Techniques for Solving Shortest Vector Problem

V. Dinesh Reddy[1], P. Ravi[2], Ashu Abdul[3], Mahesh Kumar Morampudi[4], Sriramulu Bojjagani[5]
School of Engineering and Sciences, Department of CSE
SRM University, Amaravati, AP, India[1,3,4,5]
Assistant Professor, CSE,
Anurag University, Hyderabad, Telangana, India[2]

***Abstract*—Lattice-based crypto systems are regarded as secure and believed to be secure even against quantum computers. lattice-based cryptography relies upon problems like the Shortest Vector Problem. Shortest Vector Problem is an instance of lattice problems that are used as a basis for secure cryptographic schemes. For more than 30 years now, the Shortest Vector Problem has been at the heart of a thriving research field and finding a new efficient algorithm turned out to be out of reach. This problem has a great many applications such as optimization, communication theory, cryptography, etc. This paper introduces the Shortest Vector Problem and other related problems such as the Closest Vector Problem. We present the average case and worst case hardness results for the Shortest Vector Problem. Further this work explore efficient algorithms solving the Shortest Vector Problem and present their efficiency. More precisely, this paper presents four algorithms: the Lenstra-Lenstra-Lovasz (LLL) algorithm, the Block Korkine-Zolotarev (BKZ) algorithm, a Metropolis algorithm, and a convex relaxation of SVP. The experimental results on various lattices show that the Metropolis algorithm works better than other algorithms with varying sizes of lattices.**

Keywords*—*Lattice; SVP; CVP; post quantum cryptography

I. Introduction

A lattice an abstract structure defined as the set of all integer linear combinations of some independent vectors in R_n. Over the last two centuries, mathematicians have explored the fascinating combinatorial structure of lattices, and it has also been studied from an asymptotic algorithmic viewpoint for at least three decades. Most fundamental problems are not considered to be effectively solvable on lattices. In addition, hardness results suggest that such problems can not be solved by polynomial-time algorithms unless the hierarchy of polynomial-time collapses. Cryptographic constructions based on lattice hold a clear promise for cryptography with strong security proof, as was demonstrated by Ajtai [1], who came up with a construction of cryptographic primitives based on worst-case hardness of certain lattice problems.

Two main important and very closely related hard computational problems used by cryptographers are the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP). In the former, for a lattice specified by some basis we are supposed to find nontrivial and small nonzero vector (length of vector) in the lattice. The problem CVP is an inhomogeneous variant of SVP, in which given a lattice (specified by some basis) and a vector v, one has to find the vector in L closest to v. The hardness of these problems is partially due to the fact that multiple bases can generate the same lattice. This work presents the best hardness result known for SVP, compares different algorithms solving SVP with respect to their efficiency and optimal solution.

II. Preliminaries

A. Definitions and Related Problems

Let $(b_1, \ldots, b_n)$ be a basis in $\mathbb{R}^n$ and N a norm on $\mathbb{R}^n$. Let L denote $\bigoplus_{i=1}^n \mathbb{Z}b_i, L$ is called a lattice. Set

$$\lambda(L) = \min_{v \in L\backslash\{0\}} N(v)$$

We are now able to define SVP. For L a lattice, the exact form Shortest Vector Problem, denoted by $SVP(L)$ is as follows.

$$SVP(L): \text{ Find } v \in L \text{ such that } N(v) = \lambda(L)$$

As it is often the case, we will actually be interested in algorithms solving an approximation of SVP. For $\gamma \geqslant 1$ the approximated form of $SVP(L)$, denoted by $SVP_\gamma(L)$ is

$$SVP_\gamma(L): \text{ Find } v \in L\backslash\{0\} \text{ such that } N(v) \leqslant \gamma\lambda(L)$$

Now, a closely related problem is the Closest Vector Problem $CVP(L, x)$, defined for a lattice $L \subset \mathbb{R}^n$ and $x \in \mathbb{R}^n$ as

$$CVP(L): \ Find\ v \ \in L \text{ such that } N(v - x) = \lambda(L)$$

In its exact form. As in the above we can define an approximated problem CVP_γ

$$CVP_\gamma(L): \ Find\ v \in \ L\ such\ that\ N(v - x) \leqslant \gamma\lambda(L)$$

In fact, CVP_γ solves SVP_γ. Indeed, for $i = 1, \ldots, n$ set $L^i = \mathbb{Z}b_1 \oplus \ldots \oplus 2b_i \oplus \ldots \oplus b_n$ and x_i a solution to $CVP_\gamma\left(L^i, b_i\right)$, then one of the vectors in $\{x_i - b_i\}_{i=1\ldots n}$ is a solution to $SVP_\gamma(L)$. For we have the following claim:

Claim 1. Let x_i be a solution to $CVP\left(L^i, b_i\right)$ then $\{x_i - b_i\}_{i=1,\ldots,n}$ contains a solution to $SVP(L)$
Proof. Indeed, let $x = \sum n_i b_i$ be any solution to $SVP(L)$, as $N\left(\frac{x}{2}\right) < N(x)$ we get that $\frac{x}{2} \notin L$ and there is $i \in \{1, \ldots, n\}$ such that n_i is odd. Then, $x + b_i \in L^i$ Therefore, the set $\{x_i - b_i\}_{i=1,\ldots,n}$ contains a element of norm less than or equal to $N(x)$.

