Lattice Algorithms for Integer Programming

Huck Bennett*

July 21, 2024

These notes present two enumeration-based algorithms for the Integer Programming Problem. First, they present a slightly modified version of Lenstra's algorithm [Len83], and show how to improve the $2^{O(n^3)}$ runtime of his algorithm to $2^{O(n^2)}$ using similar techniques to his, and even to $n^{O(n \log n)}$ time using additional techniques. They then show how to modify this algorithm slightly to get Kannan's algorithm [Kan87], which runs in time $n^{cn+o(n)}$.[1] Here n is the dimension of the integer program and $c > 0$ is an absolute constant. Kannan's algorithm remains the fastest known algorithm for integer programming up to the value of c, which has been improved somewhat.

1 The Integer Programming Problem

The feasibility version of the *integer programming problem* (IP) is to decide whether there exist integers $x_1, \ldots, x_n$ satisfying a system of inequalities

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n &\leq b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n &\leq b_2 \\ &\vdots \\ a_{m,1}x_1 + a_{m,2}x_2 + \cdots + a_{m,n}x_n &\leq b_m \ , \end{aligned}$$

where the values $a_{i,j}, b_i \in \mathbb{Q}$ for $i \in \{1, \ldots, m\}$, $j \in \{1, \ldots, n\}$ are given as input. One can define this problem more concisely in vectorized form by asking whether, given $A \in \mathbb{Q}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{Q}^m$ as input, there exists $\boldsymbol{x} \in \mathbb{Z}^n$ such that $A\boldsymbol{x} \leq \boldsymbol{b}$. Here the inequalities between the vectors are considered component-wise.[2]

Integer programming is one of the most important problems in discrete optimization. There are easy integer programming formulations of many NP-hard problems (e.g., 3-SAT and vertex cover), and so it is not hard to show that integer programming is NP-hard. Additionally, one can show that integer programming is in NP (and hence NP-complete) by using an integer vector $\boldsymbol{x}$ satisfying $A\boldsymbol{x} \leq \boldsymbol{b}$ as a certificate of feasibility.[3]

Despite its NP-hardness, it is natural to ask whether there is an algorithm for integer programming that runs in polynomial time (in the input length S) when the dimension n is fixed. In a pair of seminal papers, Hendrik Lenstra [Len83] answered this question in the affirmative by giving a $(2^{O(n^3)} \cdot \mathrm{poly}(S))$-time algorithm for integer programming, and then Kannan [Kan87] gave an improved algorithm running in $(n^{cn+o(n)} \cdot \mathrm{poly}(S))$-time for some constant $c > 0$.

*Oregon State University. huck.bennett@oregonstate.edu

[1] Here and in some of what follows we ignore polynomial factors in the input length when discussing runtimes.

[2] One may also consider the *optimization version* of integer programming, where the input additionally contains a cost vector $\boldsymbol{c} \in \mathbb{Z}^n$, and the goal is to compute the minimum value of $\langle \boldsymbol{c}, \boldsymbol{x} \rangle$ over all $\boldsymbol{x} \in \mathbb{Z}^n$ satisfying $A\boldsymbol{x} \leq \boldsymbol{b}$, assuming such an $\boldsymbol{x}$ exists. It is possible to reduce the optimization version of integer programming to the feasibility version by repeatedly adding one additional inequality of the form $\langle \boldsymbol{c}, \boldsymbol{x} \rangle \leq d$ to the original constraints $A\boldsymbol{x} \leq b$ and then querying the feasibility integer programming oracle on the new constraints. Indeed, by binary searching on $d \in \mathbb{Z}$, this reduction shows how to solve the optimization version of IP using an oracle for the feasibility version with only polynomial slowdown.

[3] Showing that a feasible integer program always has such an $\boldsymbol{x}$ whose coefficients are polynomially bounded in the input length takes some additional work but can be done.

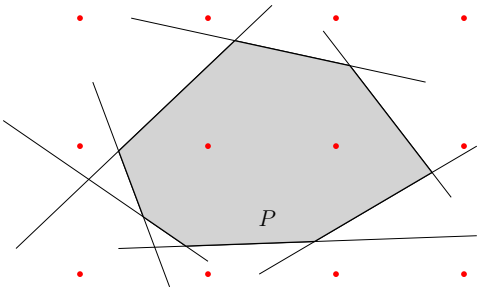

Figure 1: A geometric view of the integer programming problem. The goal is to decide whether the polytope $P = \{\boldsymbol{x} \in \mathbb{R}^n : A\boldsymbol{x} \leq \boldsymbol{b}\}$ (shown in gray) contains a point in $\mathbb{Z}^n$ (shown in red). In this example, $P \cap \mathbb{Z}^n$ contains two points.

