Faster SVP in Polynomial Space

Yansong Feng[1], Yiming Gao[2], and Jiaqi Liu[1]

[1] Academy of Mathematics and Systems Science, Chinese Academy of Sciences
{fengyansong,ljqi}@amss.ac.cn
[2] School of Cyber Science and Technology, University of Science and Technology of China
qw1234567@mail.ustc.edu.cn

Abstract. Kannan's algorithm, as analyzed by Hanrot and Stehlé in 2007, solves the exact Euclidean shortest vector problem in polynomial space and $n^{\frac{n}{2e}+o(n)}$ time. In the classical setting with polynomial space, we obtain the first improvement on this bound via a randomized algorithm that runs in $n^{\frac{n}{4e}+o(n)}$ time.
The main idea is to represent a fixed shortest vector in many ways as a difference of samples, thereby enabling the low-space collision search of Lyu and Zhu (SODA 2023) to replace exhaustive enumeration in the original analysis.

Keywords: Lattice · Shortest vector problem · Enumeration

1 Introduction

A full-rank lattice $\mathcal{L} \subseteq \mathbb{R}^n$ is the set of integer linear combinations of a basis $\boldsymbol{B} = [\boldsymbol{b}_1 \ \cdots \ \boldsymbol{b}_n]$. Given such a basis, the Euclidean shortest vector problem (SVP) asks for a nonzero vector of minimum norm in $\mathcal{L}$. Algorithms for SVP play an important role in lattice cryptanalysis [27], and their cost informs concrete security estimates for lattice-based cryptography [28]. We study exact SVP in the classical polynomial-space setting.

Kannan's algorithm is the standard deterministic algorithm in this setting [21]. It recursively computes Hermite–Korkine–Zolotarev (HKZ) reductions of projected sublattices to obtain a quasi-HKZ basis, then enumerates all lattice vectors of norm at most $\|\boldsymbol{b}_1\|_2$. The enumeration tree can be traversed in polynomial space [11,30]. Kannan bounded the running time by $n^{n+o(n)}$, and Helfrich improved the analysis to $n^{n/2+o(n)}$ [21,17]. Hanrot and Stehlé obtained the bound $n^{\frac{n}{2e}+o(n)}$ for the same algorithm [15]. Their analysis exploits the Gram–Schmidt shape of a quasi-HKZ basis to bound the number of integer points in the enumeration ellipsoids directly, instead of using an enclosing parallelepiped. The quasi-HKZ preprocessing itself is already part of Kannan's recursion. Hanrot and Stehlé later constructed HKZ bases on which Kannan's enumeration procedure requires $n^{\frac{n}{2e}+o(n)}$ bit operations [16]. This matches the leading exponent of the upper bound for that procedure, but does not rule out a different search that finds a shortest vector without traversing the entire tree.

A different line of work obtains single-exponential time by allowing exponential memory. The celebrated AKS sieve proposed by Ajtai, Kumar and Sivakumar gave a randomized $2^{O(n)}$-time and $2^{O(n)}$-space algorithm [2]. Later, Micciancio and Voulgaris gave a deterministic Voronoi-cell algorithm using $\widetilde{O}(4^n)$ time and $\widetilde{O}(2^n)$ space [25]. Aggarwal, Dadush, Regev, and Stephens-Davidowitz used discrete Gaussian sampling to reduce the randomized bound to $2^{n+o(n)}$ time and space [1]. Recently, several improvements to randomized classical exact-SVP algorithms have yielded progressively smaller time exponents, culminating in $2^{n/2+o(n)}$ time, all using $2^{n/2+o(n)}$ space [14,19,13,18]. Under heuristic assumptions, Becker, Ducas, Gama and Laarhoven proposed an algorithm with running time $2^{0.292n+o(n)}$ and also using exponential space [7]. These results therefore remain in the exponential-space setting.

Other work seeks to speed up lattice enumeration itself. Chen et al. give a classical time–space tradeoff [10]. At its balanced point, both time and space are $n^{\frac{n}{4e}+o(n)}$. Aono, Nguyen, and Shen use quantum enumeration to reach the same time bound with polynomial space [4]. This raises the question:

Can SVP be solved classically in $n^{\frac{n}{4e}+o(n)}$ time using only polynomial space?

The answer is yes. We give a randomized classical algorithm that, given any full-rank integer basis $\boldsymbol{B}$, returns a shortest nonzero lattice vector with probability at least 2/3. It uses polynomial space and runs in $n^{\frac{n}{4e}+o(n)}$ time up to a polynomial factor in the input bit length. The precise statement appears in Theorem 5.

The key observation concerns Kannan's recursion: it ultimately needs only one shortest vector, even though enumeration searches the entire ball. For a quasi-HKZ basis, we replace this search with the low-space collision procedure of Lyu and Zhu [24]. An exact sampler provides access to two large random arrays of lattice points without storing them explicitly, and the quasi-HKZ shape bound controls their length. With constant probability, the two arrays contain many pairs whose difference is a fixed shortest vector. For sufficiently many of these pairs, a random grid assigns the two lattice points in each pair to the same cell. The corresponding array entries are then labeled identically and thus form collisions for the low-space search. Bounds on repeated samples and on the number of points in a grid cell control the total number of collisions. Repeating the grid construction and collision search $2^{O(n)}$ times gives constant success probability. The resulting square-root saving over exhaustive enumeration reduces the exponent from $1/2e$ to $1/4e$. Finally, embedding this subroutine in Kannan's recursion extends the method to arbitrary full-rank integer bases.

Our use of many representations is conceptually reminiscent of the representation technique introduced by Howgrave-Graham and Joux for subset sum [20] and further developed by Becker, Coron, and Joux [6]. In their subset sum setting, a fixed solution is given many combinatorial representations so that one representation survives suitable modular filters. In our SVP setting, a fixed shortest lattice vector has many representations as a difference of two array entries, and a random grid turns some of these representations into label collisions for the low-space search.

Organization. Section 2 develops the two tools used throughout: shape bounds for quasi-HKZ bases and a low-space collision lemma. Section 3 uses rounded Gram–Schmidt sampling and a random grid to construct an implicit label array for the collision search. Section 4 applies this construction first to a quasi-HKZ basis and then, through Kannan's recursion, to an arbitrary full-rank lattice basis. Section 5 summarizes the result and discusses directions for future work.

2 Preliminaries

We write $[m] := \{1, \dots, m\}$ and use $\langle \cdot, \cdot \rangle$ and $\|\cdot\|_2$ for the Euclidean inner product and norm, respectively. For a rational matrix $\boldsymbol{B}$, let $|\boldsymbol{B}|$ denote its binary encoding length. The symbols log and $\log_2$ denote the natural and binary logarithms, respectively. Throughout the recursion, n is fixed and $d \le n$ is the current rank. Projected bases lie in $\mathbb{Q}^n$ and are scaled uniformly to integer bases when needed.

2.1 Lattices

Let $\boldsymbol{B} = [\boldsymbol{b}_1 \ \cdots \ \boldsymbol{b}_d] \in \mathbb{R}^{n \times d}$ have linearly independent columns. The rank-d lattice generated by the basis $\boldsymbol{B}$ is defined as

$$\mathcal{L}(\boldsymbol{B}) := \{\boldsymbol{B}\boldsymbol{x} : \boldsymbol{x} \in \mathbb{Z}^d\}.$$

We write $\mathcal{L} = \mathcal{L}(\boldsymbol{B})$ when the basis is clear. The length of its nonzero shortest vector is denoted by

$$\lambda_1(\mathcal{L}) := \min_{\boldsymbol{v} \in \mathcal{L} \setminus \{0\}} \|\boldsymbol{v}\|_2.$$

Definition 1. *Given a basis $\boldsymbol{B}$, the shortest vector problem (SVP) asks for a vector $\boldsymbol{v} \in \mathcal{L}(\boldsymbol{B}) \setminus \{0\}$ satisfying $\|\boldsymbol{v}\|_2 = \lambda_1(\mathcal{L}(\boldsymbol{B}))$.*

Given $\boldsymbol{B}$, let $\boldsymbol{b}_1^*, \dots, \boldsymbol{b}_d^*$ be its Gram–Schmidt orthogonalization. Set

$$r_i := \|\boldsymbol{b}_i^*\|_2 \quad \text{for } i \in [d],$$

and

$$\mu_{j,i} := \frac{\langle \boldsymbol{b}_j, \boldsymbol{b}_i^* \rangle}{\|\boldsymbol{b}_i^*\|_2^2} \quad \text{for } i < j.$$

Then, for $j \in [d]$,

$$\boldsymbol{b}_j = \boldsymbol{b}_j^* + \sum_{i<j} \mu_{j,i} \boldsymbol{b}_i^*.$$

For $i \in [d]$, let π_i denote the orthogonal projection onto $\operatorname{span}(\boldsymbol{b}_1, \dots, \boldsymbol{b}_{i-1})^\perp$, with π_1 the identity. Then $\boldsymbol{b}_i^* = \pi_i(\boldsymbol{b}_i)$, and

$$\pi_i(\mathcal{L}) = \mathcal{L}\big(\pi_i(\boldsymbol{b}_i), \dots, \pi_i(\boldsymbol{b}_d)\big)$$

is the ith *projected lattice* of $\boldsymbol{B}$.

For $\boldsymbol{x} \in \mathbb{Z}^d$ and $i \in [d]$, set

$$c_i(\boldsymbol{x}) := \sum_{j>i} \mu_{j,i} x_j, \qquad y_i(\boldsymbol{x}) := x_i + c_i(\boldsymbol{x}).$$

Then $\boldsymbol{B}\boldsymbol{x} = \sum_i y_i(\boldsymbol{x})\boldsymbol{b}_i^*$, so $y_i(\boldsymbol{x})$ is the coefficient of $\boldsymbol{b}_i^*$ in $\boldsymbol{B}\boldsymbol{x}$. Orthogonality gives

$$\|\boldsymbol{B}\boldsymbol{x}\|_2^2 = \sum_i r_i^2 y_i(\boldsymbol{x})^2.$$

The basis $\boldsymbol{B}$ is *size reduced* if $|\mu_{j,i}| \le 1/2$ for all $i < j$.

We next introduce the definitions of HKZ and quasi-HKZ bases [15].

Definition 2. *A size-reduced basis is* HKZ reduced *if $\boldsymbol{b}_i^*$ is a shortest vector of $\pi_i(\mathcal{L})$ for every $i \in [d]$. For $d \ge 2$, a size-reduced basis is* quasi-HKZ *if $r_2 \ge \|\boldsymbol{b}_1\|_2/2$ and the projected basis $[\pi_2(\boldsymbol{b}_2) \ \cdots \ \pi_2(\boldsymbol{b}_d)]$ is HKZ reduced.*

The following Hanrot–Stehlé product bound will be used below [15, Theorem 3].

Lemma 1. *Let $\rho_1, \dots, \rho_m$ be the Gram–Schmidt lengths of an HKZ-reduced basis of rank m. For every nonempty $I \subseteq [m]$, writing $k := |I|$,*

$$\frac{\rho_1^k}{\prod_{i\in I} \rho_i} \le m^{\frac{k}{2}(1+\log(m/k))} \le m^{\frac{m}{2e}+k/2}.$$

For the shape bounds below, assume $d \ge 2$. Write $\lambda_1 := \lambda_1(\mathcal{L})$, and set

$$s_i := \max\left\{1, \frac{\|\boldsymbol{b}_1\|_2}{\sqrt{d}\, r_i}\right\} \quad (i \in [d]), \qquad P := \prod_{i=1}^{d} s_i.$$

Now we introduce the shape bounds for quasi-HKZ bases in the following Lemma 2, which bounds these scale parameters and compares $\|\boldsymbol{b}_1\|_2$ with λ_1.

Lemma 2. *For a quasi-HKZ basis of rank $d \ge 2$,*

$$P \le 2^{O(d)} d^{\frac{d}{2e}} = d^{\frac{d}{2e}+o(d)}, \qquad \max_{i\in[d]} s_i = d^{o(d)}, \qquad \|\boldsymbol{b}_1\|_2 \le 2\lambda_1.$$

Proof. Let $S := \{i : s_i > 1\}$. Since $s_1 = 1$, we have $S \subseteq \{2, \dots, d\}$. If $S = \varnothing$, then $P = 1$. Otherwise, set $k := |S|$. Applying Lemma 1 with $m = d-1$ and $\rho_j = r_{j+1}$ to the HKZ-reduced basis $[\pi_2(\boldsymbol{b}_2) \ \cdots \ \pi_2(\boldsymbol{b}_d)]$ and the index set $\{i-1 : i \in S\}$ gives

$$\frac{r_2^k}{\prod_{i\in S} r_i} \le (d-1)^{(d-1)/(2e)+k/2}.$$

Using $\|\boldsymbol{b}_1\|_2 \le 2r_2$, we obtain

$$\begin{aligned} P &= \frac{\|\boldsymbol{b}_1\|_2^k}{d^{k/2}\prod_{i\in S} r_i} \\ &\le \frac{2^k}{d^{k/2}}(d-1)^{(d-1)/(2e)+k/2} \le 2^d d^{\frac{d}{2e}}. \end{aligned}$$

For each $i \in \{2, \dots, d\}$, the same lemma applied to the projected basis with $I = \{i-1\}$ gives

$$\frac{r_2}{r_i} \le (d-1)^{\frac{1}{2}(1+\log(d-1))}.$$

Together with $\|\boldsymbol{b}_1\|_2 \le 2r_2$ and $s_1 = 1$, this yields

$$\max_{i \in [d]} s_i \le 2d^{(\log d)/2} = d^{O(\log d)} = d^{o(d)}.$$

Finally, let $0 \neq \boldsymbol{v} \in \mathcal{L}$. If $\boldsymbol{v}$ is parallel to $\boldsymbol{b}_1$, then $\boldsymbol{v} = t\boldsymbol{b}_1$ for some $t \in \mathbb{Z} \setminus \{0\}$, so $\|\boldsymbol{v}\|_2 \ge \|\boldsymbol{b}_1\|_2$. Otherwise, $\pi_2(\boldsymbol{v}) \in \pi_2(\mathcal{L}) \setminus \{0\}$. Since $\boldsymbol{b}_2^*$ is shortest in $\pi_2(\mathcal{L})$,

$$\|\boldsymbol{v}\|_2 \ge \|\pi_2(\boldsymbol{v})\|_2 \ge r_2 \ge \|\boldsymbol{b}_1\|_2/2.$$

Thus $\lambda_1 \ge \|\boldsymbol{b}_1\|_2/2$, which proves $\|\boldsymbol{b}_1\|_2 \le 2\lambda_1$. □

2.2 Low-space collision search

Our algorithm uses a low-space collision procedure on an implicitly represented array. This idea goes back to Floyd's cycle-finding algorithm [22] and its use in the Pollard–ρ method [29] to find a collision among successive iterates. Beame, Clifford, and Machmouchi [5] adapted this approach to array collisions in the random-oracle model. Chen, Jin, Williams, and Wu [9] instantiated this procedure with a short-seed pseudorandom hash family, thereby removing its random-oracle requirement in the polylogarithmic-space regime. Lyu and Zhu [24] subsequently simplified and strengthened the analysis.

To define a collision walk, fix an array $\boldsymbol{u} \in [N]^L$ and a hash $h : [N] \to [L] \cup \{\perp\}$, and set

$$f_h(i) := h(u_i) \qquad (i \in [L]).$$

Starting from $x_0 := s \in [L]$, iterate

$$x_0 \longrightarrow x_1 = f_h(x_0) \longrightarrow x_2 = f_h(x_1) \longrightarrow \cdots.$$

The walk stops at $\perp$ or the first repeated index. The set of visited indices is the *orbit* of the walk, denoted by $\mathcal{O}(h, s)$. If $u_i = u_j$ for distinct $i, j \in \mathcal{O}(h, s)$, then $f_h(i) = f_h(j)$. Because the walk stops at the first repeated index, $\{i, j\}$ is the only pair of visited indices with the same successor. A cycle finder [5, Theorem 2.1] recovers this pair using $O(|\mathcal{O}(h, s)|)$ array accesses and $O(\log L)$ additional bits, without storing the orbit. It outputs the recovered pair only if its two array values agree.

The *second frequency moment* of $\boldsymbol{u}$ is

$$F_2(\boldsymbol{u}) := \sum_{a \in [N]} |\{i \in [L] : u_i = a\}|^2.$$

We use the following orbit bounds of Lyu and Zhu [24, Lemmas 3 and 4].

Lemma 3. *Let $\boldsymbol{u} \in [N]^L$, where $N = L^{O(1)}$. There is a distribution on hashes $h : [N] \to [L] \cup \{\perp\}$ with $O(\log^3 L)$-bit seeds and $\mathrm{poly}(\log L)$-time evaluation. Let h be sampled from this distribution, and let s be chosen independently and uniformly from $[L]$. For distinct $i, j \in [L]$,*

$$\Pr_{h,s}[i \in \mathcal{O}(h,s)] = O(1/\sqrt{L}),$$
$$\Pr_{h,s}[\{i,j\} \subseteq \mathcal{O}(h,s)] = \Omega\big(1/F_2(\boldsymbol{u})\big).$$

A *duplicate pair* is an index pair $\{i, j\}$ with $i \neq j$ and $u_i = u_j$. The orbit bounds control the expected length of a walk and its chance of finding a fixed duplicate pair. Repeating the walk gives a search bound for any set $\mathcal{G}$ of such pairs.

Lemma 4. *Let $\boldsymbol{u}$ be a random-access array of length L over a universe of size $L^{O(1)}$. Assume that*
(i) $F_2(\boldsymbol{u}) \le p \le L^2$;
(ii) $\mathcal{G}$ is a set of duplicate pairs with $|\mathcal{G}| \ge r_0 \ge 1$.
There is a randomized algorithm that, given random access to $\boldsymbol{u}$ and the bounds p, r_0, outputs a sequence of duplicate pairs. The sequence contains a pair in $\mathcal{G}$ with constant probability. On every run, the algorithm makes

$$\widetilde{O}\left(\sqrt{L}\, p/r_0\right)$$

array accesses and uses $O(\log^3 L)$ working bits in addition to any space needed to evaluate an entry u_i.

Proof. Sample h and s as in Lemma 3 and run the cycle finder described above. Summing the first bound of Lemma 3 over $i \in [L]$ gives

$$\mathbb{E}_{h,s}[|\mathcal{O}(h,s)|] = O(\sqrt{L}).$$

Thus each trial uses $O(\sqrt{L})$ expected array accesses. By the second bound, each fixed duplicate pair is reached and thus output with probability $\Omega(1/F_2(\boldsymbol{u}))$. A single orbit contains at most one duplicate pair, so the output events for pairs in $\mathcal{G}$ are disjoint. Thus one trial outputs a pair in $\mathcal{G}$ with probability

$$\Omega\big(|\mathcal{G}|/F_2(\boldsymbol{u})\big) \ge \Omega(r_0/p).$$

Run $\Theta(p/r_0)$ independent trials and output every duplicate pair found. The resulting sequence contains a pair in $\mathcal{G}$ with constant probability, and its expected total array-access count is $O(\sqrt{L}\, p/r_0)$. Let A be the total array-access count when all these trials run to completion. Let $\eta > 0$ be a success lower bound. For $T := C\sqrt{L}\, p/r_0$ with sufficiently large C, Markov's inequality gives

$$\Pr[A > T] \le \frac{\mathbb{E}[A]}{T} \le \eta/2.$$

Thus, truncation at T accesses yields success probability at least $\eta/2$ and a worst-case access bound T. The trial counter uses $O(\log L)$ bits since $p/r_0 \le L^2$, and the hash seed and orbit state use $O(\log^3 L)$ bits. □

The walk may also output pairs outside $\mathcal{G}$. We may check each output pair with a polynomial-space predicate and retain the best accepted pair, while continuing the prescribed trials. The collision search needs only $\boldsymbol{u}, p, r_0$. The set $\mathcal{G}$ is used only in the success analysis.

3 Sampling and grid collisions

To apply the collision search of Lemma 4 to a quasi-HKZ basis, we construct two implicit arrays of lattice samples and assign a short label to each sample. The collision search is run on their concatenated label array: its tth entry is the label of the tth sample and is computed only when queried. We need many equal-label pairs whose underlying samples differ by a fixed shortest vector. At the same time, the label array's second frequency moment, which counts ordered pairs of positions with equal labels, must remain small. This section gives such a construction.

Fix a quasi-HKZ integer basis $\boldsymbol{B}$ of rank $d \geq 2$, write $\mathcal{L} := \mathcal{L}(\boldsymbol{B})$ and $\lambda_1 := \lambda_1(\mathcal{L})$, and let s_i, P be the shape parameters from Section 2. Our starting point is a finitely supported distribution ν on $\mathcal{L}$ with two properties. First, every lattice point should have small probability. Second, ν should have enough overlap with its translate by every lattice vector $\boldsymbol{Bz}$ satisfying $\|\boldsymbol{Bz}\|_2 \leq \|\boldsymbol{b}_1\|_2$. This range contains every shortest vector because $\lambda_1 \leq \|\boldsymbol{b}_1\|_2$. For independent samples $\boldsymbol{V}$ and $\boldsymbol{W}$ drawn from ν, the overlap is

$$\sum_{\boldsymbol{u}\in\mathcal{L}} \nu(\boldsymbol{u})\nu(\boldsymbol{u}-\boldsymbol{Bz}) = \Pr[\boldsymbol{V}-\boldsymbol{W}=\boldsymbol{Bz}].$$

The overlap bound gives many pairs, with one sample from each of two arrays, whose difference is a shortest vector. The point-mass bound limits repeated sample values.

The construction proceeds in four steps. First, Section 3.1 introduces rounded Gram–Schmidt coordinates and bounds their change under a short lattice translation in a weighted norm. Second, Section 3.2 defines a finite product distribution on these coordinates that can be sampled exactly and has enough overlap with every short translate. Third, Section 3.3 uses short random seeds to define two implicit sample arrays. The two distribution bounds give many pairs with the desired difference while controlling repeated sample values. Finally, Section 3.4 uses a random grid to assign both points of many target pairs the same cell label. A packing argument converts the bound on repeated sample values into a bound on the label array's second frequency moment, and a final hash compresses the labels into the range required by Lemma 4.

3.1 Rounded Gram–Schmidt coordinates

We introduce rounded Gram–Schmidt coordinates and prove that they define a bijection of $\mathbb{Z}^d$. Unlike the original coefficient coordinates, whose change need not be controlled by $\|\boldsymbol{Bz}\|_2$, these coordinates give a bounded weighted coordinate

change for every short lattice translation. This bound underlies the overlap analysis below.

Define $\lfloor t \rceil := \lfloor t + 1/2 \rfloor$. Then $\lfloor t + k \rceil = \lfloor t \rceil + k$ for $t \in \mathbb{R}$ and $k \in \mathbb{Z}$. For $\boldsymbol{x} \in \mathbb{Z}^d$ and $i \in [d]$, define

$$\kappa_i(\boldsymbol{x}) := x_i + \lfloor c_i(\boldsymbol{x}) \rceil,$$
$$\kappa(\boldsymbol{x}) := (\kappa_1(\boldsymbol{x}), \ldots, \kappa_d(\boldsymbol{x})).$$

Thus $\kappa_i(\boldsymbol{x}) = \lfloor y_i(\boldsymbol{x}) \rceil$. We call $\kappa(\boldsymbol{x})$ the *rounded Gram–Schmidt coordinates* of $\boldsymbol{B}\boldsymbol{x}$.

We round each scale s_i up to a power of two so that the coordinate weights can be computed exactly. Let a_i be the least power of two satisfying

$$a_i \geq 1, \qquad a_i^2 d r_i^2 \geq \|\boldsymbol{b}_1\|_2^2.$$

Set

$$\alpha_i := a_i^{-2}, \qquad \Pi := \prod_i a_i, \qquad Q := \max_i a_i.$$

These quantities are exactly computable from the rational squared Gram–Schmidt lengths, and they satisfy

$$s_i \leq a_i < 2s_i, \qquad P \leq \Pi < 2^d P, \qquad Q < 2 \max_{i \in [d]} s_i = d^{o(d)}, \tag{1}$$

and

$$\alpha_i \leq \min\left\{1, \frac{d r_i^2}{\|\boldsymbol{b}_1\|_2^2}\right\}. \tag{2}$$

The next lemma proves that κ is bijective and bounds the coordinate change induced by a short lattice translation.

Lemma 5. *The map $\kappa : \mathbb{Z}^d \to \mathbb{Z}^d$ is a bijection. For fixed $\boldsymbol{z} \in \mathbb{Z}^d$, define $T_{\boldsymbol{z}} : \mathbb{Z}^d \to \mathbb{Z}^d$ and its coordinate changes by*

$$T_{\boldsymbol{z}}(\boldsymbol{k}) := \kappa(\kappa^{-1}(\boldsymbol{k}) + \boldsymbol{z}),$$
$$\delta_i(\boldsymbol{k}) := T_{\boldsymbol{z}}(\boldsymbol{k})_i - k_i \qquad (\boldsymbol{k} \in \mathbb{Z}^d,\ i \in [d]).$$

For every $\boldsymbol{k} \in \mathbb{Z}^d$ and $i \in [d]$,

$$|\delta_i(\boldsymbol{k}) - y_i(\boldsymbol{z})| \leq 1.$$

If $\|\boldsymbol{B}\boldsymbol{z}\|_2 \leq \|\boldsymbol{b}_1\|_2$, then

$$\sum_i \alpha_i \delta_i(\boldsymbol{k})^2 \leq 4d.$$

Proof. Fix $\boldsymbol{k} \in \mathbb{Z}^d$. Since $c_i(\boldsymbol{x})$ depends only on $x_{i+1}, \ldots, x_d$, define $\boldsymbol{x}$ by backward substitution:

$$x_i := k_i - \left\lfloor \sum_{j=i+1}^{d} \mu_{j,i} x_j \right\rceil, \qquad i = d, d-1, \ldots, 1.$$

The resulting $\boldsymbol{x} \in \mathbb{Z}^d$ satisfies $\kappa(\boldsymbol{x}) = \boldsymbol{k}$, so κ is surjective. Every preimage of $\boldsymbol{k}$ must satisfy the same recursion, so it is unique and κ is also injective.

Let $\boldsymbol{x}$ denote this preimage and set $\theta(t) := \lfloor t \rceil - t$, so $|\theta(t)| \le 1/2$. Using $c_i(\boldsymbol{x}+\boldsymbol{z}) = c_i(\boldsymbol{x}) + c_i(\boldsymbol{z})$ and $y_i(\boldsymbol{z}) = z_i + c_i(\boldsymbol{z})$, we obtain

$$\begin{aligned}\delta_i(\boldsymbol{k}) - y_i(\boldsymbol{z}) &= \kappa_i(\boldsymbol{x}+\boldsymbol{z}) - \kappa_i(\boldsymbol{x}) - y_i(\boldsymbol{z}) \\ &= \lfloor c_i(\boldsymbol{x}) + c_i(\boldsymbol{z}) \rceil - \lfloor c_i(\boldsymbol{x}) \rceil - c_i(\boldsymbol{z}) \\ &= \theta(c_i(\boldsymbol{x}) + c_i(\boldsymbol{z})) - \theta(c_i(\boldsymbol{x})).\end{aligned}$$

Thus $|\delta_i(\boldsymbol{k}) - y_i(\boldsymbol{z})| \le 1$.

Now suppose that $\|\boldsymbol{B}\boldsymbol{z}\|_2 \le \|\boldsymbol{b}_1\|_2$. By Equation (2),

$$\begin{aligned}\sum_i \alpha_i y_i(\boldsymbol{z})^2 &\le \frac{d}{\|\boldsymbol{b}_1\|_2^2} \sum_i r_i^2 y_i(\boldsymbol{z})^2 \\ &= \frac{d}{\|\boldsymbol{b}_1\|_2^2} \|\boldsymbol{B}\boldsymbol{z}\|_2^2 \le d.\end{aligned}$$

Also, $\sum_i \alpha_i \le d$. Combining these estimates with $|\delta_i(\boldsymbol{k}) - y_i(\boldsymbol{z})| \le 1$ gives

$$\begin{aligned}\sum_i \alpha_i \delta_i(\boldsymbol{k})^2 &\le 2\sum_i \alpha_i y_i(\boldsymbol{z})^2 + 2\sum_i \alpha_i \big(\delta_i(\boldsymbol{k}) - y_i(\boldsymbol{z})\big)^2 \\ &\le 2d + 2\sum_i \alpha_i \le 4d.\end{aligned}$$

□

3.2 Constructing the sampling distribution

We now construct ν as a product distribution in the rounded Gram–Schmidt coordinates, using Gaussian-like weights in each coordinate. By Lemma 5, translation by a short lattice vector changes these coordinates by only a bounded amount in the weighted norm, so the product distribution keeps enough overlap with every such translate. We truncate each coordinate distribution to obtain finite support. We then approximate each one by a distribution that can be sampled exactly from a fixed number of random bits. The resulting distribution ν has a small maximum point mass and a uniform lower bound on its overlap with every short translate. Theorem 1 gives the precise bounds, which will be used to analyze the sample arrays in Section 3.3.

For coordinate i, truncate the support and round the weight $2^{-\alpha_i k^2}$ down to an integer power of two. Set

$$I_i := \{k \in \mathbb{Z} : \alpha_i k^2 \le 16d\}, \qquad e_i(k) := \lceil \alpha_i k^2 \rceil, \qquad w_i(k) := 2^{-e_i(k)}.$$

The set I_i is the finite support for coordinate i. Normalize the weights by setting

$$Z_i := \sum_{k \in I_i} w_i(k), \qquad p_i(k) := \frac{w_i(k)}{Z_i} \quad (k \in I_i).$$

We first bound Z_i, $|I_i|$, and the smallest point mass of p_i.

Lemma 6. *For each $i \in [d]$,*

$$\frac{a_i}{4} \le Z_i \le 4a_i, \qquad |I_i| \le 9\sqrt{d}\, a_i, \qquad \min_{k\in I_i} p_i(k) \ge \frac{2^{-16d}}{4a_i}.$$

Proof. Since $\alpha_i = a_i^{-2}$, a direct count gives

$$|I_i| = 2\lfloor 4\sqrt{d}\, a_i \rfloor + 1 \le 9\sqrt{d}\, a_i.$$

Moreover, $e_i(k) \le 16d$ and $w_i(k) \ge 2^{-16d}$ for $k \in I_i$. Since $w_i(k) \le 2^{-k^2/a_i^2}$, comparison with the Gaussian integral gives

$$Z_i \le \sum_{k\in\mathbb{Z}} 2^{-k^2/a_i^2} \le 1 + a_i\sqrt{\pi/\log 2} < 4a_i.$$

For the lower bound, the fact that a_i is a power of two gives

$$\big|\{k \in \mathbb{Z} : |k| \le a_i/4\}\big| = 2\lfloor a_i/4 \rfloor + 1 \ge a_i/2.$$

For each such k, we have $\alpha_i k^2 \le 1/16$, so $k \in I_i$, $e_i(k) \le 1$, and $w_i(k) \ge 1/2$. Hence

$$Z_i \ge \sum_{|k|\le a_i/4} w_i(k) \ge \frac{a_i}{4}.$$

The lower bound on $p_i(k)$ follows from $w_i(k) \ge 2^{-16d}$ and $Z_i \le 4a_i$. □

We next round the probabilities of p_i so that a coordinate can be sampled from a fixed-length random bit string. The construction assigns an integer number of bit strings to each $k \in I_i$, while preserving all point masses within a constant factor. Set

$$\omega_i(k) := 2^{16d} w_i(k) = 2^{16d - e_i(k)} \in \mathbb{Z}_{\ge 1} \quad (k \in I_i),$$
$$\Omega_i := \sum_{k\in I_i} \omega_i(k) = 2^{16d} Z_i, \qquad \ell_i := \lceil \log_2(2\Omega_i) \rceil.$$

For each $k \in I_i$, first set

$$\widetilde{m}_i(k) := \left\lfloor \frac{2^{\ell_i}\omega_i(k)}{\Omega_i} \right\rfloor, \qquad D_i := 2^{\ell_i} - \sum_{k\in I_i} \widetilde{m}_i(k).$$

As a sum of $|I_i|$ fractional parts, $0 \le D_i < |I_i|$. It counts the random inputs left unassigned by the rounded-down counts. Thus the following adjustment is well defined. Increase $\widetilde{m}_i(k)$ by one for the first D_i elements of I_i in increasing order. Denote the resulting counts by $m_i(k)$. The rounded probabilities are

$$\widehat{p}_i(k) := \frac{m_i(k)}{2^{\ell_i}} \qquad (k \in I_i).$$

The next lemma gives the pointwise approximation guarantee for $\widehat{p}_i(k)$ and the complexity of sampling from $\widehat{p}_i$.

Lemma 7. *For each $i \in [d]$, the distribution $\widehat{p}_i$ satisfies*

$$\frac{1}{2}p_i(k) \le \widehat{p}_i(k) \le \frac{3}{2}p_i(k) \qquad (k \in I_i).$$

A sample from $\widehat{p}_i$ uses $\ell_i = O(d + \log a_i)$ random bits. It can be generated with two passes over I_i in $d^{o(d)}\,\mathrm{poly}(|\boldsymbol{B}|)$ time and polynomial space.

Proof. Since $\omega_i(k)/\Omega_i = p_i(k)$, the construction gives

$$\sum_{k\in I_i} m_i(k) = 2^{\ell_i}, \qquad \widetilde{m}_i(k) = \lfloor 2^{\ell_i} p_i(k) \rfloor,$$
$$m_i(k) \in \{\widetilde{m}_i(k), \widetilde{m}_i(k)+1\}.$$

Thus $|m_i(k) - 2^{\ell_i}p_i(k)| \le 1$. Also, $2^{\ell_i} \ge 2\Omega_i$ and $\omega_i(k) \ge 1$ give

$$2^{\ell_i}p_i(k) = \frac{2^{\ell_i}\omega_i(k)}{\Omega_i} \ge 2.$$

Therefore

$$|\widehat{p}_i(k) - p_i(k)| = 2^{-\ell_i}|m_i(k) - 2^{\ell_i}p_i(k)| \le 2^{-\ell_i} \le \frac{1}{2}p_i(k),$$

which proves the pointwise bounds.

For $e \in \{0, \dots, 16d\}$, let

$$N_{i,e} := |\{k \in I_i : e_i(k) = e\}|.$$

A first pass over $k \in I_i$ computes these $16d+1$ multiplicities, and hence

$$\Omega_i = \sum_{e=0}^{16d} N_{i,e} 2^{16d-e},$$
$$D_i = 2^{\ell_i} - \sum_{e=0}^{16d} N_{i,e} \left\lfloor \frac{2^{\ell_i} 2^{16d-e}}{\Omega_i} \right\rfloor.$$

Choose U uniformly from $\{0, \dots, 2^{\ell_i} - 1\}$. During a second pass over $k \in I_i$ in increasing order, output the least $k \in I_i$ such that

$$U < \sum_{u\in I_i,\, u\le k} m_i(u).$$

Exactly $m_i(k)$ values of U produce k, so the output has distribution $\widehat{p}_i$.

By Lemma 6,

$$\Omega_i = 2^{16d} Z_i \le 2^{16d+2} a_i, \qquad \ell_i = O(d + \log a_i).$$

Each pass has $|I_i| \le 9\sqrt{d}\,Q = d^{o(d)}$ iterations by Equation (1). Each iteration uses $O(1)$ exact arithmetic operations on integers of $O(\ell_i + \log a_i)$ bits. Since $\log a_i \le \log Q = o(d \log d)$ and $d \le |\boldsymbol{B}|$, these operations cost $\mathrm{poly}(|\boldsymbol{B}|)$ bit operations. Thus the two passes take $d^{o(d)}\,\mathrm{poly}(|\boldsymbol{B}|)$ time. The values $N_{i,e}$ and the cumulative sum use $O(d\ell_i)$ bits. All remaining stored data use $\mathrm{poly}(|\boldsymbol{B}|)$ bits. □

Let $\widehat{\nu} := \bigotimes_{i=1}^{d} \widehat{p}_i$ be the product distribution on $\prod_i I_i$. Define the bijection $\Phi : \mathbb{Z}^d \to \mathcal{L}$ and the induced distribution ν by

$$\Phi(\boldsymbol{k}) := \boldsymbol{B}\kappa^{-1}(\boldsymbol{k}), \qquad \nu(\Phi(\boldsymbol{k})) := \widehat{\nu}(\boldsymbol{k}) = \prod_i \widehat{p}_i(k_i).$$

Their supports are

$$\operatorname{supp}(\widehat{\nu}) = \prod_i I_i, \qquad \operatorname{supp}(\nu) = \Phi\Big(\prod_i I_i\Big).$$

The following theorem bounds the point masses of ν and its overlap under every short lattice translation.

Theorem 1. *The distribution ν satisfies*

$$\|\nu\|_\infty \le \frac{2^{3d}}{\Pi}.$$

Let $\boldsymbol{V}$ and $\boldsymbol{W}$ be independent samples from ν. For every $\boldsymbol{z} \in \mathbb{Z}^d$ satisfying $\|\boldsymbol{B}\boldsymbol{z}\|_2 \le \|\boldsymbol{b}_1\|_2$,

$$\rho_{\boldsymbol{z}} := \Pr[\boldsymbol{V} - \boldsymbol{W} = \boldsymbol{B}\boldsymbol{z}] = \sum_{\boldsymbol{v} \in \mathcal{L}} \nu(\boldsymbol{v})\nu(\boldsymbol{v} - \boldsymbol{B}\boldsymbol{z}) \ge \frac{2^{-20d}}{\Pi}.$$

Proof. By Lemma 6, $Z_i \ge a_i/4$. Since $w_i(k) \le 1$, $p_i(k) = w_i(k)/Z_i \le 4/a_i$ for $k \in I_i$. By Lemma 7, $\widehat{p}_i(k) \le (3/2)p_i(k) \le 6/a_i$. Since Φ is a bijection,

$$\|\nu\|_\infty = \|\widehat{\nu}\|_\infty \le \frac{6^d}{\Pi} \le \frac{2^{3d}}{\Pi}.$$

Fix $\boldsymbol{z} \in \mathbb{Z}^d$ with $\|\boldsymbol{B}\boldsymbol{z}\|_2 \le \|\boldsymbol{b}_1\|_2$, and set

$$\mathcal{C} := \{\boldsymbol{k} \in \mathbb{Z}^d : |k_i| \le a_i/4 \text{ for all } i\},$$
$$|\mathcal{C}| = \prod_{i=1}^{d} \big(2\lfloor a_i/4 \rfloor + 1\big) \ge \frac{\Pi}{2^d} \ge \frac{\Pi}{4^d}.$$

For $\boldsymbol{k} \in \mathcal{C}$, $\alpha_i k_i^2 \le 1/16$ for every i. Hence $\mathcal{C} \subseteq \prod_i I_i$ and $\sum_i \alpha_i k_i^2 \le d/16$. Let $\boldsymbol{k}' := T_{\boldsymbol{z}}(\boldsymbol{k})$. Writing $\delta_i(\boldsymbol{k}) = k_i' - k_i$, Lemma 5 gives $\sum_i \alpha_i \delta_i(\boldsymbol{k})^2 \le 4d$. Therefore

$$\sum_i \alpha_i (k_i')^2 \le 2\sum_i \alpha_i k_i^2 + 2\sum_i \alpha_i \delta_i(\boldsymbol{k})^2 \le \frac{d}{8} + 8d = \frac{65d}{8} < 16d.$$

Thus $\boldsymbol{k}' = T_{\boldsymbol{z}}(\boldsymbol{k}) \in \prod_{i=1}^{d} I_i$.

Since $e_i(t) = \lceil \alpha_i t^2 \rceil \le \alpha_i t^2 + 1$,

$$\sum_i e_i(k_i) \le \sum_i \alpha_i k_i^2 + d \le \frac{17d}{16} < 2d,$$
$$\sum_i e_i(k_i') \le \sum_i \alpha_i (k_i')^2 + d \le \frac{73d}{8} < 10d.$$

By Lemmas 6 and 7, for $t \in I_i$,

$$\widehat{p}_i(t) \geq \frac{p_i(t)}{2} = \frac{2^{-e_i(t)}}{2Z_i} \geq \frac{2^{-e_i(t)}}{8a_i}.$$

Consequently,

$$\widehat{\nu}(\boldsymbol{k}) \geq \frac{2^{-3d-\sum_i e_i(k_i)}}{\Pi} \geq \frac{2^{-5d}}{\Pi},$$
$$\widehat{\nu}(\boldsymbol{k}') \geq \frac{2^{-3d-\sum_i e_i(k'_i)}}{\Pi} \geq \frac{2^{-13d}}{\Pi}.$$

Since κ and $\boldsymbol{x} \mapsto \boldsymbol{x}+\boldsymbol{z}$ are bijections, $T_{\boldsymbol{z}}$ is bijective. Moreover, $\Phi(T_{\boldsymbol{z}}(\boldsymbol{k}))-\Phi(\boldsymbol{k}) = \boldsymbol{B}\boldsymbol{z}$. Hence

$$\rho_{\boldsymbol{z}} \geq \sum_{\boldsymbol{k}\in\mathcal{C}} \widehat{\nu}(\boldsymbol{k})\widehat{\nu}(T_{\boldsymbol{z}}(\boldsymbol{k})) \geq \frac{\Pi}{4^d}\frac{2^{-5d}}{\Pi}\frac{2^{-13d}}{\Pi} = \frac{2^{-20d}}{\Pi}.$$

□

Generating one lattice point according to the distribution ν uses

$$\ell := \sum_i \ell_i = O(d^2 + \log \Pi)$$

random bits. Split these bits into consecutive blocks of lengths $\ell_1, \ldots, \ell_d$, and use the ith block to generate k_i according to $\widehat{p}_i$ as in Lemma 7. For $\boldsymbol{k} = (k_1, \ldots, k_d)$, recover $\boldsymbol{x} = \kappa^{-1}(\boldsymbol{k})$ by backward substitution. Size reduction gives

$$|x_i| \leq |k_i| + \frac{1}{2}\sum_{j>i} |x_j| + \frac{1}{2},$$

so each coefficient has $O(d + \log Q + \log d)$ bits. Combining Lemma 7 with this coefficient bound, computing the resulting lattice point $\Phi(\boldsymbol{k})$ takes $d^{o(d)}\,\mathrm{poly}(|\boldsymbol{B}|)$ time and polynomial space.

3.3 Many representations of a shortest vector

We now turn the two bounds on ν into high-probability properties of two sample arrays $\boldsymbol{X}$ and $\boldsymbol{Y}$. For the analysis, fix a shortest vector $\boldsymbol{v}^*$. The arrays are generated without knowing $\boldsymbol{v}^*$. The overlap lower bound makes the number Z of pairs satisfying $\boldsymbol{X}_i - \boldsymbol{Y}_j = \boldsymbol{v}^*$ large with high probability. The maximum point-mass bound limits repeated sample values and hence controls F_{pt}, the second frequency moment of the concatenated sample array. Theorem 2 gives the lower bound on Z, and Theorem 3 gives the upper bound on F_{pt}. We then show how a short random seed for each array lets any entry be recomputed on demand.

Set

$$m := 2^{64n}\Pi. \tag{3}$$

Let

$$\boldsymbol{X} := (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_m) \in \mathcal{L}^m,$$
$$\boldsymbol{Y} := (\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_m) \in \mathcal{L}^m$$

be independent random arrays. Within each array, the entries are pairwise independent with marginal distribution ν.

Write $\boldsymbol{v}^* = \boldsymbol{B}\boldsymbol{z}^*$, so $\|\boldsymbol{v}^*\|_2 \le \|\boldsymbol{b}_1\|_2$. Concatenate the two arrays as

$$\boldsymbol{U} := (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_m, \boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_m).$$

Define the *target-pair count* Z and the second frequency moment F_{pt} by

$$Z := \sum_{i,j \in [m]} \mathbf{1}[\boldsymbol{X}_i - \boldsymbol{Y}_j = \boldsymbol{v}^*],$$
$$F_{\mathrm{pt}} := F_2(\boldsymbol{U}) = \sum_{\boldsymbol{x} \in \mathcal{L}} \big|\{t \in [2m] : \boldsymbol{U}_t = \boldsymbol{x}\}\big|^2.$$

The quantity Z counts representations of $\boldsymbol{v}^*$ as $\boldsymbol{X}_i - \boldsymbol{Y}_j$, while F_{pt} sums the squared multiplicities of lattice points in $\boldsymbol{U}$.

Theorem 2. *For all sufficiently large n, the two arrays satisfy*

$$Z \ge \frac{m^2\, 2^{-20d}}{2\Pi}$$

with probability at least $1 - 2^{-\Omega(n)}$.

Proof. Set

$$z_0 := \frac{m^2\, 2^{-20d}}{2\Pi}.$$

Write $\rho := \rho_{\boldsymbol{z}^*}$. By Theorem 1, $\mathbb{E}[Z] = m^2\rho \ge 2z_0$. For each pair of indices, define

$$\xi_{ij} := \mathbf{1}[\boldsymbol{X}_i - \boldsymbol{Y}_j = \boldsymbol{v}^*].$$

Then $Z = \sum_{i,j \in [m]} \xi_{ij}$, and

$$\mathrm{Var}[Z] = \sum_{i,j \in [m]} \mathrm{Var}[\xi_{ij}] + \sum_{\substack{(i,j),(i',j') \in [m]^2 \\ (i,j) \ne (i',j')}} \mathrm{Cov}(\xi_{ij}, \xi_{i'j'}).$$

The first sum contains the m^2 terms for which the two index pairs coincide. Since ξ_{ij} is Bernoulli with mean ρ,

$$\sum_{i,j \in [m]} \mathrm{Var}[\xi_{ij}] = m^2\rho(1-\rho) \le m^2\rho.$$

We divide the covariance terms in the second sum according to their endpoints.

Case 1: disjoint endpoints. If $i \neq i'$ and $j \neq j'$, then $\boldsymbol{X}_i, \boldsymbol{X}_{i'}$ are independent, as are $\boldsymbol{Y}_j, \boldsymbol{Y}_{j'}$. The two arrays are independent, so these four variables are jointly independent. Hence ξ_{ij} and $\xi_{i'j'}$ are independent, and their covariance is zero.

Case 2: one shared endpoint. If $i = i'$ and $j \neq j'$, then

$$\mathbb{E}[\xi_{ij}\xi_{ij'}] = \sum_{\boldsymbol{x}\in\mathcal{L}} \nu(\boldsymbol{x})\nu(\boldsymbol{x}-\boldsymbol{v}^*)^2 \leq \|\nu\|_\infty \rho.$$

The case $j = j'$ and $i \neq i'$ is symmetric and gives the same bound. In either case, $\mathrm{Cov}(\xi_{ij}, \xi_{i'j'}) \leq \|\nu\|_\infty \rho$. There are $2m^2(m-1)$ ordered pairs of indicators sharing exactly one endpoint. Therefore

$$\mathrm{Var}[Z] \leq m^2\rho + 2m^2(m-1)\|\nu\|_\infty\rho \leq m^2\rho + 2m^3\|\nu\|_\infty\rho.$$

Chebyshev's inequality gives

$$\begin{aligned}\Pr[Z < \tfrac{1}{2}\mathbb{E}[Z]] &\leq \frac{4\,\mathrm{Var}[Z]}{\mathbb{E}[Z]^2} \\ &\leq 4\left(\frac{1}{m^2\rho} + \frac{2\|\nu\|_\infty}{m\rho}\right) \leq 2^{-\Omega(n)},\end{aligned}$$

using $m/\Pi = 2^{64n}$, $d \leq n$, and the bounds on ρ and $\|\nu\|_\infty$ from Theorem 1. Since $z_0 \leq \frac{1}{2}\mathbb{E}[Z]$, this proves the claim. □

We next bound F_{pt}, the second frequency moment of the sample points.

Theorem 3. *The two arrays satisfy*

$$F_{\mathrm{pt}} \leq 16\left(2m + \frac{4m^2\,2^{3d}}{\Pi}\right)$$

with probability at least 15/16.

Proof. Set

$$f_0 := 16\left(2m + \frac{4m^2\,2^{3d}}{\Pi}\right).$$

By definition,

$$F_{\mathrm{pt}} = \sum_{s,t\in[2m]} \mathbf{1}[\boldsymbol{U}_s = \boldsymbol{U}_t].$$

The $2m$ terms with $s = t$ contribute $2m$. For $s \neq t$, pairwise independence gives

$$\Pr[\boldsymbol{U}_s = \boldsymbol{U}_t] = \sum_{\boldsymbol{x}\in\mathcal{L}} \nu(\boldsymbol{x})^2 \leq \|\nu\|_\infty.$$

There are $2m(2m-1)$ ordered pairs with $s \neq t$. Hence

$$\begin{aligned}\mathbb{E}[F_{\mathrm{pt}}] &= 2m + \sum_{\substack{s,t\in[2m]\\ s\neq t}} \Pr[\boldsymbol{U}_s = \boldsymbol{U}_t] \\ &\le 2m + 2m(2m-1)\|\nu\|_\infty \\ &\le 2m + \frac{4m^2\, 2^{3d}}{\Pi} = \frac{f_0}{16}.\end{aligned}$$

Markov's inequality gives $\Pr[F_{\mathrm{pt}} > f_0] \le 1/16$, which proves the claim. □

By a union bound, Theorems 2 and 3 imply that, for all sufficiently large n, $Z \ge z_0$ and $F_{\mathrm{pt}} \le f_0$ hold simultaneously with probability at least

$$1 - 2^{-\Omega(n)} - 1/16 \ge 3/4.$$

Generating a lattice point according to ν uses an ℓ-bit random string. To generate the two arrays from short seeds, set

$$\bar{\ell} := \max\{\ell, \lceil \log_2 m \rceil\}$$

and fix an ordered $\mathbb{F}_2$-basis of $\mathbb{F}_{2^{\bar{\ell}}}$. For $j \in [m]$, let t_j be represented by the $\bar{\ell}$-bit encoding of $j-1$, so t_j is computed from j rather than stored. For each $\boldsymbol{A} \in \{\boldsymbol{X}, \boldsymbol{Y}\}$, choose an independent random affine polynomial

$$h_{\boldsymbol{A}}(t) = a_{\boldsymbol{A}} t + b_{\boldsymbol{A}}, \qquad a_{\boldsymbol{A}}, b_{\boldsymbol{A}} \in \mathbb{F}_{2^{\bar{\ell}}},$$

with independent uniform coefficients. The first ℓ coordinates of $h_{\boldsymbol{A}}(t_j)$ provide the random bits used to generate $\boldsymbol{A}_j$. These bit strings are pairwise independent and uniform, so the entries within each array are pairwise independent with marginal distribution ν. Storing the two coefficients uses $2\bar{\ell}$ bits per array and makes repeated queries consistent [8]. The field construction and evaluations take deterministic polynomial time and space [31].

3.4 From short differences to collisions

The arrays $\boldsymbol{X}$ and $\boldsymbol{Y}$ now contain many pairs with difference $\boldsymbol{v}^*$, but the collision search detects equal labels rather than lattice differences. We bridge this gap by assigning a label to each array entry in three steps. A random signed Hadamard transform spreads a short difference across the coordinates, and a randomly shifted grid assigns the two lattice points defining that difference to the same cell with probability at least $\gamma = 2^{-O(n)}$. Every cell has bounded diameter, so a lattice packing bound limits the number of distinct lattice points with the same cell label. Combined with the bound on F_{pt} from Section 3.3, this controls the second frequency moment of the cell-label array. Finally, a pairwise-universal hash maps the cell labels to the range required by the collision search. Theorem 4 combines these facts: many target pairs receive the same label, while the second frequency moment of the label array remains small.

We first define the random grid. Set

$$\bar{n} := 2^{\lceil \log_2 n \rceil} \le 2n,$$

and identify $\mathbb{Z}^n$ with its embedding in $\mathbb{Z}^{\bar{n}}$ obtained by appending $\bar{n} - n$ zero coordinates. Let $\boldsymbol{H}_{\bar{n}}$ be the Sylvester Hadamard matrix. Choose independent uniform signs $\varepsilon_j \in \{-1, 1\}$, and set

$$\boldsymbol{D}_{\boldsymbol{\varepsilon}} := \operatorname{diag}(\varepsilon_1, \dots, \varepsilon_{\bar{n}}), \qquad \boldsymbol{S} := \boldsymbol{H}_{\bar{n}} \boldsymbol{D}_{\boldsymbol{\varepsilon}}.$$

Since $\boldsymbol{S}^{\mathsf{T}} \boldsymbol{S} = \bar{n} \boldsymbol{I}_{\bar{n}}$, every vector $\boldsymbol{v}$ with $\|\boldsymbol{v}\|_2 \le \|\boldsymbol{b}_1\|_2$ satisfies

$$\|\boldsymbol{S}\boldsymbol{v}\|_2 = \sqrt{\bar{n}} \|\boldsymbol{v}\|_2 \le \sqrt{\bar{n}} \|\boldsymbol{b}_1\|_2.$$

Choose the least power of two W satisfying

$$W^2 \ge 128 \|\boldsymbol{b}_1\|_2^2 \log_2 \bar{n}. \tag{4}$$

This condition can be checked exactly because $\log_2 \bar{n} = \lceil \log_2 n \rceil$ is an integer. Independently choose a shift $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_{\bar{n}})$ whose coordinates are independent and uniform on $\{0, \dots, W-1\}$. Write $g := (\boldsymbol{D}_{\boldsymbol{\varepsilon}}, \boldsymbol{\sigma})$ for the resulting random grid. Define the *cell label* of $\boldsymbol{x} \in \mathbb{Z}^{\bar{n}}$ as the integer tuple $\operatorname{cell}_g(\boldsymbol{x}) \in \mathbb{Z}^{\bar{n}}$ with

$$\operatorname{cell}_g(\boldsymbol{x})_j := \left\lfloor \frac{(\boldsymbol{S}\boldsymbol{x})_j + \sigma_j}{W} \right\rfloor, \qquad j \in [\bar{n}].$$

The level sets of cell_g form the cells of the random grid.

The next lemma bounds the collision probability for $\|\boldsymbol{v}\|_2 \le \|\boldsymbol{b}_1\|_2$ and the cell diameter.

Lemma 8. *Let $\boldsymbol{x}, \boldsymbol{v} \in \mathbb{Z}^{\bar{n}}$ satisfy $\|\boldsymbol{v}\|_2 \le \|\boldsymbol{b}_1\|_2$, and set*

$$\gamma := \frac{1}{2}(3/4)^{\bar{n}} = 2^{-O(n)}.$$

Then

$$\Pr_g[\operatorname{cell}_g(\boldsymbol{x}) = \operatorname{cell}_g(\boldsymbol{x} - \boldsymbol{v})] \ge \gamma.$$

Moreover, any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}^{\bar{n}}$ in the same cell satisfy

$$\|\boldsymbol{x} - \boldsymbol{y}\|_2 < W.$$

Proof. The collision claim is immediate when $\boldsymbol{v} = 0$, so assume $\boldsymbol{v} \ne 0$. For fixed $j \in [\bar{n}]$,

$$(\boldsymbol{S}\boldsymbol{v})_j = \sum_{k=1}^{\bar{n}} (\boldsymbol{H}_{\bar{n}})_{j,k} \varepsilon_k v_k.$$

The summands are independent and have mean zero. Since every Hadamard entry has absolute value one,

$$\mathbb{E}_{\boldsymbol{D}_{\boldsymbol{\varepsilon}}}[(\boldsymbol{S}\boldsymbol{v})_j] = 0,$$

$$\operatorname{Var}_{\boldsymbol{D}_{\boldsymbol{\varepsilon}}}[(\boldsymbol{S}\boldsymbol{v})_j] = \sum_{k=1}^{\bar{n}} v_k^2 = \|\boldsymbol{v}\|_2^2 \le \|\boldsymbol{b}_1\|_2^2.$$

Hoeffding's inequality therefore gives

$$\begin{aligned}\Pr_{\boldsymbol{D}_{\boldsymbol{\varepsilon}}}\left[|(\boldsymbol{S}\boldsymbol{v})_j| > \frac{W}{4}\right] &\le 2\exp\left(-\frac{W^2}{32\|\boldsymbol{v}\|_2^2}\right)\\ &\le 2\exp\left(-\frac{W^2}{32\|\boldsymbol{b}_1\|_2^2}\right).\end{aligned}$$

A union bound and Equation (4) give, since $\bar{n} \ge 2$,

$$\begin{aligned}\Pr_{\boldsymbol{D}_{\boldsymbol{\varepsilon}}}\left[\max_{j\in[\bar{n}]}|(\boldsymbol{S}\boldsymbol{v})_j| > \frac{W}{4}\right] &\le 2\bar{n}\exp\left(-\frac{W^2}{32\|\boldsymbol{b}_1\|_2^2}\right)\\ &\le 2\bar{n}\exp(-4\log_2\bar{n}) \le 4e^{-4} < \frac{1}{2}.\end{aligned}$$

Thus, with probability at least $1/2$,

$$\max_{j\in[\bar{n}]}|(\boldsymbol{S}\boldsymbol{v})_j| \le W/4.$$

Fix a sign matrix $\boldsymbol{D}_{\boldsymbol{\varepsilon}}$ satisfying this bound. For each j, set

$$a_j := (\boldsymbol{S}\boldsymbol{x})_j, \qquad b_j := (\boldsymbol{S}(\boldsymbol{x}-\boldsymbol{v}))_j.$$

Both quantities are integers. Moreover,

$$|a_j - b_j| = |(\boldsymbol{S}\boldsymbol{v})_j| \le W/4.$$

Exactly $|a_j - b_j|$ integers $\sigma_j \in \{0,\dots,W-1\}$ make $a_j+\sigma_j$ and $b_j+\sigma_j$ belong to different intervals $[kW,(k+1)W)$, where $k\in\mathbb{Z}$. Hence

$$\Pr_{\sigma_j}\left[\left\lfloor\frac{a_j+\sigma_j}{W}\right\rfloor = \left\lfloor\frac{b_j+\sigma_j}{W}\right\rfloor\right] = 1 - \frac{|a_j-b_j|}{W} \ge \frac{3}{4}.$$

The shifts are independent, so

$$\Pr_{\boldsymbol{\sigma}}[\mathrm{cell}_g(\boldsymbol{x}) = \mathrm{cell}_g(\boldsymbol{x}-\boldsymbol{v}) \mid \boldsymbol{D}_{\boldsymbol{\varepsilon}}] \ge (3/4)^{\bar{n}}.$$

Averaging over $\boldsymbol{D}_{\boldsymbol{\varepsilon}}$ proves the collision bound.

If $\boldsymbol{x},\boldsymbol{y}$ share a cell, each coordinate of $\boldsymbol{S}(\boldsymbol{x}-\boldsymbol{y})$ has absolute value less than W. Hence

$$\sqrt{\bar{n}}\|\boldsymbol{x}-\boldsymbol{y}\|_2 = \|\boldsymbol{S}(\boldsymbol{x}-\boldsymbol{y})\|_2 < \sqrt{\bar{n}}W.$$

Dividing by $\sqrt{\bar{n}}$ gives $\|\boldsymbol{x}-\boldsymbol{y}\|_2 < W$. □

The following Lemma 9 gives an upper bound for the number of lattice points in each cell.

Lemma 9. *Given a quasi-HKZ basis with rank d, every cell contains at most*

$$K := \left(1 + 128\left\lceil\sqrt{\log_2 n}\right\rceil\right)^d = \exp(O(d\log\log n))$$

distinct points in $\mathcal{L}$.

Proof. Let $\mathcal{Q}$ be the set of lattice points in a fixed cell. The claim is immediate if $\mathcal{Q} = \varnothing$, so fix $\boldsymbol{x}_0 \in \mathcal{Q}$. By Lemma 8, every $\boldsymbol{x} \in \mathcal{Q}$ satisfies $\|\boldsymbol{x} - \boldsymbol{x}_0\|_2 < W$. Moreover, the open balls of radius $\lambda_1/2$ centered at the points of $\mathcal{Q}$ are pairwise disjoint, and all lie in the ball of radius $W + \lambda_1/2$ centered at $\boldsymbol{x}_0$. Comparing d-dimensional volumes in the lattice span gives

$$|\mathcal{Q}| \le \left(\frac{W + \lambda_1/2}{\lambda_1/2}\right)^d = \left(1 + \frac{2W}{\lambda_1}\right)^d.$$

The minimality of W in Equation (4) and $\log_2 \bar{n} = \lceil \log_2 n \rceil \le 2 \log_2 n$ give

$$W < 2\sqrt{128}\,\|\boldsymbol{b}_1\|_2 \sqrt{\log_2 \bar{n}} \le 32 \|\boldsymbol{b}_1\|_2 \sqrt{\log_2 n}.$$

Together with $\|\boldsymbol{b}_1\|_2 \le 2\lambda_1$ from Lemma 2, this implies

$$\frac{2W}{\lambda_1} < 128 \sqrt{\log_2 n} \le 128 \left\lceil \sqrt{\log_2 n} \right\rceil.$$

Substitution proves $|\mathcal{Q}| \le K$. Finally,

$$\left\lceil \sqrt{\log_2 n} \right\rceil = \left\lceil \sqrt{\lceil \log_2 n \rceil} \right\rceil,$$

so K is exactly computable. □

Fix the arrays $\boldsymbol{X}$ and $\boldsymbol{Y}$, and fix a grid g. Let $n_{\boldsymbol{x}}$ be the multiplicity of $\boldsymbol{x}$ in $\boldsymbol{U}$. Since each cell C contains at most K distinct points, applying Cauchy–Schwarz within each cell and then summing over the cells gives

$$F_{\text{cell}} := \sum_C \left(\sum_{\boldsymbol{x} \in C} n_{\boldsymbol{x}} \right)^2 \le K \sum_{\boldsymbol{x}} n_{\boldsymbol{x}}^2 = K F_{\text{pt}}. \tag{5}$$

To apply Lemma 4, represent each of the $2m$ cell labels by a canonical bit string of a common polynomial length. To compress these labels to a universe of size $(2m)^{O(1)}$, choose h independently and uniformly from a pairwise-universal family of binary affine maps to $\{0,1\}^{q_h}$, where $q_h := \lceil 8 \log_2(2m) \rceil$. This range has size less than $2(2m)^8$.

Define

$$Z_g := \sum_{\substack{i,j \in [m] \\ \boldsymbol{X}_i - \boldsymbol{Y}_j = \boldsymbol{v}^*}} \mathbf{1}[\text{cell}_g(\boldsymbol{X}_i) = \text{cell}_g(\boldsymbol{Y}_j)].$$

Apply h to the cell labels in the order of $\boldsymbol{U}$ to obtain the compressed label array

$$\boldsymbol{C}_{g,h} := \big(h(\text{cell}_g(\boldsymbol{U}_t))\big)_{t \in [2m]}.$$

The expected number over h of ordered pairs with unequal cell labels but equal compressed labels is at most

$$\frac{(2m)^2}{2^{q_h}} \le (2m)^{-6}.$$

Equal cell labels always have equal compressed labels. Hence, by Markov's inequality, h creates no equality between unequal cell labels with probability at least $1 - (2m)^{-6} \geq 63/64$.

Theorem 4. *Assume*

$$Z \geq z_0, \qquad F_{\mathrm{pt}} \leq f_0,$$

and set

$$r_0 := \left\lfloor \frac{\gamma z_0}{4} \right\rfloor, \qquad p_0 := \min\{(2m)^2, K f_0\}.$$

If g and h are sampled independently as above, then

$$\Pr_{g,h}[F_2(\boldsymbol{C}_{g,h}) \leq p_0 \text{ and } Z_g \geq r_0] \geq \frac{\gamma}{8}.$$

Proof. Since $m = 2^{64n}\Pi$, $d \leq n$, $\bar{n} \leq 2n$, and $\Pi \geq 1$, the definitions give

$$\frac{\gamma z_0}{4} \geq 2^{108n-4}(3/4)^{2n} \longrightarrow \infty.$$

Hence $r_0 \geq 1$ for all sufficiently large n.

For each target pair, its two lattice points are assigned to the same cell with probability at least γ by Lemma 8, so $\mathbb{E}_g[Z_g] \geq \gamma Z$. Since $0 \leq Z_g \leq Z$, we also have $\mathbb{E}_g[Z_g^2] \leq Z\mathbb{E}_g[Z_g]$. Paley–Zygmund now gives

$$\Pr_g[Z_g \geq \tfrac{1}{2}\mathbb{E}_g[Z_g]] \geq \frac{(\mathbb{E}_g[Z_g])^2}{4\mathbb{E}_g[Z_g^2]} \geq \frac{\mathbb{E}_g[Z_g]}{4Z} \geq \frac{\gamma}{4}.$$

Let $\mathcal{A} := \{Z_g \geq \frac{1}{2}\mathbb{E}_g[Z_g]\}$. Whenever $\mathcal{A}$ occurs,

$$Z_g \geq \gamma Z/2 \geq \gamma z_0/2 \geq r_0.$$

The cell multiplicities sum to $2m$, so $F_{\mathrm{cell}} \leq (2m)^2$. Equation (5) gives $F_{\mathrm{cell}} \leq K f_0$. Thus $F_{\mathrm{cell}} \leq \min\{(2m)^2, K f_0\} = p_0$ for every g.

Let $\mathcal{H}_g$ be the event that h is injective on the distinct cell labels represented in the two arrays. The compression bound above gives $\Pr_h[\mathcal{H}_g \mid g] \geq 63/64$ for every g. If both $\mathcal{A}$ and $\mathcal{H}_g$ occur, then

$$F_2(\boldsymbol{C}_{g,h}) = F_{\mathrm{cell}} \leq p_0, \qquad Z_g \geq r_0.$$

Therefore,

$$\Pr_{g,h}[\mathcal{A} \cap \mathcal{H}_g] \geq \frac{63}{64}\Pr_g[\mathcal{A}] \geq \frac{63\gamma}{256} \geq \frac{\gamma}{8}.$$

□

4 The complete SVP algorithm

We now use the sampler and the collision search in Section 3 to solve SVP, which is formally stated in the following Theorem 5.

Theorem 5. *There is a classical algorithm that, given any full-rank integer basis $\boldsymbol{B} \in \mathbb{Z}^{n\times n}$, returns a shortest nonzero vector of $\mathcal{L}(\boldsymbol{B})$ with probability at least $2/3$. On every execution, it uses polynomial space and performs at most*

$$\operatorname{poly}(|\boldsymbol{B}|)\, n^{\frac{n}{4e}+o(n)}$$

bit operations, where $|\boldsymbol{B}|$ is the input bit length.

We first solve the problem for a quasi-HKZ basis in Section 4.1. By Theorem 2, the sample arrays contain many pairs whose difference is a shortest vector. The random grid in Theorem 4 turns many of these pairs into collisions while keeping the second frequency moment small. We can therefore apply Lemma 4 to find one such pair. We then use this solver in Kannan's recursion to handle an arbitrary basis in Section 4.2.

4.1 Solving SVP on a quasi-HKZ basis

Hanrot and Stehlé show that Kannan's method enumerates all lattice vectors in a prescribed ball, which satisfies the shape bounds Lemma 2. Since we need only one shortest vector, we replace this enumeration by collision search. Our subroutine samples two implicit arrays once and then repeats the grid search. On each grid, it searches for equal cell labels, checks the corresponding difference exactly, and stores the shortest valid difference.

The next theorem gives the correctness, success probability, and cost of this subroutine.

Theorem 6. *Given a quasi-HKZ basis $\boldsymbol{B} \in \mathbb{Z}^{n\times d}$ of rank $d \le n$, the solver always returns a vector $\boldsymbol{v} \in \mathcal{L}(\boldsymbol{B}) \setminus \{\boldsymbol{0}\}$ satisfying $\|\boldsymbol{v}\|_2 \le \|\boldsymbol{b}_1\|_2$. With probability at least $2/3$, this vector is shortest. The solver uses polynomial space and performs at most*

$$2^{O(n)} \exp(O(d \log\log n))\, d^{d/(4e)+o(d)} \operatorname{poly}(|\boldsymbol{B}|)$$

bit operations.

Proof. The initial vector is $\boldsymbol{b}_1$. Every sampled point belongs to $\mathcal{L}(\boldsymbol{B})$, so every stored difference does as well. The exact test excludes zero and requires norm at most $\|\boldsymbol{b}_1\|_2$, so the output is always a nonzero lattice vector no longer than $\boldsymbol{b}_1$.

Success probability. Fix a shortest vector $\boldsymbol{v}^*$. Since $\boldsymbol{b}_1$ is a nonzero lattice vector, $\|\boldsymbol{v}^*\|_2 \le \|\boldsymbol{b}_1\|_2$. By Theorems 2 and 3 and a union bound, for all sufficiently large n, with probability at least $3/4$ the two arrays contain at least z_0 pairs whose difference is $\boldsymbol{v}^*$, and their second frequency moment is at most f_0. Fix arrays with these two properties.

Input: A quasi-HKZ basis $\boldsymbol{B} = [\boldsymbol{b}_1 \ \cdots \ \boldsymbol{b}_d] \in \mathbb{Z}^{n\times d}$ of rank $d \le n$.
Output: A nonzero vector $\boldsymbol{v} \in \mathcal{L}(\boldsymbol{B})$ satisfying $\|\boldsymbol{v}\|_2 \le \|\boldsymbol{b}_1\|_2$. With probability at least 2/3, it is a shortest vector of $\mathcal{L}(\boldsymbol{B})$.

1. Set up the sampler for ν from Section 3 and use the array length m from Equation (3). Let γ be the lower bound from Lemma 8. Two lattice points at distance at most $\|\boldsymbol{b}_1\|_2$ share a grid cell with probability at least γ.
2. Draw seeds for the two implicit arrays $\boldsymbol{X}, \boldsymbol{Y}$. Set $\boldsymbol{v} \leftarrow \boldsymbol{b}_1$.
3. Repeat a fixed, sufficiently large constant multiple of $1/\gamma$ times:
   (a) Draw a fresh grid g and compression map h. Use the numerical bounds p_0, r_0 from Theorem 4 in the search of Lemma 4 on the combined compressed label array.
   (b) For every returned cross-array pair $(\boldsymbol{X}_i, \boldsymbol{Y}_j)$, recompute the two samples. Accept the pair only if
   $$\mathrm{cell}_g(\boldsymbol{X}_i) = \mathrm{cell}_g(\boldsymbol{Y}_j) \qquad \text{and} \qquad 0 < \|\boldsymbol{X}_i - \boldsymbol{Y}_j\|_2 \le \|\boldsymbol{b}_1\|_2.$$
   (c) If an accepted pair gives a shorter vector than $\boldsymbol{v}$, replace $\boldsymbol{v}$ by its difference.
4. Return $\boldsymbol{v}$.

All array, grid, compression, and walk seeds used above are mutually independent.

Fig. 1. The SVP solver for a quasi-HKZ basis.

For a fresh grid and compression map, Theorem 4 gives the following with probability at least $\gamma/8$: the compressed label array has second frequency moment at most p_0 and contains at least r_0 target pairs. Conditioned on this event, the two hypotheses of Lemma 4 hold with $p = p_0$ and $\mathcal{G}$ equal to the set of target pairs. Hence the collision search returns a target pair with constant probability. The pair passes the exact test: its points share a cell, and their difference is the nonzero vector $\boldsymbol{v}^*$ of norm at most $\|\boldsymbol{b}_1\|_2$.

Thus one outer iteration succeeds with probability $\Omega(\gamma)$. The iterations use independent seeds, so a sufficiently large constant multiple of $1/\gamma$ iterations has failure probability at most 1/9. Including the probability that the arrays have the two properties above, the success probability is at least $(3/4)(8/9) = 2/3$. When a target pair is inspected, either the current candidate is already shortest or it is replaced by $\boldsymbol{v}^*$. No later accepted vector can be shorter.

Complexity. We first bound the number of array accesses. The parameters in Theorem 4 satisfy $r_0 \ge \gamma z_0/8$ and $p_0 \le K f_0$. The definitions of z_0, f_0 in Theorems 2 and 3, together with $m = 2^{64n}\Pi$ from Equation (3), give $f_0/z_0 = 2^{O(d)}$. It follows that

$$\frac{p_0}{r_0} \le \gamma^{-1} K 2^{O(d)}.$$

The access bound in Lemma 4, applied to an array of length $2m$, is therefore

$$\widetilde{O}\Big(\sqrt{m}\,\gamma^{-1} K 2^{O(d)}\Big)$$

per outer iteration. We use the same fixed access limit even when the grid does not satisfy the lemma's hypotheses. Thus the bound holds for every choice of the seeds. The $O(\gamma^{-1})$ outer iterations add the second loss in γ, for a total of

$$\widetilde{O}\Big(\sqrt{m}\,\gamma^{-2}K2^{O(d)}\Big)$$

array accesses.

It remains to bound the three factors in this expression. By Lemma 8, $\gamma^{-2} = 2^{O(n)}$. By Lemma 9, $K = \exp(O(d \log\log n))$. Finally, Equation (3) gives $\sqrt{m} = 2^{32n}\sqrt{\Pi}$. Equation (1) gives $\Pi < 2^d P$, while Lemma 2 gives $P \le 2^{O(d)} d^{d/(2e)}$. Hence

$$\sqrt{m} \le 2^{O(n)} d^{d/(4e)}.$$

By Lemma 7, sampling all d coordinates takes $d^{o(d)}\,\mathrm{poly}(|\boldsymbol{B}|)$ time and polynomial space. Evaluating the seed polynomial and computing the cell label add only polynomial work. Each walk reports at most one pair, so the access bound also covers all exact checks. The total number of bit operations is therefore

$$2^{O(n)} \exp(O(d \log\log n))\, d^{d/(4e)+o(d)}\, \mathrm{poly}(|\boldsymbol{B}|).$$

The arrays are stored by their seeds. The space bound in Lemma 7 covers sample evaluation, and Lemma 4 uses only $O(\log^3(2m))$ additional bits for the walk. The bounds above give $\log m = O(n + d \log d)$. Thus the grid, compression map, current samples, and best vector all have polynomial-size descriptions. This proves the claimed space bound.

□

4.2 Kannan's recursion for arbitrary bases

To handle an arbitrary basis, we use Kannan's recursion [21,15]. It first recursively reduces the projected basis $[\pi_2(\boldsymbol{b}_2)\ \cdots\ \pi_2(\boldsymbol{b}_d)]$ of $\pi_2(\mathcal{L}(\boldsymbol{B}))$ to an HKZ-reduced basis. Whenever the norm condition $\|\boldsymbol{b}_2^*\|_2 \ge \|\boldsymbol{b}_1\|_2/2$ fails, Gauss reduction replaces $\boldsymbol{b}_1$ by a shorter vector, and the process repeats. Once the basis is quasi-HKZ, the solver in Theorem 6 finds a shortest vector with probability at least $2/3$. We run it n^2 times and keep the shortest result. We put this vector first and recursively reduce the corresponding projected basis once more. When every group of solver runs contains a shortest vector, the resulting basis is HKZ reduced.

Figure 2 gives the recursive procedure Reduce. At the top level, we save $\boldsymbol{b}_1$, call Reduce($\boldsymbol{B}$), and return the first vector of the result. If the call exceeds the time or space bound proved below, we stop and return the saved vector.

Input: A rank-d integer lattice basis $\boldsymbol{B} = [\boldsymbol{b}_1 \ \cdots \ \boldsymbol{b}_d] \in \mathbb{Z}^{n \times d}$.
Output: A basis of the same lattice, HKZ reduced if each group of solver runs finds a shortest vector.

1. LLL-reduce $\boldsymbol{B}$.
2. If $d = 1$, return $\boldsymbol{B}$.
3. Repeat until:
4. Compute the projections of $\boldsymbol{b}_2, \ldots, \boldsymbol{b}_d$ orthogonally to $\boldsymbol{b}_1$.
5. Apply REDUCE to this projected basis.
6. Lift the result to lattice vectors and size-reduce them against $\boldsymbol{b}_1$.
7. If $\|\boldsymbol{b}_2^*\|_2 \geq \|\boldsymbol{b}_1\|_2/2$, leave the loop.
8. Gauss-reduce $(\boldsymbol{b}_1, \boldsymbol{b}_2)$ and repeat.
9. Run the solver in Fig. 1 n^2 times. Divide each output by the gcd of its coefficients in $\boldsymbol{B}$. Let $\boldsymbol{v}$ be the shortest resulting lattice vector.
10. Set $\boldsymbol{B} \leftarrow \mathrm{LLL}(\boldsymbol{v}, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_d)$, processing $\boldsymbol{v}$ first and deleting dependent vectors.
11. Compute the projections of $\boldsymbol{b}_2, \ldots, \boldsymbol{b}_d$ orthogonally to $\boldsymbol{b}_1$.
12. Apply REDUCE to this projected basis.
13. Lift the result to lattice vectors, size-reduce them against $\boldsymbol{b}_1$, and return $\boldsymbol{B}$.

Fig. 2. Kannan's recursive reduction REDUCE.

The projected bases have rational entries. Before each recursive call, we clear their denominators. To lift the result, we record the integer basis changes, apply them to the corresponding unprojected vectors $\boldsymbol{b}_2, \ldots, \boldsymbol{b}_d$, and size-reduce against the first vector. All solver calls use independent seeds.

The next lemma bounds the number of loop rounds. It uses the standard analysis of Kannan's recursion [21,23,17,15].

Lemma 10. *For $d \geq 2$, the loop in Fig. 2 produces a quasi-HKZ basis in $O(\log d)$ rounds.*

Proof. After recursively reducing the projected basis and lifting the result, the basis is size reduced and its projected basis is HKZ reduced. By Definition 2, only the norm test remains. We show that it can fail only $O(\log d)$ times.

Write $\lambda_1 := \lambda_1(\mathcal{L}(\boldsymbol{B}))$. If the test fails, size reduction gives $\|\boldsymbol{b}_2\|_2 < \|\boldsymbol{b}_1\|_2/\sqrt{2}$. Thus Gauss reduction replaces the first vector by $\boldsymbol{b}_1'$ with $\|\boldsymbol{b}_1'\|_2 < \|\boldsymbol{b}_1\|_2/\sqrt{2}$. A shortest vector cannot be parallel to $\boldsymbol{b}_1$, because every lattice vector in that direction is an integer multiple of $\boldsymbol{b}_1$, whereas $\boldsymbol{b}_2$ is shorter. Consequently, the projection of every shortest vector is nonzero, so the HKZ-reduced projected basis gives $\|\boldsymbol{b}_2^*\|_2 \leq \lambda_1$. The two-dimensional area bound for Gauss reduction now gives

$$\|\boldsymbol{b}_1'\|_2^2 \leq 2\|\boldsymbol{b}_1\|_2\|\boldsymbol{b}_2^*\|_2 \leq 2\|\boldsymbol{b}_1\|_2\lambda_1,$$

$$\frac{\|\boldsymbol{b}_1'\|_2}{\lambda_1} \leq \sqrt{\frac{2\|\boldsymbol{b}_1\|_2}{\lambda_1}}.$$

LLL starts with $\|\boldsymbol{b}_1\|_2/\lambda_1 \leq 2^{(d-1)/2}$. Repeated square roots make this ratio bounded by a constant after $O(\log d)$ failed rounds. Each subsequent failure

reduces $\|\boldsymbol{b}_1\|_2$ by a factor of at least $\sqrt{2}$, so only constantly many more are possible before $\|\boldsymbol{b}_1\|_2 < \lambda_1$. Since $\boldsymbol{b}_1$ is always a nonzero lattice vector, this cannot happen, and the loop must stop.

The basis changes preserve the lattice. The usual exact implementation of LLL and recursive lifting keeps the intermediate bases and recorded changes polynomial in the original input size. □

Proof (of Theorem 5). We first show that the recursion gives an HKZ-reduced basis when it finds a shortest vector at every call. We then bound the probability that a search fails and the total cost.

Correctness. Suppose that, at every recursive call, at least one of the n^2 solver runs returns a shortest vector. We prove by induction on the rank that REDUCE returns an HKZ-reduced basis of the same lattice. For $d = 1$, the returned rank-one basis is HKZ reduced. Assume henceforth that $d \geq 2$. By induction, the recursive calls in the loop return HKZ-reduced projected bases. Thus Lemma 10 shows that the loop stops after $O(\log d)$ rounds with a quasi-HKZ basis.

By Theorem 6, each solver output is a nonzero lattice vector. Dividing its coefficients by their gcd keeps it nonzero and in the lattice and cannot increase its length. At least one solver output is shortest and is unchanged by gcd division, since a nontrivial common divisor would yield a shorter nonzero lattice vector. Every resulting vector has norm at least that of this shortest output. Hence the shortest resulting vector $\boldsymbol{v}$ is shortest. Its coefficients have gcd one, so it can be extended to a basis.

LLL preserves the lattice because the input list still contains the old basis. It also keeps $\boldsymbol{v}$ first: no nonzero lattice vector is shorter, so the LLL test never swaps it out of the first position. The last recursive call reduces the new projected basis, and lifting makes the full basis size reduced. By Definition 2, a shortest first vector and an HKZ-reduced projected basis give an HKZ-reduced basis. This proves the induction and the correctness of the top-level output.

Success probability. Consider the next group of n^2 solver runs. Fix the full history before these runs and suppose that every earlier group found a shortest vector. The argument above shows that the current input is quasi-HKZ. For this fixed input, Theorem 6 bounds the failure probability of one run by $1/3$. The runs use independent seeds, so the probability that all n^2 fail is at most 3^{-n^2}.

Before the first such failure, Lemma 10 bounds the number of loop rounds by $O(\log d)$ at rank d. There is one lower-rank call per round and one more after inserting $\boldsymbol{v}$. The number of recursive calls is therefore at most

$$\prod_{d=2}^{n} O(\log(d+1)) = \exp(O(n \log\log n)).$$

A union bound over the possible first failure bounds the probability of any such failure by

$$\exp(O(n \log\log n))\, 3^{-n^2} = 2^{-\Omega(n^2)} < \frac{1}{3}.$$

Thus all the chosen vectors are shortest with probability at least $2/3$.

Complexity. First suppose that every group finds a shortest vector. Here $|\boldsymbol{B}|$ denotes the original input length. As noted in the proof of Lemma 10, the intermediate bases have bit length polynomial in $|\boldsymbol{B}|$. By Theorem 6, each run at rank d therefore takes

$$2^{O(n)} \exp(O(d \log\log n))\, d^{d/(4e)+o(d)} \operatorname{poly}(|\boldsymbol{B}|)$$

bit operations. The n^2 repetitions add only a polynomial factor. All other work at each node is polynomial. The recursion has $\exp(O(n \log\log n)) = n^{o(n)}$ nodes, and $d \le n$, so the total time is

$$n^{n/(4e)+o(n)} \operatorname{poly}(|\boldsymbol{B}|).$$

The solver uses polynomial space by Theorem 6. We run the calls one after another, and each recursive call decreases the rank. At most n levels are active at once, so the total space is polynomial.

Choose the top-level time and space limits to cover these bounds. They are never reached when every group finds a shortest vector. On any other execution they bound the cost, and stopping returns the saved nonzero vector $\boldsymbol{b}_1$. The success probability remains at least $2/3$, which proves the theorem. □

5 Conclusion and Future Work

We presented a randomized polynomial-space algorithm for exact SVP with running time $n^{\frac{n}{4e}+o(n)}$. The main idea is to replace the exhaustive enumeration step in Kannan's algorithm [21,15] with the low-space collision search of Lyu and Zhu [24]. For this approach to work, we design a sampling distribution so that a shortest vector can be written in many ways as the difference of two samples, while the number of irrelevant collisions stays under control.

This work leaves two natural directions for future research. The first is to improve the running time while keeping the space polynomial. Possible approaches include better sampling distributions, faster low-space collision methods, and a reduction in the loss caused by the random grid. It would also be interesting to obtain a similar improvement with a deterministic algorithm.

The second direction is to study whether these ideas can be used in practice, for example within the General Sieve Kernel (G6K) framework and implementation [3]. The current algorithm is not meant for direct implementation. In particular, the array length $m = 2^{64n}\Pi$ is very large, and the random grid must be sampled $O(1/\gamma)$ times, where γ is exponentially small. Moreover, each array entry is computed only when it is needed, so every access requires exact sampling, recomputation, and several hash evaluations. The quasi-HKZ preprocessing adds another large cost.

These costs come from parameter choices and tools used to prove the asymptotic bound, and a practical version would likely use a different setting. Possible changes include much smaller parameters, heuristic sampling and pruning, collision detection in batches, and reuse of the vector database already maintained

by G6K [12,26,3]. An interesting question is whether the main idea, i.e., giving a short vector many representations and then finding a useful collision, still works well after these changes.

AI disclosure. GPT-5.6 Sol was used interactively throughout the preparation of this manuscript, including early research, organization, exposition, and drafting. At an early stage, the model proposed a classical polynomial-space approach with running time $n^{\frac{3n}{8e}+o(n)}$, based on the result of Chen et al. [10]. Further exploration led to the present algorithm, which uses the low-space collision search of Lyu and Zhu [24], improves the running time to $n^{\frac{n}{4e}+o(n)}$, and gives a simpler analysis. The authors checked and substantially revised the final manuscript and remain solely responsible for all mathematical claims, proofs, references, and conclusions.

References

1. Aggarwal, D., Dadush, D., Regev, O., Stephens-Davidowitz, N.: Solving the shortest vector problem in 2^n time via discrete gaussian sampling. In: Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC). pp. 733–742. ACM (2015). https://doi.org/10.1145/2746539.2746606, https://arxiv.org/abs/1412.7994
2. Ajtai, M., Kumar, R., Sivakumar, D.: A sieve algorithm for the shortest lattice vector problem. In: Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC). pp. 601–610. ACM (2001). https://doi.org/10.1145/380752.380857
3. Albrecht, M.R., Ducas, L., Herold, G., Kirshanova, E., Postlethwaite, E.W., Stevens, M.: The general sieve kernel and new records in lattice reduction. In: Advances in Cryptology – EUROCRYPT 2019, Part II. Lecture Notes in Computer Science, vol. 11477, pp. 717–746. Springer (2019). https://doi.org/10.1007/978-3-030-17656-3_25, https://eprint.iacr.org/2019/089
4. Aono, Y., Nguyen, P.Q., Shen, Y.: Quantum lattice enumeration and tweaking discrete pruning. In: Advances in Cryptology – ASIACRYPT 2018. Lecture Notes in Computer Science, vol. 11272, pp. 405–434. Springer (2018). https://doi.org/10.1007/978-3-030-03326-2_14
5. Beame, P., Clifford, R., Machmouchi, W.: Element distinctness, frequency moments, and sliding windows. In: Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS). pp. 290–299. IEEE Computer Society (2013). https://doi.org/10.1109/FOCS.2013.39, https://arxiv.org/abs/1309.3690
6. Becker, A., Coron, J.S., Joux, A.: Improved generic algorithms for hard knapsacks. In: Advances in Cryptology – EUROCRYPT 2011. Lecture Notes in Computer Science, vol. 6632, pp. 364–385. Springer (2011). https://doi.org/10.1007/978-3-642-20465-4_21
7. Becker, A., Ducas, L., Gama, N., Laarhoven, T.: New directions in nearest neighbor searching with applications to lattice sieving. In: Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). pp. 10–24. SIAM (2016). https://doi.org/10.1137/1.9781611974331.ch2

8. Carter, J.L., Wegman, M.N.: Universal classes of hash functions. Journal of Computer and System Sciences **18**(2), 143–154 (1979). https://doi.org/10.1016/0022-0000(79)90044-8
9. Chen, L., Jin, C., Williams, R.R., Wu, H.: Truly low-space element distinctness and subset sum via pseudorandom hash functions. In: Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms (SODA). pp. 1661–1678. SIAM (2022). https://doi.org/10.1137/1.9781611977073.67, https://arxiv.org/abs/2111.01759
10. Chen, Y., Chen, Z., Guo, T., Sun, C., Wen, W., Yu, Y.: Time memory trade-off for enumeration. Cryptology ePrint Archive, Paper 2025/2227 (2025), https://eprint.iacr.org/2025/2227
11. Fincke, U., Pohst, M.: Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. Mathematics of Computation **44**(170), 463–471 (1985). https://doi.org/10.1090/S0025-5718-1985-0777278-8
12. Gama, N., Nguyen, P.Q., Regev, O.: Lattice enumeration using extreme pruning. In: Advances in Cryptology – EUROCRYPT 2010. Lecture Notes in Computer Science, vol. 6110, pp. 257–278. Springer (2010). https://doi.org/10.1007/978-3-642-13190-5_13, https://www.iacr.org/archive/eurocrypt2010/66320257/66320257.pdf
13. Gao, Y., Feng, Y., Hu, H.: Discrete gaussian sampling meets BDGL decoding: Solving the shortest vector problem in $2^{0.5596n+o(n)}$ time. Cryptology ePrint Archive, Paper 2026/1844 (2026), https://eprint.iacr.org/2026/1844
14. Gao, Y., Feng, Y., Hu, H.: Solving the shortest vector problem in $2^{0.7314n+o(n)}$ time via discrete gaussian sampling on superlattices. Cryptology ePrint Archive, Paper 2026/1587 (2026), https://eprint.iacr.org/2026/1587
15. Hanrot, G., Stehlé, D.: Improved analysis of kannan's shortest lattice vector algorithm. In: Advances in Cryptology – CRYPTO 2007. Lecture Notes in Computer Science, vol. 4622, pp. 170–186. Springer (2007). https://doi.org/10.1007/978-3-540-74143-5_10, https://arxiv.org/abs/0705.0965
16. Hanrot, G., Stehlé, D.: Worst-case hermite-korkine-zolotarev reduced lattice bases. arXiv preprint arXiv:0801.3331 (2008)
17. Helfrich, B.: Algorithms to construct Minkowski reduced and Hermite reduced lattice bases. Theoretical Computer Science **41**, 125–139 (1985). https://doi.org/10.1016/0304-3975(85)90067-2
18. Hhan, M.: Finding a shortest vector and more in $2^{n/2+o(n)}$ time using q-ary coset difference tree. Cryptology ePrint Archive, Paper 2026/1859 (2026), https://eprint.iacr.org/2026/1859
19. Hhan, M.: Solving the shortest vector problem in $2^{0.6039n}$ time via mid-point Hessian. arXiv preprint arXiv:2608.02478 (2026), https://arxiv.org/abs/2608.02478
20. Howgrave-Graham, N., Joux, A.: New generic algorithms for hard knapsacks. In: Advances in Cryptology – EUROCRYPT 2010. Lecture Notes in Computer Science, vol. 6110, pp. 235–256. Springer (2010). https://doi.org/10.1007/978-3-642-13190-5_12
21. Kannan, R.: Minkowski's convex body theorem and integer programming. Mathematics of Operations Research **12**(3), 415–440 (1987). https://doi.org/10.1287/moor.12.3.415
22. Knuth, D.E.: The Art of Computer Programming, Volume II: Seminumerical Algorithms. Addison-Wesley (1969)
23. Lenstra, A.K., Lenstra, Jr., H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen **261**, 515–534 (1982). https://doi.org/10.1007/BF01457454

24. Lyu, X., Zhu, W.: Time–space tradeoffs for element distinctness and set intersection via pseudorandomness. In: Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms (SODA). pp. 5243–5281. SIAM (2023). https://doi.org/10.1137/1.9781611977554.ch190, https://arxiv.org/abs/2210.07534
25. Micciancio, D., Voulgaris, P.: A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations. SIAM Journal on Computing **42**(3), 1364–1391 (2013). https://doi.org/10.1137/100811970
26. Micciancio, D., Walter, M.: Fast lattice point enumeration with minimal overhead. In: Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). pp. 276–294. SIAM (2015). https://doi.org/10.1137/1.9781611973730.21, https://eprint.iacr.org/2014/569
27. Nguyen, P.Q., Stern, J.: The two faces of lattices in cryptology. In: Cryptography and Lattices – CaLC 2001. Lecture Notes in Computer Science, vol. 2146, pp. 146–180. Springer (2001). https://doi.org/10.1007/3-540-44670-2_12
28. Peikert, C.: A decade of lattice cryptography. Foundations and Trends in Theoretical Computer Science **10**(4), 283–424 (2016). https://doi.org/10.1561/0400000074, https://web.eecs.umich.edu/~cpeikert/pubs/lattice-survey.pdf
29. Pollard, J.M.: A monte carlo method for factorization. BIT **15**(3), 331–334 (1975). https://doi.org/10.1007/BF01933667
30. Schnorr, C.P., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Mathematical Programming **66**, 181–199 (1994). https://doi.org/10.1007/BF01581144
31. Shoup, V.: New algorithms for finding irreducible polynomials over finite fields. Mathematics of Computation **54**(189), 435–447 (1990). https://doi.org/10.1090/S0025-5718-1990-0993933-0, https://www.shoup.net/papers/detirred.pdf