When N is the l^2 norm on $\mathbb{R}^n, CVP$ has another formulation that allows us to use convex optimization techniques. To this end, let B be the matrix given by the basis $(b_1, \ldots, b_n)$ and c a vector in $\mathbb{R}^n$. Then, $CVP(L, c)$ is equivalent to

$$\begin{array}{ll} \text{minimize} & x^T B^T B x - 2c^T x \\ \text{subject to} & x \in \mathbb{Z}^n \end{array}$$

This is readily seen by expanding out the quantity $\|Bx - c\|_2$. Following (ref), we will apply a convex relaxation to this problem to get an efficient randomized algorithm solving CVP_γ.

Finally, a last related problem is the Hermite Shortest Vector problem denoted by $HSVP$. Again, let $B \in \mathbb{R}^{n\times n}$ be the matrix given by $(b_1, \ldots, b_n)$, for $\gamma \geqslant 1$, $HSVP_\gamma(L)$ is the following problem.

$$HSVP_\gamma(L): \text{ Find } v \in L \text{ such that } N(v) \leqslant \gamma|\det(B)|$$

According to a theorem of Minkowski's, HSVP and SVP are in fact closely related. The LLL algorithm described in Section III-A actually solves $HSVP_\gamma$.

B. Hardness: Average and Worst Case

An interesting feature of lattice problem is there hardness. M.Ajtai in his seminal papers [1], [2] showed in particular that worst-case hardness is related to average-case hardness and that SVP is NP-hard for randomized reductions. This makes lattice problems particularly interesting as basis for crypto systems. Example of such systems are systems of Ajtai and Dwork [3], Goldreich, Goldwasser and Halevi [4], the NTRU cryptosystem [5], and the first fully homomorphic encryption scheme by Gentry [6]. In this section we state a number of hardness results without proofs. For a more precise treatment see also [7]. The main result of worst-case to average case NP-hardness is as follows.

Theorem 1 Suppose there is a randomized polynomial time algorithm $\mathcal{A}$ such that for all $n, \mathcal{A}$ returns a vector of $\Lambda(X)$ of length $\leqslant n$ with probability $\frac{1}{n^{O(1)}}$ for $\Lambda(X)$ chosen at random in $\Lambda_{n,m,q}$ where $m = \alpha n \log(n)$ and $q = n^\beta$ for appropriate α, β. Then, there exist a randomized polynomial time algorithm $\mathcal{B}$ and constants c_1, c_2, c_3 such that for all $n, (b_1, \ldots, b_n)$ basis of $\mathbb{R}^n$ and $L := \oplus\mathbb{Z}b_i, \mathcal{B}$ performs the following with high probability [3]:
1) Finds a basis $\{\beta_1, \ldots, \beta_n\}$ for L such that

$$\max\|\beta_i\| \leqslant n^{c_1} \min_{(c_i)\ basis\ of\ L} \max\|c_i\|\,;$$

2) finds an estimate $\bar{\lambda}$ of $\lambda(L)$ such that,

$$\frac{\lambda_1(L)}{n^{c_2}} \leqslant \tilde{\lambda} \leqslant \lambda(L)$$

3) If moreover, L has a n^{c_3} unique shortest vector, finds this vector.

We are now oing to explain the content of this theorem. First of all, we must explain a number of concepts and notation.

Here, $\|.\|$ refers to the l^2 -norm.

A randomized algorithm is an algorithm that uses a degree of randomness as part of its process. As such, it does not guarantee success, but the probability of success for a given input size can be computed. A randomized algorithm is said to perform an event with high probability if the probability of this event goes to 1 as the input size goes to $+\infty$

We turn now to the definition of $\Lambda(X)$. Let $n, m, q \in \mathbb{N}, (\mathbb{Z}/q\mathbb{Z})^{n\times m}$ the space of matrices with entries in $(Z/q\mathbb{Z})$, and $\Omega_{n,m,q}$ the uniform distribution on $(\mathbb{Z}/q\mathbb{Z})^{n\times m}$ Now, for any $X \in (\mathbb{Z}/q\mathbb{Z})^{n\times m}$, set $\Lambda(X)$ as the lattice $\{y \in \mathbb{Z}^m \mid Xy \equiv 0[q]\}$ and $\Lambda_{n,m,q}$ the space of such lattices. We see $\Lambda_{n,m,q}$ as a probability space by choosing X according to $\Omega_{n,m,q}$ and computing $\Lambda(X)$. For any $X, \Lambda(X)$ is indeed a lattice, as $q(\mathbb{Z}^m) \subset \Lambda(X)$.

Finally, for $c \geqslant 1$ a lattice L endowed with a norm N is said to have a c-unique shortest vector v if the following holds

$$\{v \in L \mid N(v) \leqslant c\lambda(L)\} = \{v, -v\}$$

This result is called a worst-case to average case hardness because it links the lack of a polynomial ranodmized algorithm over 'all' lattices to the lack of such algorithm for a small class of lattices, namely $\Lambda_{n,m,q}$. We also note that similar results exist for related problems such as CVP. In the following, we list further hardness results.

Theorem 2 SVP is NP -hard under the l_∞ -norm. Moreover, CVP is NP-hard under the l_p -norm for all $p \geqslant 1$ [8]..

