ldots, w_n)^T = (1, \ldots, 1, \mathbf{x})^T,$$

where the vector $(w_1, \ldots, w_n)^T$ is integral since $\mathbf{S}^{-1}$ and $(z_1, \ldots, z_n)^T$ are integral. Using (16) we obtain

$$\mathbf{B}'_{\text{NULL}}(w_1, \ldots, w_n)^T = \begin{pmatrix} \mathbf{C}'_1\mathbf{w}_1 \\ \mathbf{E}'_1\mathbf{w}_1 + \mathbf{E}'_2\mathbf{w}_2 \end{pmatrix} = (1, \ldots, 1, \mathbf{x})^T.$$

From the first part of the proof we know that $(w_1, \ldots, w_k)^T$ is unique, and from the second part that $(w_1, \ldots, w_k)^T$ is integral. This completes the proof. ■

Remarks: The matrix $\mathbf{C}'_2$ consists of zeros only. This implies that vectors formed by the columns of $\mathbf{E}'_2$ all belong to N($\mathbf{A}$). We can check, in polynomial time, whether the system $\mathbf{Ax} = \mathbf{d}$ has a feasible solution by deriving the Hermite normal form of $\mathbf{A}$, see for instance Schrijver [16], Chapter 5.3.

Once we have obtained the vector $\mathbf{x}_{\text{D}}$ as described in Theorem 4, we can check if it satisfies the bound constraints. If $\mathbf{x}_{\text{D}}$ violates the bound constraints we branch on linear combinations of vectors in N($\mathbf{A}$), i.e., on the columns of $\mathbf{E}'_2$.

Example 2 (continued). In our example we obtain the basis $\mathbf{B}'_{\text{NULL}}$ as given below.

$$\mathbf{B}'_{\text{NULL}} = \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 2 & 1 & -3 & -1 & 0 & 1 \\ 2 & 0 & -3 & 3 & -3 & 0 \\ 1 & 0 & 4 & 1 & 1 & 2 \\ 0 & -2 & 3 & 0 & 0 & -4 \\ 1 & 5 & -2 & 1 & 1 & 1 \\ -2 & 1 & 4 & -2 & -2 & -2 \end{array}\right)$$

We have partitioned the matrix according to (16). Here $\mathbf{w}_1 = \mathbf{1}$. The vector $\mathbf{x}_{\text{D}} = \mathbf{E}'_1\mathbf{w}_1 = (0,-1,5,1,4,3)^T$. If we subtract the second column of $\mathbf{E}'_2$: $(0,-3,1,0,1,-2)^T$, from $\mathbf{x}_{\text{D}}$ we obtain a vector

$$\mathbf{x} = (0,2,4,1,3,5)^T$$

that satisfies all bound constraints. ■

6 Computational experience

We solved thirteen instances of problem (1). Eight of the instances were feasible and five infeasible. The instances starting with "P" in Table 1 were obtained from Philips Research Labs. The instances starting with "F" are the Frobenius instances of Cornuéjols et al. [3]. Here we used the Frobenius number as right-hand side a_0. The two other instances, starting with "E", were derived from F3 and F4. The information in Table 1 is interpreted as follows. In the first two columns, "Instance" and "n", the instance names and the dimension of the instances are given. An "F" in column "Type" means that the instance is feasible, and an "N" that it is not feasible. In the two columns of LP-based branch-and-bound, "LP B&B", the number of nodes and the computing time are given. In the "# Nodes" column, $500,000^*$ means that we terminated the search after 500,000 nodes without reaching a result. Two asterisks after the number of nodes indicate that a rounding error occurred, i.e., that the rounded solution given by the algorithm did not satisfy the diophantine equation. In both cases we do not report on the computing times since no result was obtained. In the three columns corresponding to our algorithm, "Algorithm", the column "Heur." gives the number of vectors belonging to N($\mathbf{a}$) that was used in the integer linear combination of vectors added to the vector $\mathbf{x}_{\text{D}}$ by the heuristic in order to obtain a feasible solution. A zero in this column therefore means that the vector $\mathbf{x}_{\text{D}}$ was feasible. For the infeasible instances the heuristic obviously failed, and therefore the sign "–" is given in the column. In that case we turn to the branching phase. Here, a one in the column "# Nodes" means that we solved the problem in the root node by using logical implications. The computing times are given in seconds on a 144MHz Sun Ultra-1. For the LP-based branch-and-bound we used CPLEX version 4.0.9 [5], and in our algorithm we used LiDIA, a library for computational number theory [11], for computing the reduced basis.

Our results indicate that the instances are rather trivial once they are represented in a good way. Using the basis $\mathbf{e}_j$ and branching on variables as in LP-based branch-and-bound is clearly not a good approach here, but it is the standard way of tackling integer programs. Using basis reduction seems to give a more natural representation of the problem. For our instances the computing times were very short, and, contrary to LP-based branch-and-bound, we avoid round-off errors. It is also worth noticing that the infeasibility of instances F1–F5 was particularly quickly verified using our algorithm.

			LP B&B		Algorithm		
Instance	n	Type	# Nodes	Time (s)	Heur.	# Nodes	Time (s)
P1	5	F	420**	–	1		$< 10^{-5}$
P2	5	F	327	0.09	1		$< 10^{-5}$
P3	4	F	75	0.05	0		0.01
P4	5	F	313**	–	1		$< 10^{-5}$
P5	5	F	231	0.11	2		$< 10^{-5}$
P6	5	F	313**	–	1		0.01
E1	6	F	3,271	0.97	0		$< 10^{-5}$
E2	7	F	500,000*	–	0		$< 10^{-5}$
F1	5	N	500,000*	–	–	1	$< 10^{-3}$
F2	6	N	500,000*	–	–	2	0.01
F3	6	N	500,000*	–	–	1	$< 10^{-3}$
F4	7	N	500,000*	–	–	1	0.01
F5	8	N	500,000*	–	–	5	0.01

Table 1: Results of the computational experiments.

References

[1] W. Cook, T. Rutherford, H.E. Scarf, D. Shallcross (1993). An implementation of the generalized basis reduction algorithm for integer programming. *ORSA Journal on Computing* 5, 206–212.

[2] D. Coppersmith (1997). Small solutions to polynomial equations, and low exponent RSA vulnerability. *Journal of Cryptology* 10, 233–260.

[3] G. Cornuéjols, R. Urbaniak, R. Weismantel, L. Wolsey (1997). Decomposition of integer programs and of generating sets. In: R. Burkard, G. Woeginger (eds.), *Algorithms – ESA '97*. Lecture Notes in Computer Science 1284, pp 92–103, Springer-Verlag.

[4] M.J. Coster, A. Joux, B.A. LaMacchia, A.M. Odlyzko, C.P. Schnorr (1992). Improved low-density subset sum algorithms. *Computational Complexity* 2, 111–128.

[5] CPLEX Optimization Inc. (1989). Using the CPLEX Callable Library.

[6] B. de Fluiter (1993). *A Complexity Catalogue of High-Level Synthesis Problems*. Master's Thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology.

[7] G. Havas, B.S. Majewski, K.R. Matthews (1996). Extended gcd and Hermite normal form algorithms via lattice basis reduction. Working paper, Department of Mathematics, The University of Queensland, Australia.

[8] J.C. Lagarias, A.M. Odlyzko (1985). Solving low-density subset sum problems. *Journal of the Association for Computing Machinery* 32, 229–246.

[9] A.K. Lenstra, H.W. Lenstra, Jr., L. Lovász (1982). Factoring polynomials with rational coefficients. *Mathematische Annalen* 261, 515–534.

[10] H.W. Lenstra, Jr. (1983). Integer programming with a fixed number of variables. *Mathematics of Operations Research* 8, 538–548.

[11] LiDIA – A library for computational number theory. TH Darmstadt / Universität des Saarlandes, Fachbereich Informatik, Institut für Theoretische Informatik.
`http://www.informatik.th-darmstadt.de/pub/TI/LiDIA`

[12] L. Lovász, H.E. Scarf (1992). The generalized basis reduction algorithm. *Mathematics of Operations Research* 17, 751–764.

[13] M.C. McFarland, A.C. Parker, R. Camposano (1990). The high-level synthesis of digital systems. *Proceedings of the IEEE* 78, 301–318.

[14] C.P. Schnorr, M. Euchner (1994). Lattice basis reduction: improved practical algorithms and solving subset sum problems. *Mathematical Programming* 66, 181–199.

[15] C.P. Schnorr, H.H. Hörner (1995). Attacking the Chor-Rivest Cryptosystem by improved lattice reduction. In: L.C. Guillou, J.-J Quisquater (eds.), *Advances in Cryptology – EUROCRYPT '95*. Lecture Notes in Computer Science 921, pp 1–12, Springer Verlag.

[16] A. Schrijver (1986). *Theory of Integer and Linear Programming*. Wiley, Chichester.

[17] W.F.J. Verhaegh, P.E.R. Lippens, E.H.L. Aarts, J.H.M. Korst, J.L. van Meerbergen, A. van der Werf (1992). Modeling periodicity by PHIDEO steams. *Proceedings of the Sixth International Workshop on High-Level Synthesis*, ACM/SIGDA, IEEE/DATC, 256–266.