Theorem 1.1 ([Kan87])**.** *There exists an algorithm that, given $A \in \mathbb{Q}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{Q}^m$ as input, decides whether there exists $\boldsymbol{x} \in \mathbb{Z}^n$ satisfying $A\boldsymbol{x} \leq \boldsymbol{b}$ in $n^{cn+o(n)} \cdot \text{poly}(S)$ time for some absolute constant $c > 0$, where S is the size of the input.*

Kannan originally showed the above theorem with $c = 9/2$, which was quickly improved to $c = 5/2$ by Frank and Tardos [FT87] who showed how to keep the bit lengths of the numbers used in Kannan's algorithm polynomially bounded (Kannan's original paper showed an upper bound of $n^{2n+o(n)}$ on these bit lengths). This was then further improved to $c = 3/2$ by Helfrich [Hel85]. More recently, Dadush [Dad12] showed how to achieve $c = 1$, which to the best of our knowledge remains the best known as of 2022. It remains a major open question whether there is a $2^{O(n)}$-time algorithm for integer programming.

1.1 A Geometric View and a Generalization to Other Convex Bodies

We may equivalently formulate integer programming as the problem of determining whether the polytope $P = \{\boldsymbol{x} \in \mathbb{R}^n : A\boldsymbol{x} \leq \boldsymbol{b}\}$ contains an integer point, i.e., whether $P \cap \mathbb{Z}^n$ is non-empty. We will focus on this geometric view of the problem, and we will make two additional simplifying assumptions about P in the remainder of the notes: (1) that P is bounded, and (2) that $\text{vol}_n(P) > 0$. We can make these assumptions essentially without loss of generality since we can ensure that they hold using an efficient preprocessing step, which roughly amounts to solving a linear program to ensure that P is bounded and non-empty and to performing dimension reduction if P is lower-dimensional.

In fact, Kannan's algorithm naturally extends to the more general problem of determining whether a convex body K with a "nice representation" contains an integer point. A *convex body* is a closed, bounded, convex set with $\text{vol}_n(K) > 0$.

We may easily represent a polytope either in terms of its facets as the intersection of m half spaces, where a *half space* is the set of points $\boldsymbol{x} \in \mathbb{R}^n$ satisfying a single linear constraint $\langle \boldsymbol{a}, \boldsymbol{x} \rangle \leq b$ for $\boldsymbol{a} \in \mathbb{R}^n$, $b \in \mathbb{R}$. However, it is less clear how to represent an arbitrary convex body K, and what a nice representation of such a body K might be. One natural way of representing such a K is via a membership oracle, which is an algorithm that answers queries of the form, "Is $\boldsymbol{x} \in K$?" for vectors $\boldsymbol{x}$. However, this representation is stronger than what we'll need.

For our purposes all we will need is a *weak separation oracle*, which intuitively answers membership queries about a convex body K approximately. More precisely, a *weak separation oracle* for a convex body K is an efficient algorithm that for a fixed $\varepsilon > 0$ returns YES if a query vector $\boldsymbol{x}$ is in K and returns NO if $\inf_{\boldsymbol{y} \in K} \|\boldsymbol{x} - \boldsymbol{y}\| > \varepsilon$. In the case where the algorithm returns NO, we also require that it certifies the fact by giving a hyperplane separating $\boldsymbol{x}$ and K. We will now present Kannan's algorithm for convex bodies K of this form.

Theorem 1.2 (Kannan's algorithm for convex bodies, [Kan87])**.** *There exists an algorithm that, given a convex body $K \subseteq \mathbb{R}^n$ represented by an efficient weak separation oracle as input, decides whether $K \cap \mathbb{Z}^n$ is*

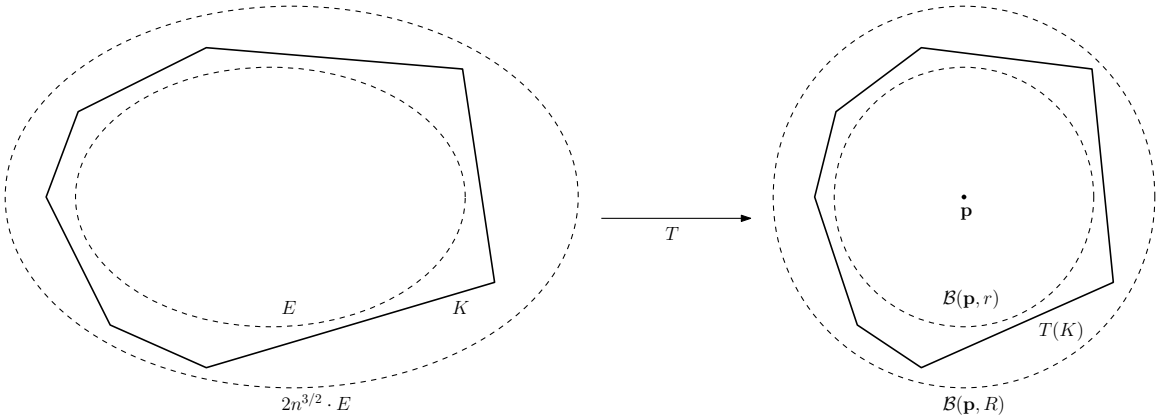

Figure 2: A visual depiction of John's Theorem for a convex body K (left), and the result after applying a linear transformation T to obtain a "ball-like" convex body $T(K)$ satisfying $\mathcal{B}(\boldsymbol{p}, r) \subseteq T(K) \subseteq \mathcal{B}(\boldsymbol{p}, R)$ with $R/r = 2n^{3/2}$ (right).

non-empty in $n^{cn+o(n)} \cdot \mathrm{poly}(S)$ *time for some absolute constant* $c > 0$*, where* S *is the size of the input.*

We note in passing that Theorem 1.2 together with a few straightforward observations also implies $(n^{cn+o(n)} \cdot \mathrm{poly}(S))$-time algorithms for the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP) on lattices, although for SVP this is slightly misleading because Kannan's algorithm itself actually needs to solve SVP in one of its steps.

2 A Simplified Enumeration Algorithm

We now describe an enumeration algorithm for integer programming that is a mix of Lenstra's algorithm and Kannan's algorithm. Let $K \subseteq \mathbb{R}^n$ be a convex body represented by a weak separation oracle.

2.1 Making K ball-like

A theorem of John [Joh48] says that *every* convex body $K \subseteq \mathbb{R}^n$ is "ellipsoid-like" in the sense that it can be approximated by a pair of concentric ellipsoids of similar size. Indeed, John's Theorem says that every such K contains some inner ellipsoid E, and is contained in a congruent outer ellipsoid $n \cdot E$. Together, E and $n \cdot E$ "sandwich" K, and approximate its shape; see the left side of Figure 2. In fact, we will need the inner and outer ellipsoids to be efficiently computable, which is achievable when the scaling factor n of the outer ellipsoid is increased to a slightly larger polynomial; this comes from an argument of Lovász appearing in [Len83, Section 2].

Fact 2.1 (Algorithmic John Ellipsoids, [Joh48], [Len83, Section 2])**.** *For every convex body* $K \subseteq \mathbb{R}^n$ *represented by a weak separation oracle, there exists an efficiently computable ellipsoid* $E \subseteq \mathbb{R}^n$ *such that* $E \subseteq K \subseteq 2n^{3/2} \cdot E$*.*

The first step of the algorithm is to compute E, $2n^{3/2}E$ using Fact 2.1, and then apply a linear transformation T to E, $2n^{3/2} \cdot E$, and K in order to obtain balls $\mathcal{B}(\boldsymbol{p}, r)$, $\mathcal{B}(\boldsymbol{p}, R)$ centered at $\boldsymbol{p}$ of radii r, R, $R/r = 2n^{3/2}$, such that $\mathcal{B}(\boldsymbol{p}, r) \subseteq T(K) \subseteq \mathcal{B}(\boldsymbol{p}, R)$ (see Figure 2). Applying T to K makes it "ball-like" and reduces the problem of deciding whether a general, ellipsoid-like convex body K contains a point in $\mathbb{Z}^n$ to the problem of deciding whether a ball-like convex body $T(K)$ contains a point in some (typically different) lattice $\mathcal{L} := T(\mathbb{Z}^n)$. Indeed, because T is a full-rank linear mapping, $\mathcal{L} = T(\mathbb{Z}^n)$ is a lattice, and $K \cap \mathbb{Z}^n$ is non-empty if and only if $T(K) \cap \mathcal{L}$ is non-empty.

2.2 Finding a Solution or Finding Structure

Define the *Gram-Schmidt decay* of a basis B as

$$\eta(B) := \max_{1 \le j \le i \le n} \frac{\|\tilde{\boldsymbol{b}}_j\|}{\|\tilde{\boldsymbol{b}}_i\|} \tag{1}$$

In particular, for all $i \in [n]$

$$\|\tilde{\boldsymbol{b}}_i\| \le \eta(B) \cdot \|\tilde{\boldsymbol{b}}_n\| \ . \tag{2}$$

Furthermore, we recall that Babai's algorithm (size-reduction) implies that for any basis B and $\boldsymbol{t}$

$$\mathrm{dist}(\boldsymbol{t}, \mathcal{L}) \le \frac{1}{2} \cdot \big(\sum_{i=1}^{n} \|\tilde{\boldsymbol{b}}_i\|^2\big)^{1/2} \ . \tag{3}$$

We now consider two cases, which form a dichotomy closely related to the so-called Flatness Theorem of Khinchin [Khi48]:

1. (Solution.) If $r \ge \frac{1}{2} \cdot \big(\sum_{i=1}^{n} \|\tilde{\boldsymbol{b}}_i\|^2\big)^{1/2}$ then output YES.
2. (Structure.) Otherwise,

$$r < \frac{1}{2} \cdot \big(\sum_{i=1}^{n} \|\tilde{\boldsymbol{b}}_i\|^2\big)^{1/2} \le \frac{1}{2} \cdot \big(\sum_{i=1}^{n} (\eta(B) \cdot \|\tilde{\boldsymbol{b}}_n\|)^2\big)^{1/2} \le \frac{\sqrt{n}}{2} \cdot \eta(B) \cdot \|\tilde{\boldsymbol{b}}_n\| \ ,$$

where we have used Equation (2). It follows that

$$\|\tilde{\boldsymbol{b}}_n\| > \frac{2r}{\sqrt{n} \cdot \eta(B)} \ . \tag{4}$$

Correctness in the first case holds by Equation (3), which implies that $\mathcal{L} \cap \mathcal{B}(\boldsymbol{p}, r)$ is non-empty, and hence that $\mathcal{L} \cap T(K)$ is non-empty. Moreover, we remark that Babai's algorithm actually *finds* a point $\boldsymbol{y} \in \mathcal{L}$ within distance $\frac{1}{2} \cdot \big(\sum_{i=1}^{n} \|\tilde{\boldsymbol{b}}_i\|^2\big)^{1/2}$ of any target point, and so applying it with target $\boldsymbol{p}$, we can in turn efficiently find a lattice point $\boldsymbol{y}$ within distance r of $\boldsymbol{p}$ (i.e., a point $\boldsymbol{y} \in T(K) \cap \mathcal{L}$) when in the case corresponding to Item 1. Intuitively, the second case in Equation (4) says that $\|\tilde{\boldsymbol{b}}_n\|$ is large. In the next section we will show why this structure is useful.

2.3 Recursing on $(n-1)$-dimensional subproblems

The goal of this section is to show how to decide whether $T(K) \cap \mathcal{L}$ is non-empty by recursing on $(n-1)$-dimensional subproblems and why this recursive algorithm is relatively efficient when Equation (4) holds. Let $\mathcal{L}' = \mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n-1})$, and note that

$$\mathcal{L} = \cup_{x_n \in \mathbb{Z}} (\mathcal{L}' + x_n \boldsymbol{b}_n) \ .$$

That is, we can partition $\mathcal{L}$ into $(n-1)$-dimensional lattice shifts $\mathcal{L}' + x_n \boldsymbol{b}_n$, and so we can decide whether $T(K) \cap \mathcal{L}$ is non-empty by deciding whether $T(K) \cap (\mathcal{L}' + x_n \boldsymbol{b}_n)$ is non-empty for some $x_n \in \mathbb{Z}$.

The key idea is that the distance between two hyperplanes $\mathrm{span}(\mathcal{L}') + x_n \boldsymbol{b}_n$ and $\mathrm{span}(\mathcal{L}') + (x_n + 1) \cdot \boldsymbol{b}_n$ is $\|\tilde{\boldsymbol{b}}_n\|$, and so when $\|\tilde{\boldsymbol{b}}_n\|$ is large only a few such hyperplanes (and corresponding lattice shifts $\mathcal{L}' + x_n \boldsymbol{b}_n$) could possibly intersect $T(K)$. See Figure 3.

Let π_1 denote the identity map and for $i = 2, \ldots, n$ let π_i denote projection onto $\mathrm{span}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{i-1})^\perp = \mathrm{span}(\tilde{\boldsymbol{b}}_i, \ldots, \tilde{\boldsymbol{b}}_n)$. Then because projection is a linear operator and can only decrease length, any lattice point $\boldsymbol{y} = \sum_{i=1}^{n} x_i \boldsymbol{b}_i$ for $x_1, \ldots, x_n \in \mathbb{Z}$ must satisfy

$$\|\boldsymbol{y} - \boldsymbol{p}\| \ge \|\pi_n(\boldsymbol{y} - \boldsymbol{p})\| = |x_n - \langle \boldsymbol{p}, \tilde{\boldsymbol{b}}_n \rangle / \langle \tilde{\boldsymbol{b}}_n, \tilde{\boldsymbol{b}}_n \rangle| \cdot \|\tilde{\boldsymbol{b}}_n\| \ , \tag{5}$$

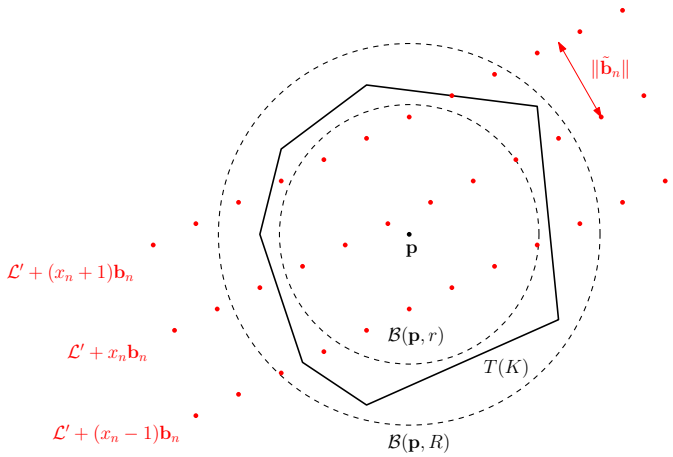

Figure 3: Partition $\mathcal{L}$ into shifts $\mathcal{L}' + x_n\boldsymbol{b}_n$ of the sublattice $\mathcal{L}' := \mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n-1})$, where $x_n \in \mathbb{Z}$. We can decide whether $T(K) \cap \mathcal{L}$ is non-empty by identifying a superset of the shifts $\mathcal{L}' + x_n\boldsymbol{b}_n$ that intersect $T(K)$ and then recursing on the $(n-1)$-dimensional subproblems they induce. Because the distance between the hyperplanes $\text{span}(\mathcal{L}') + x_n\boldsymbol{b}_n$ is $\|\tilde{\boldsymbol{b}}_n\|$, at most $2R/\|\tilde{\boldsymbol{b}}_n\| + 1$ of them can intersect $T(K) \subseteq \mathcal{B}(p, R)$.

where we note that $\tilde{\boldsymbol{b}}_n = \pi_n(\boldsymbol{b}_n)$. Furthermore, any point $\boldsymbol{y} \in T(K)$ must satisfy $\|\boldsymbol{y} - \boldsymbol{p}\| \leq R$, and so by Equation (5) x_n must satisfy

$$|x_n - \langle \boldsymbol{p}, \tilde{\boldsymbol{b}}_n \rangle / \langle \tilde{\boldsymbol{b}}_n, \tilde{\boldsymbol{b}}_n \rangle| \leq R/\|\tilde{\boldsymbol{b}}_n\| \ . \tag{6}$$

Because $x_n \in \mathbb{Z}$, by Equation (6) it can be one of at most $2R/\|\tilde{\boldsymbol{b}}_n\| + 1$ different values. Combining the lower bound on $\|\tilde{\boldsymbol{b}}_n\|$ from Equation (4) with the fact that $R/r = 2n^{3/2}$, we have that

$$\frac{R}{\|\tilde{\boldsymbol{b}}_n\|} \leq \frac{R \cdot \sqrt{n} \cdot \eta(B)}{2r} = n^2 \cdot \eta(B) \ .$$

Therefore, at most $O(n^2 \cdot \eta(B))$ lattice shifts $\mathcal{L}' + x_n\boldsymbol{b}_n$ for $x_n \in \mathbb{Z}$ can intersect $T(K)$. These shifts are also efficiently computable; they are those with $|x_n - \langle \boldsymbol{p}, \tilde{\boldsymbol{b}}_n \rangle / \langle \tilde{\boldsymbol{b}}_n, \tilde{\boldsymbol{b}}_n \rangle| \leq n^2 \cdot \eta(B)$. In order to decide whether $\mathcal{L} \cap T(K)$ is non-empty, it therefore suffices to decide whether

$$|(\mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n-1}) + x_n\boldsymbol{b}_n) \cap T(K)| = |\mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n-1}) \cap (T(K) - x_n\boldsymbol{b}_n)| > 0$$

for $O(n^2 \cdot \eta(B))$ values $x_n \in \mathbb{Z}$.

The overall runtime of the algorithm is then captured by the recurrence

$$T(n) \leq n^2 \cdot \eta(B) \cdot T(n-1) + \text{poly}(n) \ ,$$

which solves to $T(n) = (n \cdot \eta)^{O(n)}$. When B is LLL-reduced, we have that $\eta(B) \leq 2^{O(n)}$, and when B is HKZ-reduced we have by Homework 3, Question 1 that $\eta(B) \leq n^{O(\log n)}$. The overall algorithm therefore requires $2^{O(n^2)}$ time using an LLL-reduced basis B or $n^{O(n \log n)}$ time using an HKZ-reduced basis B as preprocessing.

Remark 2.2. We remark that the algorithm in this section is very similar to Lenstra's algorithm (although its analysis is inspired more by Kannan's analysis of his algorithm). Indeed, the only difference between Lenstra's algorithm and the algorithm described here instantiated with an LLL-reduced basis B is that Lenstra's algorithm permutes the vectors in B to obtain $B' = (\boldsymbol{b}'_1, \ldots, \boldsymbol{b}'_n)$ so that $\|\boldsymbol{b}'_n\| = \max_i \|\boldsymbol{b}'_i\| = \max_i \|\boldsymbol{b}_i\|$ whereas we *do not* do this and just use the LLL-reduced basis B as is. Lenstra's analysis is worse (and achieves an overall runtime of $2^{O(n^3)}$) because he lower-bounds $\|\tilde{\boldsymbol{b}}_n\|$ using the orthogonality defect rather than using $\eta(B)$ directly.

3 Kannan's Algorithm

In this section we show how to modify the algorithm from Section 2 to obtain an $n^{O(n)}$-time algorithm for integer programming. This algorithm and its analysis are due to Kannan [Kan87]. We note again that, up to the constant in the $O(\cdot)$ in the exponent, this remains the fastest general algorithm known for integer programming.

Roughly speaking, Kannan's main insight was that while the *worst-case* Gram-Schmidt decay $\eta(B)$ of an HKZ-reduced basis is at most quasipolynomial (specifically, at most $n^{O(\log n)}$), Minkowski's theorem implies that the (geometric) *average* Gram-Schmidt decay of such bases is at most polynomial (specifically, at most $\sqrt{n}$). This difference improves the $n^{O(n \log n)}$ runtime of the enumeration algorithm in Section 2 to a $n^{O(n)}$.

3.1 HKZ-Reduced Bases

Unlike Section 2, Kannan's algorithm essentially requires the use of an HKZ-reduced basis B, which can be done as part of a preliminary step. We recall that an *HKZ-reduced basis* $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ of a lattice $\mathcal{L}$ is one that satisfies the following two conditions:

1. $\|\boldsymbol{b}_1\| = \lambda_1(\mathcal{L})$, i.e., $\boldsymbol{b}_1$ is a shortest non-zero vector in $\mathcal{L}$, and
2. $(\pi_i(\boldsymbol{b}_i), \ldots, \pi_i(\boldsymbol{b}_n))$ is an HKZ-reduced basis for $i = 2, \ldots, n$.

Define $\mathcal{L}_i := \mathcal{L}(\pi_i(\boldsymbol{b}_i), \ldots, \pi_i(\boldsymbol{b}_n))$. Note that $\pi_i(\boldsymbol{b}_i) = \tilde{\boldsymbol{b}}_i$, and so by definition $\tilde{\boldsymbol{b}}_i \in \mathcal{L}_i$ and $\|\tilde{\boldsymbol{b}}_i\| = \lambda_1(\mathcal{L}_i)$. Additionally, note that

$$\det(\mathcal{L}_i) = \prod_{j=i}^{n} \|\tilde{\boldsymbol{b}}_j\| \ . \tag{7}$$

for $i = 1, \ldots, n$, although this is true whether or not B is HKZ-reduced.

Notice that computing an HKZ-reduced basis is polynomial-time equivalent to solving exact SVP. Indeed, this is trivial in one direction since the first column $\boldsymbol{b}_1$ of an HKZ-reduced basis is a shortest non-zero lattice vector, and in the other direction computing an HKZ-reduced basis can essentially be reduced to making n calls to an exact SVP oracle, one on each of the lattices $\mathcal{L}_1, \ldots, \mathcal{L}_n$.

In his original paper [Kan87], Kannan describes a similar algorithm to the one we present here for integer programming to solve SVP and compute an HKZ-reduced basis. In fact, [Kan87] has three different sections dedicated to solving each of the three problems SVP, CVP, and integer programming in $n^{O(n)}$ time using enumeration techniques. As it turns out, there are asymptotically faster, $2^{O(n)}$-time algorithms for solving SVP (see [AKS01, MV13, ADRS15]). We will not describe any of these algorithms here, but note that computing an HKZ-reduced basis is not the bottleneck in finding a faster integer programming algorithm.

3.2 Setup for Kannan's Algorithm

Kannan's algorithm starts with the same steps described in Section 2.1. Namely, it uses the algorithmic version of John's Theorem (Fact 2.1) to reduce the problem of deciding whether $K \cap \mathbb{Z}^n$ is non-empty for some convex body K to the problem of whether deciding $T(K) \cap \mathcal{L}$ is non-empty for $\mathcal{L} := T(\mathbb{Z}^n)$ and concentric balls $\mathcal{B}(\boldsymbol{p}, r), \mathcal{B}(\boldsymbol{p}, R)$ such that $\mathcal{B}(\boldsymbol{p}, r) \subseteq T(K) \subseteq \mathcal{B}(\boldsymbol{p}, R)$ and $R/r = 2n^{3/2}$.

After that, Kannan's algorithm computes an HKZ-reduced basis $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ of $\mathcal{L}$. For the rest of this section, B refers to this basis.

3.3 Finding a solution or finding structure, take two

Let i be an index such that $\|\tilde{\boldsymbol{b}}_i\| = \max_{j \in [n]} \|\tilde{\boldsymbol{b}}_j\|$. We next use similar analysis to Section 2.2 to find a solution or find structure.

1. (Solution.) If $r \geq \sqrt{n}/2 \cdot \|\tilde{\boldsymbol{b}}_i\|$ then output YES.

2. (Structure.) Otherwise, $r < \sqrt{n}/2 \cdot \|\tilde{\boldsymbol{b}}_i\|$, which implies that $\|\tilde{\boldsymbol{b}}_i\| > 2r/\sqrt{n}$.

In Item 1 we know that

$$\operatorname{dist}(\boldsymbol{p}, \mathcal{L}) \le \frac{1}{2} \cdot (\sum_{j=1}^{n} \|\tilde{\boldsymbol{b}}_j\|^2)^{1/2} \le \frac{\sqrt{n}}{2} \cdot \|\tilde{\boldsymbol{b}}_i\| \le r \ ,$$

and so $\mathcal{L} \cap \mathcal{B}(\boldsymbol{p}, r)$ is non-empty. It follows that the output in that case is correct.

3.4 Recursing on lower-dimensional subproblems

We now assume that $\|\tilde{\boldsymbol{b}}_i\| > 2r/\sqrt{n}$ holds as in Item 2 above. The algorithm proceeds by identifying a set of candidate (partial) lattice vectors $x_i\boldsymbol{b}_i + \cdots + x_n\boldsymbol{b}_n$, specified by the (partial) coefficient vector $(x_i, \ldots, x_n)^T \in \mathbb{Z}^{n-i+1}$, and for each of these recurses on the $(i-1)$-dimensional subproblem of determining whether $\mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{i-1}) \cap (T(K) - \sum_{j=i}^{n} x_j\boldsymbol{b}_j)$ is non-empty. We note that in Section 2.3 we always enumerate a single coefficient x_n at a time and then recurse on $(n-1)$-dimensional subproblems. On the other hand, here we will enumerate a "block" of coefficients $x_i, \ldots, x_n \in \mathbb{Z}$ at a time and recurse on potentially much lower-dimensional subproblems.

Analogously to Equation (5), we have that any lattice point $\boldsymbol{y} = \sum_{i=1}^{n} x_i\boldsymbol{b}_i$ for $x_1, \ldots, x_n \in \mathbb{Z}$ must satisfy

$$\|\boldsymbol{y} - \boldsymbol{p}\| \ge \left| \left(x_j + \sum_{k=j+1}^{n} \langle \boldsymbol{b}_k, \tilde{\boldsymbol{b}}_j \rangle / \langle \tilde{\boldsymbol{b}}_j, \tilde{\boldsymbol{b}}_j \rangle \right) - \langle \boldsymbol{p}, \tilde{\boldsymbol{b}}_j \rangle / \langle \tilde{\boldsymbol{b}}_j, \tilde{\boldsymbol{b}}_j \rangle \right| \cdot \|\tilde{\boldsymbol{b}}_j\|$$

for every $j \in [n]$, where $x_j + \sum_{k=j+1}^{n} \langle \boldsymbol{b}_k, \tilde{\boldsymbol{b}}_j \rangle / \langle \tilde{\boldsymbol{b}}_j, \tilde{\boldsymbol{b}}_j \rangle$ and $\langle \boldsymbol{p}, \tilde{\boldsymbol{b}}_j \rangle / \langle \tilde{\boldsymbol{b}}_j, \tilde{\boldsymbol{b}}_j \rangle$ are the respective projections of $\boldsymbol{y}$ and $\boldsymbol{p}$ onto $\operatorname{span}(\tilde{\boldsymbol{b}}_j)$. Again as before, any point $\boldsymbol{y} \in T(K)$ must satisfy $\|\boldsymbol{y} - \boldsymbol{p}\| \le R$, and so x_j must satisfy

$$\left| \left(x_j + \sum_{k=j+1}^{n} \langle \boldsymbol{b}_k, \tilde{\boldsymbol{b}}_j \rangle / \langle \tilde{\boldsymbol{b}}_j, \tilde{\boldsymbol{b}}_j \rangle \right) - \langle \boldsymbol{p}, \tilde{\boldsymbol{b}}_j \rangle / \langle \tilde{\boldsymbol{b}}_j, \tilde{\boldsymbol{b}}_j \rangle \right| \le R / \|\tilde{\boldsymbol{b}}_j\| \ .$$

Furthermore, we have that for $j = i, \ldots, n$

$$\frac{R}{\|\tilde{\boldsymbol{b}}_j\|} = \frac{2n^{3/2} \cdot r}{\|\tilde{\boldsymbol{b}}_j\|} < \frac{n^2 \cdot \|\tilde{\boldsymbol{b}}_i\|}{\|\tilde{\boldsymbol{b}}_j\|} \ , \tag{8}$$

where the second inequality uses the assumption from Section 3.3, Item 2.

Therefore for any $j \in [n]$ and any fixed choices of $x_{j+1}, \ldots, x_n$, there are at most $2n^2 \cdot \|\tilde{\boldsymbol{b}}_i\| / \|\tilde{\boldsymbol{b}}_j\| + 1$ possible choices for $x_j \in \mathbb{Z}$. It follows that the total number of possible choices for $(x_i, \ldots, x_n)$ is at most

$$\begin{aligned} \prod_{j=i}^{n} \left(2n^2 \cdot \frac{\|\tilde{\boldsymbol{b}}_i\|}{\|\tilde{\boldsymbol{b}}_j\|} + 1 \right) &\le \prod_{j=i}^{n} \left(3n^2 \cdot \frac{\|\tilde{\boldsymbol{b}}_i\|}{\|\tilde{\boldsymbol{b}}_j\|} \right) \\ &\le (3n)^{2(n-i+1)} \cdot \frac{\lambda_1(\mathcal{L}_i)^{n-i+1}}{\det(\mathcal{L}_i)} \\ &\le (3n)^{2(n-i+1)} \cdot (n-i+1)^{(n-i+1)/2} \\ &\le (3n)^{5/2 \cdot (n-i+1)} \ . \end{aligned} \tag{9}$$

Here the second inequality uses the fact that B is HKZ-reduced and hence that $\|\tilde{\boldsymbol{b}}_i\| = \lambda_1(\mathcal{L}_i)$ together with Equation (7). The third inequality uses Minkowski's theorem, which asserts that $\lambda_1(\mathcal{L}') \le \sqrt{n'} \cdot \det(\mathcal{L}')^{1/n'}$ for rank-n' lattices $\mathcal{L}'$, with both sides of the inequality raised to the $(n-i+1)$th power, which is the rank of $\mathcal{L}_i$.

We then get the following recurrence for the number of coefficient vectors $\boldsymbol{x} \in \mathbb{Z}^n$ considered by Kannan's algorithm:

$$T(n) \leq \max_{i \in [n]} (3n)^{5/2 \cdot (n-i+1)} \cdot T(i-1) + \text{poly}(n) \ .$$

This solves to $T(n) \leq n^{5/2 \cdot n + o(n))}$, which is also an upper bound on the number of arithmetic operations performed by the algorithm. As discussed in the introduction, getting a proper runtime upper bound of $n^{O(n)}$ additionally requires bounding the bit lengths of the numbers used in the algorithm, but we omit discussing that here.

References

[ADRS15] Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. Solving the shortest vector problem in 2^n time using discrete gaussian sampling: Extended abstract. In *STOC*, 2015.

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Symposium on Theory of Computing (STOC)*, 2001.

[Dad12] D. Dadush. *Integer Programming, Lattice Algorithms, and Deterministic Volume Estimation.* PhD thesis, Georgia Institute of Technology, 2012.

[FT87] András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987.

[Hel85] Bettina Helfrich. Algorithms to construct Minkowski reduced and Hermite reduced lattice bases. *Theor. Comput. Sci.*, 41:125–139, 1985.

[Joh48] Fritz John. Extremum problems with inequalities as subsidiary conditions. In *Studies and Essays Presented to R. Courant on his 60th Birthday, January 8, 1948*, pages 187–204. Interscience Publishers, Inc., New York, N. Y., 1948.

[Kan87] R. Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, 1987.

[Khi48] A. Ya. Khincin. A quantitative formulation of the approximation theory of Kronecker. *Izvestiya Akad. Nauk SSSR. Ser. Mat.*, 12:113–122, 1948.

[Len83] H. W. Lenstra. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8(4):538–548, 1983.

[MV13] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. *SIAM J. Comput.*, 42(3):1364–1391, 2013. Preliminary version in STOC 2010.