Theorem 3 For all γ, CVP_γ is NP-hard under any l_p -norm [9].

Building on results by Ajtai [2], Micciancio proved the following.

Theorem 4 For all $\epsilon > 0, SVP_{\sqrt{2}-\epsilon}$ is NP -hard under randomized polynomial time reductions [10].

Finally:

Theorem 5 There is $c > 0$ such that CVP is NP-hard to approximate within a factor $n^{\frac{c}{log(log(n))}}$, where n is the dimension of the lattice [11].

Many other results related to hardness exist. We only decided to mention some of the most fundamental ones.

III. Solving SVP

This section presents solving SVP and related results. Even though many hardness results have been found, we can still solve approximate SVP within some reasonable constant depending on the input size. Usually, the 'reasonable constant' is not known theoretically to be 'reasonable' but only empirically. For instance, the first algorithm on this list, LLL, is only known to solve SVP within a factor exponential in the input size. However, results with LLL are in practice much better than the bound.

A. Lenstra-Lenstra-Lováz Algorithm

This section presents the Lenstra-Lenstra-Lovász reduction basis algorithm, by Lenstra, Lenstra and Lovász, see [12]. Originally, the purpose of this algorithm is to factor polynomials, we will not consider this problem here.

Moreover, it is important to note that the techniques used were developed by Hermite, Minkowksy and others to study Siegel sets and lattices in algebraic groups. This is why LLL is in fact an algorithm solving $HSVP$. $HSVP$ and SVP are related thanks to the following results due to Minkowski.

Theorem 6 (Minkowski). Let $n \in \mathbb{N}$ and L be any lattice in $\mathbb{R}^n$, then

$$\lambda(L) \leqslant \gamma_n(\det(L))^{\frac{1}{n}}$$

where γ_n is some constant depending only on n. More precisely, γ_n is known to satisfy

$$\sqrt{\frac{n}{2\pi e}} \leqslant \gamma_n \leqslant \sqrt{\frac{n}{\pi e}}$$

This theorem admits a short and elementary proof but can also be seen as a consequence of Minkowski's convex body theorem. For proofs and further details see [13] and [14]. In particular, we get the following important fact.

Claim 2. For $\gamma \geqslant 1, HSVP_{\gamma_n\gamma}(L) \Rightarrow SVP_\gamma(L)$

The Gram-Schmidt orthogonalization method is critical in the LLL algorithm, we briefly recall what this method entails. Let $(V, \langle\cdot,\cdot\rangle)$ be a vector space endowed with a scalar product and $\|\cdot\|$ the l_2 -norm on V given by this scalar product. For any basis $(b_1, \ldots, b_n)$ we define inductively its orthogonalization $(b_1^*, \ldots, b_n^*)$ as

$$b_i^* = b_i - \sum_{j<i} \langle b_i, b_j^* \rangle b_j^*$$

This new basis $(b_1^*, \ldots, b_n^*)$ is orthogonal i.e. $\langle b_i^*, b_j^* \rangle = 0$ whenever $i \neq j$. More-over, $\operatorname{span}(b_1, \ldots, b_r) = \operatorname{span}(b_1^*, \ldots, b_r^*)$ for all $1 \leqslant r \leqslant n$. In the following, we will pay particular attention to the Gram-schmidt coefficients $\mu_{i,j} := \frac{\langle b_i, b_j \rangle}{\|b_j^*\|^2}$ for $i > j$. Moreover, let B^* be the matrix with columns the vectors $(b_1^*, \ldots, b_n^*)$ and L the lattice generated by $(b_1, \ldots, b_n)$. Then, it is readily seen that vol $(L) = \|b_1^*\| \cdot \ldots \cdot \|b_n^*\|$. (For vol $(L) = |\det(B^*)|$ as for all i we have $\operatorname{span}(b_1, \ldots, b_r) = \operatorname{span}(b_1^*, \ldots, b_r^*)$, but $\operatorname{diag}\left(\|b_1^*\|^{-1}, \ldots, \|b_n^*\|^{-1}\right) B^* \in O_n(\mathrm{R})$ so $\det(B^*) = \|b_1^*\| \cdot \ldots \cdot \|b_n^*\|$.) Now, assume that $\frac{\|b_i\|}{\|b_{i-1}\|}$ is bounded below by some constant $c > 0$. As $b_1 = b_1^*$, we get that

$$\|b_1\|^n \leqslant c^{-n(n-1)/2} \|b_1^*\| \cdot \ldots \cdot \|b_n^*\| = c^{-n(n-1)/2} \operatorname{vol}(L)$$

So b_1 is a solution to $HSVP_{c-(n-1)/2}(L)$.
In order to make use of this observation, we define the following condition.

Definition 1 (Lovász' condition). A basis $(b_1, \ldots, b_n)$ satisfies Lovasz' condition if there is $\delta \in \left(\frac{1}{4}, 1\right]$ such that for all $2 \leqslant i \leqslant n$

$$\frac{\|b_i\|^2}{\|b_{i-1}\|^2} \leqslant \delta - \mu(i, i-1)^2$$

Where $(b_1^*, \ldots, b_n^*)$ is the Gram-Schmidt orthogonalization of $(b_1, \ldots, b_n)$.

For the sake of simplicity, we removed the necessary parts that consist in updating the Gram-Schmidt orthogonalization

```
initialization;
Require: a basis $(b_1, \ldots, b_n)$ of $L$ and $\delta \in \left(\frac{1}{4}; 1\right]$
Ensure: the output basis $(b_1, \ldots, b_n)$ of $L$ satisfies
 Lovász' condition
$i \leftarrow 2$
while $i \leqslant n$ do
    $b_i \leftarrow b_i - \sum_{j<i} \lfloor \mu_{i,j} \rceil b_j^*$
    if $\|b_i^*\|^2 \geqslant \left(\delta - \mu_{i,i-1}^2\right) \|b_{i-1}\|^2$ then
        $i \leftarrow i + 1$
    else
        $swap\ b_i,\ b_{i-1}$;
        $i \leftarrow \max(2, i-1)$;
    end
end
```
Algorithm 1: LLL basis reduction algorithm

and Gram-Schmidt coefficients. Therefore, the previous algorithm is only a sketch of the Lenstra-Lenstra-Lovász reduction algorithm. Let us now state the main result about this algorithm, namely its complexity and the properties of the output basis.

Theorem 7 Let $(b_1, ..., b_n)$ be a basis generating a lattice L and $\delta \in \left(\frac{1}{4}, 1\right]$. Given these as input, the LLL algorithm terminates in poly $\left(d, (1-\delta)^{-1}, \log\left(\max\|b_i\|\right)\right)$ and the output is a basis $(\beta_1, \ldots, \beta_n)$ generating L such that:

$$\frac{\|\beta_1\|}{\operatorname{vol}(L)^{1/n}} \leqslant \left(\frac{1}{\delta - \frac{1}{4}}\right)^{(d-1)/4}$$

In particular, for $\delta = 1 - \epsilon$ with e small, the LLL algorithm terminates in $poly\left(d, \frac{1}{\epsilon}, \log\left(\max\|b_i\|\right)\right)$ time and the output satisfies:

$$\frac{\|\beta_1\|}{\operatorname{vol}(L)^{1/n}} \leqslant \left(\frac{4}{3} + O(\epsilon)\right)^{(d-1)/4} \simeq 1.07^{d-1}$$

Empirically, it has been observed that for large n, that $\frac{\|\beta_1\|}{\operatorname{vol}(L)^{1/6}}$ is around 1.02^n which is a considerable improvement [15].

This result is about how well LLL solves the Hermite Shortest Vector Problem [12]. According to a remark made earlier, we obtain the following result concerning our main interest, the Shortest Vector Problem.

Corollary 1. With the same input as above, the output $(\beta_1, \ldots, \beta_n)$ satisfies:

$$\frac{\|\beta_1\|}{\operatorname{vol}(L)^{1/n}} \leqslant \left(\frac{1}{\delta - \frac{1}{4}}\right)^{(d-1)/2}$$

Or for $\epsilon > 0$ small and $\delta = 1 - \epsilon$ we get that:

$$\frac{\|\beta_1\|}{\operatorname{vol}(L)^{1/n}} \leqslant \left(\frac{4}{3} + O(\epsilon)\right)^{(d-1)/2} \approx 1.15^{d-1}$$

More precisely, we can get the following complexity.

Theorem 8 (Lenstra-Lenstra-Lovaisz, 1982, [L.LL 82]). The LLL algorithm has complexity $O\left(n^6 \log\max\|b_i\|\right)$.

Now, we briefly mention an improvement of LLL due to Schnorr and Euchner, see [16]. In the previous algorithm, when the swap function is applied, instead of swapping b_i and b_{i-1} we apply what Schnorr and Euchner call deep insertions. This entails to inserting b_i at the index j where j is the smallest index satisfying:

$$\delta \left\|b_j^*\right\| \geqslant \|b_i\|$$

The result of this change is shorter outputs but longer running time.

B. Block Korkine-Zolotarev Algorithm

The Block korkine-Zootarev (BKZ) algorithm is another algorithm solving SVP_γ. Even though it also runs in polynomial time, its purpose is to achieve better accuracy than LLL, therefore has longer running time. After breaking down how LLL algorithm works, we could see that it uses an SVP-oracle for lattices in dimension 2 called Gauss's reduction. Given more efficient oracles or higher dimensional SVP -oracles we could hope for a more efficient algorithm. This algorithm was proposed by Schnorr and Euchner in their paper [17].

Along this line of thought, given an SVP-oracle for all dimensions up to some integer k the BKZ algorithm finds a shorter vector than LLL does. Unsurprisingly, running BKZ takes more time than running LLL. Here, there is a clear tradeoff between the shortness of the output and the complexity of the algorithm. Moreover, as of today it is not known theoretically if BKZ terminates in polynomial time, only empirical results are known.

Here, an SVP-oracle is a mean by which we are able to solve the exact SVP. Many techniques were developed to solve SVP, but as one could guess from NP-hardness results, all these methods have at least exponential time complexity. Therefore, it is not deemed feasible to run these algorithms in high dimension. Among such oracles are enumeration algorihtms which are some kind of greedy algorithms that dates back to Pohst [18], sieving techniques which are applications of Monte-carlo methods (just as the next section is) due to Micciancio and Voulgaris in [19], and a method based on Voronoi cells due to the same authors in [20]. Even though these methods have exponential complexity, they can still be used for lattices in reasonably large dimension.

Let us now turn to the exposition of BKZ algorithm. To this end, we first need to define the notion of Korkine-Zolotarev reduced basis and a KZ-reduction algorithm. Suppose we have a SVP -oracle $\mathcal{O}$ up to dimension k and, for any basis $(b_1,\ldots,b_k)$ generating a lattice L define $\pi_i : \mathbb{R}^k \to \operatorname{span}(b_1,\ldots,v_i)^\perp$ as the orthogonal projection and $L_i = \pi_{i-1}(L)$. A basis $(b_1,\ldots,b_n)$ is called KZ-reduced if

$$||b_i^*|| = \lambda(L_i)$$

where $(b_1^*,\ldots,b_n^*)$ is the Gram-Schmidt reduction of $(b_1,\ldots,b_n)$. (Note that $\pi_i(b_i) = b_i^*$.) As for all i there is $\alpha_i \in [-1/2; 1/2]$ such that $\alpha_i b_i^\infty + b_{i+1}^* \in L_i$ we get by an induction argument the following theorem.

Theorem 9. Let $(b_1,\ldots,b_n)$ be a KZ -reduced basis. Then,

$$\|b_1\| = \lambda(L) \text{ and } \frac{\|b_1\|}{\|b_k\|} \leqslant k^{(1+\log(k))/2}$$

Moreover, we have the following KZ-reduction algorithm.

```
Result: Write here the result
initialization;
Require: a basis $B = (b_1,\ldots,b_n)$ for $L$ and a $SVP$ -oracle $\mathcal{O}$ for up to $k$ dimensions.
Ensure: The output basis is KZ-reduced.
$m \leftarrow c(R)$
for $i = 1$ to $k$ do
    call $\mathcal{O}$ to find $b_i^* \in \pi_i(L)$ of length $\lambda(\pi_i(L))$
    lift $b_i^*$ into $b_i \in L$ such that $\|b_i^* - b_i\|$ is minimal;
    change $(b_{i+1},\ldots,b_k)$ such that $(b_1,\ldots,b_k)$ generates $L$
end
```

Algorithm 2: KZ-reduction algorithm

Note that step 4 may look as hard as solving CVP but is actually a lot easier thanks to the properties of orthogonal projections. Now, BKZ is as follows.

Algorithm 3 Schnorr and Euchner's BKZ-reduction algorithm
Require: a basis $B = (b_1,\ldots,b_n)$ for $L, \mathrm{ak} \in \mathbb{N}, \delta \in \left(\frac{1}{4}, 1\right)$ and a SVP -oracle $\mathcal{O}$ for up to k dimensions.
Ensure: The output basis satisfies Lovisz' conditon with factor δ and $\|b_i^*\| = \lambda\left(\pi_{i-1}\left(\bigoplus_{j=1}^k b_{i+j-1}\right)\right)$ for i from 1 to $d-k+1$

```
initialization;
Require: a basis $B = (b_1,\ldots,b_n)$ for $L, \mathrm{ak} \in \mathbb{N}, \delta \in \left(\frac{1}{4}, 1\right)$ and a $SVP$ -oracle $\mathcal{O}$ for up to $k$ dimensions.
Ensure: The output basis satisfies Lovisz' conditon with factor $\delta$ and $\|b_i^*\| = \lambda\left(\pi_{i-1}\left(\bigoplus_{j=1}^k b_{i+j-1}\right)\right)$ for $i$ from 1 to $d-k+1$
repeat
    for $i = 1$ to $d-k+1$ do
        KZ-reduce the basis $\pi_{i-1}(b_i,\ldots,b_{i+k-1})$;
        Lift up to the closest vectors in $L$ and complete the basis.
    end
until no changes occur;
```

Algorithm 3: Schnorr and Euchner's BKZ-reduction algorithm

The theoretical complexity computation of this algorithm is not computed, but the following can be proved.

Theorem 10 Given a basis $(b_1,\ldots,b_n)$ generating L and an SVP oracle $\mathcal{O}$ for up to k dimensions, the BKZ algorithm outputs a basis $(\beta_1,\ldots,\beta_n)$ generating L that satisfies:

$$\frac{\|b_1\|}{\lambda(L)} \leqslant \left(k^{\frac{1+\log(k)}{2k-2}}\right)^{n-1} \text{ and } \frac{\|b_1\|}{\operatorname{vol}(L)^{1/d}} \leqslant \sqrt{\gamma_k}\left(k^{\frac{1+\log(\omega)}{2k-2}}\right)^{n-1}$$

C. Metropolis-Hasting's Algorithm

The next algorithm is a randomized algorithm applying the well-known Metropolis algorithm, a kind of Monte-Carlo method, to lattice problems. For a precise account on this method see for instance [21]. The following Metropolis algorithm is due to Ajitha, Biswas, and Kurur in [22]. This algorithm returns an approximate solution with respect to the euclidean norm, denoted by $\| \cdot \|$.

The search space is defined as follows.

Once again let $(b_1, \ldots, b_n)$ be a matrix of R^n, L the lattice it generates and B the matrix given by $(b_1, \ldots, b_n)$.

The main idea behind this Metropolis-Hastings is to work on a bounded set of vectors with integer entries and pair each of these vectors v to a cost $c(v) := \|Bv\|$. If $w \in L$ has norm less than or equal to k then $w = Bv$ where v has integer entries bounded above by $M = (\alpha n)^n$, where $\alpha = \max_{i,j}(|b_{ij}|, k)$. Thus, it is enough to look at vectors bounded above by M. As we want to modify the vector step by step, it is interesting to work with more than one vector at a time. Let us now define the Markov chain we will be working on.

Fix parameters $k \in \mathbb{R}, m \in \mathbb{N}$ and set M as above. The search space S consists of matrices $A' := [A \mid I]$ with integer entries bounded above by M where A is a $n \times m$ matrix and I is the $n \times n$ identity matrix. It remains only to describe the transition probability of the Markov chain. We first describe the neighbourhood of a given the state, the actual transition matrix will be described later on in the pseudo-code of the Metropolis-Hastings algorithm.

A matrix $S' = [S \mid I]$ is in the neighbourhood $N(R')$ of $R' = [R \mid I]$ if

(i) S is equal to R up to swapping two columns;
(ii) S is equal to R up to multiplying a column by -1
(iii) we can get from R' to S' by $r_i \leftarrow r_i \pm 2^l r_j$ for any $1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n+m, i \neq j$

Two interesting features of S are:

- For any $R' \in S, \#N(R') = O(m^2 \log(M))$
- For any $R', S' \in S$, there is a path between R' and S' of length $O(mn \log(M))$

Let us now turn to the pseudo-code and the actual description of the transition probability.

```
initialization;
Require: a basis $B = (b_1, \ldots, b_n)$ for $L$ and $K \in \mathbb{Q}$
Ensure: Matrix $R$ with integer entries such that $BR$
  has a column $c$ with $\|c\| \leqslant K$.
$m \leftarrow c(R)$
while $m > K$ do
    Select $S$ a neighbour of $R$ by performing one of
      the following operations.
        • Swap two columns with probability $\frac{C_2^m}{\#N(R)}$;
        • Multiply a column by -1 with probability $\frac{m}{d}$
        • Add $2^i$ times a column of $R$ to another column of $R$
          with probability $\frac{d - C_3^m - m}{d} \cdot p_i$
    if if $m > c(R)$ then
        $m \leftarrow c(R)$;
    end
    if $S \in S$ then
        $R \leftarrow S$ with probability $\min\left(\frac{e^{-c(S)/T}}{e^{-c(R)/T}}, 1\right)$;
    end
end
```
Algorithm 4: Metropolis-Hasting's algorithm for SVP

Here, (p_i) is a probability distribution on $\mathbb{N}$ such that there is $n_0 \in \mathbb{N}$ satisfying $p_i = 0$ for all $i \geqslant n_0$. This algorithm seems to give better results than LLL for lattices in low dimensional vector spaces. However, in higher dimension such methods will have to deal with the following geometric issue, the so-called *curse of dimensionality*. Let $R > \epsilon > 0$ then

$$\frac{\operatorname{vol}(B_{\mathbb{R}^n}(R))}{\operatorname{vol}(B_{\mathbb{R}^n}(R-\epsilon))} \to_{n \to +\infty} 0$$

Therefore, as n goes to $+\infty$ 'most' of the states of the markov chain built in the Metropolis algorithm will lie in the annulus $B_{\mathbb{R}^n}(R) \backslash B_{\mathbb{R}^n}(R-\epsilon)$ and the markov chain might take more and more time before hitting any point inside $B_{\mathbb{R}^n}(R-\epsilon)$. For more on related heuristics see [23].

Finally, note that a similar method is widely used to solve exact instances of SVP. The version of this algorithm used to solve SVP is called a sieve method and is often used as an SVP -oracle in the BKZ algorithm, see for instance the following paper by Micciancio and Voulgaris [19].

D. Convex Relaxation

This section presents the convex relaxation (see [24] for definitions and results on convex relaxation) to obtain another randomized algorithm to solve CVP_γ (thus, SVP_γ as well). As mentioned in the first part of this report, CVP_γ admits another formulation, more precisely $\mathrm{CVP}(L, c)$ is equivalent to

(1)
$$\begin{gathered}\text{minimize} \quad x^T B^T B x - 2c^T x \\ \textit{subject to } x \in \mathbb{Z}^n\end{gathered}$$

.
Where B be the matrix given by the basis $(b_1, \ldots, b_n)$ and c a vector in $\mathbb{R}^n$. A convex relaxation consists in relaxing the hard condition $x \in \mathbb{Z}^n$ to a looser condition so that the problem considered is now a convex optimization problem. Here, the method is due to Park and Boyd in [25]. The convex relaxation is in this particular case an instance of semidefinite programming, for more on this topic see [26]. We first relax Problem 1 into a non-convex problem

(2)
$$\begin{gathered}\text{minimize} \quad x^T B^T B x - 2c^T x \\ \text{subject to } x_i(x_i - 1) \geqslant 0, \forall i\end{gathered}$$

Now, set $P := B^T B$ we can reformulate Problem 2

$$minimize Tr(PX) - 2c^T x$$

(3) subject to $\operatorname{diag}(X) \geqslant x$

$$X = xx^T$$

Finally, relax condition $X = xx^T$ of Problem 3 to $X \geq xx^T$ which means that $X - xx^T$ is semidefinite positive. But $X \geq xx^T$ is equivalent to $\begin{bmatrix} X & x \\ x^T & 1 \end{bmatrix} \geq 0$. So we get the following relaxation.

(4)

$$\begin{array}{ll} \text{minimize} & \operatorname{Tr}(PX) - 2c^T x \\ \text{subject to} & \operatorname{diag}(X) \geqslant x \end{array}$$

$$\begin{bmatrix} X & x \\ x^T & 1 \end{bmatrix} \geq 0$$

Now, this is a semi definite relaxation that can be solved in polynomial time.

```
initialization;
**Require:** $P = B^T B, c \in \mathbb{R}^n, K \in \mathbb{N}$
Solve (4) to get $X^*$ and $x^*$
$\Sigma \leftarrow X^* - x^* x^{4T}$
Find Cholesky factorisation $LL^T = \Sigma$;
$x^{\text{best}} \leftarrow 0$;
$f^{\text{best}} \leftarrow 0$
**for** k=1,2, ..., K **do**
    $z^{(k)} \leftarrow x^* + Lw$ where $w \sim \mathcal{N}(0, I)$
    $x^{(k)} \leftarrow \text{round}\left(z^{(k)}\right)$
    **if** $f^{best} > f\left(x^{(k)}\right)$ **then**
        $x^{best} \leftarrow x^{(k)}$;
            $f^{best} \leftarrow f\left(x^{(k)}\right)$
    **end**
**end**
```

Algorithm 5: Randomized algorithm for suboptimal solution

Where the Cholesky factorisation is an algorithm computing the square root of a positive semidefinite matrix in $O\left(n^3\right)$ [27].

```
initialization;
**Require**: $x \in \mathbb{Z}^n, P = B^T B, c \in \mathbb{R}^n$.
$g \leftarrow 2(Px + c)$
**repeat**
    Find index $i$ and integer $a$ minimizing $a^2 P_{ii} + cg_i$;
    $x_i \leftarrow x_i + c_3$
    $g \leftarrow g + 2cP_i$
**until** $\operatorname{diag}(P) \geqslant |g|$;
```

Algorithm 6: Greedy descent algorithm

This last algorithm runs in polynomial time as well. Combining these two algorithms we get a approximate solution such that for all integer $a \in \mathbb{Z}$ and all index i, x is a better solution than $x + ae_i$

IV. Experimental Results

This section describes how the said approaches perform on benchmark instances. We see that the output of the LLL algorithm is near to the shortest vector which is $2^{\frac{n-1}{2}}$ of the shortest vector for the given lattice, it can be used for polynomial factoring etc. LLL algorithm complexity is $O\left(n^6 \cdot \log^3 \beta\right)$, where β is $\max_{1 \leq i \leq n} ||b_i||$. We ran tests to compare LLL algorithm and BKZ algorithm on data from [22], [21] and [24].

For experimental analysis a class of SVP instances are generated using the techniques developed by Richard Lindner and Michael Schneider [28]. They have given sample bases for Modular, Random, ntru, SWIFT and Dual Modular lattices of dimension 10. We have tested our code for all these instances and found that our algorithm works faster and gives shorter lattice vector when compared to LLL. The tested results are given in the Table I.

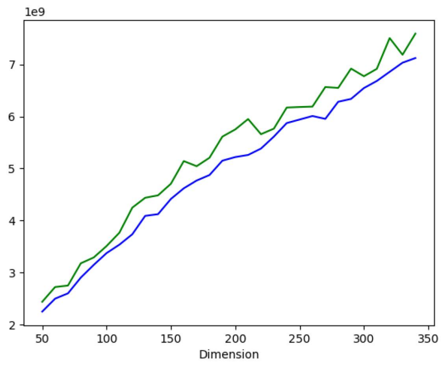

Fig. 1. Norm of the Output/ Dimension; Blue: BKZ / Green: LLL.

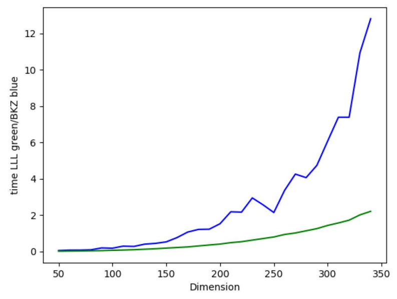

Fig. 2. Running time/ Dimension; Blue: BKZ / Green: LLL.

A. LLL/BKZ

Fig. 1 and Fig. 2 compare the basis reduction algorithms LLL and BKZ for a given family of Lattices parametrized by their dimension. Moreover, BKZ given an enumeration oracle for blocks of size 30. In particular, we see that with these parameters, the BKZ algorithm needs considerably more time to get a slightly shorter vector. To some extent, the LLL algorithm already seems to be efficient. A comparative results for running time of BKZ and LLL algorithms for varying dimension of lattice is given in Fig. 2.

TABLE I. Results of LLL and MH for Various Lattices

n	Input size	t (LLL: MH algo)	Norm (LLL: MH algo)
15	150	1234.58: 1147.2279	0.016: 201.651
20	8	3: 2.8284	0.2680: 0.052
25	8	1.73: 1.73	0.008: 0.004
30	8	4.123: 3.8729	0.008: 0.008
50	100	20.49: 8.66	0.108: 291.892

B. Metropolis-Hasting's Method

For a specific class of lattices [21], we have tested LLL and MH algorithms and the results are shown in Table I. Moreover, these data seem to support the curse of dimensionality heuristic as we can see the times needed to solve cases with large input size is high compared to small input size.

C. Convex Relaxation

The algorithm from [24] returns an approximate solution to CVP shown in Table II. To test their method, they computed solutions for instances randomly generated : all entries ar $\sim \mathcal{N}(0, 1)$ then normalized so that the solution to problem (1) without the integer constraint has value -1. In what follows, n will denote the dimension, opt the percentage of outputs that were optimal solutions and t the time it took to get this output. All values are averages.

TABLE II. Results for CVP for Random Instances

n	t	opt
50	0.397	90%
60	0.336	94%
70	0.402	89%

Because of the NP -hardness of CVP it is hard to compute the quantity opt in higher dimensions. However, the break down of running times of the method for larger $n's$ presented in Table III.

TABLE III. Running Times of the Convex Relaxation

n	t_{total}	SDP	Random sampling	Greeedy 1-opt
50	0.397	0.296	0.065	0.036
60	0.336	0.201	0.084	0.051
70	0.402	0.249	0.094	0.058
100	0.690	0.380	0.193	0.117
500	20.99	12.24	4.709	4.045
1000	135.1	82.38	28.64	24.07

In spite of the theoretical running time being $O\left(n^3\right)$ we can note that the total running time seems to grow subcubically.

V. Conclusion

In this paper we have discussed Shortest Vector Problem, Closest Vector Problem and their average case and worst case hardness results. Further, this work presented solving SVP using the LLL algorithm, BKZ algorithm, a Metropolis algorithm and a convex relaxation. We have compared the performance of these algorithms on various lattices by varying input sizes and the results we have obtained are fairly encouraging. The experimental results on various lattices shows that Metropolis algorithm works better than other algorithms.

Acknowledgment

The authors would like to thank the reviewers for their valuable comments.

References

[1] M. Ajtai, "Generating hard instances of lattice problems," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 99–108.

[2] ——, "The shortest vector problem in l2 is np-hard for randomized reductions," in *Proceedings of the 30th annual ACM symposium on Theory of computing*, 1998, pp. 10–19.

[3] M. Ajtai and C. Dwork, "A public-key cryptosystem with worst-case/average-case equivalence," in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, 1997, pp. 284–293.

[4] O. Goldreich, S. Goldwasser, and S. Halevi, "Public-key cryptosystems from lattice reduction problems," in *Proceedings of the Annual International Cryptology Conference*. Springer, 1997, pp. 112–131.

[5] J. Hoffstein, J. Pipher, and J. H. Silverman, "Ntru: A ring-based public key cryptosystem," in *Proceedings of the International Algorithmic Number Theory Symposium*. Springer, 1998, pp. 267–288.

[6] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 2009, pp. 169–178.

[7] J.-y. Cai, "The complexity of some lattice problems," in *Proceedings of the International Algorithmic Number Theory Symposium*. Springer, 2000, pp. 1–32.

[8] J. C. Lagarias, "The computational complexity of simultaneous diophantine approximation problems," *SIAM Journal on Computing*, vol. 14, no. 1, pp. 196–209, 1985.

[9] S. Arora, L. Babai, J. Stern, and Z. Sweedyk, "The hardness of approximate optima in lattices, codes, and systems of linear equations," *Journal of Computer and System Sciences*, vol. 54, no. 2, pp. 317–331, 1997.

[10] D. Micciancio, "The shortest vector in a lattice is hard to approximate to within some constant," *SIAM journal on Computing*, vol. 30, no. 6, pp. 2008–2035, 2001.

[11] I. Dinur, G. Kindler, R. Raz, and S. Safra, "Approximating cvp to within almost-polynomial factors is np-hard," *Combinatorica*, vol. 23, no. 2, pp. 205–243, 2003.

[12] A. K. Lenstra, H. W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Mathematische annalen*, vol. 261, pp. 515–534, 1982.

[13] H. Minkowski, "U rather the positive square forms and "u over chain break ä similar algorithms," *Journal f "u r pure and applied mathematics (Crelles Journal)*, vol. 1891, no. 107, pp. 278–297, 1891.

[14] R. Kannan, "Minkowski's convex body theorem and integer programming," *Mathematics of operations research*, vol. 12, no. 3, pp. 415–440, 1987.

[15] N. Gama and P. Q. Nguyen, "Predicting lattice reduction," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2008, pp. 31–51.

[16] C.-P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Mathematical programming*, vol. 66, no. 1, pp. 181–199, 1994.

[17] ——, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Mathematical programming*, vol. 66, no. 1-3, pp. 181–199, 1994.

[18] M. Pohst, "On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications," *ACM Sigsam Bulletin*, vol. 15, no. 1, pp. 37–44, 1981.

[19] D. Micciancio and P. Voulgaris, "Faster exponential time algorithms for the shortest vector problem," in *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*. SIAM, 2010, pp. 1468–1480.

[20] ——, "A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations," *SIAM Journal on Computing*, vol. 42, no. 3, pp. 1364–1391, 2013.

[21] C. P. Robert and G. Casella, "The metropolis—hastings algorithm," in *Monte Carlo statistical methods*. Springer, 2004, pp. 267–320.

[22] S. K. