FROBENIUS NUMBERS BY LATTICE POINT ENUMERATION

David Einstein
Structured Decisions Corporation, West Newton, Mass., USA
einstein@sdcorp.net

Daniel Lichtblau
Wolfram Research, Inc., Champaign, Illinois, USA
danl@wolfram.com

Adam Strzebonski
Wolfram Research, Inc., Champaign, Illinois, USA
adams@wolfram.com

Stan Wagon
Macalester College, St. Paul, Minnesota, USA
wagon@macalester.edu

Received: , Accepted: , Published:

Abstract

The *Frobenius number* $g(A)$ of a set $A = (a_1, a_2, \ldots, a_n)$ of positive integers is the largest integer not representable as a nonnegative linear combination of the a_i. We interpret the Frobenius number in terms of a discrete tiling of the integer lattice of dimension $n-1$ and obtain a fast algorithm for computing it. The algorithm appears to run in average time that is softly quadratic and we prove that this is the case for almost all of the steps. In practice, the algorithm is very fast: examples with $n = 4$ and the numbers in A having 100 digits take under one second. The running time increases with dimension and we can succeed up to $n = 11$.

We use the geometric structure of a fundamental domain $\mathcal{D}$, having a_1 points, related to a lattice constructed from A. The domain encodes information needed to find the Frobenius number. One cannot generally store all of $\mathcal{D}$, but it is possible to encode its shape by a small set of vectors and that is sufficient to get $g(A)$. The ideas of our algorithm connect the Frobenius problem to methods in integer linear programming and computational algebra.

A variation of these ideas works when $n = 3$, where $\mathcal{D}$ has much more structure. An integer programming method of Eisenbrand and Rote can be used to design an algorithm for $g(a_1, a_2, a_3)$ that takes soft linear time in the worst case. We present a variation of the method that we have implemented and that can be viewed in two ways: as having provably soft linear time, but not guaranteed to work (we know of no instances in which it fails), or as an algorithm that always works and appears to be softly linear in the worst case. At the other end, when n is beyond 11 we can get upper and lower bounds that are far better than what was known.

Our ideas lead to new theoretical results. The first is a simple characterization of triples A such that the ratio of the number of nonrepresentable positive integers to $g(A)+1$ is exactly $1/2$: the condition holds iff some member of A is representable in terms of the other two, reduced by their gcd. We also obtain new Frobenius formulas. Here's a quadratic formula that is easy to discover experimentally: For $a \geq 16$, $g(a, a+1, a+4, a+9) = \frac{1}{9}(a^2 + c_k a) - d_k$, where k is the mod-9 residue of a and c_k and d_k are defined, respectively, by the lists $\{18, 17, 16, 15, 14, 13, 12, 20, 19\}$ and $\{2, 3, 4, 1, 1, 2, 1, 1, 1\}$ starting from $k = 0$. Our methods prove this, as well as similar formulas for longer sequences.

Contents

1. Introduction

Given a vector of positive integers, $A = (a_1, a_2, \ldots, a_n)$, the *Frobenius number* $g(A)$ is the largest M for which there is not a nonnegative integer vector $\boldsymbol{X}$ so that $\boldsymbol{X} \cdot A = M$. Equivalently, $g(A)$ is the largest number not in the additive semigroup generated by A. We always assume $\gcd(A) = 1$, for otherwise $g(A)$ does not exist. The *Frobenius problem* refers to two problems: (1) computing $g(A)$; (2) determining, for a given M, whether a representation exists and if it does, finding one solution. Problem (2) is related to a class of problems known as postage-stamp or change-making problems, since those real-world problems ask for nonnegative solutions to a linear Diophantine equation. The literature on the Frobenius

problem is large; see [36]. We will focus in this paper on the determination of $g(A)$, but will also discuss some variations to the Aardal–Lenstra algorithm for (2); that algorithm can solve the representation problem and we will show how such an algorithm can be used to speed up computations of $g(A)$. It is not hard to prove that $g(A)$ is finite (this follows from Theorem 1). In full generality, the determination of the Frobenius number is $\mathcal{NP}$-hard [36], but when n is fixed it is not.

Sometimes it is more convenient to consider the *positive Frobenius number* $G(A)$, which is the largest M for which there is not a positive integer vector $\boldsymbol{X}$ so that $\boldsymbol{X} \cdot A = M$. It is easy to see that $G(A) = g(A) + \sum A$.

The set A is called a *basis* and is usually given in sorted order, though that is not essential. Since arithmetic modulo a_1 dominates almost all work on the Frobenius number, it is convenient to adopt the following notation: write $A = (a, b_1, \ldots, b_m)$, where $m = n - 1$; we use B to denote $(b_1, \ldots, b_m)$ and sometimes write the basis as (a, B). We use the term *representation*, as in "M has a representation in terms of A", to mean that M is a nonnegative linear combination of entries in A. When we refer to random input, we mean vectors A of random integers that are generated in increasing order so that $b_m < 10a$; in other words, a random basis consists of integers having the same order of magnitude.

In 1884 Sylvester proved that $g(a, b) = ab - a - b$ (so $G(a, b) = ab$). A century later Greenberg [20] showed how to compute $g(a, b, c)$ efficiently; his method was independently discovered by Davison in 1994 [15]. This method has quadratic bit-complexity and can easily handle triples of 100-digit numbers. Kannan [21] showed that, for fixed n, there is a polynomial-time algorithm to compute the Frobenius number, but the method has complexity of the form $O((\log a)^{n^n})$ and so is not practical.

In this paper we present a method that, with rare exceptions, can find $g(A)$ quickly for fixed n. The main object of study is a certain polytope $\mathcal{D}$ in $\mathbb{N}^m$ that is a fundamental domain for a tiling of $\mathbb{Z}^m$ using translations from the lattice of points $\boldsymbol{X}$ such that $\boldsymbol{X} \cdot B$ is divisible by a. When $n = 3$ this domain is just a hexagon in the shape of an L, and its study by Greenberg and Davison led to an excellent algorithm to obtain two vectors (*protoelbows* in our terminology) that encode the shape of the domain; it is then immediate to go from these two vectors to the Frobenius number. In higher dimensions the domain is more complicated—even for n fixed at 4 the number of extreme points of $\mathcal{D}$ is not bounded—but it is possible, with rare exceptions, to describe $\mathcal{D}$ using a relatively small number of vectors. The crux of our method is that we find those vectors and then use them to get the extreme points of $\mathcal{D}$, one of which yields the Frobenius number.

Our algorithm is complex, with several steps: it uses some algorithms from the literature (such as a method of finding the domination kernel of a set of vectors) and some algorithms that are developed here. To summarize the overall complexity situation, all the steps (except one, the lattice point enumeration) of our algorithm have a worst-case running time that is softly quadratic (i.e., $O((\log a)^{2+\epsilon})$) in terms of the length of the input and the parameter $n_P(A)$, which counts the number of *protoelbows*. The parameter can be very large, but computations show that for random input it is quite small; indeed, for fixed n its expected

value (and the expected value of its integer powers) appears to be bounded as a grows. Under the two assumptions that $n_P(A)$ and its powers are bounded on average and that the lattice point enumeration is quadratic time on average, our algorithm runs in softly quadratic time on average. Both of these assumptions are well supported by numerical evidence. Details of this analysis are in §16.

The algorithm can be implemented in an environment that has linear programming tools (such as *Mathematica* [48] or *lattE* [25]), and works well for n up to 11. The method does slow down as n increases (e.g., 44 hours for $n = 11$), but for $n \leq 8$ it is quite efficient, even for 100-digit integers. Moreover, the first step of our algorithm yields reasonably tight bounds on $g(A)$ and can be used when n is larger than 10, up to about 25.

There are several consequences of our approach to the Frobenius problem.

- We show how to use the fundamental domain to compute, when $n = 3$, a rational form for $F(A)$, the generating function for the nonrepresentables: $\sum\{x^i : i \in \mathbb{N}$ and not representable using $A\}$. Setting x to 1 then quickly yields the exact value of $N(A)$, the number of nonrepresentable integers.

- The preceding formula for $N(A)$ was essentially known, but we use the generating function to obtain some new theoretical results. Theorem 9 gives a simple characterization of triples $A = (a, b_1, b_2)$ for which the number of nonrepresentables is exactly one half of $1 + g(A)$, extending an old theorem of Nijenhuis and Wilf [31].

- Our analysis of the $n = 3$ case allows us to develop and implement a new way to compute $g(a, b, c)$ that we conjecture to be softly linear in the worst case. Thus it significantly outperforms the quadratic Greenberg–Davison algorithm, being able to handle million-digit inputs in three minutes. A main idea here is the use of a fast method for lattice reduction based on the "half-GCD" algorithm, which is known to be softly linear. Moreover, we point out that ideas of Eisenbrand and Rote can be used to design an algorithm (called ILP–ER; not implemented) that is provably softly linear in the worst-case. This leads to the remarkable conclusion that computing $g(a, b, c)$ is not much harder than computing $g(a, b)$, which involves but a single multiplication. If Λ is input length, multiplication takes $O(\Lambda \log \Lambda \operatorname{loglog} \Lambda)$ while the ILP–ER method is $O(\Lambda(\log \Lambda)^2 \operatorname{loglog} \Lambda)$, the extra factor arising from the difference between multiplication and gcd extraction.

- We investigate the quadratic sequence $a, a + 1, a + 4, a + 9, \ldots, a + M^2$ in an attempt to generalize work on arithmetic sequences. We find that patterns exist for the Frobenius number of such sequences and we can prove that these patterns hold by finding algebraic forms for the elbows and corners that define the fundamental domain. One example: $g(a, a + 1, a + 4, a + 9) = \frac{1}{9}a^2 + 2a - 2$ if $a \equiv 0 \pmod 9$ and $a \geq 18$, with similar forms for the other eight congruence classes. We can prove similar formulas for $M = 4, 5, 6$, and 7; but there is one surprise as experiments point to a pattern that holds out to $M = 27$, but then changes.

- We include a discussion of the connection of toric Gröbner bases to the Frobenius problem.

Sections 3–12 describe our algorithm. Sections 13 and 14 describe enhancements for speed. Sections 15, 17, 18, and 19 discuss other aspects of the problem, applications, and connections to Gröbner basis theory. Section 16 summarizes the complexity situation. And

the final section lists some open questions. All computer timings use *Mathematica* version 5.2 and are CPU timings on a Macintosh 1.25 GHz PowerPC G4, except where indicated otherwise (Table 3 in §16).

Acknowledgements. We are grateful to Matthias Beck, Dale Beihoffer, Neils Lauritzen, Bjarke Roune, and Kevin Woods for helpful discussions related to the Frobenius problem. We are grateful to Roune for an early version of [37]. We thank two anonymous referees for their many helpful comments.

2. Notation and Terminology

Because our methods rely on several objects and notations, we list the cast of characters.

$\mathbb{Z}$: the integers, $\{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$
$\mathbb{N}$: the natural numbers, $\{0, 1, 2, 3, \ldots\}$
A: a Frobenius basis, which is a list of n positive integers having no nontrivial common divisor and, usually, assumed to be sorted $g(A)$: the Frobenius number of A
$G(A)$: the positive Frobenius number of A; $G(A) = g(A) + \sum A$
$\tilde{O}(\Lambda^p)$: the intersection of the classes $O(\Lambda^{p+\epsilon})$ for $\epsilon > 0$; when p is 1 or 2, this is called *softly linear* or *softly quadratic*, respectively.

a: the first entry of A
B: the entries of A with the first deleted; written $(b_1, \ldots, b_m)$
m: the number of entries in B; $m = n - 1$

L: the homogeneous lattice in $\mathbb{Z}^m$ consisting of vectors $\boldsymbol{X}$ such that $\boldsymbol{X} \cdot B \equiv 0 \pmod{a}$
L_B: the expanded lattice: $L_B = \{B\boldsymbol{v} : \boldsymbol{v} \in L\}$
L^+: the nonnegative part of L, excluding the zero vector: $L^+ = L \cap (\mathbb{N}^m \backslash \{\boldsymbol{0}\})$
$\mathcal{D}$: the fundamental domain for the lattice L
V: a reduced basis of the lattice L
V_B: the expanded basis: $V_B = \{B\boldsymbol{v} : \boldsymbol{v} \in V\}$

K: the bounding box for the domain $\mathcal{D}$; the vector of axial elbows
$\boldsymbol{k}$: the bounding vector or axial vector: the boundary information of the bounding box
$\boldsymbol{k}^+$: the vector that is an upper bound on $\boldsymbol{k}$ obtained by the center-line method
$K^{\pm}$: the expanded bounding box: the reflection of K across all coordinate axes
$\boldsymbol{e}_j$: the coordinate vector $(0, 0, \ldots, 1, 0, \ldots, 0)$ with a 1 in the jth coordinate
$\boldsymbol{i}$: a multiplier vector in $\mathbb{Z}^m$ for generating vectors in L; $\boldsymbol{i} \cdot V \in L$
W: the weight function on $\mathbb{Z}^m$ determined by B: $W(\boldsymbol{X}) = \boldsymbol{X} \cdot B$
$w_{\min}$: the minimal weight, which is the smallest value of $W(\boldsymbol{X})$ for $\boldsymbol{X} \in L^+$

domination: for $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{Z}^m$, $\boldsymbol{X}$ dominates $\boldsymbol{Y}$ if $X_i \geq Y_i$ for each i
protoelbows: vectors $\boldsymbol{X}$ in $L \cap K^{\pm}$such that $W(\boldsymbol{X}) > 0$ or $W(\boldsymbol{X}) = 0$ and first nonzero entry is negative
$n_P(A)$, or just n_P: number of protoelbows

$n_P^+(A)$, or just n_P^+: number of protoelbows when only bounds $\boldsymbol{k}^+$ on the bounding box are used, as opposed to the true box $K^\pm$
axial protoelbow: a protoelbow having one positive entry
preelbows: protoelbows with negative entries replaced by 0
elbows: points $\boldsymbol{X}$ in $\mathbb{N}^m$ that are not in $\mathcal{D}$ but such that each $\boldsymbol{X} - \boldsymbol{e}_i \in \mathcal{D}$ when $X_i \geq 1$
$n_E(A)$, or just n_E: number of elbows
cone: the cone of $\boldsymbol{X} \in \mathbb{N}^m$ is the set of vectors that dominate $\boldsymbol{X}$
domain determined by vectors $\mathcal{E}$: $\mathbb{N}^m \backslash \cup \{\text{cone}(\boldsymbol{X}) : \boldsymbol{X} \in \mathcal{E}\}$
corners: points $\boldsymbol{C} \in \mathcal{D}$ such that each $\boldsymbol{C} + \boldsymbol{e}_j \notin \mathcal{D}$
Frobenius corner: a corner of maximal weight
domination kernel of a set of vectors S: the set that remains after the removal of any vector $\boldsymbol{Y}$ in S that dominates some $\boldsymbol{X} \in S$ with $\boldsymbol{X} \neq \boldsymbol{Y}$

$N(A)$: the number of positive integers that are nonrepresentable by A using coefficients in $\mathbb{N}$
$\omega(A)$: the ratio $N(A)/(1 + g(A))$, which is the proportion of nonrepresentables
$F(x)$: the generating function for the nonrepresentables: $\sum\{x^i : i \in \mathbb{N}$ and not representable using $A\}$
$L(A)$: a heuristic estimate of a lower bound on $g(A)$: $L(A) = \left(\frac{1}{2}n! \prod A\right)^{1/m} - \sum A$
Reps(A): the set of representable integers: $\{M \in \mathbb{N}$: there are nonnegative integers $\boldsymbol{X}$ so that $M = \boldsymbol{X} \cdot A\}$

3. The Fundamental Domain

Each Frobenius basis yields a tiling of the integer lattice $\mathbb{Z}^m$; understanding the shape of a fundamental domain of this tiling leads to an understanding of the Frobenius number. The tiling arises from the additive group homomorphism induced by the basis. If $A = (a, B)$, then B induces a homomorphism $W : \mathbb{Z}^m \to \mathbb{Z}$, where $W(\boldsymbol{X}) = \boldsymbol{X} \cdot B$; $W(\boldsymbol{X})$ is called the *weight* of $\boldsymbol{X}$. Reducing modulo a yields a homomorphism $W_a : \mathbb{Z}^m \to \mathbb{Z}/a\mathbb{Z}$, where $W_a(\boldsymbol{X})$ is the *reduced weight* of $\boldsymbol{X}$. We always use $\{0, 1, \ldots, a-1\}$ to represent the integers modulo a. The extended Euclidean algorithm implies that, for any integer k, there is a vector $\boldsymbol{X} \in \mathbb{Z}^m$ so that $\boldsymbol{X} \cdot A = k$, and hence W and W_a are surjective. The kernel of W_a, a lattice that we call L, is known as the homogeneous lattice for B modulo a, meaning that it consists of all vectors $\boldsymbol{X}$ so that $\boldsymbol{X} \cdot B \equiv 0 \pmod{a}$. We will call two vectors *equivalent* if they have the same reduced weight. Getting an integer basis for L quickly is standard. For general linear systems one would use the Hermite normal form, but here we have only one linear Diophantine equation and so the basis can be obtained when $m = 2$ by the extended Euclidean algorithm, and then by induction for larger m. For more on the Hermite normal form see [17, 43].

We will have occasion to use the *expanded lattice* L_B, which is simply the pointwise product BL; that is, $L_B = \{B\boldsymbol{X} : \boldsymbol{X} \in L\}$.

We will often use the nonnegative vectors in $\mathbb{Z}^m$, so we use $\mathbb{N}^m$ for the semigroup of all vectors in $\mathbb{Z}^m$ with no negative entries. We use L^+ to denote $L \cap \mathbb{N}^m \backslash \{\mathbf{0}\}$, where $\mathbf{0}$ is the zero vector. Because the reduced weight homomorphism is surjective, the set of equivalence

classes $\mathbb{Z}^m / L$ is isomorphic to the integers modulo a. We wish to choose one vector in each class to form a set $\mathcal{D}$ which will give a concrete shape to this particular version of the group $\mathbb{Z}/a\mathbb{Z}$. The shape of $\mathcal{D}$ is intimately connected to the Frobenius problem. Note that every equivalence class contains nonnegative vectors: for if $\boldsymbol{X} \in \mathbb{Z}^m$ then $\boldsymbol{X} \sim \boldsymbol{X} + ka$ and if k is large enough, the new vector is nonnegative. We need the *domination ordering* in $\mathbb{N}^m$.

Definition. $\boldsymbol{Y}$ *dominates* $\boldsymbol{X}$ ($\boldsymbol{X} \leq \boldsymbol{Y}$) if each coordinate of $\boldsymbol{X}$ is no greater than the corresponding coordinate of $\boldsymbol{Y}$. If a set $\mathcal{S}$ has distinct vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$, and $\boldsymbol{Y}$ dominates $\boldsymbol{X}$, then $\boldsymbol{Y}$ is called a *dominator* in $\mathcal{S}$, and $\boldsymbol{X}$ is *dominated* in $\mathcal{S}$. These definitions extend to $\mathbb{Z}^m$ and we will make brief use of domination in that larger space.

A natural way to choose a vector in each equivalence class is to choose the one that has minimal weight among all vectors with no negative entries. There always is such, since any residue class mod a has an entry that is larger than $g(A)$. However, there can be several vectors having the same minimal weight; when that happens we will choose the one that is lexicographically *last*.

Definition. We say $\boldsymbol{X} \in \mathbb{N}^m$ is *irreducible* if (a) $W(\boldsymbol{X})$ is minimal among all weights of equivalent vectors in $\mathbb{N}^m$, and (b) $\boldsymbol{X}$ is lexicographically *last* among all vectors in $\mathbb{N}^m$ having the same weight. Note that the zero vector is irreducible.

Definition. We let $\mathcal{D}$ be the set of irreducible vectors; then $\mathcal{D}$ is a choice set for the equivalence relation. Note that $\mathcal{D}$ does depend on the ordering of A; permuting A's entries yields different domains $\mathcal{D}$. We call $\mathcal{D}$ the *fundamental domain* of A.

Much work on the Frobenius problems takes place in the realm of a certain directed, weighted, circulant graph $\mathcal{G}(A)$ whose vertices are the integers mod a. Each b_i defines a edges of weight b_i, of the form $v \to v + b_i \pmod{a}$ for vertices v. Then $G(A)$ is simply the diameter of $\mathcal{G}(A)$. One can therefore use Dijkstra's algorithm (done by Nijenhuis in 1979) or modifications (see [7] for a comprehensive treatment of the graph approach; that paper presents two fast shortest-path algorithms that can solve the Frobenius problem quickly so long as $a \leq 10^7$). Central to the work in [7] is the concept of a canonical shortest path tree called the *critical tree*. That is defined by taking, for each vertex v, the parent such that the edge from parent to v has the least weight among all edges occurring in any shortest-path to v. The connection with the work here is that, assuming A is in sorted order, the domain $\mathcal{D}$ can be given a natural tree structure so that it becomes *identical* to the critical tree. It is noteworthy that $\mathcal{D}$ embodies so much structure: it is a cyclic group, a lattice quotient, a shortest-path tree in a circulant graph, and a nonconvex geometric polytope.

We will state and prove some simple facts about $\mathcal{D}$, but here are the highlights:

- $\mathcal{D}$ has a group structure isomorphic to $\mathbb{Z}/a\mathbb{Z}$; $\mathcal{D}$ can be given a graph structure so that it is isomorphic to the critical tree.
- $\mathcal{D}$ (viewed in $\mathbb{R}^m$ by considering each point as a unit cube emanating outward from the point) is a continuous solid in the first orthant with no holes and a monotonic character: $\mathcal{D}$ is closed downward under domination ($\boldsymbol{X} \leq \boldsymbol{Y} \in \mathcal{D}$ implies $\boldsymbol{X} \in \mathcal{D}$).

- $\mathcal{D}$ is a fundamental region for a tiling of $\mathbb{Z}^m$ with respect to vectors in L.
- A geometric understanding of the shape of $\mathcal{D}$ yields the Frobenius number.
- There is a relatively small collection of vectors (we call them *elbows*) that completely describes the shape of $\mathcal{D}$. When n is fixed, the size of this set does not (with rare exceptions) change much. Thus the generation of data to describe $\mathcal{D}$'s shape takes about the same time whether the numbers in A have 6 digits or 100 digits. The number of elbows appears to be, on average, polynomial in $\log a$.

This last property is the one that gives us a fast algorithm. This property is the exact opposite of how the graph-based methods of [7] work. When a is fixed, those methods take hardly any more time as n increases, so they work fine when, say, $a = 10000$ and $n = 500$. But the methods of this paper are, in practice, restricted to $n \leq 11$, and the performance does not change much as a changes. When a and n are both of modest size the graph methods are faster.

Proposition 1. The collection of reduced weights of vectors in $\mathcal{D}$ is just $\{0, \ldots, a-1\}$. For any $\boldsymbol{X} \in \mathbb{Z}^m$ there is a unique $\boldsymbol{Y} \in \mathcal{D}$ that is equivalent to $\boldsymbol{X}$. The size of $\mathcal{D}$ is exactly a. $\mathcal{D}$ is closed downward under domination.

Proof. All properties except the last follow from the induced group isomorphism between $\mathcal{D}$ and $\mathbb{Z}/a\mathbb{Z}$. For the last, suppose $\boldsymbol{Y} \leq \boldsymbol{X} \in \mathcal{D}$. If $\boldsymbol{Y} \notin \mathcal{D}$ then there is some $\boldsymbol{Z} \in \mathbb{N}^m$ equivalent to $\boldsymbol{Y}$ so that either $W(\boldsymbol{Z}) < W(\boldsymbol{Y})$ or $W(\boldsymbol{Z}) = W(\boldsymbol{Y})$ and $\boldsymbol{Z}$ is lexicographically later than $\boldsymbol{Y}$. But then $\boldsymbol{Z} + (\boldsymbol{X} - \boldsymbol{Y})$ has this exact relation to $\boldsymbol{X}$, showing that $\boldsymbol{X} \notin \mathcal{D}$, contradiction. Thus $\mathcal{D}$ is downward closed under domination. □

The group properties imply that $\mathcal{D}$, when translated by vectors in L, tiles $\mathbb{Z}^m$. We will show a picture of such a tiling shortly. Further, we can give $\mathcal{D}$ a graph structure that turns it into the critical tree (as defined in [7]). For each $\boldsymbol{X} \in \mathcal{D}$, let $\boldsymbol{Y} = \boldsymbol{X} - \boldsymbol{e}_i$, where i corresponds to the first nonzero entry of $\boldsymbol{X}$; then place a directed edge of weight b_i from $\boldsymbol{Y}$ to $\boldsymbol{X}$. It is not hard to see that this tree structure is isomorphic to the critical tree. More precisely, each $\boldsymbol{X} \in \mathcal{D}$ describes a critical path to the vertex $W_a(\boldsymbol{X})$ in the graph $\mathcal{G}(A)$. Now we turn to the geometry of $\mathcal{D}$.

Definition. A *corner* in $\mathcal{D}$ is a point $\boldsymbol{C}$ in $\mathcal{D}$ so that, for each i, $\boldsymbol{C} + \boldsymbol{e}_i$ is not in $\mathcal{D}$. An *elbow* is a point $\boldsymbol{X} = (x_i)$ that is not in $\mathcal{D}$ but is such that, for each i, either $x_i = 0$ or $\boldsymbol{X} - \boldsymbol{e}_i$ is in $\mathcal{D}$. The *dimensionality* of an elbow is the number of nonzero entries. Thus the 1-dimensional elbows, or *axial elbows*, are the first points along the m axes in $\mathbb{N}^m$ that are not in $\mathcal{D}$. The number of elbows is denoted $n_E(A)$, and it is always at least m, because there are always exactly m axial elbows. The vector $\boldsymbol{k}$ consisting of the positive entries in all the axial elbows (that is, the sum of the axial elbows) is called the *bounding vector*. The smallest rectangle with sides parallel to the axes containing the origin and the bounding vector is called the *bounding box*, and denoted K. If both the bounding vector and its negative are used to form the box, we call it the *expanded bounding box*, $K^{\pm}$. An elbow having no zeroes (an m-dimensional elbow) is called an *interior elbow* (because it is inside the bounding box).

It is easy to see that the fundamental domain is just the complement of cones of its elbows; precisely: $\mathcal{D} = \mathbb{N}^m \setminus \cup \{\text{cone}(\boldsymbol{X}) : \boldsymbol{X} \text{ an elbow of } \mathcal{D}\}$. For a basis having no more than about 25 entries we will be able to compute the bounding vector; for bases having no more than 11 entries, we will be able to compute the complete set of elbows and corners. The fundamental domain and its relation to the Frobenius problem has appeared in work of Killingbergtrø [23] and Owens [34] and, in a different form, in work of Scarf and Shallcross [38] (see [36] pp. 67) and Sturmfels et al [44]. For example, this last paper has the result, mentioned in §4, that the interior elbow, if it exists, is unique. In any case, the connection to the Frobenius number is easy to prove.

Theorem 1. If $\boldsymbol{C}$ is a point in $\mathcal{D}$ whose weight is maximal, then $\boldsymbol{C}$ is a corner and $W(\boldsymbol{C}) - a = g(A)$.

Proof. If $\boldsymbol{C}$ were not a corner, then one of $\boldsymbol{C} + \boldsymbol{e}_i$ would have greater weight. To prove that the Frobenius number is as claimed, we will show that for each $\boldsymbol{X}$ in $\mathcal{D}$, $W(\boldsymbol{X}) - a$ is not representable while each $ka + W(\boldsymbol{X})$, where $k \geq 0$, is. The latter assertion is obvious, as the number is given in the form of a representation. For the first assertion, suppose $W(\boldsymbol{X}) - a = \alpha\ a + \boldsymbol{Y} \cdot B$ with $\boldsymbol{Y}$ and α nonnegative. Then $W(\boldsymbol{X}) = (\alpha + 1)\ a + W(\boldsymbol{Y})$. It follows that $\boldsymbol{Y}$ is equivalent to $\boldsymbol{X}$ and has smaller weight, contradicting $\boldsymbol{X} \in \mathcal{D}$. This means that $g(A) = \max\{W(\boldsymbol{X}) - a : \boldsymbol{X} \in \mathcal{D}\}$, and so $g(A) = W(\boldsymbol{C}) - a$. □

A corner of maximum weight can therefore be called a *Frobenius corner*. It turns out that the number of interior elbows is either 0 or 1; a proof is in §4. The following use of the domain to get a lower bound is due to Killingbergtrø [23].

Corollary 1. For any basis A with n elements, $g(A) > (m! \prod A)^{1/m} - \sum A$.

Proof. Let W be such that the plane defined by $x_1 b_1 + x_2 b_2 + \cdots + x_m b_m = W$ cuts off a first-orthant simplex with volume a. Then, because $\mathcal{D}$ also has volume a, there must be a corner $\boldsymbol{C}$ of $\mathcal{D}$ so that $\boldsymbol{C} + \boldsymbol{1}$ is outside the simplex, and so $(\boldsymbol{C} + \boldsymbol{1}) \cdot B > W$. But the plane has axis-intercepts W/b_i, giving the volume of the simplex it bounds as $(m!)^{-1} \prod (W/b_i)$. It follows that $W = (m! a \prod b_i)^{1/m}$. Because W is a lower bound on the weight of $\boldsymbol{C} + \boldsymbol{1}$, this yields the claimed bound, since $g(A) \geq \boldsymbol{C} \cdot B - a > W - a - \sum B$. □

In [7] it was suggested, with much evidence, that the somewhat larger function $L(A) = ((n!/2) \prod A)^{1/m} - \sum A$ is a good rough estimator of $g(A)$, and observed that $L(A)$ is very often a lower bound on $g(A)$. Davison [15] showed that $g(A) \geq L(A)$ whenever $|A| = 3$, a result that is stronger than the bound of Corollary 1. The bound of the corollary is nicely universal; in §15 we show how to get much larger lower bounds by finding actual corners.

Example: The Two-Dimensional Domain

When $n = 3$, the domain $\mathcal{D}$ is quite simple: if there is one interior elbow, $\mathcal{D}$ is an L-shaped hexagon; if there is no interior elbow, it degenerates to a rectangle. Therefore $n_E(A)$ is 3 in general, and 2 in the degenerate case. The degenerate case occurs precisely when the basis

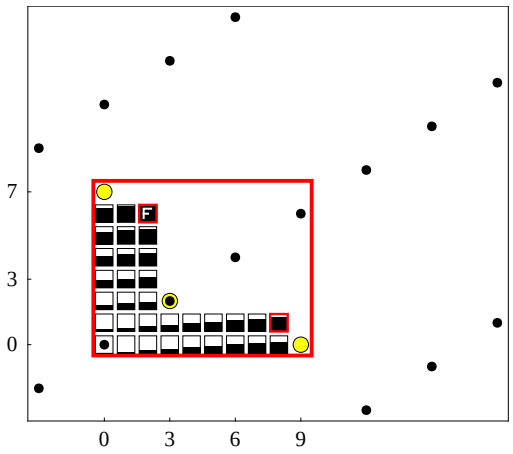

Figure 1: The squares are the members of the fundamental domain for $A = (33, 89, 147)$. The black dots are points in the lattice L, the yellow disks are the three elbows, the squares with red borders are the two corners, and the corner with a white F is the Frobenius corner. The filling in the domain squares indicates the ratio of the weight to the maximum weight; thus the Frobenius corner $(2, 6)$ is a fully filled square and the Frobenius number is $(2, 6) \cdot (89, 147) - 33 = 1027$.

has the property that one of its entries is representable in terms of the other two, reduced by their gcd; a proof is in §18 (see also §5).

The number of corners is 2 in general, 1 in the degenerate case. If the only elbows are the axial ones, $(x_1, 0)$ and $(0, y_2)$, then the unique corner is $(x_1 - 1, y_2 - 1)$ and $g(A)$ is $x_1 b_1 + y_1 b_2 - a - b_1 - b_2$. If there is a non-axial elbow (x_3, y_3), then there are two corners, $(x_1 - 1, y_3 - 1)$ and $(x_2 - 1, y_2 - 1)$, and $g(A)$ is determined by which of these has the greater weight. Algorithms for quickly determining the elbows exist [20, 15], though our general methods handle this case well, and are faster for very large numbers (details in §5).

Figure 1 illustrates an example where $A = (33, 89, 147)$. The points in $\mathcal{D}$ are the squares, each one filled according to its weight (the maximum weight is a fully filled square). The elbows are circles and the two corners are marked by red borders, with the Frobenius corner $(2, 6)$ marked with a white F; $g(A)$ is therefore $2 \cdot 89 + 6 \cdot 147 - 33 = 1027$. The two axial elbows are $(9, 0)$ and $(0, 7)$, and there is one interior elbow, $(3, 2)$. The black dots show the points of the lattice L and the red frame is the bounding box K.

Example: The Three-Dimensional Domain

When $n = 4$ the domain $\mathcal{D}$ is a 3-dimensional staircase. Figure 2 shows $\mathcal{D}$ when $A = (100, 239, 543, 609)$. The 9 elbows are in yellow and the corners are green cubes except for the blue Frobenius corner.

It can happen that the number of elbows is very large, which causes problems for our methods. Szekely and Wormald [45] showed that such examples exist, but their work is in terms of generating functions. An example from their paper (slightly modified) is $A =$

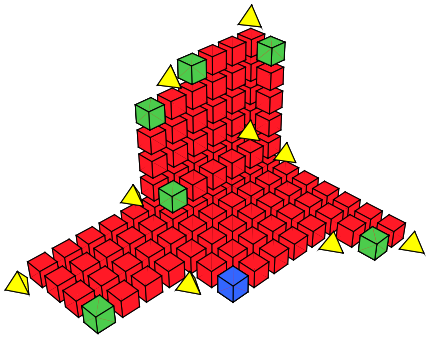

Figure 2: The boxes form the fundamental domain for $A = (100, 239, 543, 609)$. The elbows are the yellow tetrahedra. The corners are green, except the Frobenius corner $(6, 5, 0)$, which is blue. Therefore $g(A)$ is $(6, 5, 0) \cdot (239, 543, 609) - 100 = 4049$.

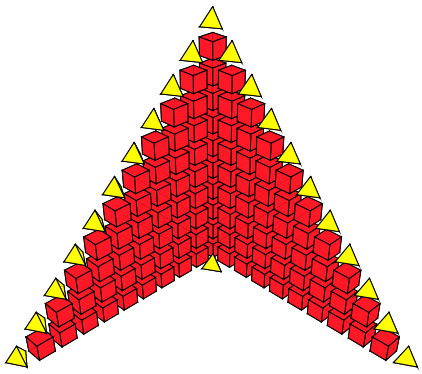

Figure 3: The fundamental domain and elbow set for $A = (100, 101, 110, 111)$. There are 22, or $2 + 2\sqrt{a}$, elbows.

$(w^2, w^2 + 1, w^2 + w, w^2 + w + 1)$. Figure 3 shows the corresponding domain when $w = 10$, and $A = (100, 101, 110, 111)$. It is easy to see that there are precisely $2w + 2$ elbows for this basis. Proof: Show that each of the points $(0, 0, w)$, $(1, 1, 0)$, and $\{(0, w - i, i), (w - i, 0, i)\}$, $i = 0, \ldots, w - 1$ is outside $\mathcal{D}$ (the reductions are easy to find; e.g., $(0, w - i, i) \cdot B = (w - i + 1, i, 0, 0) \cdot A$); the domain these points determine has size $w + 2\sum_{i=1}^{w-1} i = w^2$, so each must in fact be an elbow: The number of elbows is therefore $2 + 2\sqrt{a}$, which means that an approach that enumerates all elbows will run into time and memory problems on such examples when a is large. On the other hand, such issues do not seem to arise in random examples.

4. Protoelbows and Preelbows

Our main mission is to find all the elbows of $\mathcal{D}$. We will do this by finding a finite set of vectors in $\mathbb{Z}^m$ that contain, in their positive entries, a vector that might be an elbow. These integer

vectors will be called protoelbows and their positive parts preelbows. It will then be possible to get the elbows from the set of preelbows. First we need a lemma on the relationship between an elbow $\boldsymbol{X}$, which is not in $\mathcal{D}$, and the vector in $\mathcal{D}$ that is equivalent to $\boldsymbol{X}$.

Lemma 1 (Orthogonality Lemma). If $\boldsymbol{X}$ is an elbow and $\boldsymbol{Y} \in \mathcal{D}$ is equivalent to $\boldsymbol{X}$, then $\boldsymbol{X}$ and $\boldsymbol{Y}$ are orthogonal. These two vectors are nonnegative, so for each coordinate i, at most one of X_i, Y_i is nonzero.

Proof. Assume the conclusion is false for some i. Then $\boldsymbol{X} - \boldsymbol{e}_i$ has the same reduced weight as $\boldsymbol{Y} - \boldsymbol{e}_i$. Moreover, by downward closure of $\mathcal{D}$, $\boldsymbol{Y} - \boldsymbol{e}_i \in \mathcal{D}$. This means that $\boldsymbol{X} - \boldsymbol{e}_i \notin \mathcal{D}$, contradicting the assumption that $\boldsymbol{X}$ is an elbow. □

It follows easily from this lemma that the interior elbow has reduced weight 0, and this implies that the interior elbow, when it exists, is unique. (If there were another, subtract 1 from an entry in which they differ to obtain distinct equivalent points in $\mathcal{D}$.) The orthogonality lemma is the key that allows entry to the world of the elbows.

Definition. A *protoelbow* is a vector $\boldsymbol{X}$ in $L \cap K^{\pm}$ such that $W(\boldsymbol{X}) \geq 0$ and if $W(\boldsymbol{X}) = 0$, then the first nonzero entry in $\boldsymbol{X}$ is negative. A *preelbow* is a protoelbow with all of its negative entries set to 0. A protoelbow having only one positive coordinate is an *axial protoelbow*. The number of protoelbows is denoted $n_P(A)$.

Note that the zero vector is not a protoelbow. Protoelbows can be thought of as reducing relations that prove that a certain vector is not in $\mathcal{D}$. For example, suppose $n = 6$ and $A = (30, 31, 43, 55, 67, 98)$. The fact that the weight of $(0, -2, 2, 2, -1)$ is positive and 0 (mod 30) can be viewed as saying that $W(0, 0, 2, 2, 0) = 244 \equiv 184 = W(0, 2, 0, 0, 1)$ (mod 30). We learn from this that $\boldsymbol{X} = (0, 0, 2, 2, 0)$ is not in $\mathcal{D}$, and so has the potential of being an elbow. In short, a protoelbow provides us, after zeroing out the negatives, with a vector $\boldsymbol{X}$ that meets two necessary conditions for elbow-ness: $\boldsymbol{X}$ is not in $\mathcal{D}$, and there is an equivalent vector of lesser weight that is orthogonal to $\boldsymbol{X}$. The protoelbow concept appears in [34], where Owens showed how, when a is small, these vectors, which he called *zeroes*, could be used to find the Frobenius corner.

By the Orthogonality Lemma, every elbow occurs as the positive part of a protoelbow. For if $\boldsymbol{X}$ is an elbow then $\boldsymbol{X}$ is equivalent to some $\boldsymbol{Y} \in \mathcal{D}$ and, by orthogonality and the size of the bounding box, $\boldsymbol{X}$ and $-\boldsymbol{Y}$ combine to form a protoelbow. So once we have the set of protoelbows in hand, we can just zero out the negatives to get a set of what we call *preelbows*. This set must contain the set of elbows.

No elbow can dominate another one. Thus it seems reasonable that removing dominators from the set of preelbows will yield the set of elbows. To be precise, given a set $\mathcal{S}$ of vectors in $\mathbb{N}^m$, its *domination kernel* consists of those vectors in $\mathcal{S}$ that do not dominate another vector in $\mathcal{S}$.

Proposition 2. The domination kernel of the preelbows is exactly the set of elbows.

Proof. As noted above, any elbow is in the set of preelbows. An elbow $\boldsymbol{v}$ is a vector that lies in $K^{\pm}$ and has the property that any vector in $\mathbb{N}^m$ that it dominates is in $\mathcal{D}$. Thus $\boldsymbol{v}$

cannot dominate another preelbow. So no elbow would be removed from the preelbow set when the domination kernel is formed. On the other hand, suppose $\boldsymbol{v}$ is a preelbow (indeed, any point in K but not in $\mathcal{D}$) but not an elbow. Move down from $\boldsymbol{v}$ in the first coordinate until just hitting $\mathcal{D}$ (or a coordinate hyperplane). Then move down in the second direction the same way. Continue in all directions. The endpoint of such an orthogonal tour must be an elbow, and hence a preelbow by the first part of the proof. Thus $\boldsymbol{v}$ dominates another preelbow and so will be removed. □

Returning to the example above, where $A = (30, 31, 43, 55, 67, 98)$, there are 93 protoelbows. Once the negatives are zeroed out, some duplicates are introduced and the number of preelbows is down to 85. The domination kernel of this set has size 10, and that is the set of elbows, namely

$(0,0,0,0,1)$, $(0,0,0,2,0)$, $(0,0,1,1,0)$, $(0,0,2,0,0)$, $(0,1,1,0,0)$, $(0,2,0,0,0)$, $(5,0,1,0,0)$, $(5,1,0,1,0)$, $(6,0,0,1,0)$, $(7,0,0,0,0)$.

From the axial elbows in this set we see that the bounding vector is $(7,2,2,2,1)$.

While the preceding comments on protoelbows, preelbows, and domination are enough to formulate a workable algorithm to find all elbows, there are several improvements that can be made with a little more theoretical delving into the elbow world. A starting point of our algorithm is to find the m axial elbows, but in fact there is one more elbow that can be found, and it yields useful information. Let $w_{\min}$ be the smallest weight of a vector in L^+; call a vector in L^+ *minimal* if its weight equals $w_{\min}$. Thus there is always at least one minimal vector, but there can be more. The optimization methods that we will use to find the axial elbows can also be used to find the set of minimal vectors. Now, it turns out that there is always an elbow in this set.

Proposition 3. If $\boldsymbol{M}$ is the lexicographically last minimal vector, then $\boldsymbol{M}$ is an elbow.

Proof. Either $\boldsymbol{M}$ is in the bounding box or not. If it is not, then there is some axial elbow $\boldsymbol{F}$ that $\boldsymbol{M}$ dominates. If it is, then $\boldsymbol{M}$ is a protoelbow and, because it is nonnegative, it is a preelbow. Therefore, in this case too, there is an elbow $\boldsymbol{F}$ that it dominates. Let $\boldsymbol{F'}$ be a protoelbow that yielded $\boldsymbol{F}$ (so $W(\boldsymbol{F'}) \geq 0$ and $\boldsymbol{F}$ agree wherever either is positive). Consider the vector $\boldsymbol{M'} = \boldsymbol{M} - \boldsymbol{F'}$. Because $\boldsymbol{M}$ dominates $\boldsymbol{F}$, $\boldsymbol{M'}$ must be nonnegative. Also, $\boldsymbol{M'}$ is in L. Therefore $\boldsymbol{M'} \in L^+$ and so either $\boldsymbol{M'}$ is the zero vector or its weight equals that of $\boldsymbol{M}$. In the latter case $W(\boldsymbol{F'})$ must be zero and so, because $\boldsymbol{F'}$ is a protoelbow, its first nonzero entry must be negative. But this means that $\boldsymbol{M'}$ is lexicographically later than $\boldsymbol{M}$, contradicting the choice of $\boldsymbol{M}$. Therefore $\boldsymbol{M'}$ is the zero vector, so $\boldsymbol{M} = \boldsymbol{F'}$. But then $\boldsymbol{F'}$ is a nonnegative protoelbow, and so it must equal the elbow $\boldsymbol{F}$ that it corresponds to. It follows that $\boldsymbol{M} = \boldsymbol{F}$, so $\boldsymbol{M}$ is an elbow. □

This proposition will allow us to find one more elbow in addition to the axial elbows. So in cases where finding all elbows is impractical, we can use this set of elbows to get reasonably tight upper and lower bounds on $f(A)$. Let us use *minimal elbow* to refer to the lexicographically last minimal vector. Since the minimal elbow is unique, it follows from the next corollary that the number of interior elbows is either 0 or 1.

Corollary 2. If $\boldsymbol{X}$ is the interior elbow, then $\boldsymbol{X}$ is minimal, and in fact $\boldsymbol{X}$ is the minimal elbow.

Proof. We know (see comment after Lemma 1) that $W(\boldsymbol{X}) \equiv 0 \pmod{a}$, so $\boldsymbol{X}$ is equivalent to the zero vector. And there is no other elbow equivalent to the zero vector, for if $\boldsymbol{Y}$ were such, subtract $\boldsymbol{e}_i$, where $Y_i \neq 0$, from $\boldsymbol{X}$ and $\boldsymbol{Y}$ to get distinct vectors in $\mathcal{D}$ with the same reduced weight. But $\boldsymbol{M}$, the lexicographically last minimal vector is an elbow and, because it is in the lattice L^+, $\boldsymbol{M}$ is equivalent to the zero vector. Therefore $\boldsymbol{M}$ must be the interior elbow. □

We will use the minimal elbow in our algorithms, since every Frobenius basis has one, something that is not true for interior elbows. For most cases of interest the minimal elbow will in fact be the interior elbow. In any case, we can state an algorithm for determining if an interior elbow exists and, if so, finding it. First find $\boldsymbol{Y}$, the minimal elbow. If $\boldsymbol{Y}$ is interior, then it is the interior elbow. If $\boldsymbol{Y}$ is not interior, then there is no interior elbow. Of course, for this to work we need to understand how to find the minimal elbow, a point that will be covered in the detailed discussion of the main algorithmic steps that follows.

It is also useful to classify the protoelbows according to their weight. A protoelbow having weight 0 is called *null*; a protoelbow whose weight is greater than $w_{\min}$ is called a *heavy* protoelbow; the ones that are neither heavy nor null are called *light*.

How Many Protoelbows Are There

We have seen (Fig. 3) that the number of elbows can be as large as $\sqrt{a}$ when $n = 4$, and so n_P, the number of protoelbows, is at least that large. Such cases are exceptional, and for random data this type of shape does not occur. Because the size of n_P is central to the performance and analysis of our Frobenius algorithm, we present here some experiments with $n = 4$ to justify the view that, when n is fixed, n_P is, on average, constant. Letting a take on the values 10^{10}, 10^{100}, and 10^{1000} and averaging over 2000 trials in each case, the mean numbers of protoelbows and elbows are as in Table 1. Because we also need to know about the behavior of powers of these parameters, some data on that is included. We show the results from 10000 trials also because of the concern that large spikes could cause the means to diverge. But the evidence is that the spikes, which do occur (see Fig. 4), show up too infrequently to have an impact on the means. Further experiments show that this behavior persists for larger n. In the experiment below a was taken to be a random integer near the power of 10.

While every so often an example will turn up with a large number of protoelbows, the mean values show no obvious tendency to rise as a increases; indeed the computed averages sometimes *decrease*. This could be because the spikes become less of a factor as a rises; one imagines that that is because the sample space is so vast that the chance of finding unusual behavior decreases significantly.

a	10^{10}	10^{100}	10^{1000}
n_P averaged over 2000 (and 10000) trials	22.8 (23.4)	23.2 (23.0)	23.7
n_E averaged over 2000 (and 10000) trials	11.5 (11.3)	11.6 (11.4)	11.5
n_P^2 averaged over 2000 (and 10000) trials	2218 (2318)	1594 (1428)	1655
n_E^2 averaged over 2000 (and 10000) trials	432 (225)	315 (205)	173
n_P^3 averaged over 2000 (and 10000) trials	1.83 10^6 (2.43 10^6)	743000 (464000)	578756
n_E^3 averaged over 2000 (and 10000) trials	204052 (45627)	81448 (21427)	4330

Table 1: An experiment with $n = 4$ and a varying up to 10^{1000}.

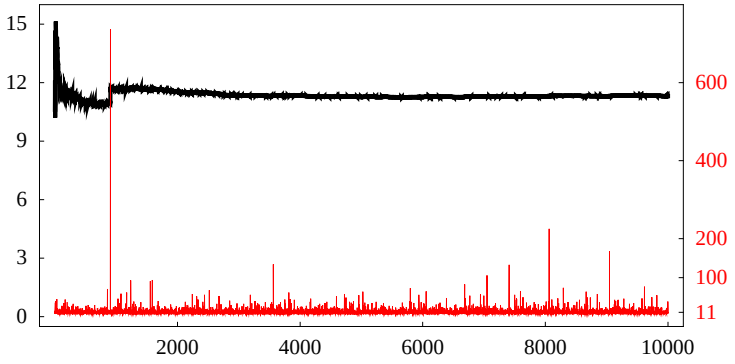

Figure 4: The black curve shows the cumulative average for 10000 counts of elbows for a near 10^{10} and $n = 4$. The red curve shows the values of n_E, the elbow counts, scaled according to the right-hand ticks. Spikes do occur—the first large one bumps up the average—but they do not seem to occur often enough to affect the long-term average. The assumption that n_E is constant on average is critical to the performance of our Frobenius algorithm.

This boundedness-on-average is a nice point of coherence with the $n = 3$ case, where $n_E \leq 3$. It is the controlled behavior of n_P and n_E—the fact that the fundamental domain can almost always be described by a relatively small number of vectors—that allows our algorithms to work. Nevertheless, as n increases, then n_P increases too, and this is what brings our approach to its knees when n reaches 11: the number of protoelbows is larger than 10^8 (see Fig. 12).

5. A Special Case: $n = 3$

When $n = 3$ we have already seen that there are either two or three elbows and $\mathcal{D}$ is therefore either a rectangle or an L-shaped hexagon. But there are several other phenomena that make the $n = 3$ case special, and we summarize them here (for more detail in the $n = 3$ case see §18). Let (x_1, y_1) and (x_2, y_2) be the two active axial protoelbows; by this we mean that the axial elbows are $(x_1, 0)$ and $(0, y_2)$ but we need to be precise about y_1 and x_2. Let $(0, Y)$ be the point in $\mathcal{D}$ whose weight is the same mod-a as that of $(x_1, 0)$; then $y_1 = -Y$; similarly for the other axis. In other words, y_1 is the minimal second coordinate in absolute value

among protoelbows of the form (x_1, Y). And we let (x_3, y_3) denote the interior elbow, if it exists (in which case it is the minimal elbow).

Let M denote the matrix formed by the two active axial protoelbows. We distinguish two cases: If either y_1 or x_2 is 0, then we are in the *degenerate* case; otherwise the *normal* case. A subtle point is that this definition is not invariant under permutation: $(6, 7, 8)$ is degenerate while $(7, 8, 6)$ is not (see end of §18 for more on this point). Authors interested in computing $g(a, b, c)$ have often assumed that the numbers are pairwise relatively prime and in increasing order, and that c is not representable in terms of a and b. But we are here interested in understanding the domain for any triple.

Let $\boldsymbol{s}$ be the sum of the two active axial protoelbows, which must be a lattice point in L. Moreover, by the definition of protoelbow (which is always in the expanded bounding box), we always have that $|y_1| \leq y_2$ and $|x_1| \leq x_2$, and so $\boldsymbol{s}$ lies in the first quadrant. Let $\mathcal{D}^*$ consist of lattice points that lie in the rectangle with vertices $(0, 0)$, $(x_1 - 1, 0)$, $(0, y_2 - 1)$, and $(x_1 - 1, y_2 - 1)$ (inclusive), and then removing the cone determined by $\boldsymbol{s}$; $\mathcal{D}^*$ is either an L-shaped hexagon or a rectangle and it is easy to see that the number of points in $\mathcal{D}^*$ is $x_1y_2 - x_2y_1$. Further, in the normal case we have the strict inequalities $|y_1| < y_2$ and $|x_2| < x_1$, which means that $\boldsymbol{s}$ has no zero entry; this was proved in [12] and we state their main result here.

Proposition 4. In the normal case $|y_1| < y_2$ and $|x_2| < x_1$. Moreover, $\mathcal{D}^*$ contains exactly one point in each equivalence class (i.e., the residues $\boldsymbol{X} \cdot B$ are complete modulo a as $\boldsymbol{X}$ varies in $\mathcal{D}^*$). Therefore $\mathcal{D}^*$ has size a.

Now, in the normal case $\boldsymbol{s}$ is a lattice vector in the first quadrant, with no zero entry. Therefore $\boldsymbol{s}$ must be a protoelbow. But Proposition 4 tells us that $\boldsymbol{s}$ is in fact an elbow, for otherwise $\mathcal{D}^*$ would have size smaller than a; therefore $\boldsymbol{s}$ is the interior elbow and $\mathcal{D}^* = \mathcal{D}$. Further, the count of points in $\mathcal{D}^*$ means that $x_1y_2 - x_2y_1 = a$, and so we learn that $\det(M) = a$. And this means that M is a basis for the lattice L. We need a well-known fact relating lattice bases to a determinant computation, which we state it as a theorem. The proof (omitted) is a simple argument based on an Hermite normal form construction for a particular basis of the lattice,.

Theorem 2. A set $\mathcal{U}$ of m vectors in L is a basis for L iff $\det(\mathcal{U}) = \pm a$.

For larger n one can ask whether the m axial protoelbows form a basis for L: sometimes they do, sometimes they do not. For example, if $A = (4, 5, 6, 7)$, then the axial protoelbows are $(2, -1, 0)$, $(0, 2, 0)$, and $(0, -1, 2)$ with determinant 8. So the axial protoelbows are not a basis for L; alternatively, just observe that the unique solution to $\boldsymbol{X} \cdot M = (1, -2, 1)$ is not integral, where $(1, -2, 1)$ is one of the basis vectors for L.

In the degenerate case, suppose first that the axial protoelbows are $(x_1, 0)$ and (x_2, y_2). Then $(x_1, 0)$ must be the minimal elbow: it lies in L^+ and if (α, β) were in $L^+ \backslash \{(x_1, 0)\}$ and had smaller or equal weight then $0 \leq \alpha < x_1$ and the vector $(x_1, 0) - (\alpha, \beta) = (x_1 - \alpha, -\beta)$ would contradict the minimality of x_1. The other case is identical. Thus the minimal elbow lies on an axis and the domain is therefore a rectangle. We learn from this that $x_1y_2 = a$,

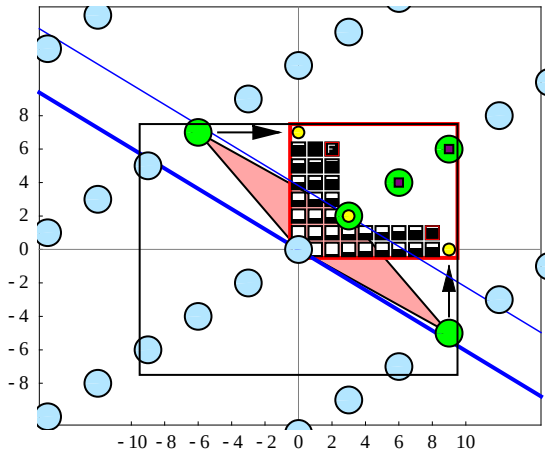

Figure 5: All disks are lattice points. The heavy blue line marks vectors of weight zero, with positive weight above. The lattice points inside the expanded bounding box and having positive weight are the five protoelbows, shown in green. The two with arrows are the axial protoelbows, and the arrows point to the axial elbows; the axial protoelbows are a basis for the lattice. The small purple squares are preelbows that are not elbows. The thin blue line is the contour for the minimal weight; this line passes through the minimal elbow, which is also the interior elbow. The pink parallelogram shows that the sum of the axial protoelbows is the interior elbow.

and because $x_1y_2 = x_1y_2 - x_2y_1$, the two axial protoelbows are again a basis for L and again $\mathcal{D}^* = \mathcal{D}$.

Continuing with the earlier example where $A = (33, 89, 147)$, Figure 5 shows all the protoelbows and preelbows. While some of the geometry is much simpler when $n = 3$, this diagram does show many aspects of the general case. The red rectangle is the bounding box K; the larger black one is $K^{\pm}$. The colored disks show all points of the lattice, with the blue line indicating points having weight exactly 0. The protoelbows, five of them indicated by green disks, are all the lattice points above the blue line and in $K^{\pm}$. The small purple squares indicate the preelbows that are not elbows and the yellow disks mark the three elbows. The arrows show the transformation from protoelbow to elbow. The two corners are outlined in red, the Frobenius corner is shown by a white F, and the black filling indicates weight in proportion to the maximum weight.

In the case of triples one can relate the shape of $\mathcal{D}$ to arithmetic properties of A. We do this in §18 where we show that several conditions are equivalent. For the geometry of $\mathcal{D}$ the noteworthy equivalence is: There is a permutation of A that is degenerate (i.e., $\mathcal{D}$ is a rectangle) iff some element of A is representable in terms of the other two reduced by their common divisor.

One additional useful fact is that, if the x_1 in (x_1, y_1) is known, then y_1 can be quickly computed.

Lemma 2. For each of the vectors (x_1, y_1), (x_2, y_2), knowledge of a positive entry is enough to compute the other entry of the vector using a few arithmetic operations and one call to the extended Euclidean algorithm.

Proof. Suppose x_1 is known. Then $(0, y_1)$ is the point in $\mathcal{D}$ whose weight is the same mod-a as the weight of $(x_1, 0)$. So to find y_1 we need to minimize $\beta \geq 0$ so that $\beta b \equiv \rho \pmod{a}$ where ρ is the least nonnegative mod-a residue of $x_1 b$; y_1 is then $-\beta$. Let $d = \gcd(a, b)$; let β_0 be the particular solution (reduced mod a) to the $\beta b \equiv \rho$ congruence given by $\left(\frac{b}{d}\right)^{-1} \frac{\rho}{d}$ where the inverse is modulo $\frac{a}{d}$. This inverse is where the extended Euclidean algorithm is used. Then standard elementary number theory shows that the general solution to the congruence is $\beta \equiv \beta_0 + ia/d \pmod{a}$ where $i = 1, 2, \ldots, d$. Since the right side is between 0 and $2a$, the value with the smallest residue is given by either the first one, $\beta_0 + \frac{a}{d}$, or the first one greater than a, which is $\left\lceil \frac{a-\beta_0}{a/d} \right\rceil \frac{a}{d} + \beta_0$. So we take the minimum of these two, reducing mod a first. The other case is similar. □

Our general elbow algorithm can be modified using some of the special structure that exists when $n = 3$. To repeat the main notation, recall that (x_1, y_1) and (x_2, y_2) are the active protoelbows for the axial elbows (meaning that the axial elbows are $(x_1, 0)$ and $(0, y_2)$), and (x_3, y_3) is the interior elbow if it exists. The main relationships are that $(x_3, y_3) = (x_1, y_1) + (x_2, y_2)$, $\det((x_1, y_1), (x_2, y_2)) = a$, and $g(A) = \max[\{(x_1, y_1 + y_2), (x_1 + x_2, y_2)\} \cdot B] - \sum A$ (this last is by computing the weight of the corners and taking the largest).

We first show how a recent planar integer-linear programming (ILP) algorithm of Eisenbrand and Rote determines the active protoelbows. We use this particular method in order to guarantee soft linear complexity.

Finding the Elbows and Frobenius Number by the Eisenbrand–Rote ILP Method

Input. A Frobenius basis $A = (a, b, c)$.
Output. The set of elbows and the Frobenius number of A.

Step 1. Form the homogeneous basis H and then use lattice reduction to obtain a reduced basis V. Using the straightforward extended Euclidean algorithm approach for finding H is adequate and has complexity $\tilde{O}(\log a)$. To get V one can use, instead of the classic LLL method, a special, much faster, planar lattice reduction; the complexity is known to be $\tilde{O}(\log a)$ [18, 28].

Step 2. Use the planar ILP algorithm of Eisenbrand and Rote [19] to determine the axial elbow $(x_1, 0)$. Note that this is a 2-variable problem with unknowns being the multipliers (i_1, i_2) where $(i_1, i_2) \cdot V$ is a generic lattice point: Minimize the first coordinate of $(i_1, i_2) \cdot V = (x_1, y_1)$ subject to the constraints $x_1 \geq 1$, $x_2 \leq 0$, and $(x_1, y_1) \cdot B \geq 1$. Once x_1 is computed, use Lemma 2 to compute y_1.

Step 3. Use (x_1, y_1) to determine (x_2, y_2) by the algebraic techniques of Theorem 3 below.

Step 4. If y_1 or x_2 is zero then the elbow set is $\{(x_1, 0), (0, y_2)\}$ (see end of first paragraph of this section); otherwise it is $\{(x_1, 0), (0, y_2), (x_1 + x_2, y_1 + y_2)\}$.

Step 5. In all cases the Frobenius number is $\max[\{(x_1, y_1 + y_2), (x_1 + x_2, y_2)\} \cdot B] - \sum A$.

The algorithm as presented, with Eisenbrand–Rote used once in step 2, has time complexity $\tilde{O}(\log a)$ in the worst case, since that is the worst-case time for step 1, for the Eisenbrand–Rote method, and for the arithmetic in Lemma 2 (the Euclidean algorithm is

softly linear [40]). We call this the ILP–ER method. The performance is remarkable since this shows that computing the Frobenius number of a triple is asymptotically not very much more time-consuming than using the ancient formula $ab - a - b$ when $n = 2$.

For a fast practical approach we present a simple heuristic for determining the active protoelbows that has the same complexity as ILP–ER but avoids implementing the ER step. One main idea is that a small expansion of the reduced basis for L is very likely to have one of the vectors (x_1, y_1), (x_2, y_2), (x_3, y_3) in it, and with one in hand, we can determine the other two by pure algebra. Note that these three vectors are the three active protoelbows, in that zeroing out the negatives in them yields the set of elbows. In the degenerate case there are only the two active protoelbows, (x_1, y_1) and (x_2, y_2). We first prove an algebraic theorem that shows why it is sufficient to have one active protoelbow.

Theorem 3. Given $A = (a, b, c)$, let (x_1, y_1), (x_2, y_2), (x_3, y_3) be, respectively, the two active axial protoelbows and the interior elbow (if it exists). From any one of these one can deduce the values of the others by a bounded number of arithmetic operations (we allow the asymptotically fast variants of the Euclidean algorithm as an operation since the point is to be certain we have complexity that is soft linear and, indeed, no worse than a constant times asymptotically fast GCD).

Proof. Recall from comments following Proposition 4 that (x_3, y_3), when it exists, is just the sum of the other two. Let $\boldsymbol{v}_1 = (x_1, y_1)$; we will show how to get (x_2, y_2).

Let V be a basis for the homogeneous lattice L; if $\det(V) = -a$ interchange the rows of V so that the determinant is $+a$. Let $\boldsymbol{i}$ be the coefficients of $\boldsymbol{v}_1$ w.r.t. V: so $\boldsymbol{i} \cdot V = \boldsymbol{v}_1$; $\boldsymbol{i}$ is just $\boldsymbol{v}_1 V^{-1}$. Find two integers $\boldsymbol{u} = (u_1, u_2)$ such that $\det(\boldsymbol{i}, \boldsymbol{u}) = 1$, which is easy by elementary number theory, and define $\boldsymbol{W}$ to be $\boldsymbol{u} \cdot V$. The pair $(\boldsymbol{v}_1, \boldsymbol{W}) = (\boldsymbol{i} \cdot V, \boldsymbol{u} \cdot V) = (\boldsymbol{i}, \boldsymbol{u}) \cdot V$. Applying determinants yields $\det[(\boldsymbol{v}_1, \boldsymbol{W})] = \det[(\boldsymbol{i}, \boldsymbol{u})] \cdot \det(V) = \det(V) = a$.

Now set $\boldsymbol{v}(j) = j\boldsymbol{v}_1 + \boldsymbol{W}$. We know that $\det(\boldsymbol{v}_1, \boldsymbol{W}) = a$, and this means that (x_2, y_2) must equal $\boldsymbol{v}(j)$ for some integer j. This is because $\boldsymbol{v}_1$ and $\boldsymbol{W}$ generate the lattice, which contains (x_2, y_2). So $(x_2, y_2) = j\boldsymbol{v}_1 + r\boldsymbol{W}$ for some coefficients j and r. This means that $a = \det(\boldsymbol{v}_1, (x_2, y_2)) = \det(\boldsymbol{v}_1, j\boldsymbol{v}_1 + r\boldsymbol{W}) = r \det(\boldsymbol{v}_1, \boldsymbol{W}) = ra$, so $r = 1$, as claimed.

We know (x_2, y_2) satisfies five inequalities: $(x_2, y_2) \cdot B \geq 0$, $x_2 \leq 0$, $y_2 \geq 1$, and, by Proposition 4, $x_1 + x_2 \geq 0$ and $y_1 + y_2 \geq 0$. The second of these five is just $j \leq -W_1/x_1$ and the fourth is $x_1 + jx_1 + W_1 \geq 0$. These combine to force $-\frac{W_1}{x_1} - 1 \leq j \leq -\frac{W_1}{x_1}$, which means that j, which must be as large as possible in order to minimize the second coordinate of $j\boldsymbol{v}_1 + \boldsymbol{W}$ (the coefficient of j in the second coordinate is nonpositive), must be $\lfloor -W_1/x_1 \rfloor$. This gives (x_2, y_2). It is possible that $y_1 = 0$, which means that the second coordinate of $\boldsymbol{v}_j$ is independent of j. Then the choice of j must be to maximize the negative quantity x_2; but since the coefficient of j in x_2 is always strictly positive, this again indicates that the choice should maximize j.

The other two cases are similar, with the constraints yielding the proper j value as a quotient. We omit the details except to say that if $\boldsymbol{v}_3 = (x_3, y_3)$ is in hand then one defines $\boldsymbol{W}$ so that $\det(\boldsymbol{W}, \boldsymbol{v}_3 - \boldsymbol{W}) = \pm a$. □

Corollary 3. Any of the three active protoelbows can be certified in softly linear time.

Proof. Let us show how to certify (x_1, y_1). First verify that $x_1 \geq 1$, $y_1 \leq 0$, $(x_1, y_1) \cdot B \geq 1$ and $(x_1, y_1) \cdot B$ is divisible by a. All this means that the corresponding elbow $(x_1, 0)$ is an upper bound on the true axial elbow and so its cone is excluded from $\mathcal{D}$. Use Theorem 3 to compute (x_2, y_2) and verify that $x_2 \leq 0$, $y_2 \geq 1$, $(x_1, y_1) \cdot B \geq 0$, and $(x_1, y_1) \cdot B$ is divisible by a. This means that $(0, y_2)$ is an upper bound on the second axial elbow and its cone is excluded from $\mathcal{D}$. Note that the construction of Theorem 3 guarantees $x_1 y_2 - y_1 x_2 = a$.

If neither y_1 nor x_2 is 0, we check that $(x_3, y_3) = (x_1, y_1) + (x_2, y_2)$ lies in the first quadrant. Because $(x_3, y_3) \in L$, its cone is excluded from $\mathcal{D}$. The area is the domain determined by the three elbows is therefore correct, and this certifies x_1 and y_2; the nonpositive entries are certified because if the true values were any larger, the determinant would be larger, and therefore incorrect.

If y_1 is 0, then there is no need for further certification because Theorem 3 guarantees $x_1 y_2 = a$, which certifies x_1 and y_2. Moreover, the theorem always produces the optimal value of x_2 to accompany y_2, so that number is correct too. Finally, it is easy to certify the 0, since one need only check that $0 \cdot c \equiv x_1 b \pmod{a}$, as this means that $(0, 0)$ is the point in $\mathcal{D}$ equivalent to $(x_1, 0)$. The case $x_2 = 0$ is similar.

The certification process for (x_2, y_2) or (x_3, y_3) is entirely analogous. □

We now present a heuristic that is very good at finding at least one of the active protoelbows quickly. In more than a million trials this method, which we call *LLLMult* because of its use of small multiples, has not failed and it has the same worst-case performance as ILP–ER. And the absolute times are very fast: it is nearly instantaneous up to 1000 digits and can handle million-digit inputs in a couple of minutes. Another way of viewing the complexity of LLLMult is by saying that, if it falls into step 7 with failure, we resort to ILP–ER or Greenberg–Davison; since failure is so rare, this would yield an algorithm that, we conjecture, has average-case complexity $\tilde{O}(\log a)$.

There are four ideas in this heuristic. First, because the two active axial protoelbows form a basis (call it P), we can compute $U = P \cdot V^{-1}$ for several thousand cases, where V is the usual reduced basis of L, to see which unimodular matrices U arise. It turns out that, because V is often "close" to being P, the matrices in U have small integer entries, and we can find 20 of them, say, that cover about 90% of the cases. This the algorithm can start by just trying $U \cdot V$ for these 20 to see if the desired two vectors result. If not, we move to the second step, which is to take small multiples of the vectors in V to see if two of them are two of the active protoelbows. If this fails, we can see if we have captured at least one of the active protoelbows by trying the procedure of Corollary 3. And if all this fails (we know of no examples) we can use an ILP method to find (x_1, y_1) by integer optimization.

For the bound on the multipliers in the second phase we try 3 first, and if that fails we move up to 20. Examples that require a multiplier bound greater than 3 are very rare. We found one such in two million trials, and increasing the bound to 4 then succeeds. The

paucity of failures can be explained roughly as follows. The axial protoelbows satisfy certain constrained minimality of one component. It would be surprising in general if these arose with large component in the other position. Hence for most cases one might expect these to be attained as small multiples of shortest independent lattice generators. As the interior elbow is their sum, it too will be so attained in general.

Finding the Elbows and Frobenius Number by an LLL Multiplier Heuristic when $n = 3$

Input. A Frobenius basis $A = (a, b, c)$.
Output. The set of elbows and the Frobenius number of A.

Step 1. Form the homogeneous basis H and use fast planar lattice reduction to obtain a reduced basis V. If a vector in V has negative weight (i.e., dot product with (b, c)), replace it with its negative. If then $\det V = -a$, reverse the order of the vectors so that $\det V = a$.

Step 2. For each of 25 unimodular matrices U obtained by experiment, check to see if $U \cdot V$ contains the two active axial protoelbows. If the two vectors, in reverse sorted order, are $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, one need only check that the sign patterns are right, the weights are at least 1 for $\boldsymbol{v}_1$ and 0 for $\boldsymbol{v}_2$, and the sum $\boldsymbol{v}_1 + \boldsymbol{v}_2$ lies in the first quadrant. The determinant is automatically a, so these conditions are enough to certify correctness as in Corollary 3. This step succeeds in over 90% of the cases.

Step 3. Form the set $\mathcal{S}$ of small multiples $\{i_i, i_2\} \cdot V$ where the coefficients vary from -3 to 3. If a vector in this set has negative weight, multiply it by its negative.

Step 4. Let $\boldsymbol{v}_1$ be the vector in $\mathcal{S}$ with sign pattern $(+, -)$ and weight at least 1, and with smallest first coordinate. Let $\boldsymbol{v}_2$ be the vector in $\mathcal{S}$ with sign pattern $(-, +)$ and with smallest last coordinate. Let $\boldsymbol{v}_3$ be the vector in $\mathcal{S}$ with sign pattern $(+, +)$ and with smallest weight. Here "+" means positive and "−" means nonpositive. It is possible that in some of these three cases there will be no such vectors.

Step 5. If step 4 yields at least two vectors, check the pairs among them to see if any of the determinants of $\{\boldsymbol{v}_1, \boldsymbol{v}_2\}$, $\{\boldsymbol{v}_3 - \boldsymbol{v}_2, \boldsymbol{v}_2\}$, or $(\boldsymbol{v}_1, \boldsymbol{v}_3 - \boldsymbol{v}_1)$ equals a. If so, use the certification process of Corollary 3; if this verifies correctness of the two vectors, we can get the third active protoelbow (if it exists) and then compute $g(a, b, c)$ as $\max[\{(x_1, y_1 + y_2), (x_1 + x_2, y_2)\} \cdot B] - \sum A$. We are done.

Step 6. If step 5 fails to find a pair of the active protoelbows, check each single vector $\boldsymbol{v}_i$ from step 5 in turn, using Corollary 3 to see if it is the active protoelbow matching its sign pattern. Stop if successful and compute $g(a, b, c)$ as in step 5.

Step 7. If step 6 fails, start over at step 2 with the bound on multiples increased from 3 to, say 20. We know of no examples where this fails.

Step 8. If step 7 fails after the increase to 20, use a variation on branch-and-bound ILP to carry out step 2 of the ILP–ER algorithm. While we have no proof that it is softly linear like ILP–ER, it appears to be so in practice.

Implementation Notes. 1. When forming the multiples in step 3, we omit coefficients such as $(2, 2)$, since $(2, 2) \cdot V$ will be larger in any positive entry than $(1, 1) \cdot V$ which is already included.

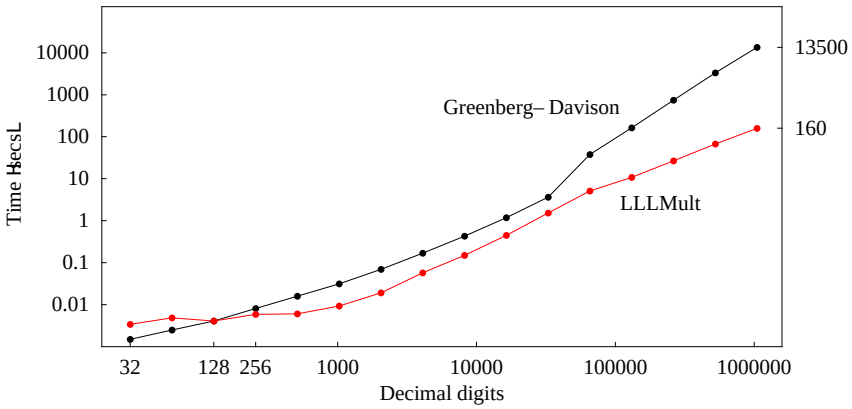

Figure 6: Comparison of the Greenberg–Davison method for $g(A)$ when $|A| = 3$ with the LLLMult method (red), for inputs having up to one million digits. The LLLMult data was averaged over 10 trials per data point. The scale is logarithmic in both axes.

2. The ILP in step 8 works as follows. Recall we have a reduced basis comprised of $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$, and we seek an axial protoelbow $\boldsymbol{v}_1$ that we know is an integer combination of these. Moreover we have linear constraints that must be satisfied and a minimality condition that must be met by the positive (that is, first) coordinate. We set up the equation $\boldsymbol{v}_1 = i_1\boldsymbol{V}_1 + i_2\boldsymbol{V}_2$. Then the ILP problem to solve is to minimize $v_{1,1}$ subject to constraints $v_{1,1} \geq 1$, $v_{1,2} \leq 0$, $\boldsymbol{v}_1 \cdot B \geq 1$. These translate immediately into conditions on i_1 and i_2. We solve this by solving relaxed LPs within a standard branch-and-bound loop [41]. While we cannot prove it, our experience indicates the complexity of this step to be softly linear in the bit size of the Frobenius basis A. We believe this is due to use of a reduced basis $\{\boldsymbol{V}_1, \boldsymbol{V}_2\}$, as that seems to keep the number of branch-and-bound nodes small.

The complexity of LLLMult up to step 7 is provably softly linear because the reduction is, and the other steps only use a small number of arithmetic or Euclidean algorithm operations. Step 8 is a problem, but (a) we have never found an example requiring that step, and (b) experiments show that step 8 is softly linear, though we cannot prove it. Thus we conjecture that, perhaps with a constant larger than 3 in step 3, the algorithm up to step 7, which we know to be softly linear, is a correct algorithm for $g(a, b, c)$. And we also conjecture that the algorithm through step 8, which we know is correct, works in worst-case soft linear time when using ILP as described in [2] or [29] and coupled with the planar lattice reduction of [18] (implementation details shown in [28]).

In practice the performance of LLLMult is excellent; it can handle million-digit inputs in under three minutes. Figure 6 compares the performance of LLLMult to the fastest way we have been able to implement the Greenberg–Davison method, which is softly quadratic. It is possible that one could find faster implementations of GD, but we suspect that even the best implementation, while it might be softly linear, would still be slower than LLLMult. At one million digits LLLMult is better by a factor of about 80.

6. A Typical Case: $n = 4$

Once n is 4 or more the fundamental domain becomes more complicated as there is no bound on the number of elbows. The main thrust of this paper is that one can devise an algorithm that can find all the elbows and corners, and so understand the geometry of the domain, even when there are millions of elbows. Here is an example to clarify the central concepts. Let $A = (50, 69, 89, 103)$. Figure 7 shows the fundamental domain in gray with the elbows in yellow and the corner as gray boxes wrapped in cyan. Figure 8 shows more of the construction. The protoelbows are green, with black lines connecting the axial protoelbows to the axial elbows. Only the protoelbows that correspond to one of the seven elbows are shown (the interior one at $(3, 2, 5)$ coincides with the yellow elbow); there are four additional protoelbows whose corresponding preelbows dominate one of the elbows. The red frame shows the expanded bounding box. The zero-weight plane is shown in light blue.

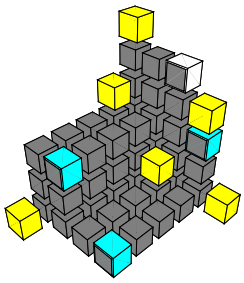

Figure 7: The fundamental domain and set of elbows (yellow) for $A = (50, 69, 89, 103)$. The corners are in cyan with the Frobenius corner in white. The axial elbows are $(5, 0, 0)$, $(0, 4, 0)$, and $(0, 0, 5)$ so the bounding vector $\boldsymbol{k}$ is $(5, 4, 5)$.

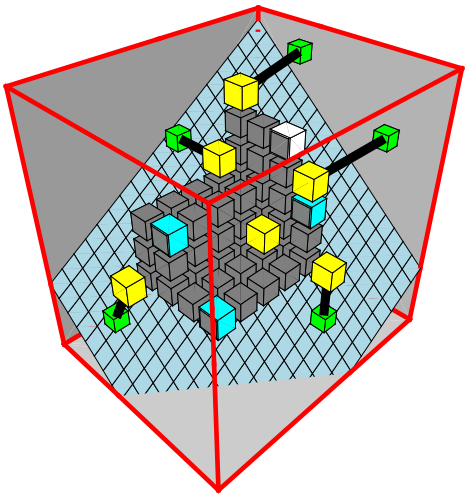

Figure 8: The same example as in Figure 7, but shown with the bounding box, the set of protoelbows that yield elbows (green), and the plane marking the vectors of weight 0. Black lines connect the protoelbows, $(5, -1, -2)$, $(0, 4, -2)$, $(-4, -1, 5)$, $(-4, 3, 3)$, $(1, -2, 3)$, to the corresponding elbows. The interior elbow at $(1, 2, 1)$ is also a protoelbow (as interior elbows always are).

7. Finding All Elbows: The Vanilla Algorithm

Our attack on the Frobenius number is similar to how it is done when $n = 3$: find the elbows and use them to find the corners. When $n = 3$ there are always two axial elbows and, usually, one interior elbow. These points can be found directly by number theory (extended Euclidean algorithm; see [20, 15]) or other methods (§5). And once the elbows are in hand, it is immediate to get the (one or two) corners and determine the Frobenius corner. But the situation is much more complicated when $n \geq 4$; for example, the number of elbows, even when restricted to one of the coordinate planes, is not bounded, as pointed out in §3.

Here is an overview of how to find all elbows and corners, and hence the Frobenius number. We call this the *vanilla algorithm*, because it attacks each step in the most straightforward way. When we discuss the individual steps in detail we will see how many of them can be enhanced to improve speed or cut down on memory requirements. For example, in §13 we show how to compute the axial elbows exactly by integer-linear programming (ILP), and knowing them (that is, knowing the bounding box) cuts down on the search time significantly. But in fact we can very quickly bound the axial elbows by the simple center-line method and so, in the vanilla algorithm, we will work only with upper bounds on the axial elbows. This algorithm is therefore quite self-contained and already more efficient than any other known methods for getting $g(A)$.

Throughout this paper we work with integer lattices. To make such work computationally feasible, we usually require that the lattices be reduced using the Lenstra–Lenstra–Lovász algorithm [26]. Roughly speaking, this is in order to keep the size of integers as small as possible, so as to reduce sizes of search spaces and make for convenient searching directions.

The Vanilla Algorithm for Elbows, Corners, and the Frobenius number

Input. A Frobenius basis A.
Output. The set of all elbows, the set of all corners, and the Frobenius corner, whose weight, less a, is the Frobenius number $f(A)$.

Step 1. Use linear Diophantine theory to find a basis for the homogeneous lattice L (this can be done by the extended Euclidean algorithm and induction ([13], ex. 2.4, [33]), or using the more general Hermite normal form [9].

Step 2. Use lattice reduction (LLL) on the basis of step 1 to get V, a reduced basis for L.

Step 3. Use the center-line algorithm to find bounds $\boldsymbol{k}^+$ on the axial elbows.

Step 4. Find the set $\mathcal{I}$ of multipliers $\boldsymbol{i} \in \mathbb{Z}^m$ contained in the bounded polytope described by the conditions $-\boldsymbol{k}^+ \leq \boldsymbol{i} \cdot V \leq \boldsymbol{k}^+$ and $W(\boldsymbol{i} \cdot V) \geq 0$. A simple recursive search, using linear programming to obtain bounds on $\boldsymbol{i}$, works well.

Step 5. Remove from $\{\boldsymbol{i} \cdot V : \boldsymbol{i} \in \mathcal{I}\}$ elements $\boldsymbol{X}$ with $W(\boldsymbol{X}) = 0$ and the first nonzero coordinate positive. The set that remains is a superset of the set of protoelbows.

Step 6. Take the lattice vectors from step 5, replace negatives with 0, and eliminate any duplicates. The set that remains is a superset of the set of preelbows.

Step 7. Turn the points of step 6 into elbows by taking the domination kernel of the set of preelbows. An algorithm due to Bentley et al [8] does this efficiently. Aside: The axial vectors in this set will be the set of true axial elbows.

Step 8. Use the elbows to determine the set of corners (recursion on dimension) and find the corner $\boldsymbol{C}$ of maximum weight.

Step 9. The weight of $\boldsymbol{C}$ is then $g(A) + a$. So $g(A) = W(\boldsymbol{C}) - a$ and $G(A) = W(\boldsymbol{C}) + \Sigma B$.

Steps 5, 6, and 9 are simple, requiring no further discussion; steps 1 and 2 use standard algorithms. But each of the other steps needs to be discussed in some detail. The bottlenecks of this algorithm are steps 4 and 8, since there can be a lot of protoelbows and elbows. However, when n is fixed the sizes of these sets grow only slowly as a grows. Step 7 is a secondary bottleneck, but takes less time than the protoelbow search. This vanilla algorithm can find $g(A)$ in reasonable time when $n \leq 7$. Beyond that, the use of the bounds rather than the true axial elbows in step 4 is a problem, and we will show in §13 how to compute the exact axial elbows. Also step 8 can be improved by arranging the search so that only the farthest corner is found, as opposed to all of them. With these and other improvements the algorithm works when $n \leq 10$ in reasonable time. When $n = 11$ the set of protoelbows is about half a billion and things start to slow down at several of the steps, but programming in a way that avoids excessive memory consumption we were able to do a case with $a = 10^{10}$ and $n = 11$ in 44 hours; there were 27037 elbows.

Example

Here is an example showing some of the steps of the vanilla algorithm. We include it so that the reader can follow some of the steps, but also to provide data for the reader interested in programming these methods, perhaps in a programming environment different than ours.

$n = 5; a = 10^{10}$; $B = (18543816066, 27129592681, 43226644830, 78522678316)$; $A = (10^{10}, B)$

Homogeneous basis of lattice L from Hermite normal form:
$\{(0, 0, 6696459318, -3686393215), (1, 0, 4198679393, -2311368216),$
$(0, 2, -2319663279, 1276971988), (0, 0, 39261339158, -21613322415)\}$

Reduced homogeneous basis V:
$\{(-165, -174, -80, -26), (30, -164, 116, -236),$
$(129, 110, -242, -179), (-229, 180, -15, -326)\}$

Axial protoelbows:
$(553, -60, -31, -63)$, $(-94, 518, -51, -64)$, $(-586, -314, 827, -25)$,
$(-195, -10, -196, 210)$

Axial elbow bounds: $(553, 0, 0, 0)$, $(0, 518, 0, 0)$, $(0, 0, 827, 0)$, $(0, 0, 0, 210)$

Bounding vector, $\boldsymbol{k}^+$: $(553, 518, 827, 210)$ (the true axial elbow vector $\boldsymbol{k}$ is $(553, 518, 358, 210)$).

The size of the superset of protoelbows resulting from step 5: 41.

The elbows obtained by the domination kernel applied to the superset of protoelbows:
$(0, 0, 0, 210)$, $(0, 0, 162, 153)$, $(0, 0, 358, 0)$,
$(0, 234, 111, 89)$, $(0, 244, 307, 0)$, $(0, 508, 0, 146)$,
$(0, 518, 0, 0)$, $(165, 174, 80, 26)$, $(259, 0, 131, 90)$, $(264, 448, 0, 83)$, $(358, 0, 0, 147)$,
$(360, 184, 276, 0)$, $(454, 0, 327, 0)$, $(459, 458, 0, 0)$, $(553, 0, 0, 0)$

The corners (24), by recursion, sorted by weight:
$(359, 243, 357, 25)$, $(453, 183, 357, 25)$,
$(552, 183, 326, 25)$, $(164, 243, 357, 88)$, $(552, 173, 130, 146)$,
$(458, 517, 79, 82)$, $(552, 457, 79, 82)$, $(164, 233, 161, 209)$,
$(258, 173, 161, 209)$, $(164, 517, 110, 145)$,
$(357, 173, 130, 209)$, $(263, 517, 79, 145)$,
$(453, 173, 357, 89)$, $(359, 517, 306, 25)$, $(552, 173, 326, 89)$,
$(458, 517, 275, 25)$, $(552, 457, 275, 25)$,
$(164, 233, 357, 152)$, $(258, 173, 357, 152)$, $(164, 517, 306, 88)$,
$(552, 447, 79, 146)$, $(164, 507, 110, 209)$, $(263, 507, 79, 209)$, $(357, 447, 79, 209)$

8. Center-line Algorithm to Bound the Bounding Box

A rather simple algorithm works surprisingly well to find bounds on the axial elbows, needed for step 3 of the vanilla algorithm. The idea is to start with a multiplier vector that has almost no chance of working and modify it in a very simple manner to one that has a certain chance of working. We call it the center-line algorithm, and it works in polynomial time. The main idea is that we seek a vector in L having a particular sign pattern. We form vectors that a priori have the correct sign pattern and annihilate X, but whose components are not generally integers. Rounding the multipliers (with respect to the basis V_B) makes them integers but they may now stray outside the allowed region. We will show that for sufficiently large multipliers this does not happen.

It is simpler to work here in the expanded lattice L_B. Start with $\boldsymbol{w} = (-1, \ldots, -1, m, -1, \ldots, -1)$, a vector in $\mathbb{Z}^m$ that has the proper sign structure for providing a bound on the axial elbow, but is almost surely not in L_B. We let $\boldsymbol{c}$ be the coordinates of this vector in the multiplier space ($\boldsymbol{c} = \boldsymbol{w} \cdot V_B^{-1}$), the idea being that we wish to look at multiples $i\boldsymbol{c}$ (upper left of Fig. 9). These are not integers, so using them as multipliers generates points that are not integers. But we can round the multiples to integers (upper right of Fig. 9). These integer multipliers then generate points of the lattice L_B (lower right of Fig. 9). The idea is to check these lattice points in order until one of them works, meaning that it is in the appropriate region to yield a bound on an axial elbow (details in step 6 below). Now, this

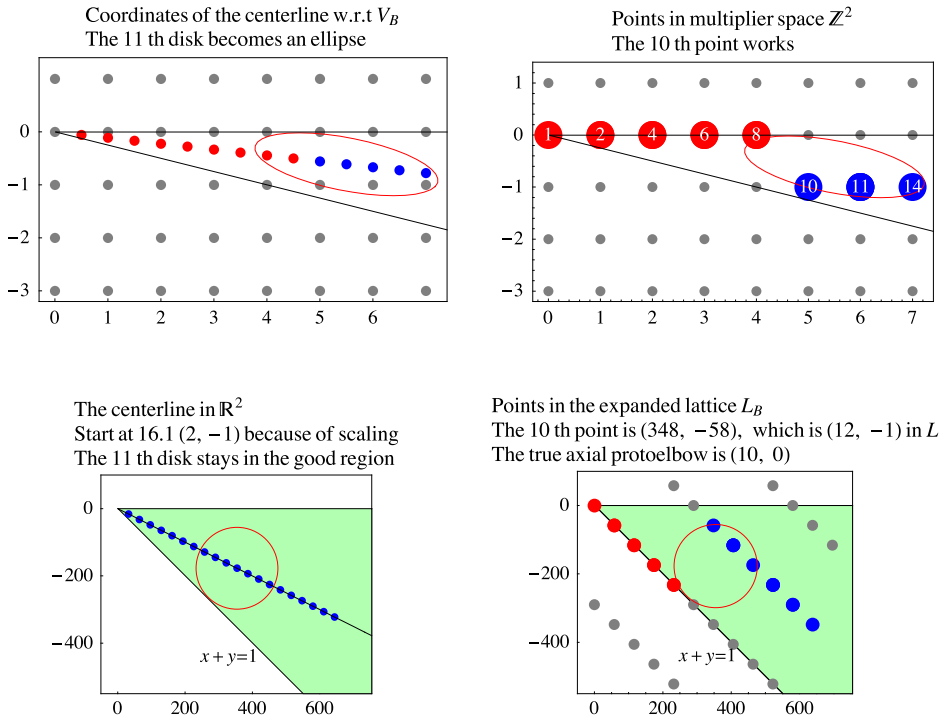

Figure 9: The workings of the center-line method for $A = (10, 29, 58)$. The 10th point (6th if we ignore repetitions) is $(348, -58) = (5, -1) \cdot V_B$ in L_B. This corresponds to $(12, -1)$ in L and means that 12 is an upper bound on the first axial elbow (which is in fact 10).

naive view will cause us to look at too many multiples, since the rounding of, say, $\boldsymbol{c}$ and $2\boldsymbol{c}$ might well be the same. Thus we scale the sequence by setting $\delta = 1/(2 \max |\boldsymbol{c}|)$ and using multiples of $\delta\boldsymbol{c}$ instead of multiples of $\boldsymbol{c}$. This scaling makes the largest entry of $\delta\boldsymbol{c}$ exactly $1/2$ in absolute value, and so multiples will generally lead to new rounded values. The points shown in Figure 9 include this scaling; the numbers in the points at upper right are the indices of first appearance; the fact that 3 is missing means that the 2nd and 3rd rounded points were the same.

It is not hard to see that this iteration must eventually halt. What is surprising is how good the results are as an upper estimate to the axial elbows. To see that it will eventually halt, note that the rounding moves the point in the lattice by no more than $\|\boldsymbol{v}_1 + \cdots + \boldsymbol{v}_m\|/2$ where the $\boldsymbol{v}_i$ form the basis V_B. Since the distances of the unrounded points in the lattice from the boundary of the good region increases without bound, we will eventually get the desired lattice vector.

The diagrams in Figure 9 show the method for a simple 2-dimensional case: $A = (10, 29, 58)$. We start at the lower left with the scaled points in $\mathbb{R}^2$. The circle around the 11th point stays within the green region, so we know we will have success at that iteration or before. The points in the coordinate space are at the upper left and right, and the circle has become an ellipse. These points are rounded in the upper right, and then brought into the lattice L^B at the lower right. The 10th point, which is the sixth distinct point, is

the first one that works.

Here is one example that shows how efficient this method is. The Frobenius basis is

$$(2361342111, 2668847064, 3509684908, 3713675653, 6372498435, 7392744209, 8135623128).$$

The center-line method yields $(210, 92, 180, 153, 98, 73)$ with the number of iterations being 10, 3, 17, 9, 6, 3, respectively. The actual axial elbow vector is $(201, 92, 154, 77, 79, 48)$.

When the ratio of the length of the longest basis vector in V_B to the length of the shortest is large then the first i that works might be quite large, and going through them one at a time is too inefficient. If we use an exponential pattern, such as $i, 2i, 4i, \ldots$, then we will of course go through the list much faster. Indeed, this will guarantee that the algorithm always halts in polynomial time. But always using powers of 2 would miss the first i by too much in too many cases. So in the algorithm below we use a function such as $i \mapsto \lceil C_{\text{mult}} \rceil i + C_{\text{add}}$, where C_{mult} is some multiplier, and C_{add} an adder. Defaulting C_{add} to 1 is no problem, but it is best to use, say, $C_{\text{mult}} = 1.1$, as a reasonable compromise between speed and tightness of bound.

Center-Line Algorithm to Bound the Bounding Box

Input. A Frobenius n-basis $A = (a, B)$; $m = n - 1$.
Output. A vector $\boldsymbol{k}^+$ that is an upper bound in each coordinate on the bounding vector $\boldsymbol{k}$.

For $j = 1, \ldots, m$ obtain k_j^+ as follows:

Step 1. Compute a basis for the homogeneous lattice L, and then reduce it to get V, a reduced basis for L, and expand it to V_B, a basis for the lattice L_B.

Step 2. Let $\boldsymbol{w} = (-1, -1, \ldots, -1, m, -1, \ldots, -1)$, a vector with m entries where the positive entry is in the jth coordinate.

Step 3. Let $\boldsymbol{c} = \boldsymbol{w} \cdot V_B^{-1} \in \mathbb{Q}^m$, the coordinates of $\boldsymbol{w}$ in the expanded lattice.

Step 4. Set the scaling factor $\delta = 1/(2 \max |\boldsymbol{c}|)$.

Step 5. Define $\boldsymbol{P}(i)$ to be $\text{round}(\boldsymbol{c} i \delta) \cdot V_B$.

Step 6. Choose a real multiplier $C_{\text{mult}} \geq 1$ and an integer adder $C_{\text{add}} \geq 1$. If $C_{\text{mult}} > 1$ then the algorithm will be polynomial-time in all cases, but might not find the first i that will work in step 7; if $C_{\text{mult}} = C_{\text{add}} = 1$ then the first i will be found and the bound will be best possible using this method, but in some cases the algorithm will not halt in polynomial time.

Step 7. Start with $i = 1$ and find the first i so that $\boldsymbol{P}(i)$ is a bound on the axial elbow (check that $\boldsymbol{P}(i)_j \geq 1$, $\boldsymbol{P}(i)_s \leq 0$ for $s \neq j$, and either $\sum \boldsymbol{P}(i) \geq 1$ or $\sum \boldsymbol{P}(i) = 0$ and the first nonzero entry of $\boldsymbol{P}(i)$ is negative). Increment i by replacing it with $\lceil C_{\text{mult}} i \rceil + C_{\text{add}}$. If i ever gets past $i_0 = 2 \text{ numerator}(\max |\boldsymbol{c}|) \prod A$, set i to i_0.

Step 8. Return the jth coordinate of $\boldsymbol{P}(i)$ found in step 6.

An implementation note: In step 3 one can use numeric values in V_B and a linear solver on $\boldsymbol{c} \cdot V_B = \boldsymbol{w}$ instead of computing the full inverse. For in an extreme case (100×100 matrix of 100-digit numbers) computing the full rational inverse will be slow. Of course, using numerics can introduce error, but the rounding that takes place in step 5 will still work and, so long as step 7 halts, the answer will be a correct bound. But for small values of n computing the full rational inverse is not only faster, it is more accurate, and so finds the correct i value sooner, thus leading to better bounds.

We can prove that when $C_{\text{mult}} > 1$ the method always halts in polynomial time. The proof will be a worst-case analysis, but that is overly pessimistic in practice. For example, in the $n = 7$ example given above the largest number of iterations needed was 17 when the multiplier was 1. The bound of the proof is $\prod A$, which is 10^{67}. The complexity proof says nothing about how good the bounds are, and we have no proof related to that. But in practice they seem to be very good (see end of section for more on this point).

We use the soft-O notation $\tilde{O}(\Lambda^w)$ to abbreviate $O(\Lambda^{w+\epsilon})$ for any $\epsilon > 0$. This notation is useful especially for multiplication of two numbers of bit length λ, since that can be done in time $O(\lambda \log \lambda \operatorname{loglog} \lambda)$ when an FFT-based method is used; and this is $\tilde{O}(\lambda)$. We always assume that A contains distinct entries and $\max A = O(a)$; this implies that $m = O(a)$.

Theorem 4. The center-line algorithm on input A with multiplier $C_{\text{mult}} > 1$ halts in time $\tilde{O}(m^5(\log a)^2)$. If the dimension of A is fixed, the complexity is $\tilde{O}((\log a)^2)$.

Proof. For the complexities we will separate out the two parts of the input, m and $\log a$. The basic algorithms used are multiplication, getting the homogeneous basis, reducing that basis, and inverting a matrix and these are all well known to work in polynomial time; see [26, 17, 43]. To be precise, the homogeneous basis can be obtained in time $\tilde{O}(m^3 \log a)$, but the numbers in the basis have size $O(\log a)$. Then getting the reduced basis V requires $O(m^3 \log a)$ arithmetic steps, with the operands in each step of size $O(m \log a)$. Thus the overall time for the lattice reduction step is $\tilde{O}(m^4(\log a)^2)$. There are m^2 integers in V, and the largest has bit length $O(m \log a)$ in the worst case (in practice we found that the entries in V are more like $(\log a)/m$).

It follows that this size bound applies also to V_B, and therefore if t is the largest integer in V_B in absolute value, t has bit length $O(m \log a)$.

Inverting a matrix takes $O(m^3)$ multiplications when done by one-step row reduction. But the numbers can expand in this computation; Cramer's rule and the Hadamard bounds show that they do not expand beyond $O(m \log m + m^2 \log a)$, which, by the remark preceding the theorem, is $O(m^2 \log a)$. Therefore the time needed to form V_B^{-1} is $\tilde{O}(m^5 \log a)$.

Next, pretend that the scaling factor δ is just 1. In that case $\prod A$ works in step 7. The algorithm looks at points $\boldsymbol{Q} = i(-1, -1, \ldots, -1, m, -1, \ldots, -1)$, a point in $\mathbb{Z}^m$ that has the correct sign structure and would give a bound if it were in the lattice L_B. When $i = \prod A$, then this point is in fact in the lattice: $\boldsymbol{Q}/B$ is an integer vector and $(\boldsymbol{Q}/B) \cdot B$ is divisible by a. In the algorithm, there will be no rounding in step 5. Thus, if the multiplier is taken to be 2, the

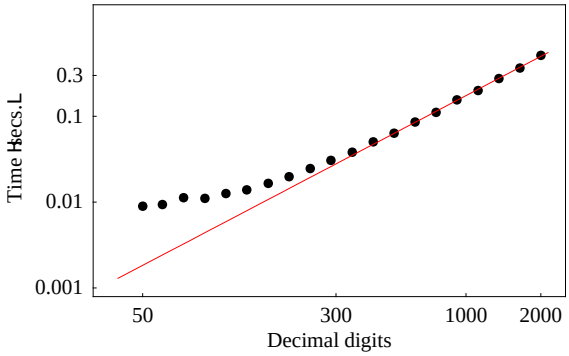

Figure 10: The time needed for the center-line method, with both scales logarithmic and 20 trials for each data point. The observed growth rate is about $O((\log a)^{1.53})$.

number of times step 7 is run is at most $\log_2 a + \sum \log_2 b_j$, which is exactly the input length. So long as the multiplier is greater than 1, this number of steps will be polynomial in the input length. To handle the scaling factor, i must kill the denominator of δ and have enough left over to get the needed $\prod A$. So i can be denominator$(\delta) \cdot \prod A = \pm 2$ numerator$(c_k) \cdot \prod A$. Now, the numerator, by Cramer's rule for the inverse, is no larger than a certain cofactor of V_B. But, by Hadamard's bound on the determinant, any such cofactor is at most $t^m m^{m/2}$. An i that works is therefore bounded by $2t^m m^{m/2} \prod A$, and the base-2 logarithm of this expression bounds the number of steps needed when C_{mult} is 2.

$$\begin{aligned} \log_2 \left(2t^m m^{m/2} \prod A \right) &= O\left(m \log_2 t + \frac{m}{2} \log_2 m + m \log a \right) \\ &= O(m^2 \log a + m \log m + m \log a) = O(m^2 \log a) \end{aligned}$$

Thus the number of i-values to try to reach an i that works, using a multiplier of 2, is $O(m^2 \log a)$. But each step involves the arithmetic of steps 5 and 7 on numbers of bit length $O(m^2 \log a)$, and so the total complexity is $\tilde{O}(m^4 (\log a)^2)$ for this part of the proof. Since the first part needed $\tilde{O}(m^5 (\log a)^2)$, the overall complexity of the algorithm is $\tilde{O}(m^5 (\log a)^2)$. When m is fixed, the general result specializes to $\tilde{O}((\log a)^2)$. □

Figure 10 shows the results of an experiment with $n = 4$ to confirm the complexity analysis. The exponent of about 1.5 is consistent with the theorem since the multiplications and other operations that *Mathematica* uses in this range may not be the asymptotically fastest.

The axial elbow bounds $\boldsymbol{k}^+$ obtained by the center-line method give an upper bound on the Frobenius number, since $g(A) \le \boldsymbol{k}^+ \cdot B - \sum A$. In a typical case with $a = 10^{10}$ and $n = 6$, $g(A) = 1.46 \cdot 10^{13}$ while this bound is $8.0 \cdot 10^{13}$. In 100 such cases, the 300 axial elbow bounds differed from the true values by a factor of 4.9 in the worst case and a mean of 1.33. The corresponding upper bound on $g(A)$, when interpreted as a bound on $\log_{10} g(A)$, had average relative error of 4.4%. So we can summarize by saying that the method is a true polynomial-time algorithm that works in all cases and provides bounds that appear to be quite good—far better than any other such method in the literature. For example, the

method of [7], which basically uses $g(a, b_1, b_2)$ as an upper bound, has a mean $\log_{10}$ relative error of 18% on these 100 examples.

But note that when n and a are of modest size we can use the bisection method (see §13) to find the true axial elbow vector $\boldsymbol{k}$, which will of course yield a tighter upper bound. So a more realistic application would use $a = 10^{100}$ and $n = 30$. As n rises the error in the method seems to increase. In a typical example of the size just mentioned it takes only a few seconds for the center-line method to give an upper bound of $1.66 \cdot 10^{24}$. We do not know $g(A)$, but we can use $L(A)$ as an estimate of a lower bound; that is $1.3 \cdot 10^{22}$ so we learn that the $\log_{10}$ relative error is probably about 9%. And this method is capable of producing reasonable bounds for giant examples. Example: $a = 10^{100}$ and $n = 100$. It takes about a minute to get the bounds where we use $C_{\text{mult}} = 2$ since the first i that works can be over one million. This yields a bound of 107.5 on the base-10 log of $g(A)$, and the true value is probably within about 4% of this. The best method from the literature would be to use the $n = 3$ case on a subset of A; this gives an upper bound near 10^{150}.

9. Finding Protoelbows

Finding Integer Points in a Polytope

Given a system $\Phi(x_1, \ldots, x_n)$ of linear inequalities describing a polytope in $\mathbb{R}^n$, we would like to find all solutions of $\Phi(x_1, \ldots, x_n)$ in $\mathbb{Z}^n$. This can be accomplished by the following simple recursive algorithm.

PolytopeIntegerPoints

Input. A system $\Phi(x_1, \ldots, x_n)$ of linear inequalities describing a polytope.
Output. The set of all solutions of $\Phi(x_1, \ldots, x_n)$ in $\mathbb{Z}^n$.

Step 1. Use linear programming to find the minimum *min* and the maximum *max* of x_n on the solution set of $\Phi(x_1, \ldots, x_n)$ in $\mathbb{R}^n$.

Step 2. If $n = 1$ return the set of all integers between *min* and *max*. Otherwise, for each integer $min \le a_n \le max$ call **PolytopeIntegerPoints** recursively to find all integer solutions of $\Phi(x_1, \ldots, x_{n-1}, a_n)$ in $\mathbb{Z}^n$.

While this algorithm works for arbitrary linear inequality systems with bounded real solution sets, it may not be very efficient. In general it is advantageous to first use the lattice reduction algorithm to find a coordinate system in $\mathbb{Z}^n$ in which hyperplanes corresponding to the inequalities are close to orthogonal. Then, at least for randomly generated systems, we obtained the best performance by using the following hybrid method. Let S be the real solution set of $\Phi(x_1, \ldots, x_n)$, and let S_1 be the set of points $p \in \mathbb{R}^n$ such that $p + U \subset S$, where U is the unit cube. To find the integer solutions of Φ in S_1 we use a recursive method, and to find the integer solutions of Φ in $S \setminus S_1$ we use a branch-and-bound type of method.

However, our experiments suggest that for the particular linear systems we get during the protoelbow computation the simple recursive algorithm **PolytopeIntegerPoints** gives the best performance. A typical system produced during the protoelbow computation consists of a set of inequalities with small coefficients which have already been transformed by lattice reduction and a single inequality with somewhat larger coefficients. Using the lattice reduction again on the whole system seems to make it harder to solve. Also, the real solution set of the inequalities does not usually have large "thin" parts, which in the hybrid algorithm are handled by a branch and bound method.

The complexity of algorithm **PolytopeIntegerPoints** is bounded by the number of lattice points in the bounding box. We have found that the number of points in the bounding box is largely independent of the size of the numbers involved. In theory, one could use Barvinok's algorithm [4] to generate a polynomial that enumerates the protoelbows. Our experiments indicate that the time taken by Barvinok's algorithm is a high-degree polynomial of the size of the numbers involved, and becomes much less efficient than **PolytopeIntegerPoints** even for problems involving only three or four digits.

Finding Protoelbows

When using the vanilla algorithm we have $\boldsymbol{k}^+$, a vector that bounds the bounding vector $\boldsymbol{k}$ from above. Then we can set up a system Φ as follows. We work in multiplier space, so the system is of the form $\Phi(i_1, \ldots, i_m)$, and it consists of the $2m + 1$ inequalities: $(i_1, \ldots, i_m) \cdot V \cdot B \geq 0$, $-\boldsymbol{k}^+ \leq (i_1, \ldots, i_m) \cdot V$, and $(i_1, \ldots, i_m) \cdot V \leq \boldsymbol{k}^+$. We are using $\boldsymbol{k}^+$ so the set of lattice points from these multipliers is a superset of the true set of protoelbows. But having this set, it is a simple matter to identify the axial elbows (i.e., $\boldsymbol{k}$), and then discard any vector that does not lie in the expanded bounding box, thus getting the exact set of protoelbows. Of course, if we have the actual bounding vector $\boldsymbol{k}$ in hand before solving Φ, then we use that at the start.

Reducing the Number of Protoelbows

A heuristic idea can be used at this point to avoid finding the complete set of protoelbows. Assume that we know the minimal weight of a nonzero lattice point having no negative entries (see §14 for its computation). Many experiments showed us that, for the cases of interest, two types of protoelbows were irrelevant to the determination of $g(A)$: those having weight 0 and those having weight greater than $w_{\min}$. We call these the *null* and *heavy* protoelbows, respectively; a protoelbow that is neither null nor heavy is called *light*. So we can speed things up substantially by setting up the protoelbow search to find only the light protoelbows (we just replace the last of the inequalities with $1 \leq ((\boldsymbol{i} \cdot V) \cdot B)/a \leq w_{\min}/a$). We include the axial elbows too, since we will have them on hand. Now, we also implemented an Aardal–Lenstra type of Frobenius instance solver (§12). Using the smaller set of protoelbows just described we can compute the corners as described in §11, find the one of greatest weight,

and subtract a to get a number g_{light} that is an upper bound on the true value of $g(A)$. But then we can simply use the instance solver to see if the equation $\boldsymbol{X} \cdot A = g_{\text{light}}$ has no nonnegative solution. If that is so, we can be certain that $g(A) = g_{\text{light}}$.

The fact that g_{light} is indeed an upper bound follows from the next lemma. Given a set of vectors $\mathcal{E}$ in $\mathbb{N}^m$, we call the set that remains after the removal of the cones of each vector in $\mathcal{E}$ the *domain determined by* $\mathcal{E}$. The domain determined by $\mathcal{E}$ is finite if and only if $\mathcal{E}$ contains vectors on each axis.

Lemma 3. Suppose $\mathcal{E}_1$ and $\mathcal{E}_2$ are sets of vectors in $\mathbb{N}^m$ so that the domains they determine are finite. Suppose that any vector in $\mathcal{E}_2$ is dominated by one in $\mathcal{E}_1$. Then any vector in the domain determined by $\mathcal{E}_2$ is dominated by a corner of D_1.

Proof. Any vector $\boldsymbol{X} \in D_2$ is in D_1 and so we can just start at $\boldsymbol{X}$ and move in a positive direction in each coordinate so as to stay in D_1 until we can move no farther. That brings us to a corner of D_1. □

It follows from this lemma that if $\mathcal{E}$ is a set of preelbows for A that includes the axial elbows, then the weight of the farthest corner in the domain determined by $\mathcal{E}$ is no less than the weight of the Frobenius corner for A. For we can just apply the lemma, with $\mathcal{E}_2$ being the full set of elbows for A.

It turns out that when a is small the heuristic often fails. Whenever that happens, we return to the protoelbow search and find the heavy and null protoelbows and use them to find all elbows, and then $g(A)$. But there is another shortcut, as we need only consider certain of the heavy elbows. If $\boldsymbol{X}$ is a heavy protoelbow and $\boldsymbol{X}$ dominates $\boldsymbol{\mu}-\boldsymbol{k}$, where $\boldsymbol{\mu}$ is the minimal elbow and $\boldsymbol{k}$ the bounding vector, then $\boldsymbol{X}$ can be ignored. For if $\boldsymbol{X}$ satisfies this condition, let $\boldsymbol{\mu'} = \boldsymbol{X} - \boldsymbol{\mu}$; $\boldsymbol{\mu'} \in L$, $\boldsymbol{\mu'}$ is dominated by $\boldsymbol{k}$ because $\boldsymbol{X}$ is, and $\boldsymbol{\mu'} = \boldsymbol{X} - \boldsymbol{\mu}$ dominates $-\boldsymbol{k}$; all this means that $\boldsymbol{\mu'}$ is a lattice point in the expanded bounding box. Because $\boldsymbol{X}$ is heavy, $W(\boldsymbol{\mu'}) = W(\boldsymbol{X}) - W(\boldsymbol{\mu}) > 0$, and so $\boldsymbol{\mu'}$ is a protoelbow. Because $\boldsymbol{X}$ dominates $\boldsymbol{\mu'}$, the preelbow corresponding to $\boldsymbol{X}$ cannot be an elbow. Further, we can ignore any protoelbow $\boldsymbol{X}$ for which $W(\boldsymbol{X'}) \geq a + g_{\text{light}}$, where $\boldsymbol{X'}$ is the nonnegative part of $\boldsymbol{X}$. This is because such a preelbow $\boldsymbol{X'}$ cannot, when its cone is removed, affect the existence of the corner whose weight is $a + g_{\text{light}}$.

The light-protoelbow heuristic is a very powerful way to speed things up and we have not found any significant examples for which it fails; that is, for all such that we know of, the numbers are so small that $g(A)$ can be computed easily using all protoelbows, or one of the methods of [7]. One such example is $(1000, 2100, 3198, 9523)$, for which there are only 7 protoelbows. The savings from this shortcut is substantial. In a typical case where $a = 10^{10}$ and $n = 8$, the total number of protoelbows is 160683 but the number of light-plus-axial protoelbows is only 74027. This reduction by about half is typical, and of course it yields a nice speed-up in the subsequent steps of the algorithm. A final point about this heuristic is that by excluding the null protoelbows we eliminate the tie-breaking issue. This means that when this heuristic is used, step 5 in the Vanilla Algorithm can be skipped.

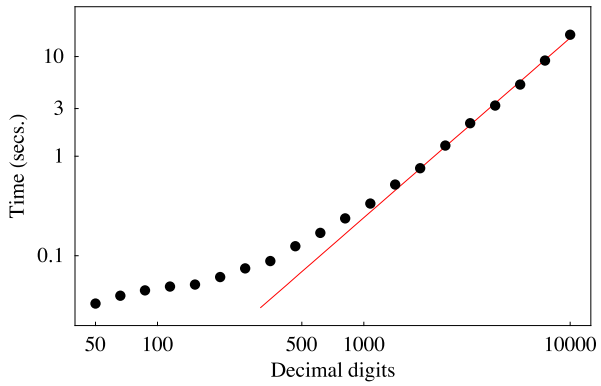

Figure 11: Computation time for finding the integer points in the polytope as a function of the number of decimal digits of a, for $n = 4$. One hundred trials were used to obtain an average for each data point. The behavior changes at around 500 digits, but the final section is strongly linear, indicating a time complexity of $O((\log a)^{1.8})$ in this range.

Complexity Of Protoelbow Enumeration

The complexity of the algorithm **PolytopeIntegerPoints** depends not only on the number of integer points in the polytope but also on geometric properties of the polytope. The rectangle $1 \le x \le n \wedge 1/3 \le y \le 2/3$ contains no integer points, but **PolytopeIntegerPoints** calls itself recursively n times to prove it. The worst-case complexity of protoelbow computation using **PolytopeIntegerPoints** is not worse than exponential in the size of the input, since the number of recursive calls is clearly bounded by ma^m. However, our experiments suggest that for n fixed and less than or equal to 7 the average complexity is much better, namely for the investigated range of $a \le 10^{10,000}$ it is subquadratic in $\log a$.

Assuming that the average number of protoelbows and geometric properties of the polytope do not change with a for fixed m, the complexity of **PolytopeIntegerPoints** grows as fast as the complexity of linear programming. Our *Mathematica* implementation solves linear programming problems using the simplex algorithm. The size of matrices the simplex algorithm operates on and the number of matrix operations depend on the number of variables and the number of constraints. For fixed m, both these numbers are fixed; hence the complexity as a function of a is proportional to the complexity of coefficient arithmetic, which is softly linear. The experiment shown in Figure 11 is not using asymptotically fastest multiplication and shows subquadratic behavior.

10. Finding Elbows

Turning the protoelbows into preelbows requires only zeroing out the negatives and eliminating duplicates. Then we need only take the domination kernel of the preelbows to get the exact set of elbows. This set completely characterizes $\mathcal{D}$. Indeed, $\mathcal{D}$ consists of those vectors

in $\mathbb{N}^m$ that do not dominate any elbow. Equivalently, $\mathcal{D} = \mathbb{N}^m \setminus \cup \{\text{cone}(\boldsymbol{v}) : \boldsymbol{v} \text{ an elbow}\}$. The concept of the domination kernel has been studied before so we can use an efficient algorithm from the literature.

Given a finite set $\mathcal{S}$ of points in $\mathbb{Z}^n$, say that $\boldsymbol{X} \in \mathcal{S}$ is *maximal* if it is not dominated by any other point in $\mathcal{S}$. Efficient algorithms for finding the maximal points are presented in [8]. Because $\boldsymbol{X}$ is in the domination kernel of $\mathcal{S}$ iff $-\boldsymbol{X}$ is maximal in $-\mathcal{S}$, those algorithms can be used on our problem. We made a few adjustments that seem to aid performance, but we basically used Algorithm M3 of [8]. First, we sort $\mathcal{S}$ according to the sum of the vectors. Then the idea is to form the (indices of vectors in the) kernel in $\mathcal{Z}$ by going through $\mathcal{S}$ one point at a time. If the point is $\boldsymbol{Q}$ then we check to see if $\boldsymbol{Q}$ dominates a point in $\mathcal{Z}$ (in which case there is nothing to be done, except that we move that $\mathcal{Z}$-point to the front of $\mathcal{Z}$ so that, subsequently, it acts sooner) or whether $\boldsymbol{Q}$ is incomparable with any point of $\mathcal{Z}$, in which case it is added to the end of $\mathcal{Z}$. Because of the initial sort, it cannot happen that $\boldsymbol{Q}$ is dominated by a point of $\mathcal{Z}$. At the end $\mathcal{Z}$ contains the indices of the domination kernel.

Bentley et al [8] report that the expected number of comparisons on purely random input of N vectors in dimension m is $O(N \log^{m-1} N)$ and they conjecture that in many cases the expected complexity is actually $O(N)$. We will assume the worst and take the complexity bound to be simply $O(n_P^2 \log a)$; this comes from just comparing each protoelbow with all the others in the most naive way.

When the number of protoelbows is large, as it is when n is 10 or 11, it might not be possible to gather them all in one place. One can arrange things so that they are looked at one at a time in such a way that the domination kernel is computed in parallel to the collection of the protoelbows. Each time the recursive algorithm finds a protoelbow that satisfies the tie-breaking condition (step 5 of the vanilla algorithm), it is turned into a preelbow E; then one checks whether E dominates any of the vectors in the kernel set being maintained. If not, E is added to the kernel and any members of the kernel set that dominate E are removed. At the end we have the true domination kernel, i.e., the set of elbows, and have never gathered all the protoelbows together. This is obviously a memory-efficient way to proceed, and it is time-efficient as well; and of course, if we are using the light-protoelbow heuristic, then only light protoelbows are looked at. All these improvements are used in the implementation that has been incorporated into *Mathematica*. And while this twist avoids the use of the Domination Kernel algorithm, that algorithm is still needed in the corner-finding algorithm of §11.

11. Finding Corners

Suppose $\mathcal{E}$ is the set of elbows of $\mathcal{D}$, the fundamental domain in $\mathbb{N}^m$ for the Frobenius basis A. To find $g(A)$ we need to determine the corners of $\mathcal{D}$; this can be done by a recursion on the dimension. In fact, if we want only a Frobenius corner then we can be more efficient and use information about the farthest corner found so far to cut down the search. But let us first consider how to find the complete set of corners. It is useful to define the notion of an

external corner: a point $\boldsymbol{C}$ so that $\boldsymbol{C}-\mathbf{1}$ is a corner (where $\mathbf{1}$ is the vector of all 1s). Since the elbows are external to $\mathcal{D}$, it is convenient to use them to find the external corners first.

Definition. If an elbow and an external corner have the same value in a coordinate i and the elbow has strictly smaller values in each of the other coordinates then we will say the elbow and the external corner are *adjacent in coordinate* i.

Lemma 4. For each coordinate and each external corner there exists at least one elbow adjacent to that corner in that coordinate.

Proof. Let $\boldsymbol{X}$ be the external corner and $\boldsymbol{X}^-$ the associated corner, $\boldsymbol{X}^- = \boldsymbol{X}-\mathbf{1}$. We know that $\boldsymbol{X}^-$ does not dominate any elbow, but $\boldsymbol{X}^- + \boldsymbol{e}_1$ does dominate an elbow because it lies outside $\mathcal{D}$. This means that there is an elbow that shares the first coordinate with $\boldsymbol{X}$ and is strictly dominated by $\boldsymbol{X}$ in all the other coordinates; the elbow is therefore adjacent to $\boldsymbol{X}$ in the first coordinate. The other coordinates follow similarly. □

Suppose we have $\mathcal{E}$, the set of elbows, in hand. The algorithm proceeds by choosing each elbow $\boldsymbol{X}$ in turn and constructing the external corners $\boldsymbol{C_X}$ that are adjacent to it in the first coordinate. To do this, we first choose $\boldsymbol{X}$, so we know the first coordinate, x_1. If $x_1 = 0$, we may ignore $\boldsymbol{X}$, since there are no external corners $\boldsymbol{C_X}$. Now we know that all the other coordinates of the external corner are strictly larger than those of the elbow, so we choose to represent the external corner as the sum of the elbow plus a vector whose first coordinate is 0 and the other coordinates are positive. To do this we translate the elbow to the origin in all coordinates but the first, dragging the other elbows along. Now we know that any elbow whose first coordinate is less than or equal to x_1 can be adjacent in any coordinate but the first to an external elbow $\boldsymbol{C_X}$. Now, as we know the first coordinate, we can ignore it in further computations, thus reducing the dimension by one and proceeding recursively. The base case is dimension 2, and that is very simple for if the elbows are $\{(x_1,y_1),(x_2,y_2),\ldots,(x_k,y_k)\}$ in lexicographic order, then $x_1 = y_k = 0$ and the set of external corners is just $\{(x_2,y_1),(x_3,y_2),\ldots,(x_k,y_{(k-1)})\}$, making a staircase from $(0,y_1)$ to $(x_k,0)$.

Here is a formal description. Note that when the elbows are translated downward some negative entries can arise, and these are replaced with 0. It can also happen that when the first coordinate is removed, two elbows can become the same, or have one dominate the other. That is handled by removing dominators from the newly defined set ($\mathcal{B}$ in the algorithm that follows).

External Corner Algorithm

Input. The set of elbows $\mathcal{E}$ of a fundamental domain $\mathcal{D}$ defined from a Frobenius basis A.
Output. The set of external corners of $\mathcal{D}$. To get the corners of $\mathcal{D}$ just subtract 1 from each entry in each external corner.
Assumptions. $\mathcal{E}$ is in lexicographic order.

Base case. $m = 2$. The sorted order means that $\mathcal{E} = \{(0,y_1),(x_2,y_2),\ldots,(x_k,0)\}$; thus there are $k-1$ external corners in the plane, given explicitly as (x_2,y_1), $(x_3,y_2),\ldots,(x_k,y_{k-1})$.

Recursive step.
Let $\mathcal{P} = \{\boldsymbol{v} \in \mathcal{E} : v_1 > 0\}$.
For each $\boldsymbol{v} \in \mathcal{P}$:
Let $\mathcal{B} = \{(u_2, \ldots, u_m) - (v_2, \ldots, v_m) : \boldsymbol{u} \in \mathcal{E} \text{ and } u_1 < v_1\}$.
Replace negative entries in any vector in $\mathcal{B}$ with 0.
Let $\mathcal{B}_{\rm DK}$ be the domination kernel of $\mathcal{B}$.
Let $\boldsymbol{C_v} = \{(v_1, \boldsymbol{u} + (v_2, \ldots, v_m)) : \boldsymbol{u}$ an external corner of $\mathcal{B}_{\rm DK}\}$.
Return $\cup\{\boldsymbol{C_v} : \boldsymbol{v} \in \mathcal{P}\}$.

The preceding description returns the set of external corners. To get the actual set of corners, one subtracts $\mathbf{1}$ from each. But really we want only the Frobenius corner or, more to the point, just its weight. It is easy to modify the algorithm to keep track of the largest weight of an external corner found so far, returning this value W at the end; then $W - \sum A$ will be the Frobenius number. This approach allows us to speed things up substantially by avoiding any recursive call that has no chance of finding an external corner with weight greater than the largest found so far. We call this the *Farthest Corner* algorithm. This completes the presentation of all the pieces needed to run the vanilla algorithm for the Frobenius number and it works fine up to $n = 7$, with the usual caveat about rare cases where the number of elbows is out of control. For example, if $A = (10^6, 1510697, 2225975, 7709798, 8775401)$ then there are 39 elbows and 41 corners and the vanilla algorithm finds $g(A) = 486333765$ in 0.4 seconds. For larger n it will be best to use various enhancements to be discussed.

When n is small it does not take too long to go from the set of elbows to the farthest corner. But things slow down as the dimension and the number of elbows rise. For example, when $n = 9$ we found that it took 30 minutes to find all the light protoelbows, 6 minutes to find the elbows defined by these, and 19 minutes to then find the farthest corner. It seems likely that there is a way to compute the farthest corner at the same time as one finds the protoelbows: as each protoelbow is found it is used, if it survives a domination check, to update a data structure that computes the farthest corner. But we have not worked out details of how to do this.

For fixed dimension m, the complexity of the Corner Algorithm in the form needed to find the Frobenius number, is $O(n_E^m \log a)$. Thus the algorithm is polynomial-time in terms of the length of its input length, which, in the case of interest, is $O(n_E \log a)$. The proof is a straightforward induction on the dimension. In the base case the algorithm just moves some numbers around and takes $O(n_E)$ steps. In dimension m induction using the fact that the domination kernel runs in time $O(n_E^2 \log a)$ for n_E many vectors of length $O(\log a)$ each, yields the expression $O(n_E^m \log a)$. In this discussion we can ignore the last step of the algorithm, forming the union, since if we want only the Frobenius number, we can just compute the weight of each corner, including duplicates.

12. Integer Linear Programming and Frobenius Instances

A central algorithm in much of our work is the ability to solve certain integer programming problems. The two that arise are first, not surprisingly, the Frobenius instance problem, where we want a solution to a Frobenius instance or a proof that none exists, and second, the integer version of linear programming (called *ILP*, for integer-linear programming), where we seek to minimize $\boldsymbol{c} \cdot \boldsymbol{i}$ ($\boldsymbol{i} \in \mathbb{Z}^m$) subject to a linear constraint $M \cdot \boldsymbol{i} \geq \boldsymbol{b}$. The main occurrence of ILP in our work is the determination of the exact axial elbows, as each such is the solution to an optimization problem. In general the Frobenius instance and Frobenius number problems have been considered quite separately. But we will show how the former can help us find the latter.

We start by describing in brief the Frobenius instance solver. Suppose we have a set $A = (a, b_1, \ldots, b_m)$ of size $n = m+1$ and a target M, a positive integer. We seek nonnegative integer multipliers $\boldsymbol{X} = (x_0, \ldots, x_m)$ such that $\boldsymbol{X} \cdot A = M$. This is an ILP problem. We attack it as follows. First we find a solution vector $\boldsymbol{X}$ over $\mathbb{Z}^n$ and a basis V for the null space (that is, m independent vectors $\boldsymbol{V}_j \in \mathbb{Z}^n$ with $\boldsymbol{V}_j \cdot A = 0$). We use multiples of the basis vectors to find a "small" solution which we still call $\boldsymbol{X}$. The tactic utilized to do this is sometimes called the "embedding method" and has been independently discovered several times. Variants appear in [32, 1, 30, 27]. In starting with a solution over $\mathbb{Z}^n$ rather than the nonnegatives $\mathbb{N}^n$ we are working with what is called an "integer relaxation" of the nonnegativity constraint.

We define variables $\boldsymbol{t} = (t_1, \ldots, t_m)$ so that solution vectors are given by $\boldsymbol{X} + \boldsymbol{t}V$. We need to impose two requirements. The first is that all components are nonnegative; the second is that all are integers. The first can be met by standard linear programming. Hence we use a branching method (see e.g. [41]) whereby we find solutions to relaxed LP problems (in this type of relaxation no integer constraints are enforced; that is, we now work over positive reals). We then branch on noninteger values in those solutions. Specifically, if a solution has, say, $t_1 = 5/4$, we create two subproblems identical to the one we just solved, but with the new constraints $t_1 \leq \lfloor 5/4 \rfloor = 1$ and $t_1 \geq \lceil 5/4 \rceil = 2$, respectively. It can be shown that this process will terminate eventually with values that are all integers, or an empty solution set if no such solution exists.

In order to keep "eventually" from becoming "a very long time" or "almost forever", we employ several tactics. One is to use a naive cutting plane scheme [41]. For example we can minimize or maximize the various coordinate values t_j, selecting one in some order or at random for each subproblem. This process amounts to finding the width of the polytope along the directions of our lattice basis vectors, and enforcing integrality of the optimized variable helps to further restrict the search space.

The diligent reader will realize that the ILP problem we just described can readily be stated with respect to any specific solution augmented by any basis of the relevant integer null space. We should explain why it is important to work with a small solution and a lattice reduced basis. The idea is that by reducing we in effect transform our coordinate system to one where the various search directions are roughly orthogonal. This helps us to avoid

the possibility of taking many steps in similar directions to search through the polytope of nonnegative solutions for one that is integer valued. Thus we explore it far more efficiently. Moreover, in starting with a small solution we begin closer to the nonnegative orthant. Heuristically this seems to make the sought-for multipliers of the null vectors relatively small, which is good for computational speed. This is discussed in [1], §2.

A further efficiency is to choose carefully the variable on which to branch. This is more or less the idea presented in [2]. One reduces the lattice of null vectors, orders them by size, and branches on the noninteger multiplier variable corresponding to the largest of these basis vectors. This has the effect of exploring the solution polytope in directions in which it is relatively thin, thus more quickly finding integer lattice points therein or exhausting the space. It turns out that this refinement is critical for handling pathological examples of the sort presented in [1]. This is also discussed in [29]. To summarize, we can solve in reasonable time (the larger the dimension m, the more time needed) the following optimization problem: Given an objective vector $\boldsymbol{c} \in \mathbb{Z}^m$ and a $k \times m$ constraint matrix M and vector $\boldsymbol{b} \in \mathbb{Z}^k$, find the vector $\boldsymbol{i} \in \mathbb{Z}^m$ such that $M \cdot \boldsymbol{i} \geq \boldsymbol{b}$ and $\boldsymbol{c} \cdot \boldsymbol{i}$ is as small as possible.

As reported in [2], this method is very efficient at solving Frobenius instances. If $A = (1000000, 1692802, 3706550, 8980199)$, and the target is 1023045256 (which is $g(A) + 2$), it takes only 0.01 seconds to get a solution $(208, 308, 55, 10)$. To illustrate a much larger problem and a novel approach to bounding $g(A)$ from below, suppose A is as follows

$$\begin{aligned} A \;=\; & (10000000000, 10451674296, 18543816066, 27129592681, 27275963647, \\ & 29754323979, 31437595145, 34219677075, 36727009883, 43226644830, \\ & 47122613303, 57481379652, 73514433751, 74355454078, 78522678316, \\ & 86905143028, 89114826334, 91314621669, 92498011383, 93095723941) \end{aligned}$$

We let the target be 350930344052, the nearest integer to $L(A)$, a heuristic estimate of a lower bound on $g(A)$ (from [7]). In 16 seconds the instance-solver returns the empty set, which proves that $g(A) \geq 350930344052$.

13. Finding the Axial Elbows

In the vanilla algorithm we simply use bounds on the axial elbows to find the set of lattice vectors with nonnegative weight. Once we have them, we form the domination kernel of the associated preelbow set to get the final elbow set and, as a byproduct, the set of axial elbows. But we can speed things up by using ILP to find the actual axial elbows first, for then the search for lattice vectors can be restricted to the box determined by the axial elbows. For example, in a typical case with $a = 10^{100}$ and $n = 7$ the volume of the box determined by the bounds only is $3 \cdot 10^{103}$ while the true bounding box has volume $2 \cdot 10^{102}$; the former is 16 times larger. When $a = 10^{10}$ it is less dramatic, but the point is that we can use ILP to

find the axial elbows and they substantially cut down both time and memory in the search for all protoelbows in step 4 of the vanilla algorithm.

Finding axial elbows turns out to be essentially the same task as Frobenius instance solving. Again we have an ILP to solve, subject to certain constraints such as nonnegativity. This time the problem is homogeneous so all solutions are null vectors. Again we can work with a lattice-reduced basis of these null vectors. We form combinations of them, impose the condition of nonnegativity but not integrality (hence solve relaxed LP problems), and branch on noninteger solution values. As with instance solving we again avail ourselves of efficiencies offered by cutting planes and by selection based on lattice vector size as in [2].

So let $\boldsymbol{k}$ be the bounding vector we seek and consider the problem of finding its jth entry, which gives the axial elbow on the jth axis. This is the vector $(0, 0, \ldots, 0, k_j, 0, \ldots, 0)$ where k_j is the smallest positive integer so that $k_j\boldsymbol{e}_j \notin \mathcal{D}$ (see Fig. 2). This means that $k_j b_j$ is representable as $\alpha a + \boldsymbol{X} \cdot B$ where $\boldsymbol{X} = (x_1, x_2, \ldots, x_m)$ is nonnegative, $x_j = 0$ (by orthogonality), and either α is positive (so that the weight of $\boldsymbol{X}$ is less than that of $k_j\boldsymbol{e}_j$) or $\alpha = 0$ but one of the entries in $(x_1, \ldots, x_{j-1})$ is positive (the tie-breaking condition). If we use x_j instead of k_j (reasonable since x_j in $\boldsymbol{X}$ must be 0), this becomes a search for the single lattice vector $(-x_1, -x_2, \ldots, -x_{j-1}, x_j, -x_{j+1}, \ldots, -x_m)$, since the weight of this vector is αa, indicating that it lies in L.

We can rewrite all this as the following constrained optimization problem, where we eliminate the minus signs in the lattice vector sought for notational ease. As explained in §12, we use a reduced basis V.

Minimize the positive integer x_j such that there exist integers i_p, x_p so that, with $\boldsymbol{X} = (x_p)$ and $\boldsymbol{i} = (i_p)$,

$$\begin{array}{lcll} \boldsymbol{X} & = & \boldsymbol{i} \cdot V & (\boldsymbol{X} \in L) \\ x_j & \geq & 1 & \text{(the objective is positive)} \\ x_p & \leq & 0 & \text{for } p \neq j \text{ (the others are nonpositive, so they yield a vector} \\ & & & \text{that will be compared to } x_j\boldsymbol{e}_j) \end{array}$$

$\boldsymbol{X} \cdot B \geq 0$ (so that the weight of the objective is no less than the weight of the others)
If $\boldsymbol{X} \cdot B = 0$, then the first nonzero entry in $\boldsymbol{X}$ is negative
(the tie-breaker when weights are equal)

It is much more convenient to state the tie-breaking condition as a single arithmetic clause. It is easy to verify that that can be done as follows.

Minimize the integer x_j such that there exist integers i_p, x_p so that, with $\boldsymbol{X} = (x_p)$ and $\boldsymbol{i} = (i_p)$,

$\boldsymbol{X} = \boldsymbol{i} \cdot V$, $x_j \geq 1$, $x_p \leq 0$ for $p \neq j$, $\boldsymbol{X} \cdot B \geq 0$
$\boldsymbol{X} \cdot B - (x_1 + x_2 + \cdots + x_{j-1}) \geq 1$

Now, to find the axial elbow for the jth axis, we need only use ILP with the following definitions of $\boldsymbol{c}$, M, and $\boldsymbol{b}$, where j is the index of the axis we are working on and $V = (v_{i,j})$. Note that a is always an upper bound on any axial elbow. And while we can use 1 as a lower bound, we can also use μ_j where $\boldsymbol{\mu}$ is the minimal elbow if we have taken the time to compute that. The dimension of the optimization problem is m, but with $m+3$ constraints. The objective is given by $\boldsymbol{c} = (v_{1j}, v_{2j}, \ldots, v_{mj})$, the jth column of V; therefore $\boldsymbol{c}\cdot\boldsymbol{i} = x_j$. The matrix M and bounding vector $\boldsymbol{b}$ are defined in the display that follows. The first m rows give the conditions on $\boldsymbol{X} = (x_j)$. The next row gives an upper bound of a on x_j. The penultimate row states that the weight of $\boldsymbol{X}$ is nonnegative and the last row encodes the tie-breaking condition $\boldsymbol{X}\cdot B - (x_1 + x_2 + \cdots + x_{j-1}) \geq 1$. For brevity we let $\mathrm{bv}(j) = b_1 v_{j,1} + \cdots + b_m v_{j,m}$ and $\mathrm{bvm}(j) = (b_1 - 1)v_{j,1} + \cdots + (b_j - 1)v_{j,j-1} + b_j v_{j,j} + \cdots + b_m v_{j,m}$

$$\begin{pmatrix} -v_{1,1} & \ldots & -v_{m,1} \\ \vdots & \ldots & \vdots \\ -v_{1,j-1} & \ldots & -v_{m,j-1} \\ v_{1,j} & \ldots & v_{m,j} \\ -v_{1,j+1} & \ldots & -v_{m,j+1} \\ \vdots & \vdots & \vdots \\ -v_{1,m} & \ldots & -v_{m,m} \\ -v_{1,j} & \ldots & -v_{m,j} \\ \mathrm{bv}(1) & \ldots & \mathrm{bv}(m) \\ \mathrm{bvm}(1) & \ldots & \mathrm{bvm}(m) \end{pmatrix} \cdot \begin{pmatrix} i_1 \\ i_2 \\ \vdots \\ i_m \end{pmatrix} \geq \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \mu_j \\ 0 \\ \vdots \\ 0 \\ -a \\ 0 \\ 1 \end{pmatrix}$$

Now we can state the algorithm.

Finding the Axial Elbows

Input. A Frobenius basis A and V, the reduced homogeneous basis.
Output. The bounding vector $\boldsymbol{k} = (k_j)$, where $k_j\boldsymbol{e}_j$ is the jth axial elbow. In fact, the algorithm finds the set of m axial protoelbows.

For each $j = 1, \ldots, m$, compute $\mathrm{ILP}(\boldsymbol{c}, M, \boldsymbol{b})$ where $\mathbf{c}$, M, and $\boldsymbol{b}$ are as defined above to determine the axial protoelbow that yields the minimum nonnegative value of k_j.

A Bisection Route to the Axial Elbows

There is another way to find the axial elbows that, in some cases, is faster than the ILP method. Suppose $k_j\boldsymbol{e}_j$ is an axial elbow. If we could tell, for a given integer h, whether $h \geq k_j$, we would have the basis of a simple bisection method for determining k_j exactly. But we *can* tell this, thanks to the Aardal–Lenstra instance solver. Let $\boldsymbol{H} = h\boldsymbol{e}_j$, with $w = W(\boldsymbol{H}) = hb_j$. Then $h < k_j$ iff none of the following has a solution: $\boldsymbol{X}\cdot A = w - a$ and $\boldsymbol{Y}\cdot B = w$ with the constraint $y_1 + \cdots + y_{j-1} \geq 1$. For if $\boldsymbol{X}$ solves the first equation, then $(x_2, \ldots, x_n)$ has weight less than w and congruent to w, showing that $\boldsymbol{H} \notin \mathcal{D}$; the converse holds too if $\boldsymbol{H}$'s failure to lie in $\mathcal{D}$ is because of an equivalent vector of smaller weight. The

second equation addresses the tie-breaking issue. For implementation, we used the following set of j calls to Frobenius instance as a way of handling the constraint. That is, $h < k_j$ if and only if none of the following j Frobenius instances has a solution:

$\boldsymbol{X} \cdot A = hb_j - a$, $\boldsymbol{X} \cdot (A \backslash \{a\}) = hb_j - b_1$, $\boldsymbol{X} \cdot (A \setminus \{a, b_1\}) = hb_j - b_2$,. . .,
$\boldsymbol{X} \cdot (A \setminus \{a, b_1, \ldots, b_{j-2}\}) = hb_j - b_{j-1}$

To turn this into a working algorithm, we need bounds on k_j. We can take 1 as a lower bound. For an upper bound we could use the trivial bound $k_j \leq a$, but this is much too large. It is better to use the upper bound found by the center-line method (see §8). Then we can use standard bisection to find the exact value of k_j for each j. When $a = 10^{10}$ and n is between 10 and 20, this method is faster than the use of ILP; but when $a = 10^{100}$, the ILP method is faster.

14. Finding the Minimal Elbow

The vanilla algorithm for $g(A)$ does not require it, but there are several applications of the minimal weight and minimal elbow, so we describe here algorithms for both. The minimal weight is in fact very useful in the heuristic that cuts down the number of protoelbows needed to get $g(A)$. Other applications are: (1) $\boldsymbol{\mu}$ is a vector that we know is an elbow; (2) by Corollary 2, $\boldsymbol{\mu}$ allows us to determine whether or not $\mathcal{D}$ has an interior elbow, which can be of theoretical interest, and (3) $\boldsymbol{\mu}$ provides a lower bound on $g(A)$ (formula in §15).

A natural approach to finding $\boldsymbol{\mu}$ has two steps: (1) Find the minimal weight by ILP; (2) use this and some bounds on the bounding box (or the actual bounding box) to set up a system for *Mathematica*'s `Reduce` function to find all vectors that are in $L^+ \cap K$ and have the right weight. That does in fact work well, but we will here give an alternative approach that uses only ILP . Since we often want $w_{\min}$ and not $\boldsymbol{\mu}$, let us first compute that. The algorithm below assumes that $\boldsymbol{k}$, the bounding vector, is known. But even if it is not, one could use the same algorithm using instead of $\boldsymbol{k}$, a vector that is known to bound $\boldsymbol{k}$ (see § 6).

Algorithm to Find the Minimal Weight

Input. A Frobenius basis A.
Output. The minimal weight $w_{\min}$ of a vector in $L^+ \cap K$.
Assumptions. The bounding vector $\boldsymbol{k}$ has been computed (and the reduced homogeneous basis V, which its computation requires, has been stored).

Step 1. Define the objective function vector $\boldsymbol{c} = (B \cdot V_1, \ldots, B \cdot V_m)$ and let M and $\boldsymbol{b}$ be the matrix and right-hand side of the following system of constraints; the first m constraints together with the last guarantee that $\boldsymbol{i} \cdot V \in L^+$, while the constraints involving k_j place $\boldsymbol{i} \cdot V$ inside K.

$$\begin{pmatrix} & V^T & \\ & -V^T & \\ \sum V_1 & \cdots & \sum V_m \end{pmatrix} \begin{pmatrix} i_1 \\ i_2 \\ \vdots \\ i_m \end{pmatrix} \geq \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ -\boldsymbol{k} \\ 1 \end{pmatrix}$$

Step 2. Return ILP($\boldsymbol{c}, M, \boldsymbol{b}$).

The next algorithm will use almost the same instance to find not only the minimal weight, but also the minimal elbow. It works by setting up a problem whose constraint $M \cdot \boldsymbol{i} \geq \boldsymbol{b}$ is identical to that of the preceding algorithm. But the objective $\boldsymbol{c}$ is more complicated. If $\boldsymbol{X}$ is the minimal elbow we seek ($\boldsymbol{X} = \boldsymbol{i} \cdot V$ where $\boldsymbol{i}$ are the coordinates), then we want to minimize:

$$(\boldsymbol{X} \cdot B)(1+k_1)(1+k_2)\cdots(1+k_m) - \\ x_1(1+k_2)(1+k_3)\cdots(1+k_m) - x_2(1+k_3)(1+k_4)\cdots(1+k_m) - \cdots - x_{m-1}(1+k_m) - x_m.$$

We omit the details that if $\boldsymbol{X}$ minimizes this objective and $\boldsymbol{Y}$ is any other vector in $L^+ \cap K$, then $W(\boldsymbol{Y}) \geq W(\boldsymbol{X})$ and if the weights are equal then $\boldsymbol{Y}$ lexicographically precedes $\boldsymbol{X}$.

Algorithm to Find the Minimal Elbow

Input. A Frobenius basis A.
Output. The minimal elbow $\boldsymbol{\mu}$ of the fundamental domain $\mathcal{D}$ and, if it exists, the interior elbow.
Assumptions. The bounding vector $\boldsymbol{k}$ has been computed (and the reduced homogeneous basis V, which its computation requires, has been stored).

Step 1. Let $\boldsymbol{b}$ and M be the vector and matrix of the constraint in the minimal weight algorithm.

Step 2. Let $\text{prod}_i = \prod\{1 + k_j : i \leq j \leq m\}$; compute these by accumulating the partial products.

Step 3. Let $\beta_j = b_j \text{prod}_1 - \text{prod}_{j+1}$, where $A = (a, b_1, \ldots, b_m)$.

Step 4. Let $\boldsymbol{c}$ be the vector whose jth entry is $\beta_j \cdot V_j$.

Step 5. Let $\boldsymbol{i} = \text{ILP}(\boldsymbol{c}, M, \boldsymbol{b})$; then $\boldsymbol{i} \cdot V$ is the minimal elbow $\boldsymbol{\mu}$. If $\boldsymbol{\mu}$ has no zeroes, then it is the unique interior elbow; otherwise there is no interior elbow.

---

15. Bounds on the Frobenius Number

Upper Bounds

The axial and minimal elbows provide excellent bounds on $g(A)$. If $\boldsymbol{k}$ and $\boldsymbol{\mu}$ are the bounding vector and minimal elbow, resp., then $g(A)$ cannot be greater than the weight of the corner

Lower Bound Method	Lower Bound	Upper Bound	Upper Bound Method
One random corner	12.50	12.89	Initial(7)
100 random domain points	12.49	13.59	Center-line
Axial	12.43	13.33	Axial
Axial plus minimal	12.43	13.30	Axial and minimal
Axial, minimal, and dual axial	12.43		
Nonrepresentable	12.30		

Table 2: Bounds for a random example with $a = 10^{10}$ and $n = 10$, with all numbers having $\log_{10}$ applied. We learn from the bounds that $12.50 \le \log_{10} g(A) \le 12.89$. In truth, $\log_{10} g(A) = 12.545$.

of the box determined by $\boldsymbol{k}$. This means $g(A) \le \boldsymbol{k} \cdot B - \sum A$, which we denote $\mathrm{UB}_{\mathrm{Axial}}$. This can be improved a little by taking the set of elbows to be $\alpha \cup \{\mu\}$ where α is the set of axial elbows, and then using the Farthest Corner algorithm of §11 to get the farthest corner according to this small set of elbows. The corresponding bound is called $\mathrm{UB}_{\mathrm{AxialMinimal}}$ and it is an improvement on $\mathrm{UB}_{\mathrm{Axial}}$. If n is too large then computing the exact axial elbows is very slow, but the center-line method gives $\boldsymbol{k}^+$, a vector that dominates $\boldsymbol{k}$, and the corner determined by that vector yields the upper bound $g(A) \le \boldsymbol{k}^+ \cdot B - \sum A = \mathrm{UB}_{\mathrm{Centerline}}$. Regarding $\mathrm{UB}_{\mathrm{Axial}}$, recall that in many cases the bisection method of getting $\boldsymbol{k}$ is much faster than the ILP method. Something else that can be tried when n is large is to observe that $g(A) \le g(a_1, a_2, \ldots, a_j)$ where $j < n$; this idea is pursued in [7], where the issue of $\gcd(a_1, a_2, \ldots, a_j) > 1$ is addressed and the following bound obtained, where $A_j = (a_1, \ldots, a_j)$ and $d = \mathrm{GCD}(a_1, \ldots, a_j)$:

$$g(A) \le d + dg(A_j/d) + g(d, a_{j+1}, \ldots, a_n)$$

If j is small, say $j = 7$, this can be done quickly; if $d = 1$ the last term is just 0 and if $d \ne 1$ then it is likely small enough to use one of the graph methods from [7]. We use $\mathrm{UB}_{\mathrm{Initial}(j)}$ for this method. Table 2 shows the results of these methods on one example where $a = 10^{10}$ and $n = 10$, together with lower bounds to be discussed. There is variation among examples as to which method is best, but the main point is that when the basis has size that allows the axial elbows to be computed, then one can get quite good bounds on $g(A)$. This can actually be quantified; see Proposition 5.

Lower Bounds

For lower bounds too there are several ideas that arise out of $\mathcal{D}$. Any $\boldsymbol{X} \in \mathcal{D}$ has weight no greater than that of the farthest corner and so yields the lower bound $\boldsymbol{X} \cdot B - a$. So each axial elbow yields a lower bound by just subtracting 1 along the axis it lives on. Similarly the minimal elbow yields a point in $\mathcal{D}$ by subtracting 1 from the first nonzero coordinate (the best choice when A is sorted). We use $\mathrm{LB}_{\mathrm{Axial}}$ for the best lower bound obtained by using the axial elbows and $\mathrm{LB}_{\mathrm{AxialMinimal}}$ when the minimal elbow is included.

One interesting enhancement to these ideas comes from considering $\boldsymbol{X}$, a protoelbow that yielded the jth axial elbow $\boldsymbol{Y}$. Then it is likely, if not certain, that $\boldsymbol{Z}$, the vector obtained

from $\boldsymbol{X}$ by turning its positive entry to 0 and taking the absolute value of its negative entries, has a weight that is the weight of a vector in $\mathcal{D}$. It appears that, especially when a is large, this almost always happens. Moreover, we can use instance solving to verify it by checking whether $W(\boldsymbol{Z}) - a$ is representable in terms of A. If it is, then $\boldsymbol{Z} \notin \mathcal{D}$ and we learn nothing. If it is not, then either $\boldsymbol{Z} \in \mathcal{D}$ or there is another vector in $\mathcal{D}$ having the same weight. This tells us that some point in $\mathcal{D}$ has weight $W(\boldsymbol{Z})$, and so the Frobenius corner has weight no smaller than this and $g(A) > W(\boldsymbol{Z}) - a$. We can check this for each axial protoelbow and add the resulting inequalities to what we know about the lower bound from the axial and minimal elbows. We call the resulting bound $\mathrm{LB}_{\mathrm{AxialMinimalDualAxial}}$. This idea illustrates a certain duality between points not in $\mathcal{D}$ and points in $\mathcal{D}$; possibly there are other important relations that could be discovered by pursuing the duality notion.

The difference between the axial upper and lower bounds can be estimated, thus giving a bound to the error when these methods are used. This idea first appeared in work of Krawczyk and Paz [24], who obtained an upper bound $\mathrm{UB}_{\mathrm{KP}}$ from a sequence of n numbers that are essentially equivalent to our m axial elbows together with $w_{\min}/\,a$. They proved that $\mathrm{UB}_{\mathrm{KP}}/\,n \le g(A) \le \mathrm{UB}_{\mathrm{KP}}$. We can do something similar for $\mathrm{UB}_{\mathrm{Axial}}$. In essence what we have done is to show how to effectively compute the numbers used by Krawczyk and Paz, though the details are not identical: $\mathrm{UB}_{\mathrm{KP}}$ is based on n numbers while we, working mod a, use only $n-1$ and so generally get a smaller upper bound.

Proposition 5. Given A with n entries, let $m = n - 1$. Then $\mathrm{UB}_{\mathrm{Axial}} \le m\ \mathrm{LB}_{\mathrm{Axial}}$ and so $\mathrm{UB}_{\mathrm{Axial}}/\,m \le g(A) \le \mathrm{UB}_{\mathrm{Axial}}$.

Proof. $\mathrm{UB}_{\mathrm{Axial}} = -a + \sum(k_j - 1)b_j \le m \max(k_j - 1)b_j \le m\ \mathrm{LB}_{\mathrm{Axial}}$. □

The preceding lower bound methods all require the exact computation of elbows and when n is large this is slow. Two techniques for getting lower bounds in reasonable time are the following, which both seek a lower bound by finding a large nonrepresentable integer.

1. $\mathrm{LB}_{\mathrm{RandomPoint}(j)}$. One can find the weight of a random point in $\mathcal{D}$ by randomly choosing r, a residue class mod-a, and then using optimization to find the smallest representable value of $Ma + r$; this value is the weight of a point in $\mathcal{D}$ and so subtracting a gives a lower bound on $g(A)$. In fact, simple bisection using Aardal–Lenstra works because within each residue class, the numbers start out being nonrepresentable and, at the weight in $\mathcal{D}$, become and forever stay representable. One can use this to get j random weights from $\mathcal{D}$, the largest of which, less a, is a lower bound on $g(A)$.

2. $\mathrm{LB}_{\mathrm{RandomCorners}(j)}$. Once one has the weight w of a point in $\mathcal{D}$ as just described, one can try to find the coordinates of the point in $\mathcal{D}$ by solving $\boldsymbol{X} \cdot A = w$. While there is in general no guarantee that the Frobenius instance solver will produce the correct $\boldsymbol{X}$ from the many available, it came as a bit of a surprise that our Aardal–Lenstra implementation seems to do so: that is, in the cases under consideration it maximizes x_1, which in turn minimizes the weight of $(x_2, \ldots, x_n)$. Thus, ignoring the tiebreak issue, this gives the correct coordinates of the $\mathcal{D}$-point. We do not really understand why it behaves this way. But because we can certify this behavior at the end, we would catch any failures. Now start at this point in $\mathcal{D}$

and move out to the boundary of $\mathcal{D}$ in the first direction, which can be done by a single ILP analogous to those used to find axial elbows. Continue through the directions in turn, ending up at a corner. Thus this is a method for finding a random corner of $\mathcal{D}$ and each such gives a lower bound.

16. Complexity and Performance

Our main algorithm for computing $g(A)$ relies on an enumeration of a set of elbows, which is computed from the light protoelbows, a subset of the full set of protoelbows. The number of protoelbows can be quite large. When $n \geq 4$, n_E, the number of elbows, can be huge (see Fig. 3). Thus from a worst-case perspective our algorithm is not polynomial time even when $n = 4$. But the other parts of the algorithm take time too, and it appears at least possible that the domination kernel computation could require $O(p^2)$ comparisons where p is the number of light protoelbows. If so our algorithm would be at least $O(a)$, which is not so good for such a complicated method (the very simple round-robin method [10] is $O(a)$ in the worst case).

But the practical point of view requires looking at average-case behavior and in that context our algorithms far outperform previous methods. For random input, the number of light protoelbows appears to grow exponentially with n, but hardly at all with increased a. Thus, for n up to 11, this number appears to be one that current computers can handle, and the other steps of the algorithm are also manageable. Figure 12 shows how the number of light protoelbows grows (the numbers beyond 10^6 in that chart are, for technical reasons, estimates only, but they are likely within 10% of the true number). The number of elbows is substantially smaller, but it is the protoelbow count that determines the time and space usage. Note that the protoelbow count grows at a rate that is slightly more than exponential in n and that there is very little change in the count as a increases from 10^{10} to 10^{100}. These counts, and many timing experiments, support the conclusion that, for fixed n, our algorithm

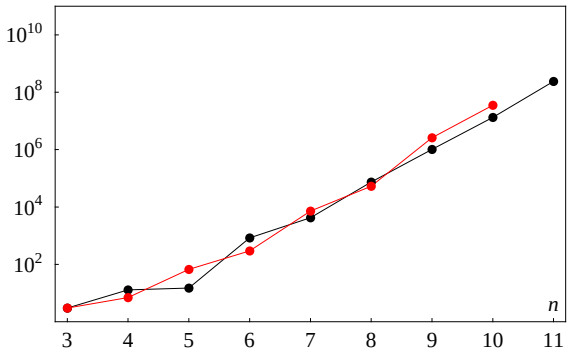

Figure 12: The number of light protoelbows as a function of basis size, for $a = 10^{10}$ (black) and $a = 10^{100}$ (red). There is very little change as a grows, but the growth is roughly exponential in n.

n	$a = 10^{10}$; Time (secs.)	$a = 10^{100}$; Time (secs.)	$a = 10^{1000}$; Time (secs.)
3	0.023	0.038	0.49
4	0.097	0.459	6.05
5	0.192	1.08	29.0
6	2.06	4.84	91
7	3.18	32.5	287
8	33.7	125	
9	311	6980	
10	6050	106650 (29.6 hrs.)	
11	156800 (43.6 hrs.)		

Table 3: Time needed to compute $g(A)$ using our main algorithm and the light protoelbow heuristic.

is polynomial-time on average, with degree $2+\epsilon$. The polynomial-time algorithm of Kannan for fixed n [21] has time complexity of the form $(\log a)^{n^n}$.

Table 3 gives the times needed for single random trials on 11-digit numbers. One second handles n up to 5; one minute up to 8, two hours up to 10, and 11 can be done in under two days. This can all be done for 100-digit numbers, or larger, and there is a slowdown of course, but it is modest. Times are based on a C-implementation using a 3.2 GHz Pentium 4 processor. Our implementation is the first that can find Frobenius numbers in this realm when $n \geq 4$.

We can be more precise about the complexity by taking a worst-case view provided we allow n_P as a parameter in the complexity function. For definiteness, we consider the $n = 4$ case and analyze each step of the vanilla algorithm as given in §7. The complexity bounds are as follows, where the observed values refer to observed averages as the input grows. They are sometimes better than the worst case, but also sometimes worse, as for the worst-case analysis we assume the theoretically best bounds.

We use Λ for the length of the input, $O(\log a)$, and recall that the notation $\tilde{O}(\Lambda^w)$ abbreviates $O(\Lambda^{w+\epsilon})$ for all positive ϵ.

Step 1. Getting a basis for the homogeneous lattice using the Hermite normal form. Worst-case $\tilde{O}(\Lambda)$; a close analysis of the Hermite normal form shows that the integers in the output have length $O(\Lambda)$; see [43]. Observed complexity: $O(\Lambda^{1.47})$.

Step 2. Reducing the basis by LLL lattice reduction. Worst-case: $\tilde{O}(\Lambda^2)$ [26]; observed: $O(\Lambda^{2.7})$. The size of each number in the reduced basis is $O(m \log a)$, which is $O(\Lambda)$; the observed sizes are $O((\log a)/m)$.

Step 3. Getting the axial elbow bounds by the center-line method, assuming step 2 already done: Worst-case: $\tilde{O}(\Lambda^2)$ (see Thm. 4); observed $O(\Lambda^{1.53})$.

Step 4. Getting the multipliers for the superset of the protoelbows. This depends on some aspects of the reduced lattice that we cannot quantify. But experiments (Fig. 11) show that when n is fixed the average time needed for this step is subquadratic as a function of the input length. So it appears that on average this step is better than $O(\Lambda^2)$. Observed complexity: $O(\Lambda^{1.74})$.

A small complication arises here because the vanilla algorithm uses axial elbow bounds (as opposed to exact axial elbows) and so produces a superset of the true protoelbow set. But, as pointed out in §9, it is a simple matter to filter this superset down to the true set, since it is trivial to locate the axial elbows within it. Still, such location and filtering takes time $O(n_P^+ \Lambda)$, where n_P^+ is the number of vectors found in the lattice point search, so we do have to check the size of n_P^+. The ratio n_P^+/n_P depends on how well the axial elbow bounds approximate the actual axial elbows (see Fig. 5) and we have already commented that the bounds are generally very good. Experiments show that this ratio is rarely larger than 5 (when $n = 4$), with a mean under 2, so it seems reasonable to assume $n_P^+ = O(n_P)$ on average.

The reason we are using n_P^+ in this analysis rather than n_P is not that n_P is hard to find. We generally use ILP to obtain the true axial elbows, and then **PolytopeIntegerPoints** yields the true protoelbows. But using the axial elbow bounds—i.e., the vanilla algorithm—avoids ILP (and questions about its complexity) completely.

Step 5. Turn the multipliers $\boldsymbol{i}$ into potential protoelbows, by forming $\boldsymbol{i} \cdot V$. Remove those with positive weight and first nonzero coordinate positive. Locate the axial elbows. Remove any vectors outside the expanded bounding box. Worst-case: For each $\boldsymbol{i}$ one needs $\tilde{O}(\Lambda)$ steps to form $\boldsymbol{i} \cdot V \cdot B$. Thus the total is $\tilde{O}(n_P^+ \Lambda)$. Observed: $O(\Lambda^{1.5})$.

Step 6. Replace negatives with 0 in the protoelbows. Worst-case and observed: $O(n_P)$.

Step 7. Getting the elbows by the Domination Kernel algorithm. The worst-case view assumes that any comparison of numbers could involve looking at every digit; the average behavior is much better. Worst-case: $O(n_P^2 \Lambda)$; observed: $O(1)$.

Step 8. Getting the corners (there are at most n_E^2) from the elbows. Worst-case: $O(n_E^3 \Lambda)$; observed $O(1)$.

Step 9. Computing the weight of each corner to find the largest. Worst-case: $\tilde{O}(n_E^2 \Lambda)$; in practice this is $\tilde{O}(n_E \Lambda)$ because the number of corners is, in all examples we have examined, close to n_E.

So, using the fact that $n_E \leq n_P$, all terms in the preceding list are $\tilde{O}(n_P^3 \Lambda^2)$ except for steps 4 and 5. The evidence is good that step 4 works in subquadratic time when n is fixed. And step 5 is in control under the assumption that $n_P^+ = O(n_P)$. Indeed, when n is 4 and a is very large, all the steps except step 2 take *much* less time than step 2, lattice reduction. So all this means that, if n_P and its powers are bounded on average (see end of §4 for the evidence) and if the protoelbow-finding steps behave on average the way we expect (evidence at end of §9), our algorithm finds the Frobenius number in just over quadratic time on average. Experiments show that in practice the vanilla algorithm has roughly the same average complexity as lattice reduction.

The analysis carries over to larger n, though the numbers are hard to examine because of the growth of n_P, which affects the constant factor in the complexity. Moreover, higher powers of n_P occur in the performance analysis of step 8. For example, when $n = 10$ the constant factor is so large that it might take 150,000-digit numbers before the LLL time

starts to dominate the protoelbow enumeration time. But it seems plausible that, when n is fixed, the vanilla algorithm finds $g(A)$ in time that on average is $\tilde{O}((\log a)^2)$. Recall that when $n = 2$ it can be done in worst-case time $\tilde{O}(\log a)$ and when $n = 3$ it can be done in worst-case time $O((\log a)^2)$.

Having a robust algorithm allows us to verify theorems and formulate new conjectures about the Frobenius number. One application is the discovery, with proof, of some formulas for Frobenius numbers of quadratic sequences; details are in the next section.

We can summarize our main algorithms as follows:

- Our implementations are the first that allow computation of $g(A)$ on random input when n is between 4 and 11, and with no restriction on the size of numbers in A.
- The methods (Frobenius number and Frobenius instance) are now included in *Mathematica*
- Special-case planar lattice reduction yields a heuristic algorithm for Frobenius numbers when $n = 3$ that has soft linear complexity in the worst case and for which no failures were found in millions of trials. Any improvements to 3-dimensional lattice reduction step would lead to corresponding improvement in our algorithm's performance when $n = 4$.
- Our methods lead to fast algorithms that yield reasonably good upper and lower bounds on $f(A)$ when n is less than about 30. Some (e.g., the center-line bound) are provably polynomial time, others are probably polynomial time.

17. Frobenius Formulas for Quadratic Sequences

Formulas for $g(A)$ are known when A is an arithmetic progression [36], §3.3, but little work has been done on quadratic sequences. For one example, consider $Q(a, M) = \{a + i^2 : i = 0, 1, \ldots, M\}$. Computations lead to a piecewise quadratic formula for $g(Q(a, M))$, with M^2 pieces. We found the patterns in the Frobenius number given in Theorem 5 and then found algebraic patterns in the sets of protoelbows, elbows, and corners that yielded proofs. Note that the first nontrivial case would be $(a, a + 1, a + 4)$, but a formula for that case was known because a general formula of Rødseth on arithmetic progressions with one number appended (see [36], p. 61); the two denominators given as 2 should be k) yields $g(A) = a(\alpha + 2) - \beta - 1 + 3\lfloor \beta/3 \rfloor$, where $a = 4\alpha + \beta$ with $0 \le \beta \le 3$. In the notation of Theorem 5 the formula is $g(a, a+1, a+4) = \frac{1}{4}(a^2 + \{8, 7, 6, 5\}a) - \{1, 2, 3, 1\}$ where the lists correspond to the mod-4 residues of a, starting with 0.

The theorem below shows that several cases of the general problem can be proved to follow the formula $\frac{1}{M^2}(a^2 + ca) - d$, where the coefficients c and d are given by the lists below, depending on M and the residue of a (mod M^2).

Theorem 5. For $M = 1, 2, 3, 4, 5, 6, 7$ and $a \ge 1, 1, 16, 24, 41, 67, 136$, resp., $g(Q(a, M)) = \frac{1}{M^2}(a^2 + c_{M,j}a) - d_{M,j}$, where j is the least nonnegative residue of a (mod M^2). The coefficients

are defined by the following lists, where $c_{M,j}$ refers to the jth entry in c_M, but the indexing starts at 0 (i.e., if $a \equiv 0 \pmod{M^2}$ the first elements of c_M and d_M are used.

$c_1 = \{-1\}$; $d_2 = \{1\}$

$c_2 = \{8, 7, 6, 5\}$; $d_2 = \{1, 2, 3, 1\}$

$c_3 = \{18, 17, 16, 15, 14, 13, 12, 20, 19\}$;
$d_3 = \{2, 3, 4, 1, 1, 2, 1, 1, 1\}$

$c_4 = \{32, 31, 30, \ldots, 25, 40, 39, 38, \ldots, 33\}$;
$d_4 = \{1, 2, 3, 4, 1, 2, 1, 1, 1, 2, 3, 4, 5, 1, 2, 1\}$

$c_5 = \{50, 49, 48, 47, 71, 70, 69, \ldots, 51\}$;
$d_5 = \{1, 2, 3, 1, 1, 2, 3, 1, 2, 3, 4, 5, 6, 1, 2, 3, 1, 2, 3, 4, 1, 2, 1, 1, 2\}$

$c_6 = \{72, 71, \ldots, 61, 96, 95, 94, \ldots, 73\}$;
$d_6 = \{1, 2, 3, 1, 2, 3, 4, 1, 1, 1, 2, 1, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 1, 2, 1, 2, 3, 1, 2, 3, 4, 1, 2, 1, 2\}$

$c_7 = \{98, 97, 96, \ldots, 68, 116, 115, 114, \ldots, 99\}$;
$d_7 = \{2, 3, 4, 1, 2, 3, 1, 1, 1, 1, 2, 3, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1,$
$2, 1, 2, 1, 1, 1, 1, 1, 1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 1, 1, 1, 2, 3, 1\}$

Proof. The case $M = 1$ is the old formula for pairs and $M = 2$ is known, as pointed out above. We will show how the proof works for $M = 3$; the others are identical. The $M = 3$ case breaks into 9 cases according to the congruence classes. Consider the first case, $a = 9k$. The key is identifying algebraic forms for the elbows, some of the protoelbows, and the corners, which we did by using our algorithm for specific values of k. The data are in Table 4.

Table 4 is essentially a complete proof of the formula for $M = 3$. The elbows and corners were first deduced by computation. But the rows beneath them prove that they are really elbows and corners. For the elbows, a protoelbow is below it having weight that is divisible by $9k$ (hence a lattice point) and positive (this proves the elbow is not in $\mathcal{D}$). This proves that the vectors in $\mathcal{E}$ are all contained in the set of preelbows. As soon as we know that the volume of the domain determined by $\mathcal{E}$ (more on that in a moment) is $9k$, we will know that $\mathcal{E}$ is the exact set of elbows. For if there were another protoelbow that yielded an elbow that was dominated by one of the proposed ones, the volume of the domain would be too small.

For the set $\mathcal{C}$ of proposed corners, the indices below each indicate which elbows (numbered 1 to 6 reading from the left) show that moving out 1 in each coordinate takes one outside $\mathcal{D}$; these are called the *active elbows* for a given (proposed) corner. For example, the first elbow, $(4, 0, 0)$ shows that $(4, 1, 0)$ is not in $\mathcal{D}$, and so is an active elbow for the first proposed corner with respect to the first coordinate. Now, we also can check that each point of $\mathcal{C}$ is in $\mathcal{D}$ by checking that there is no elbow that it dominates, for that is the only way a point gets

					Volume
Corner weights	$36k+7$	$9k^2+18k-1$	$9k^2+27k-3$	$9k^2+27k-2$	
Corners, $\mathcal{C}$	$(3,1,0)$	$(0,2,k-1)$	$(2,1,k-1)$	$(0,4,k-2)$	$9k$
Active elbow indices	$1,4,5$	$4,6,3$	$5,4,3$	$4,2,6$	
Elbows, $\mathcal{E}$	$(4,0,0)$ $(0,5,0)$	$(0,0,k)$	$(1,2,0)$ $(3,0,1)$	$(0,3,k-1)$	$9k$
Protoelbows	$(4,-1,0)$ $(-2,5,-2)$	$(0,0,k)$	$(1,2,-1)$ $(3,-3,1)$	$(-3,3,k-1)$	
Protoelbow weights	$27k$	$9k$	$9k(k+1)$	$18k$ $9k$ $9k^2$	
Protoelbow weights divided by $9k$	3	1	$k+1$	3 1 k	

Table 4: The parameters that define the fundamental domain for $A = (9k, 9k+1, 9k+4, 9k+9)$, assuming $k \geq 2$. The Frobenius corner and its weight are shown in bold; subtracting $9k$ from this maximum corner weight gives the Frobenius number.

knocked out of $\mathcal{D}$. That check is easily automated. Then once we know that the volume of the domain subtended by $\mathcal{C}$ is exactly $9k$, we will know that $\mathcal{C}$ is the full set of corners (for if there were another one, it must be in the domain subtended by $\mathcal{C}$, and this would mean the volume of the domain would be too small). Finally, using the assumption that $k \geq 2$, the weights of the corners in the first row show that the fourth corner is the Frobenius corner; therefore the Frobenius number is $9k^2+27k-2-9k$, or $9k^2+18k-2$, as claimed.

For the volume computation, consider the domain defined by a set of (proposed) corners: it is the union of the rectangular boxes in $\mathbb{N}^3$ subtended by the corners under domination. This means that a straightforward inclusion-exclusion formula can be used to compute the volume of the domain. One sums the volumes of the boxes determined by the corners, subtracts the boxes that arise as intersections of boxes defined by two corners, adds back the ones that are intersections of three, and so on. This can be programmed but is too slow if there are a lot of corners. Note that in the sets that arise, the symbol k occurs only in the last dimension. This means that all the boxes that arise have volume that is a product of integers with a linear function of k and so is a linear function of k (for this one adds 1 to the coordinates; e.g., a corner $(0,0,k)$ subtends a column of volume $1 \cdot 1 \cdot (k+1)$). We want the volume to be $9k$, so we have a linear relationship that is verified as soon as we have two cases for which it is true! So we need only check that the proposed corner set is the true corner set (for which the volume must be $9k$) in two cases where k is set to an integer. This was already done once, since that is how we found the corners in Table 4. Doing it once more shows that the volume relationship holds in general. The volume determined by the elbows can be viewed as a corner-determined-volume in the complement of the domain with respect to the bounding box. Thus the same algebraic argument applies: the volume of the domain determined by the elbows is a linear function of k. So again, two instances proves the general formula, and this proves that the proposed elbow set is the true elbow set. We learn from this that the axial elbows are $(4,0,0)$, $(0,5,0)$, and $(0,0,k)$. This is a

nice example of a situation where the truth of the general pattern follows from the truth of a small number of special cases.

Repeating all this for the other eight congruence classes completes the proof for $M = 3$. The condition $k \geq 2$ shows up in the other classes, but the conjectured formula is easily checked for small cases, yielding its truth for $a \geq 16$ (and for $a = 3$, 6, 7, or 11).

The proofs for larger values of M are identical to the one just given, with small-number computations to obtain the exact lower bound on a for which the formulas are valid. For the largest case, $M = 7$, there are 49 sets of protoelbows to compute so as to identify the necessary ones to give the elbows and corners, for a total of 447332 protoelbows. □

There are few patterns to the coefficient sets, but it certainly appears that the general form holds for all M and sufficiently large a; that is, $g(Q(a, M)) = \frac{1}{M^2}(a^2 + c\ a) - d$ for appropriate coefficients that depend on M and on j, the mod M^2 residue of a. Conjectures can be dangerous, though. The one visible pattern—$c_{M,0} = 2M^2$—is false in general! That pattern says that $g(Q(k\ M^2, M)) = k(k+2)M^2 - d_{M,0}$. We cannot use lattice methods, which are limited to small n, to check large M, but we can use the methods of [7]. Something shocking happens when M reaches 28; experiments show that the general form just mentioned for residue class 0 is valid up to 27, with $c_{M,0} = 2M^2$. But for $M = 28$ and for $k = 7, 8, 9, \ldots, 1000$, $g(Q(k\ 28^2, 28)) = k(k+1)\ 28^2 - 1$, meaning that the $c_{28,0}$ appears to be 28^2, as opposed to the expected $2 \cdot 28^2$. This shows the value of having two very different algorithms available.

These ideas might well yield uniform proofs of many of the special cases in the literature, such as arithmetic progressions or powers of 2 (see [36], §3.3). Regarding the latter, for example, computations show (we have not tried to prove it) that if $A = \{a, a+1, a+2, a+4, \ldots, a+2^M\}$ and $a = k2^M$, then the domain $\mathcal{D}$ is simply a box; the single corner is $\{1, 1, \ldots, 1, k-1\}$; and the Frobenius number is $2^M k(M+k-1) - 1$. For other congruence classes of $a \pmod{2^M}$ there are similar patterns, though more corners.

18. Generating Functions when $n = 3$

In this section we will show, given $A = (a, b_1, b_2)$, how to quickly compute the rational function $p(x)/q(x)$ that is the generating function for the set $\mathrm{Reps}(A)$. One application will be a method for computing $N(A)$, the number of nonrepresentable positive integers.

Sylvester proved that for a basis of size 2 exactly one-half of the possibilities (interval from 0 to $g(A)$, inclusive) are nonrepresentable: $N(a, b) = (ab - a - b + 1)/2$. The computation of $N(A)$ in general can be carried out (at least when n is not too large, say $n \leq 6$) using a generating-functions approach pioneered by Barvinok and Woods [5]. Before treating the general case we discuss the $n = 3$ case in some detail, as there we can apply the generating function both to compute $N(A)$ (though Tinaglia [46] had found a formula for this already) and to improve a theorem of Nijenhuis and Wilf [31] to completely characterize the triples for which $N(A) = (1 + g(A))/2$.

Let $q(x) = \frac{1}{(1-x^a)(1-x^{b_1})(1-x^{b_2})}$; $q(x)$ is a product of geometric series and so is clearly the generating function for the sequence $\{r_M\}$, where r_M is the *number* of Frobenius A-representations of M. Of much more interest is a generating function that has coefficient 1 for each representable integer. That has the form $p(x)/q(x)$, $q(x)$ as just given (see [47] and [39] for more on generating functions in this context). When $n = 3$ the numerator $p(x)$ can be given explicitly as follows (this was done by Denham [16], but our approach is different). Let w_1 and w_2 be the weights of the two axial elbows $(k_1, 0)$ and $(0, k_2)$ and let w_3 be the weight of the minimal elbow (x_3, y_3). Let $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ be the two external corners, which are $\boldsymbol{C}_1 = (k_1, y_3)$ and $\boldsymbol{C}_2 = (x_3, k_2)$ and let v_1, v_2 be their weights (if there is one corner only then only one of these will actually be an external corner, but the definition makes sense regardless). Relabel if necessary to make $v_2 \geq v_1$, so that $\boldsymbol{C}_2$ is the external Frobenius corner and $v_2 = G(A) = g(A) + \sum A$. If it happens that there is only one corner, define $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ as above and note that $\boldsymbol{C}_1$ will agree with the minimal elbow (and hence the weight v_1 will equal the minimal weight w_3).

Definition. If $A = (a, b_1, b_2)$ then $p(x)$ is defined to be $1 - x^{w_1} - x^{w_2} - x^{w_3} + x^{v_1} + x^{v_2}$.

Notes: Because v_2 is larger than the other weights that occur, the only cancellation that can arise is when v_1 equals one of the w_i. Therefore $p(x)$ has either 4 or 6 terms (see [16] for an alternative approach to this result and the theorem that follows). It is also true that $v_1 \geq \max w_i$ (proof follows from the fact (§4) that the minimal elbow is the sum of the two axial protoelbows); note that it can happen that the w_i are not distinct.

Theorem 6 [16]. When $n = 3$ and with $p(x)$ and $q(x)$ as just defined, $p(x)/q(x)$ is the generating function for the set $\mathrm{Reps}(A)$.

Proof. Assume first that the minimal elbow is not axial, so that the domain is not a rectangle. Then consider $\frac{1}{(1-t_1)(1-t_2)(1-t_3)}$. Expanded out, this represents all of $\mathbb{N}^3$ as $\sum\{t_1^i t_2^j t_3^k : i, j, k \in \mathbb{N}\}$. What we want is a representation of $\sum\{t_1^i t_2^j t_3^k : i \in \mathbb{N} \text{ and } (j, k) \text{ in } \mathcal{D}\}$, where $\mathcal{D}$ is the fundamental domain of A. We will explain in a moment why this is what is needed. Now, we can get this representation by removing the unnecessary terms. If the axial elbows are $(k_1, 0), (0, k_2)$ and the interior elbow is (x_3, y_3) we can remove the excess by considering the function $\frac{1}{(1-t_1)(1-t_2)(1-t_3)} - \frac{t_2^{k_1}}{(1-t_1)(1-t_2)(1-t_3)} - \frac{t_3^{k_2}}{(1-t_1)(1-t_2)(1-t_3)} - \frac{t_2^{x_3} t_3^{y_3}}{(1-t_1)(1-t_2)(1-t_3)}$. To see how this works, look at Figure 13. But now we have removed some points twice and need to replace them so we get $\frac{1-t_2^{k_1}-t_3^{k_2}-t_2^{x_3}t_3^{y_3}+t_2^{k_1}t_3^{y_3}+t_2^{x_3}t_3^{k_2}}{(1-t_1)(1-t_2)(1-t_3)}$. Similar reasoning shows that this same expression works when there is no interior elbow, for then the minimal elbow is either $(k_1, 0)$ or $(0, k_2)$; in either case the result is $\frac{1-t_2^{k_1}-t_3^{k_2}+t_2^{k_1}t_3^{k_2}}{(1-t_1)(1-t_2)(1-t_3)}$.

Now recall that, for each point in $\mathcal{D}$, its weight and all greater numbers congruent to it modulo a are representable and this is the full set of representables. Therefore the generating function above has exactly one point for each coordinate triple that leads to a representable number after forming the dot product with A. Since we want to work with the weights as opposed to their coordinates, we apply the transformation $t_1 = x^a$, $t_2 = x^b$, $t_3 = x^c$. This gives us the form claimed for $p(x)$. □

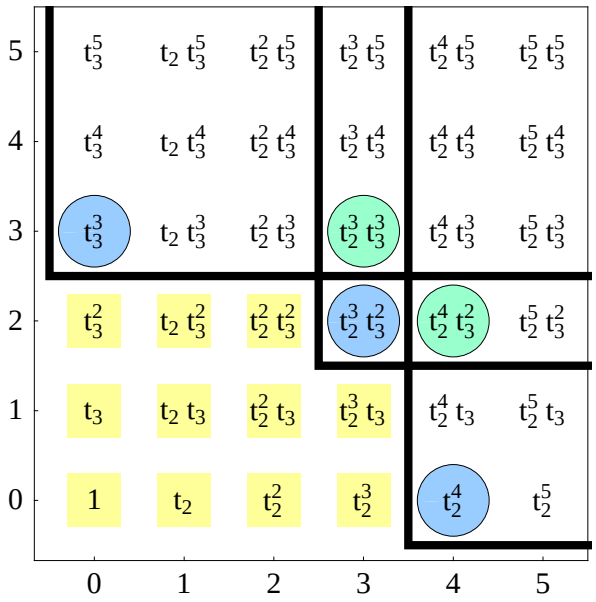

Figure 13: Here $A = \{11, 17, 24\}$ and points of $\mathbb{N}^2$ are labeled as in the proof of Theorem 7. Points in the fundamental domain $\mathcal{D}$ are shown in yellow, with the elbows in blue and external corners in green. Excluding the cones of the elbows and using the external corners to account for the multiple exclusions yields an expression for $\sum\{t_2^j t_3^k : (j,k) \in \mathcal{D}\}$. In this case the expression is $\frac{1-t_2^4-t_3^3-t_2^3t_3^2+t_2^4t_3^2+t_2^3t_3^3}{(1-t_2)(1-t_3)}$.

Note that, because $p(x) = q(x)\sum\{x^i : i \in \mathrm{Reps}(A)\}$, $p(x)$ is independent of the order of A. Now, having the generating function for $\mathrm{Reps}(A)$ in hand means that $F(x) = \frac{1}{1-x} - \frac{p(x)}{q(x)}$ is the generating function for the set of nonnegative integers having no A-representation. The largest power in $F(x)$ is therefore the Frobenius number $g(A)$ and $F(1) = N(A)$, the number of nonrepresentables; $F(1)$ can be computed from l'Hopital's rule as $\frac{p'''(1)}{3!\prod A} + \frac{3}{2} - \frac{1}{2}\sum A$.

Tinaglia [46] obtained a simpler formula for $N(a, b_1, b_2)$ and we can use that to simplify the one involving p''' as follows. Let $W = \{w_1, w_2, w_3\} = \{k_1b_1, k_2b_2, \mu\}$, where μ is the minimal weight. Then observe that $\frac{1}{3}p'''(1) = (\sum W - 2)(\prod A) - \prod W$; to see this write the difference in terms of $k_1, k_2, a, b_1, b_2, x_3, y_3$ and subtract and simplify to get an expression with a factor $k_2x_3 + (k_1 - x_3)y_3 - a$, which is 0 because the first two terms sum to the number of points in the domain. Substituting for the third derivative in the formula for $N(A)$ then yields

$$N(A) = \tfrac{1}{2}\left(\sum W - \sum A - \tfrac{\prod W}{\prod A} + 1\right) = \tfrac{1}{2}\left(\sum W - \sum A - k_1k_2\tfrac{\mu}{a} + 1\right)$$

This last expression is the one found by Tinaglia; because all the numbers of the formula can be gotten very quickly, one can compute $N(a, b_1, b_2)$ in an eyeblink even at the 100-digit level. Thus we can study the change in the average value of $\omega(A)$, which denotes the ratio of $N(A)$ to $g(A) + 1$. Using 1000 trials on triples having $a = 10^i$ and $c < 10^{i+1}$, and letting $i = 4, 10, 50, 100$, we find that the average value of ω in the four cases is 0.51179, 0.511735, 0.511414, 0.512039, respectively.

The generating function can help us completely characterize triples for which $\omega(A) = 1/2$. Recall that when $n = 2$ it always happens that $\omega(A) = 1/2$. The investigation of Frobenius bases for which $\omega = 1/2$ was started by Nijenhuis and Wilf [31], who proved several results about them, including the following sufficient condition: If c is representable in terms of $(a/d, b/d)$, where $d = \gcd(a, b)$, then $\omega(a, b, c) = \frac{1}{2}$. And they gave $A = (6, 7, 8)$, for which $g(A) + 1 = 18$ and $N(A) = 9$, as a counterexample to the converse. Theorem 8 below shows that allowing permutations leads to a necessary and sufficient condition for ω to be $\frac{1}{2}$. A central idea is an upper bound found by A. Brauer [11].

Definition. For a Frobenius basis $A = (a, b_1, \ldots, b_m)$ let $d_i = \gcd(a, b_1, \ldots, b_i)$; let $d_0 = a$. Let $\mathrm{BB}(A)$ be $\sum_{i=1}^{m} b_i d_{i-1}/d_i - \sum A$ and let $\mathrm{BB}^*(A)$ be the minimum of $\mathrm{BB}(A)$ over all permutations of A. Further, A is *sequentially redundant* if each $b_i/d_i \in \mathrm{Reps}(\{a, b_1, \ldots, b_{i-1}\}/d_{i-1})$.

We first summarize the known results.

Theorem 7. For any basis A (no restriction on n):
1. $g(A) \le \mathrm{BB}(A)$.
2. $g(A) \le \mathrm{BB}^*(A)$.
3. A is sequentially redundant iff $g(A) = \mathrm{BB}(A)$.
4. If A is sequentially redundant then $\omega(A) = 1/2$.

Proof. (1) is due to Brauer. (2) follows from (1). (3) and (4) were proved by Nijenhuis and Wilf, who also showed that $\omega(A) = 1/2$ is equivalent to A satisfying the Gorenstein condition; since we do not use that condition, we refer the reader to [31] for its definition. □

Now, by bringing permutations into the picture, we can show that several assertions for triples are fully equivalent.

Theorem 8. For $A = (a, b_1, b_2)$, the following are equivalent:
1. $\omega(A) = 1/2$.
2. $g(A) = \mathrm{BB}^*(A)$.
3. There is a permutation of A for which the fundamental domain is a rectangle.
4. There is a permutation of A for which the minimal elbow has one entry that is 0.
5. There is a permutation of A that is sequentially redundant.
6. $p(x)$ is symmetric in that $p(x) = p(1/x)x^{g(A)}$.

Proof. $2 \Leftrightarrow 5 \Rightarrow 1$ are in [31] and $4 \Leftrightarrow 3$ is just a restatement of the definition of axial and minimal elbows.

$5 \Rightarrow 4$: We prove that if A is sequentially redundant then A has only two elbows. So suppose $d = \gcd(a, b_1)$ and $b_2 = \alpha(a/d) + \beta(b_1/d)$. It follows that $(a/d, 0)$ and $(-\beta, d)$ are in the lattice L. This implies that, if (k_1, k_2) is the bounding vector, then $k_1 \le a/d$ and $k_2 \le d$. But the product of these two bounds is a, the area of $\mathcal{D}$, so the two inequalities must be equalities. Further, the rectangle defined by the bounding vector must equal $\mathcal{D}$, and therefore there is no non-axial elbow. (This proof works for any n.)

$3 \Rightarrow 5$: Suppose all elbows are axial. There is no interior elbow. Then the minimal elbow μ is an axial elbow. Assume $\mu = (\alpha, 0)$. Since $\mathcal{D}$ is a rectangle, α divides a. Let $d = a/\alpha$ and

let the other axial protoelbow be $(-x, d)$. Then $db_2 - xb_1$ is nonnegative and divisible by a, and so b_2 is representable in terms of a/d and $b_1/\,d$. If $\mu = (0, \alpha)$ then the same argument shows that b_1 is representable. Note that permutations are essential here.

$1 \Rightarrow 6$: In general (first observed by Nijenhuis and Wilf) if $x + y = g(A)$ then at most one of x or y is representable. Therefore if $\omega(A) = 1/2$, it must be that exactly one is representable. We can now get the nonrepresentables generating function $F(x)$ in two ways: First, we have $F(x) = \frac{1}{1-x} - \frac{p(x)}{q(x)}$. But the comment just made means that $F(x) = 1 + x + x^2 + \cdots + x^g - x^g F\left(\frac{1}{x}\right)$, where $g = g(A)$: $1 + x + x^2 + \cdots + x^g - x^g F\left(\frac{1}{x}\right)$. But this is $\frac{1-x^{g+1}}{1-x} - x^g\left(\frac{1}{1-\frac{1}{x}} - \frac{p(1/x)}{q(1/x)}\right)$. The difference of the expressions for $F(x)$ simplifies to $(p(x) - x^{G(A)}p(1/x))/\,q(x)$ and so, knowing that this difference is 0, gives (6).

We now have $3 \Rightarrow 5 \Rightarrow 4 \Rightarrow 1 \Rightarrow 6$ and $5 \Leftrightarrow 2 \Rightarrow 1$. We conclude with $6 \Rightarrow 3$. The sign pattern of $p(x)$, if there is no cancellation, is $+---++$ while that of $p(1/x)x^{g(A)}$ is $++---+$. So these cannot be equal. So the only way (6) can hold is if there is cancellation; then the sign pattern is $+--+$ and $p(x) = 1 - x^{w_{i_1}} - x^{w_{i_2}} + x^{v_2}$. We need to deduce from this that there is a permutation of A for which the domain is a rectangle.

So now we are in the situation that the domain is not a rectangle (otherwise we are done) and one of the two external corners has the same weight as an axial elbow. Suppose the external corner (x_3, k_2) has the same weight as the axial elbow $(k_1, 0)$. So $x_3b_1 + k_2b_2 = k_1b_1$. Now permute the basis into $A_1 = (b_1, b_2, a)$. We know that $k_2b_2 = (k_1 - x_3)b_1$. Let (m_1, m_2) be the minimal elbow for A_1. We will prove that one of the m_i is 0, which means $\mathcal{D}(A_1)$ is a rectangle. Suppose not. Then $(m_1, m_2) \cdot (b_2, a)$ is the minimal weight of a point in L^+ for A_1; let γ be such that $\gamma b_1 = m_1b_2 + m_2a$. But $(k_2, 0)$ is in L^+, so $\gamma b_1 = m_1b_2 + m_2a \leq k_2b_2 = (k_1 - x_3)b_1 < k_1b_1$. This means that $(\gamma, 0) \in \mathcal{D}(A)$. But the A-weight of this point is $\gamma b_1 = m_1b_2 + m_2a$; because $m_2 \neq 0$, this means that $(0, m_1)$ has lesser A-weight and is equivalent to $(\gamma, 0)$, contradiction. The other case, where the external corner (k_1, y_3) has the same weight as the axial elbow $(0, k_2)$, is essentially identical. $\square$

Condition (5) is very fast to check, but this doesn't really help computationally since, for random data at 10^6 or beyond, it is very rare that A is degenerate. For the record, the following is the algorithm: Given (a, b_1, b_2) determine whether $b_2 \in \mathrm{Reps}(a/d, b_1/\,d)$, where $d = \gcd(a, b_1)$ by the following well-known technique. Let $(a', b') = (a, b_1)/\,d$; let b_2^* be the least nonnegative residue of $b_2 \pmod{a'}$. Let M be the first multiple of b' in this residue class: $M = kb'$ where k is least nonnegative residue mod a' of $-b_2^*(a')^{-1}$. Then repeat for the other two choices of redundant number.

As observed earlier, $A = (6, 7, 8)$ is an example for which ω is $\frac{1}{2}$ but 8 is not representable in terms if 6 and 7. Theorem 8 above shows that the example behaves properly provided one allows permutations; reorder A as $(6, 8, 7)$ and observe that $7 = 3 + 4$. Moving up to $n = 4$, we do not know how to characterize bases A for which $\omega(A) = 1/2$. Consider the example $A = (5, 7, 9, 11)$. Then $\omega(A) = 1/2$ but no permutation of A is sequentially redundant or has a domain which is a cube.

The idea of using the elbows to compute the generating function appears to work when $n \geq 4$ and the elbow set is not too large. That is, when n is between 4 and 7, we have a method that produces the generating function reasonably quickly (and therefore also $N(A)$), but we do not yet have a complete proof of correctness. We can, however, show quite easily how to go from the generating function to $N(A)$ in the general case.

Suppose $F(x)$ is the nonrepresentable generating function. Then $F(x) = \frac{1}{1-x} - \frac{p(x)}{q(x)}$, and $N(A)$ is simply $F(1)$. As noted earlier, this can be calculated by l'Hopital's rule, and simple algebra leads to

$$N(A) = \frac{(-1)^{n-1}p^{(n)}(1)}{n! \prod A} + \frac{n}{2} - \frac{1}{2}\sum A.$$

Thus if one has $p(x)$ it is very easy to get $N(A)$.

19. Connections with Toric Gröbner Bases

In this section we show how to create the fundamental domain using toric Gröbner bases. The method is reasonably efficient for the sorts of problems we have considered in this paper in dimensions 4 to 6 and, with effort, handles sets of 7, with elements in the set as large as 10^{40} or so.

We first remark that the Frobenius instance problem can be solved using well known integer programming techniques involving toric Gröbner bases. Details may be found in [14] or [35]; a *Mathematica* implementation is in [22]. Finding a fundamental domain using toric bases is a bit different from integer programming via such bases. The main point to notice is that the fundamental domain has a staircase structure, as do the leading exponent vectors of a Gröbner basis.

Recall that protoelbows have both positive and negative coordinates and correspond to certain "minimal" equivalences (that is, reducing relations) in the lattice. These are lattice points where a positive combination of one subset of elements equals a positive combination of a complementary subset. In Gröbner basis terms these will appear as exponent vectors of binomial pairs in the basis; each term goes with one of the element subsets, and powers are the multipliers in the combinations. We describe the process in brief below.

As with instance solving a la [14] one might set up relations (that is, generating polynomials for a toric ideal) of the form $x_j - t^{a_j}$ and eliminate t. For better efficiency we will use an improvement, due to Pottier [35], in the Gröbner basis computation of the elimination ideal. Instead of $x_j - t^{a_j}$ we work with polynomials $x^{e_j^+} - x^{e_j^-}$ where e_j^+ and e_j^- are the positive and negative parts respectively of the jth generator for the null space of $A = \{a_1, a_2, \ldots, a_n\}$. As we can use any basis for this purpose we choose one that is lattice reduced so as to keep down the exponent sizes. We augment by a polynomial $1 - u \prod x_j$ that in effect allows us to invert negative exponents, and use a monomial ordering that eliminates the new variable u.

For example, suppose our set is $A = (1854, 2712, 2266, 7857)$. The null space is spanned by the vectors $(-11, 0, 9, 0)$, $(-10, 7, -21, 6)$, and $(-1, -25, 3, 8)$.

We have three generators and they have four components. These vectors give rise to the set of generating polynomials below. Note that there are three such (plus a polynomial for the power inversion), and they use four variables. That is, polynomials correspond to null vector generators, and variables correspond to components of those generators.

$$\{-x_1^{11}+x_3^9, -x_1^{10}x_3{}^{21}+x_2^7x_4^6, -x_1x_2^{25}+x_3^3x_4^8, -1+ux_1x_2x_3x_4\}$$

Our Gröbner basis computed with respect to degree reverse lexicographic order in the principle variables, and eliminating the inversion variable u, is as below.

$$\{x_1^{11}-x_3^9, x_1x_2^{25}-x_3^3x_4^8, x_3^{30}-x_1x_2^7x_4^6, x_1^{10}x_3^{21}-x_2^7x_4^6,\\ x_2^{25}x_3^6-x_1^{10}x_4^8, -x_2^{32}+x_1^9x_3^{24}x_4^2, x_2^{18}x_3^{27}-x_4^{14}, x_2^{50}x_3^3-x_1^9x_4^{16}, x_2^{57}-x_1^8x_3^{27}x_4^{10}\}$$

From this basis we next want to find a new one that will enforce the "lexically last" provision of the fundamental domain definition. As our lattice is now represented by exponent vectors in a toric ideal this amounts to an inverse lexicographical term ordering. But this is not a well-founded ordering for monomials because it has constants larger than any power products (it is an ordering appropriate for a local ring). For this we use a standard tactic of homogenizing (see, e.g., [6], §10.6). We make the homogenizing variable largest and then work with a degree-based term order. As we want an inverse lexicographic order we use the customary graded reverse-lexicographic (grading by total degree is fine because we homogenized; we'll arrive at an inverse lexicographic order upon dehomogenizing). We now compute a new Gröbner basis with respect to this order and then dehomogenize. Note that this second basis computation tends to be quite fast compared to the first, hence is not problematic in regard to efficiency.

For the example under discussion this new basis, after dehomogenization, is as below.

$$\{-x_1^{11}+x_3^9, -x_1x_2^{25}+x_3^3x_4^8, x_3^{30}-x_1x_2^7x_4^6, x_2^{25}x_3^6-x_1^{10}x_4^8,\\ -x_1^{10}x_3^{21}+x_2^7x_4^6, x_2^{32}-x_1^9x_3^{24}x_4^2, -x_2^{18}x_3^{27}+x_4^{14}, x_2^{50}x_3^3-x_1^9x_4^{16}, x_2^{57}-x_1^{41}x_4^{10}\}$$

We now recover our full set of reducing relations, as integer vectors, from the basis.

$$(-11,0,9,0), (-1,-25,3,8), (-1,-7,30,-6), (-10,25,6,-8), (-10,7,-21,6),\\ (-9,32,-24,-2), (0,-18,-27,14), (-9,50,3,-16), (-41,57,0,-10)$$

Recall that our algorithm works modulo a_1. Hence we drop first elements of these vectors to obtain, finally, our protoelbows.

$$(0,9,0), (-25,3,8), (-7,30,-6), (25,6,-8), (7,-21,6),\\ (32,-24,-2), (-18,-27,14), (50,3,-16), (57,0,-10)$$

Protoelbows by Gröbner Bases

Input. A Frobenius basis $A=\{a_1,a_2,\ldots,a_n\}$.
Output. The set of all protoelbows.

Step 1. Compute a lattice reduced basis for the integer null space of A.

Step 2. Separate the null vectors into positive and negative parts.

Step 3. Create difference binomials whose monomial exponents are respectively the positive and negative parts from step 2. Note that this requires one variable for each component of the null vectors.

Step 4. Augment with a binomial that is the difference of a new variable and the product of those already in use. Compute a Gröbner basis with respect to a term order that eliminates this new variable.

Step 5. Homogenize the resulting basis. To achieve inverse lexicographic ordering we
(i) Prepend the homogenizing variable to the reversed list of variables.
(ii) Compute a new Gröbner basis using the degree-reverse-lexicographic term ordering with respect to this modified variable list.
Dehomogenize the result.

Step 6. Convert from binomials to exponent vectors.

Step 7. Orient each exponent vector (i.e., possibly multiply by -1) so that the positive part has smaller weight (as defined in §3) than the negative part. In case of a tie decide based on which is smaller in inverse lexicographic ordering. This step is needed so that we obtain the correct positive parts for preelbows.

Step 8. Remove first components (because our fundamental domain is defined by relations modulo a_1). The resulting vectors give the protoelbows.

Quite recently Roune [37] has simplified and considerably improved on this method. A key idea is to use the set A as a homogenizing vector, instead of the oft-used (unweighted) total degree. As the null vectors are already homogeneous with respect to this vector, the Gröbner basis computation becomes considerably more efficient. There are some other differences spelled out in that paper. Coupled with dedicated toric ideal software for the actual computations, [37] obtains some spectacular results.

20. Open Questions

Open problems that seem tractable arise in both the algorithmic and theoretical contexts.

Algorithmic questions

Problem 1. Suppose $n = 4$. Is there an implementable algorithm that is polynomial-time in the worst case?

Problem 2. Work out a way to get the Frobenius corner at the same time as the elbows are computed, or perhaps avoiding elbow computations altogether, thus eliminating the recursive

approach that starts with all elbows and then gets the farthest corner. Possibly ideas of ILP and branch-and-bound could be used here.

Theoretical questions

Problem 1. Can one tell from the basis whether it might trigger a situation where the number of elbows is exceedingly large?

Problem 2. Suppose $n = 4$. Is there a characterization of bases A for which $\omega(A) = 1/2$? For which the fundamental domain is a cuboid?

Problem 3. The fact that there is at most one interior elbow restricts the possibilities for the shape of the fundamental domain. For example, a domain that is essentially a simplex is not possible. This explains why Killingbergtrø's bound, which uses a simplex of volume a to approximate $\mathcal{D}$ is not sharp (see Cor. 1). Can one improve the geometrical argument of Corollary 1 to make use of the interior elbow fact so as to sharpen the lower bound on $g(A)$?

Problem 4. Find a pattern in the coefficient sets of Theorem 5 in §17. Then extend that theorem by finding and proving a general formula for $g(a, a+1, a+4, \ldots, a+M^2)$.

Problem 5. Prove our conjecture that when n is fixed there is a constant C_n such that the expected number of protoelbows for inputs that are random between a and $10a$ is asymptotically bounded by C_n as a grows. The evidence that $C_4 \leq 12$ is strong.

Note. *Mathematica* code to generate sets of protoelbows, elbows, corners, and so on is available from the fourth author. *Mathematica* now has built-in code to find Frobenius numbers and solve Frobenius instances using both the ideas of this paper and [7].

References

[1] K. Aardal, C. A. J. Hurkens, and A. K. Lenstra. Solving a system of linear diophantine equations with lower and upper bounds on the variables. *Mathematics of Operations Research*, **25** (2000) 427–442.

[2] K. Aardal and A. K. Lenstra. Hard equality constrained knapsacks. Proceedings of the 9th Conference on Integer Programming and Combinatorial Optimization (IPCO 2002), W. J. Cook and A. S. Schulz, eds. *Lecture Notes in Computer Science*, **233** (2002) 350–366. Springer-Verlag.

[3] K. Aardal, R. Weismantel, and L. A. Wolsey. Non-standard approaches to integer programming. *Disc. Applied Math.*, **123** (2002) 5–74.

[4] A. I. Barvinok and J. Pommersheim. An algorithmic theory of lattice points in polyhedra. In: *New Perspectives in Algebraic Combinatorics*, Math. Sci. Res. Inst. Publ. 38, Cambridge Univ. Press, Cambridge (1999) 91–147.

[5] A. I. Barvinok and K. Woods. Short rational generating functions for lattice point problems. *J. Amer. Math. Soc.*, **16** (2003) 957–979.

[6] T. Becker, W. Weispfenning, and H. Kredel. *Gröbner Bases: A Computational Approach to Computer Algebra*. Graduate Texts in Mathematics, **141** (1993) Springer-Verlag, Berlin.

[7] D. Beihoffer, A. Nijenhuis, J. Hendry, and S. Wagon. Faster algorithms for Frobenius numbers. *Elec. J. Combinatorics*, **12** (2005) #R27

[8] J. L. Bentley, K. L. Clarkson, and D. B. Levine. Fast linear expected-time algorithms for computing maxima and convex hulls. *Algorithmica*, **9** (1993) 168–183.

[9] W. A. Blankenship. Algorithm 288: Solution of simultaneous linear diophantine equations. *Communications of the ACM*, **9**(7) (1966) 514.

[10] S. Böcker and Z. Lipták, The money changing problem revisited: Computing the Frobenius number in time $O(ka_1)$, Computing and Combinatorics Conference (COCOON), Kunming, China (2005) 965–974.

[11] A. Brauer. On a problem of partitions. *Amer. J. of Math.*, **64** (1942) 299–312.

[12] A. Brauer and J. E. Shockley. On a problem of Frobenius. *Journal für Reine und Angewandte Mathematik*, **211**(3/4) (1962) 215–220.

[13] D. Bressoud and S. Wagon. *A Course in Computational Number Theory*. Key College Publ. Emeryville, Calif., (2003).

[14] P. Conti and C. Traverso. Buchberger algorithm and integer programming. Proceedings of the 9th International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (AAAAECC 91). *Lecture Notes In Computer Science*, **539** 130–139.

[15] J. L. Davison. On the linear diophantine problem of Frobenius. *J. Number Theory*, **48** (1994) 353–363.

[16] G. Denham. Short generating functions for some semigroup algebras. *Electronic J. Combinatorics*, **10** (2003) #R36.

[17] P. D. Domich, R. Kannan, and L. E. Trotter, Jr. Hermite normal form computation using modulo determinant arithmetic. *Mathematics of Operations Research*, **12**(1) (1987) 50–59.

[18] F. Eisenbrand. Short vectors of planar lattices via continued fractions. *Information Processing Letters*, **79** (2001) 121–126.

[19] F. Eisenbrand and G. Rote. Fast 2-variable integer programming. *Integer Programming and Combinatorial Optimization* (IPCO 2001), K. Aardal and B. Gerards, eds. 78–89. *Lecture Notes in Computer Science*, **2081** (2001) Springer-Verlag.

[20] H. Greenberg. Solution to a linear Diophantine equation for nonnegative integers. *J. Algorithms*, **9** (1988) 343–353.

[21] R. Kannan. Lattice translates of a polytope and the Frobenius problem. *Combinatorica*, **12**(2) (1992) 161–177.

[22] D. Kapadia. Integer programming with Gröbner bases. *Mathematica* Demo notebook (2003) http://library.wolfram.com/infocenter/Demos/4825/.

[23] H. G. Killingbergtrø. Betjening av figur i Frobenius' problem (Using figures in Frobenius's problem), (Norwegian) *Normat*, **2** (2000) 75–82.

[24] H. Krawczyk and A. Paz. The diophantine problem of Frobenius: a close bound. *Disc. Applied Math.*, **23** (1989) 289–291.

[25] http://www.math.ucdavis.edu/∼latte/.

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

[27] D. Lichtblau. Revisiting strong Gröbner bases over Euclidean domains. (2003) Manuscript.

[28] Half-GCD, fast rational recovery, and planar lattice reduction. Extended version of author's Half-GCD and fast rational recovery, *Proceedings of the 2005 International Symposium on Symbolic and Algebraic Computation (ISSAC 2005)*, M. Kauers, ed. ACM Press, New York City, (2005) 231–236. http://members.wolfram.com/danl/HGCD_and_planar_lattices.pdf.

[29] D. Lichtblau. Making change and finding repfigits: Balancing a knapsack. *Proceedings of the Second International Congress on Mathematical Software (ICMS 2006)*, A. Iglesias and N. Takayama, eds. *Lecture Notes in Computer Science*, **4151** (2006) 182–193. Springer-Verlag.

[30] K. R. Matthews. Short solutions of $AX = B$ using a LLL-based Hermite normal form algorithm. (2001) Manuscript.

[31] A. Nijenhuis and H. Wilf. Representations of integers by linear forms in nonnegative integers. *J. Number Theory*, **4** (1972) 98–106.

[32] P. Nguyen. Cryptanalysis of the Goldreich–Goldwasser–Halevi cryptosystem from Crypto '97. Advances in Cryptology, *Proceedings of CRYPTO 1999*, Santa Barbara, CA, (1999). http://www.di.ens.fr/∼pnguyen/pub.html#Ng99.

[33] I. Niven, H. S. Zuckerman, and H. Montgomery. *An Introduction to the Theory of Numbers*. 2nd ed. Wiley, New York, (1991).

[34] R. W. Owens. An algorithm to solve the Frobenius problem. *Math. Mag.*, **76** (2003) 264–275.

[35] L. Pottier. Gröbner bases of toric ideals. INRIA Rapport de recherche, 2224 (1994).

[36] J. L. Ramírez Alfonsín. *The Diophantine Frobenius Problem*. Oxford University Press, Oxford, (2005).

[37] B. Roune. Solving thousand-digit Frobenius problems using Gröbner bases. Preprint (2007) http://lanl.arxiv.org/abs/math.CO/0702040.

[38] H. E. Scarf and D. F. Shallcross. The Frobenius problem and maximal lattice free bodies. *Math. Oper. Res.*, **18** (1993) 511–515.

[39] H. E. Scarf and K. M. Woods. Neighborhood complexes and generating functions for affine semigroups. *Discrete and Computational Geometry*, **35**(3) (2006) 385–403.

[40] A. Schönhage. Schnelle Berechnung von Kettenbruchentwicklungen. *Acta Informatica*, **1** (1971) 139–144.

[41] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley-Interscience Series in Discrete Mathematics and Optimization, New York, (1986).

[42] D. Shallcross. Neighbors of the origin for four by three matrices. *Math. Oper. Res.*, **17**(3) (1992) 608–614.

[43] A. Storjohann and G. Labahn. Asymptotically fast computation of Hermite normal forms of integer matrices. 1996 Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC 96), Y. N. Yakshman, ed. (1996) 259–266. ACM Press.

[44] B. Sturmfels, R. Weismantel, and G. M. Zeigler. Gröbner bases of lattices, corner polyhedra, and integer programming. *Beiträge Alg. Geom.*, **36**(2) (1995) 281–298.

[45] L. A. Szekely and N. C. Wormald. Generating functions for the Frobenius problem with 2 and 3 generators. *Math. Chronicle*, **15** (1986) 49–57. [p. 50 should appear after p. 52]

[46] C. Tinaglia. Su alcune soluzione di un problema di Frobenius in tre variabili. *Boll. U. M. I.*, **7**(2) (1988) 361–383.

[47] K. M. Woods. Rational generating functions and lattice point sets. PhD thesis, (2004) Univ. of Michigan.

[48] S. Wolfram. *The Mathematica Book (5th edition)*. Wolfram Media, Champaign, Ill., (2003).