Approximate Algorithms on Lattices with Small Determinant (Extended Abstract)

Jung Hee Cheon and Changmin Lee

Seoul National University, Korea
{jhcheon,cocomi11}@snu.ac.kr

Abstract. In this paper, we propose approximate lattice algorithms for solving the shortest vector problem (SVP) and the closest vector problem (CVP) on an n-dimensional Euclidean integral lattice L. Our algorithms run in polynomial time of the dimension and determinant of lattices and improve on the LLL algorithm when the determinant of a lattice is less than $2^{n^2/4}$. More precisely, our approximate SVP algorithm gives a lattice vector of size $\leq 2^{\sqrt{\log \det L}}$ and our approximate CVP algorithm gives a lattice vector, the distance of which to a target vector is $2^{\sqrt{\log \det L}}$ times the distance from the target vector to the lattice. One interesting feature of our algorithms is that their output sizes are independent of dimension and become smaller as the determinant of L becomes smaller. For example, if $\det L = 2^{n\sqrt{n}}$, a short vector outputted from our approximate SVP algorithm is of size $2^{n^{3/4}}$, which is asymptotically smaller than the size $2^{n/4+\sqrt{n}}$ of the outputted short vectors of the LLL algorithm. It is similar to our approximate CVP algorithm.

Keywords: LLL algorithm, SVP, CVP, Hermite normal form, Babai's nearest plane algorithm.

1 Introduction

Lattices and the LLL algorithm. A lattice is a set of all the integral linear combinations of some linearly independent vectors in $\mathbb{R}^n$. The *Shortest Vector Problem (SVP)* of a lattice is to find a shortest nonzero vector in the Euclidean norm given a basis of the lattice. This problem is easy when the dimension n of the lattice is small, but becomes exponentially hard as n increases. The best known algorithm takes $2^{n+o(n)}$ time [1]. Another class of SVP algorithms outputs an *approximate* SVP in a polynomial time of n. The most widely used algorithm for SVP is the LLL algorithm proposed by Lenstra, Lenstra, and Lovasz [19] in 1982. The LLL algorithm has a very considerable amount of applications in many areas, including computer algebra, algorithmic number theory, and cryptanalysis.

The LLL algorithm on a lattice L of dimension n runs in polynomial time in n and the determinant L, but approximates only the shortest vector. The size of output vector $\boldsymbol{b}$ satisfies $\|\boldsymbol{b}\| \leq 2^{n/4}(\det L)^{1/n}$, while a shortest nonzero vector of L is smaller than $\sqrt{n}(\det L)^{1/n}$ by the Minkowski theorem. The approximate factor $2^{n/4}$ is fixed regardless of the determinant of the lattices and therefore is relatively significant as the determinant becomes smaller.

Our Contribution. In this paper, we propose approximate lattice algorithms on integral lattices with a small determinant. Our initial idea is to reduce the SVP on a lattice L of dimension n to the SVP on a sublattice L' of L with a smaller dimension n'. In general, the determinant of a sublattice may be smaller or larger than that of the original lattice. Our approach is to use the Hermite normal form (HNF) to obtain a subspace with a bounded determinant. The HNF of an integral lattice is a unique (generalized) lower-triangular matrix, the columns of which form a basis of the lattice and it can be computed in polynomial time from any basis of the lattice [22]. We show that the subspace generated by the last m columns has a smaller determinant than that of the original lattice for any positive integer $m \leq n$. By applying LLL to this subspace of appropriate m, we obtain a short vector, the size of which is upper-bounded by $2^{\sqrt{\log \det L}+1}$, which is smaller than the shortest vector from LLL when the determinant of L is smaller than $2^{n^2/4}$. We may apply any approximate SVP algorithm to our algorithm instead of LLL, including the block Korkin-Zolotarev (BKZ) algorithm. Our algorithm with BKZ of block size polylog(λ) can be used to break the current parameter of the GGH scheme of the security parameter λ in poly$(n, \det L, \lambda)$, which is the first plausible candidate multilinear map [11].

Second, we apply this method repeatedly to a lattice obtained from an integral lattice L by permuting coordinates. Its purpose is to address the short independent vector problem (SIVP), which asks to find n linearly independent short vectors given a basis of a lattice. We remark that a permuted lattice has a different HNF, although each lattice has one unique HNF. By applying LLL on the last m columns of the permuted lattice, we obtain a short vector having a permutation that is a small vector in L. By taking a different permutation, we expect to obtain additional short vectors in L. Moreover, the subspace obtained from the last m columns consists of vectors, the first $(n-m)$ entries of which are zero. This is not sufficient to guarantee their independence, but we can choose permutations elaborately to obtain n linearly independent vectors, of which the first $(n-m)$ vectors are as small as $2^{\sqrt{\log \det L}+1}$ and the last m vectors are LLL-reduced and extensible to a basis of L. This result is also asymptotically better than that of LLL only for a lattice of small determinant of sub-quadratic size in dimension.

Third, we apply the above result to the closest vector problem (CVP) on an integral lattice, which asks to find a lattice vector closest to the target vector in $\mathbb{R}^n$ given a lattice and a target vector. CVP is one of the most fundamental lattice problems together with SVP. An exact solution of CVP takes an exponential time in dimension, but we have polynomial time approximate algorithms [4]. We propose a new CVP algorithm using Babai's nearest algorithm. Given a target vector $\boldsymbol{w}$, we apply our algorithm to obtain n linearly independent vectors $\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n$ and write $\boldsymbol{w} = \boldsymbol{w}_1 + \boldsymbol{w}_2$, where $\boldsymbol{w}_1$ is in the lattice L_1 generated by the first $(n-m)$ short vectors and $\boldsymbol{w}_2$ is in the orthogonal complement of L_1 in $\mathbb{R}^n$. Our approximate CVP algorithm applies Babai's algorithm two times, the first time to $\boldsymbol{w}_1$ and L_1 and the second to the projections of $\boldsymbol{w}_2$ and L_2 against L_1. By combining the results, we obtain a lattice vector, the distance of which to the target vector is upper-bounded by approximately $O(2^{\sqrt{\log(\det L)}})$ times the distance of the target vector to the closest lattice point, unless it is too small.

All three algorithms are asymptotically better than the previous polynomial time lattice algorithms when the determinant of the lattice has a sub-quadratic size of dimension. We remark that an $n \times n$ integral matrix with entries bounded by poly(n) has determinant $O(\text{poly}(n)^n) = 2^{O(n \log(n))}$ and is susceptible to our lattice algorithms. For example, if $\det L = 2^{n\sqrt{n}}$, a short vector outputted from our approximate SVP algorithm is of size $2^{n^{3/4}}$, which is asymptotically smaller than the size $2^{n/4+\sqrt{n}}$ of the output short vectors of LLL. It is similar to that of our approximate CVP algorithm.

Related work. Many studies have been conducted on exact algorithms for SVP and CVP [2, 3, 6, 5, 13, 16, 21, 20, 18]. In addition, there have been many attempts to improve the classical LLL and BKZ algorithms [9, 10, 25, 24, 17, 23, 26, 7]. However, no serious asymptotic improvement on approximate SVP and CVP algorithms was achieved, despite the very considerable number of their applications in optimization, computational number theory, and cryptanalysis.

Organization. In Section 2, we introduce some preliminaries related to lattice problems and algorithms and introduce HNF, which plays an important role in the next sections. In Section 3, we present our new approximate SVP algorithm combining LLL and HNF. In Section 4, we propose a new SIVP algorithm that repeats the proposed approximate SVP algorithm after coordinate permutations, and show that the results can be used to improve Babai's nearest plane algorithm for CVP.

2 Preliminaries

Throughout this paper, we use lower-case and upper-case bold letters to denote vectors and matrices, respectively. For a vector $\boldsymbol{u} \in \mathbb{R}^n$, $\|\boldsymbol{u}\|$ denotes the Euclidean norm of $\boldsymbol{u}$. For vectors $\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m \in \mathbb{R}^n$, we denote $[\boldsymbol{u}_1 \cdots \boldsymbol{u}_m]$ as an $n \times m$ matrix, the i-th column of which is $\boldsymbol{u}_i$. For a basis $\{\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n\}$ of vector space, we use $(\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n)$ to denote an ordered basis. For a set S, $\text{span}(S)$ is defined as the $\mathbb{R}$-linear vector space generated by S. In $\log_2(\cdot)$, we omit the subscript 2.

2.1 Lattices

An m-dimensional lattice of rank n is the set of all integer combinations $\{\sum_{i=1}^n x_i \boldsymbol{b}_i |\ x_i \in \mathbb{Z}\}$ of n linear independent vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{R}^m$. The set of $\boldsymbol{b}_i$'s is called a basis of L. If $n = m$, L is called a full rank lattice. A sublattice is a subset $L' \subset L$ that is also a lattice. A matrix $\boldsymbol{B} = [\boldsymbol{b}_1 \cdots \boldsymbol{b}_n]$ is a basis matrix of a lattice L. If $L \subset \mathbb{Z}^n$, we call it an integral lattice.

Given a lattice L in $\mathbb{R}^m$ with a basis matrix $\boldsymbol{B}$, the determinant of the lattice is defined as $\sqrt{\det(\boldsymbol{B}^T\boldsymbol{B})}$, where $\boldsymbol{B}^T$ is the transpose matrix of $\boldsymbol{B}$ and $\det(M)$ denotes the determinant of a square matrix M. By abusing the notations, $\det L$ is used to denote the determinant of L. The i-th successive minimum of a lattice L, denoted by $\lambda_i(L)$, is defined by

$$\lambda_i(L) = \min_r \{r : \dim(\text{span}(L \cap B(r))) \geq i\},$$

where $B(r) : \{x \in \mathbb{R}^n : \|x\| \leq r\}$.

Given a basis $\{\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n\}$, we denote by $\boldsymbol{b}_1^*, \cdots, \boldsymbol{b}_n^*$ the output of the Gram-Schmidt orthogonalization of the basis, i.e., $\boldsymbol{b}_i^*$ is the component of $\boldsymbol{b}_i$ orthogonal to $\text{span}(\boldsymbol{b}_1, \cdots, \boldsymbol{b}_{i-1})$.

2.2 Lattice Problems

Let us recall the most well-known hard problems on lattices.

- The Shortest Vector Problem (SVP): given a basis matrix B of a lattice L, compute a non-zero lattice vector $\boldsymbol{v} \in L$ such that $\|\boldsymbol{v}\| \leq \lambda_1(L)$.
- The Closest Vector Problem (CVP): given a basis matrix B of a lattice L and a vector $\boldsymbol{w} \in \mathbb{R}^n$, compute a lattice vector $\boldsymbol{v} \in L$ such that $\|\boldsymbol{w} - \boldsymbol{v}\|$ is minimal.

SVP and CVP are widely used to solve algorithmic problems such as integer programming, factoring polynomials over the rationals, solving subset sum problem, simultaneous Diophantine equations, and approximate greatest common divisor problem, to name a few.

2.3 Lattice Algorithms

There are two types of lattice algorithms. One is the exact lattice algorithms, such as enumerations [15, 13, 26] for SVP and Aggarwal *et al.*'s algorithms for CVP [1], which lead to exact solutions in exponential time of input size. The second is approximate lattice algorithms, such as the LLL algorithm and Babai's nearest plane algorithm, which result in approximate solutions, but run in polynomial time of input size. In this paper, we focus on approximate lattice reduction algorithms.

The LLL algorithm introduced by Lenstra, Lenstra, and Lovasz [19] is a basic lattice reduction algorithm to produce a short vector and a short basis. Upon input of an ordered basis $(\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots \boldsymbol{b}_n)$ of a lattice L, the LLL algorithm (with factor 3/4) outputs another ordered basis $(\boldsymbol{b}'_1, \boldsymbol{b}'_2, \cdots \boldsymbol{b}'_n)$, satisfying $\|\boldsymbol{b}'_i\| \leq 2^{n/2} \cdot \lambda_i(L)$ for each i, in polynomial time in n and $\max_i \|\boldsymbol{b}_i\|$. This new basis is called an *LLL-reduced basis*. The output is dependent on the order of basis elements.

While the Minkowski theorem guarantees $\lambda_1(L) \leq \sqrt{\gamma_n}(\det L)^{1/n}$ for the Hermite constant $\gamma_n \geq n$, the first vector of LLL satisfies $\|\boldsymbol{b}'_1\| \leq 2^{n/4} \cdot \det(L)^{1/n}$. When, upon input of a lattice L of dimension n, an algorithm outputs a lattice element of size $\alpha \det(L)^{1/n}$, α is called the *Hermite factor* of the algorithm for L. The LLL algorithm has Hermite factor $\leq 2^{n/4}$ for any lattice.

To solve the approximate CVP, Babai's nearest plane algorithm is widely used. For an input basis matrix $\boldsymbol{B} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots \boldsymbol{v}_n]$ of lattice L and $\boldsymbol{w} \in \mathbb{R}^n$, Babai's algorithm gives a lattice vector $\boldsymbol{v} \in L$, satisfying $\|\boldsymbol{v} - \boldsymbol{w}\|^2 \leq \frac{1}{4} \sum_{i=1}^{n} \|\boldsymbol{b}_i^*\|^2$, in polynomial time in n and $\max_i \|\boldsymbol{v}_i\|$. Moreover, if $\boldsymbol{B}$ is an LLL-reduced basis, then $\|\boldsymbol{v} - \boldsymbol{w}\|^2 \leq 2^n \|\boldsymbol{u} - \boldsymbol{w}\|^2$ for all $\boldsymbol{u} \in L$.

2.4 Hermite Normal Form

Approximately speaking, the Hermite normal form of a full-rank square matrix $\boldsymbol{M}$ is a lower triangular matrix obtained by column operations with integral coefficients from $\boldsymbol{M}$. A matrix with integer entries is in (column) *Hermite normal form (HNF)* if

1. (Zero rows) All nonzero columns (columns with at least one nonzero element) are at the left of any columns of all zeros (all zero columns, if any, are at the right of the matrix).
2. (Triangular) The first nonzero entry from the above (called the pivot) of a nonzero column is always strictly below the pivot of the column on the left of it. Moreover, it is positive.
3. (Modulus Reduction) All the entries in a row at the left of a pivot are nonnegative and strictly smaller than the pivot.

The HNF is used for efficiently solving linear Diophantine equations, integer programming, and various lattice problems, such as the lattice membership problem, lattice basis problem, and lattice union problems [22, 8]. Given a basis matrix $\boldsymbol{B}$, computing its HNF, denoted by HNF($\boldsymbol{B}$), takes polynomial time and space in dimension n and $\det(\boldsymbol{B})$. More precisely, for an $n \times n$ matrix with the largest entries bounded by M, the computation requires $O(n^\theta B(n \log M))$ time and $O(n^2 \log M)$ space, where n^θ is the number of arithmetic operations required to multiply two $n \times n$ matrices and $B(t)$ is an upper bound to the number of bit operations required to multiply two t-bit numbers [22].

The HNF of a matrix over $\mathbb{Z}$ is unique. We call the HNF (ordered) basis of $\boldsymbol{B}$ the columns of HNF ($\boldsymbol{B}$).

3 Lattice Reductions with HNF

In this section, we propose an algorithm to compute a short vector of a lattice by using a lattice reduction algorithm with HNF. As the determinant of the lattice is smaller, our algorithm improves on the previous lattice reduction algorithms.

3.1 Subspace from Hermite Normal Form

Currently, all approximate lattice reduction algorithms with polynomial running time in input size have exponentially large Hermite factors in the rank of the input matrix. For example, LLL has Hermite factor $2^{n/4}$ for the rank n of input matrix L. To obtain a short vector with a smaller Hermite factor, we may attempt to apply LLL to a subspace of L of smaller dimension. However, it is not always true that a subspace of smaller dimension has a determinant smaller than that of the original lattice L.

In this subsection, we show how to use HNF to obtain a sublattice having a determinant not larger than that of the original lattice.

Theorem 1. *Given a basis matrix $[\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n]$ of integral lattice L in Hermite normal form, $\det([\boldsymbol{b}_i, \cdots, \boldsymbol{b}_n]) \geq \det([\boldsymbol{b}_{i+1}, \cdots, \boldsymbol{b}_n])$ for $1 \leq i \leq n-1$.*

Proof. Suppose $\boldsymbol{b}_i$ is a vector with pivot d_i. Consider an ordered basis $\boldsymbol{B}_i := \{\boldsymbol{b}_n, \cdots, \boldsymbol{b}_i\}$ and its corresponding Gram-Schmidt basis $\{\boldsymbol{b}_n^*, \cdots, \boldsymbol{b}_i^*\}$. Then $\det(\boldsymbol{B}_i) = \prod_{j=i}^{n} \|\boldsymbol{b}_j^*\| = \|\boldsymbol{b}_i^*\| \cdot \det(\boldsymbol{B}_{i+1})$. Here $\boldsymbol{b}_i^*$ is of the form $\boldsymbol{b}_i - (c_n \boldsymbol{b}_n + \cdots + c_{i+1}\boldsymbol{b}_{i+1})$ for some $c_n, \ldots, c_{i+1} \in \mathbb{R}$ and so the first nonzero component of $\boldsymbol{b}_i^*$ is also d_i. Hence, $\|\boldsymbol{b}_i^*\| \geq d_i$ and $\det(\boldsymbol{B}_i) \geq d_i \cdot \det(\boldsymbol{B}_{i+1})$. Since d_i is a positive integer by the definition of HNF, we have $\det(\boldsymbol{B}_i) \geq \det(\boldsymbol{B}_{i+1})$. □

3.2 An Algorithm for SVP

We apply the LLL algorithm to a sublattice L' of integral lattice L with $\det(L') \leq \det(L)$ obtained as in the previous section. It is described in Algorithm 1.

Algorithm 1 LLL with HNF

```
Input: $\{\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n\}$ and $m < n$
Output: $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m\}$
1. Compute the HNF basis $(\boldsymbol{b}'_1, \cdots, \boldsymbol{b}'_n)$ of $(\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n)$.
2. Set $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$ as the output vector of LLL upon input $(\boldsymbol{b}'_{n-m+1}, \cdots, \boldsymbol{b}'_n)$
return $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$.
```

If we take an appropriate m, the first vector of the output of **Algorithm 1** is shorter than that of LLL on the original lattice when the determinant of the lattice is not large.

Theorem 2. *For an n-dimensional integral lattice L with $\log\det(L) < n^2/4$ and $m = \left\lfloor 2\sqrt{\log(\det(L))} \right\rceil$,* **Algorithm 1** *outputs an LLL-reduced basis of a certain sublattice L' of L with $\det L' \leq \det L$ in polynomial time in n and $\det(L)$. In particular, the first vector satisfies $\boldsymbol{v}_1 \in L$ such that $\|\boldsymbol{v}_1\| < 2^{\sqrt{\log\det(L)}+1/32}$.*

Proof. Let $\{\boldsymbol{b}'_1, \cdots, \boldsymbol{b}'_n\}$ be an HNF basis of lattice L. By the given conditions, $m = \left\lfloor 2\sqrt{\log(\det(L))} \right\rceil < n$. When Algorithm 1 is applied to L, the output is merely the output of LLL on the sublattice L' generated by $\{\boldsymbol{b}'_{n-m+1}, \cdots, \boldsymbol{b}'_n\}$.

Moreover, since $\det(L') \leq \det(L)$ by Theorem 1, the first vector $\boldsymbol{v}_1$ of the output satisfies

$$\|\boldsymbol{v}_1\| \leq 2^{m/4} \cdot \det(L')^{1/m} \leq 2^{m/4} \cdot \det(L)^{1/m} = 2^{\sqrt{\log(\det(L))}+\epsilon},$$

where $0 \leq \epsilon \leq 1/(16m) \leq 1/32$.

Obviously, $\boldsymbol{v}_1 \in L' \subset L$, and the running time is upper bounded by the running time of LLL and HNF. □

Another lattice reduction algorithm, the BKZ algorithm with block size β, outputs a lattice vector of a size bounded by $2(\gamma_\beta)^{\frac{n-1}{2(\beta-1)}+\frac{3}{2}} \cdot (\det L)^{\frac{1}{n}}$ in $\mathrm{poly}(n, \mathrm{size}(\boldsymbol{B})) \cdot \mathcal{C}_{HKZ}(\beta)$ times, where γ_β is the Hermite constant of rank β, $\mathrm{size}(\boldsymbol{B})$ is the size of largest entries of basis matrix $\boldsymbol{B}$ of L, and $\mathcal{C}_{HKZ}(\beta) = 2^{O(\beta)}$ is the cost of HKZ-reduction in dimension β [12, 1].

We claim that our algorithm asymptotically outperforms BKZ with block size $\beta = \mathrm{polylog}(n)$ for an n-dimensional integral lattice with determinant 2^{n^δ} for $\delta < 2$. While the output vector of BKZ with β has a Euclidean norm of at most $2^{(n \log \beta)/\beta + O(1)}$, **Algorithm 1** with BKZ gives a vector of the Euclidean norm at most $2^{n^{\delta/2}}$, which is asymptotically smaller than the previous one. We remark that when the basis entries of an n-dimensional lattice L are bounded by $2^{o(n)}$, we achieve a subquadratic determinant $2^{o(n^2)}$ of L.

Furthermore, we may plug BKZ in our algorithm (instead of LLL) to obtain a shorter lattice vector $\boldsymbol{b} \in L$. More precisely, we have $\|\boldsymbol{b}\| < 2\beta^{\sqrt{(\log_\beta \det(L))/2(\beta-1)}+\frac{3}{2}}$ when $\det L = \beta^{o(n^2)}$. The running time is upper bounded by the running time of the BKZ algorithm with block size β, $\mathrm{poly}(n, size(\boldsymbol{B})) \cdot \mathcal{C}_{HKZ}(\beta)$ and the running time of computing HNF.

3.3 Application to SVP on Ideal Lattices

The first plausible candidate of multilinear maps was suggested by Garg, Gentry, and Halevi [11] in 2013 and was followed by many subsequent research studies. However, Hu and Jia gave a cryptanalysis of this candidate and showed it to be insecure. In this paper, we present a different cryptanalysis of GGH multilinear maps based on SVP. The scheme is constructed on a ring $R := \mathbb{Z}[x]/\langle x^n + 1\rangle$ for a power n of 2, and its security relies on finding a short generator of an ideal lattice $\langle g\rangle$ generated by g in the ring. For the λ bit of security, it is assumed that $n = \Omega(\lambda^2)$ and $q = 2^{\Theta(\lambda)}$, and the determinant of the ideal $\langle g\rangle$ is bounded by $(\lambda n)^{n/2}$.

According to [11, Sec. 6.3.3], if one finds a short element $\hat{g}$ in $\langle g\rangle$ such that $\|\hat{g}\| \leq q^{3/8}$, the GGH scheme is not secure. Since a basis of the ideal lattice $\langle g\rangle$ is easily recovered in polynomial time by the zeroizing attack, we may apply our Algorithm 1 to find a short element in the lattice and break the GGH scheme. Using BKZ of block size β in Algorithm 1, we can obtain an element $\hat{g} \in \langle g\rangle$ satisfying

$$\|\hat{g}\| \leq 2\beta^{\sqrt{(\log_\beta \det(\langle g\rangle))/2(\beta-1)}+\frac{3}{2}} = 2^{O\left(\lambda\sqrt{\frac{\log\lambda \log\beta}{\beta}}\right)},$$

which is asymptotically smaller than $q^{3/8}$ for $\beta = \mathrm{polylog}(\lambda)$. Hence, a desired short element in the ideal lattice can be computed in polynomial time in λ, which yields a break of the GGH scheme with the current parameters.

4 Short Independent Vectors and CVP

In this section, we extend **Algorithm 1** to obtain more short vectors and use it to improve Babai's nearest plane algorithm for CVP.

4.1 Short Independent Vectors

For an integral lattice L, by using the LLL algorithm with HNF one can find a vector $\boldsymbol{v}$ from a sublattice $L' \subset L$ with $\|\boldsymbol{v}\| \le 2^{\sqrt{\log\det(L)}+1/32}$. We want to obtain more vectors of similar size. Since HNF is unique, the output of **Algorithm 1** is unique if we fix a lattice and m. However, if we change the order of coordinates, a different HNF and thus different outputs of **Algorithm 1** are obtained. One issue of this approach is to guarantee the independence of the second vector from the first vector.

In this subsection, we propose an algorithm to output independent short vectors by applying HNF after permuting coordinates. More precisely, we permute coordinates of vectors so that the specific m coordinates of vectors are placed in their m lowest rows and apply HNF for some integer m. We apply **Algorithm 1** and take the inverse permutation to obtain a lattice vector, of which the specific $(n-m-1)$ entries are zero and one specific entry (corr. to pivot) is nonzero. By repeating, we obtain linearly independent $(n-m)$ lattice vectors of size almost $2^{m/2}$, when $m = \lfloor 2\sqrt{\log(\det L)}\rceil$.

Now, we define a permutation $\pi_{i,j}$ from $\mathbb{Z}^n$ to $\mathbb{Z}^n$ swapping the i-th and j-th coordinates:

$$\pi_{i,j}(x_1,\cdots,x_i,\cdots,x_j,\cdots x_n) = (x_1,\cdots,x_j,\cdots,x_i,\cdots x_n).$$

Then, $\pi_{i,j}(L) = \{\pi_{i,j}(\boldsymbol{v}) \mid \boldsymbol{v} \in L\}$ is also a lattice and $\det(\pi_{i,j}(L)) = \det(L)$. The proposed algorithm is described in **Algorithm 2**.

Algorithm 2 Short Independent Vectors with LLL+HNF

```
Input: ${\boldsymbol{b}_1,\cdots,\boldsymbol{b}_n}$ and $m < n$
Output: ${\boldsymbol{v}_1,\cdots,\boldsymbol{v}_n}$
1. for $i = 1$ upto $(n-m)$ do
2.     Set $\boldsymbol{v}_i$ as the first vector of the output of Algorithm1$(\boldsymbol{b}_1,\ldots,\boldsymbol{b}_n,m)$
3.     Set $j_i$ as the position of the first nonzero component of $\boldsymbol{v}_i$
4.     Set $(\boldsymbol{v}_1,\ldots,\boldsymbol{v}_i) = (\pi_{i,j_i}(\boldsymbol{v}_1),\cdots,\pi_{i,j_i}(\boldsymbol{v}_i))$ and $(\boldsymbol{b}_1,\cdots,\boldsymbol{b}_n) = (\pi_{i,j_i}(\boldsymbol{b}_1),\cdots,\pi_{i,j_i}(\boldsymbol{b}_n))$
5. end for
6.     Set $(\boldsymbol{v}_{n-m+1},\ldots,\boldsymbol{v}_n)$ as the output of Algorithm1$(\boldsymbol{b}_1,\ldots,\boldsymbol{b}_n,m)$
7. for $i = 1$ upto $n$ do
8.     Set $\boldsymbol{v}_i = \pi^{-1}_{1,j_1} \circ \cdots \circ \pi^{-1}_{n-m,j_{n-m}}(\boldsymbol{v}_i)$
9. end for
return $\boldsymbol{v}_1,\cdots,\boldsymbol{v}_n$.
```

We show that this algorithm outputs n linearly independent vectors, not necessarily a basis of a lattice generated by $\boldsymbol{b}_1,\ldots,\boldsymbol{b}_n$. In fact, the vectors $\boldsymbol{v}_1,\ldots,\boldsymbol{v}_n$ obtained in Step 6 of **Algorithm 2** form an integral matrix

$$[\boldsymbol{v}_1 \ldots \boldsymbol{v}_n] = \begin{bmatrix} T_{n-m} & O \\ R & M_m \end{bmatrix}$$

, where T_{n-m} is an $(n-m)\times(n-m)$ lower-triangular integral matrix with nonzero diagonal entries and O is a zero matrix. Moreover, M_m is an $m\times m$ integral matrix, the columns of which form an LLL-reduced basis of a certain sublattice L' with $\det L' \le \det L$ for the original lattice L generated by $\boldsymbol{b}_1,\dots,\boldsymbol{b}_n$.

Theorem 3. *Let L be an n-dimensional integral lattice with $\det(L) < 2^{n^2/4}$, and $m = \lfloor 2\sqrt{\log(\det L)}\rceil$. One can find n independent vectors $\boldsymbol{v}_1,\dots,\boldsymbol{v}_n$ in polynomial time in n and $\det(L)$, where the first $(n-m+1)$ vectors are of size smaller than $2^{\sqrt{\log(\det L)}+1/32}$ and the last m vectors form an LLL-reduced basis of a sublattice L' of L with $\det L' \le \det L$.*

Proof. Given an integral lattice L, we apply **Algorithm 2** on any basis of L and m. Note that **Algorithm 1** outputs m linearly independent vectors starting with $(n-m)$ zero entries.

After the first round of the first loop, $\boldsymbol{v}_1$ is a vector with a nonzero first entry followed by the $(n-m)$ zero entries. After the second round of the first loop, $\boldsymbol{v}_2$ is a vector, the first $(n-m+1)$ entries of which are zero, except the second, which is nonzero. Since $j_2 \le (n-m+1)$, the first entry of $\boldsymbol{v}_1$ is unchanged and remains nonzero after permutation π_{2,j_2}. By repeating this procedure, after the entire $(n-m)$-th round of the first loop, we obtain $(n-m)$ vectors $(\boldsymbol{v}_1,\cdots,\boldsymbol{v}_{n-m})$ in the permuted lattice $\pi_{n-m,j_{n-m}}\circ\cdots\circ\pi_{1,j_1}(L)$, where $\boldsymbol{v}_i$ is a vector of which the first $(i-1)$ entries are zero and the i-th entry is nonzero. Hence, the first $(n-m)$ rows of the matrix $[\boldsymbol{v}_1\dots\boldsymbol{v}_{n-m}]$ form a lower-triangular matrix with nonzero diagonal entries and therefore $\boldsymbol{v}_1,\dots,\boldsymbol{v}_{n-m}$ are linearly independent.

On the other hand, the first $(n-m)$ rows of the matrix $[\boldsymbol{v}_{n-m+1}\cdots\boldsymbol{v}_n]$ from Step 6 of Algorithm 2 form a zero matrix. Hence, they are independent of $\boldsymbol{v}_1,\dots,\boldsymbol{v}_{n-m}$. Moreover, $\boldsymbol{v}_{n-m+1},\dots,\boldsymbol{v}_n$ are an output of Algorithm 1 and therefore an LLL-reduced basis of a certain sublattice L' of L with $\det L' \le \det L$. Hence, all $\boldsymbol{v}_i$'s form an n linearly independent set.

By inverting the permutation (note that $\pi_{i,j_i}^{-1} = \pi_{i,j_i}$), we obtain $(n-m)$ linearly independent vectors of L. By Theorem 2, we can bound the size of $\boldsymbol{v}_i$ for $1 \le i \le (n-m+1)$ by $2^{\sqrt{\log(\det L)}+1/32}$. □

We remark that we may obtain more independent small vectors. Take any m coordinates out of n and apply a row permutation to the lattice, the specific m coordinates of which are located in the last coordinates. By applying the HNF and LLL algorithms and inverting the permutation, we can obtain one short vector in the lattice, the specific $(n-m)$ coordinates of which are zero. We can repeat this procedure for each $\binom{n}{m}$ permutation. If the output vectors are randomly distributed on the set of short vectors of L, it is probable that we can find n linearly independent small lattice vectors by repeating this procedure polynomially many times. However, we do not know how to obtain a full-rank set of independent vectors efficiently, which would be an interesting problem.

4.2 Application to CVP

In this section, as an application of short independent vectors, we describe an improved Babai's nearest plane algorithm. Babai's nearest algorithm enables us to find a lattice vector $\boldsymbol{v}\in L$ close to $\boldsymbol{w}$, inductively. It is an approximate algorithm to give a lattice vector $\boldsymbol{v}$, the distance of which to target vector $\boldsymbol{w}$ is bounded by an exponential size in n. Our goal is to reduce the size of the bound using **Algorithm 2**.

Given a lattice L and a vector $\boldsymbol{w}$ in $\mathbb{R}^n$, we define

$$\operatorname{dist}(\boldsymbol{w},L) := \min_{\boldsymbol{u}\in L} ||\boldsymbol{w}-\boldsymbol{u}||.$$

Theorem 4. *Let $\boldsymbol{B} = [\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_n]$ be a basis matrix of an integral lattice L with $\det(L) < 2^{n^2/4}$. On input $\boldsymbol{B}$ and $\boldsymbol{w} \in \mathbb{R}^n$, one can find a lattice vector $\boldsymbol{v}$ such that*

$$\|\boldsymbol{v} - \boldsymbol{w}\| \leq 2^{\sqrt{\log(\det L)}+1/2} \sqrt{(n - \lfloor 2\sqrt{\log\det L}\rceil) + \operatorname{dist}(\boldsymbol{w}, L)^2}$$

in polynomial time in n and $\det(L)$.

Proof. Given a basis B of an integral lattice L, we take $m = \lfloor 2\sqrt{\log\det L}\rceil$ and apply **Algorithm 2** to obtain an independent set $B' = \{\boldsymbol{b}'_1, \cdots, \boldsymbol{b}'_n\}$. We recall that the first $(n-m+1)$ vectors have bounded size and the last m vectors are LLL-reduced and can be extended to a basis of L. We denote by L_1 the lattices generated by the first $(n-m)$ vectors. Consider the orthogonal projection maps P and $P^\perp$ on $\mathbb{R}^n$:

$$P : \mathbb{R}^n \to \langle \boldsymbol{b}'_1, \cdots \boldsymbol{b}'_{n-m}\rangle^\perp$$
$$P^\perp : \mathbb{R}^n \to \langle \boldsymbol{b}'_1, \cdots \boldsymbol{b}'_{n-m}\rangle,$$

where $\langle \boldsymbol{b}'_1, \cdots \boldsymbol{b}'_{n-m}\rangle^\perp$ is the orthogonal complement of the vector space $\langle \boldsymbol{b}'_1, \cdots \boldsymbol{b}'_{n-m}\rangle$ in $\mathbb{R}^n$. Then, an arbitrary vector $\boldsymbol{v} \in \mathbb{R}^n$ is represented by $P(\boldsymbol{v}) + P^\perp(\boldsymbol{v})$ and $\|\boldsymbol{v}\|^2 = \|P(\boldsymbol{v})\|^2 + \|P^\perp(\boldsymbol{v})\|^2$. Since, for some $c_j \in \mathbb{Q}$, $\boldsymbol{b}'_i = \sum\limits_{j=1}^{n} c_j \boldsymbol{b}_j$ for all i, one can see that $P(L)$ is also an m-dimensional lattice in $\mathbb{R}^n$.

First, we compute an LLL-reduced basis of $P(L)$ from $\{P(b_1), \cdots, P(b_n)\}$ and apply Babai's nearest plane algorithm on the basis and $P(\boldsymbol{w})$ to obtain a vector $\boldsymbol{v}_2 \in P(L)$ close to $P(\boldsymbol{w})$. Since $P(L)$ is an m-dimensional lattice in $\mathbb{R}^n$ and $\boldsymbol{v}_2 \in P(L)$, we have

$$\|\boldsymbol{v}_2 - P(\boldsymbol{w})\| \leq 2^{m/2} \operatorname{dist}(P(\boldsymbol{w}), P(L)).$$

Moreover, we have $\operatorname{dist}(P(\boldsymbol{w}), P(L)) \leq \operatorname{dist}(\boldsymbol{w}, L)$, since

$$\operatorname{dist}(P(\boldsymbol{w}), P(L)) \leq \operatorname{dist}(P(\boldsymbol{w}), P(\boldsymbol{u})) \leq \operatorname{dist}(\boldsymbol{w}, \boldsymbol{u})$$

for any vector $\boldsymbol{u} \in L$.

We write $\boldsymbol{v}_2 = \sum_{i=1}^n c_i P(\boldsymbol{b}_i)$ for some integer c_i's and take $\boldsymbol{v}_1 = \sum_{i=1}^n c_i \boldsymbol{b}_i \in L$. Then, we take $\boldsymbol{v}_3 \in L_1$ close to $P(\boldsymbol{w} - \boldsymbol{v}_1) \in L_1$ by Babai's nearest plane algorithm on the basis $\{\boldsymbol{b}'_1, \ldots, \boldsymbol{b}'_{n-m}\}$ and $P(\boldsymbol{w} - \boldsymbol{v}_1)$. Then, we have

$$\|\boldsymbol{v}_3 - P^\perp(\boldsymbol{w} - \boldsymbol{v}_1)\|^2 \leq \frac{1}{4} \sum_{i=1}^{n-m} \|\boldsymbol{b}'_i\|^2 \leq (n-m) \cdot 2^{2\sqrt{\log\det L} - 31/16},$$

where the last inequality comes from Theorem 2.

Finally, we take $\boldsymbol{v} := \boldsymbol{v}_1 + \boldsymbol{v}_3$ for $\boldsymbol{v}_1 \in L$ and $\boldsymbol{v}_3 \in L_1$. It belongs to L and

$$\begin{aligned}\|\boldsymbol{v} - \boldsymbol{w}\|^2 &= \|P(\boldsymbol{v} - \boldsymbol{w})\|^2 + \|P^\perp(\boldsymbol{v} - \boldsymbol{w})\|^2 \\ &= \|\boldsymbol{v}_2 - P(\boldsymbol{w})\|^2 + \|\boldsymbol{v}_3 - P^\perp(\boldsymbol{w} - \boldsymbol{v}_1)\|^2 \\ &\leq 2^m \operatorname{dist}(\boldsymbol{w}, L)^2 + (n-m) \cdot 2^{2\sqrt{\log\det L} - 31/16}.\end{aligned}$$

The running time of this algorithm is bounded by that of **Algorithm 2** and is twice that of Babai's nearest plane algorithm, and therefore, results in polynomial time in n and $\det(L)$. ∎

We present an algorithm described in **Theorem 4**.

Algorithm 3 Closest Vector with Babai+HNF

```
Input: $\{\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n\}$ and $\boldsymbol{w}$
Output: $\boldsymbol{v}$
1.  Compute $m = \left\lfloor 2\sqrt{\log(\det[\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n])} \right\rceil$
2.  Set $(\boldsymbol{b}'_1, \ldots, \boldsymbol{b}_n)$ as the output of Algorithm 2$(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n, m)$
3.  Compute $\boldsymbol{w}' = P(\boldsymbol{w})$
4.  for $i = 1$ upto $n$ do
5.      Set $\boldsymbol{b}''_i$ as the orthogonal projection of the vector $\boldsymbol{b}_i$ onto vector space $\langle \boldsymbol{b}'_1, \ldots, \boldsymbol{b}'_{n-m} \rangle^{\perp}$
6.  end for
7.  Set $(\boldsymbol{b}'''_1, \ldots, \boldsymbol{b}'''_n)$ as the output of LLL algorithm$(\boldsymbol{b}''_1, \ldots, \boldsymbol{b}''_n)$
8.  Set $\boldsymbol{v}_2$ as the output of Babai's nearest plane algorithm on input $(\boldsymbol{b}'''_1, \ldots, \boldsymbol{b}'''_n)$ and $\boldsymbol{w}'$
9.  for $i = 1$ upto $n$ do
10.     Set $c_i$ as the coefficient of the $\boldsymbol{b}''_i$ of the $\boldsymbol{v}_2$
11. end for
12. Set $\boldsymbol{v}_1 = c_1\boldsymbol{b}_1 + \cdots + c_n\boldsymbol{b}_n$
13. Compute $\boldsymbol{w}'' = P(\boldsymbol{w} - \boldsymbol{v}_1)$
14. Set $\boldsymbol{v}_3$ as the output of Babai's nearest plane algorithm on input $(\boldsymbol{b}'_1, \ldots, \boldsymbol{b}'_{n-m})$ and $\boldsymbol{w}''$
return $\boldsymbol{v}_1 + \boldsymbol{v}_3$
```

Consider an n-dimensional integral lattice L with determinant $2^{\beta n}$. By Gaussian heuristics [14, Sec. 6.5.3], the average distance of a random point in $\mathbb{R}^n$ to the closest lattice point is $\sqrt{\frac{n}{2\pi e}}(\det L)^{1/n}$. For a random point, our algorithm gives a lattice point, the distance of which to the target vector is bounded by approximately $2^{\sqrt{\beta n}}\sqrt{n(1+2^{\beta})}$, while it is $2^{n/2} \cdot \sqrt{n}2^{\beta}$ in the original Babai's nearest plane algorithm. Our algorithm has asymptotically smaller approximate factors when β is constant.

5 Conclusion and Open Problems

In this paper, we proposed three approximate lattice algorithms for SVP, SIVP, and CVP. Each of our algorithm runs in polynomial time of the dimension and determinant of the input lattice. When we apply them to an n-dimensional integral lattice with determinant $< 2^{n^2/4}$, the Euclidean norm of the output is bounded by $O(2^{\sqrt{\log \det L}})$, which is asymptotically smaller than $O(2^{n/4})$, that of LLL short vectors.

The sublattice generated by the last columns of the basis matrix in the Hermite normal form of an integral lattice has a determinant that is smaller than that of the original lattice. In this paper, we assumed only that the former is not less than the latter; however, we can obtain a more improved result if it is much smaller. It would be interesting to observe the behavior of the determinants of the sublattices from our algorithm for random lattices. Furthermore, our work raises a question as to how a sublattice with a smaller determinant can be found, which leads to a different direction of lattice reduction algorithms.

References

1. D. Aggarwal, D. Dadush, O. Regev, and N. Stephens-Davidowitz. Solving the shortest vector problem in 2^n time via discrete gaussian sampling. *arXiv preprint arXiv:1412.7994*, 2014.
2. M. Ajtai, R. Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Proc. of STOC*, pages 601–610. ACM, 2001.
3. M. Ajtai, R. Kumar, and D. Sivakumar. Sampling short lattice vectors and the closest lattice vector problem. In *Computational Complexity, 2002. Proceedings. 17th IEEE Annual Conference on*, pages 41–45. IEEE, 2002.
4. L. Babai. On Lovász lattice reduction and the nearest lattice point problem. *Combinatorica*, 6:1–13, 1986.
5. J. Blömer and S. Naewe. Sampling methods for shortest vectors, closest vectors and successive minima. In *Automata, Languages and Programming*, pages 65–77. Springer, 2007.
6. J. Bömer. Closest vectors, successive minima and dual-HKZ bases of lattices. In *Proceedings of the 2000 International Colloquium on Automata, Languages and Programming (ICALP 2000)*, volume 1853 of *LNCS*, pages 248–259. Springer, 2000.
7. D. Dadush, O. Regev, and N. Stephens-Davidowitz. On the closest vector problem with a distance guarantee. In *Computational Complexity (CCC), 2014 IEEE 29th Conference on*, pages 98–109. IEEE, 2014.
8. S. D. Galbraith. *Mathematics of public key cryptography.* Cambridge University Press, 2012.
9. N. Gama, N. Howgrave-Graham, H. Koy, and P. Q. Nguyen. Rankin's Constant and Blockwise Lattice Reduction. In *Advances in Cryptology – Proceedings of CRYPTO '06*, volume 4117 of *LNCS*, pages 112–130. Springer, 2006.
10. N. Gama and P. Q. Nguyen. Finding short lattice vectors within mordell's inequality. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 207–216. ACM, 2008.
11. S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices. In *Proc. of EUROCRYPT*, volume 7881 of *LNCS*, pages 1–17. Springer, 2013.
12. G. Hanrot, X. Pujol, and D. Stehlé. Terminating bkz. *IACR Cryptology ePrint Archive*, 2011:198, 2011.
13. B. Helfrich. Algorithms to construct Minkowski reduced and Hermite reduced bases. *Theoretical Computer Science*, 41:125–139, 1985.
14. J. Hoffstein, J. C. Pipher, J. H. Silverman, and J. H. Silverman. *An introduction to mathematical cryptography.* Springer, 2008.
15. R. Kannan. Improved algorithms for integer programming and related lattice problems. In *Proc. of the 15th Symposium on the Theory of Computing*, pages 193–206. ACM, 1983.
16. R. Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, 1987.
17. H. Koy and C. P. Schnorr. Segment LLL-reduction of lattice bases. In *Proceedings of the 2001 Cryptography and Lattices Conference (CALC'01)*, volume 2146 of *LNCS*, pages 67–80. Springer, 2001.
18. T. Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. Technical report, Cryptology ePrint Archive, Report 2014/744, 2014.
19. A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.
20. D. Micciancio and P. Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. In *Proc. of STOC*, pages 351–358. ACM, 2010.
21. D. Micciancio and P. Voulgaris. Faster exponential time algorithms for the shortest vector problem. In *Proc. of SODA*. ACM, 2010.
22. D. Micciancio and B. Warinschi. A linear space algorithm for computing the Hermite normal form. In *Proceedings of ISSAC*, pages 231–236. ACM, 2001.
23. P. Q. Nguyen and D. Stehlé. Floating-point LLL revisited. In *Proceedings of Eurocrypt 2005*, volume 3494 of *LNCS*, pages 215–233. Springer, 2005.
24. C. P. Schnorr. A more efficient algorithm for lattice basis reduction. *Journal of Algorithms*, 9(1):47–62, 1988.
25. C. P. Schnorr. Fast LLL-type lattice reduction. *Information and Computation*, 204:1–25, 2006.
26. C. P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. *Mathematics of Programming*, 66:181–199, 1994.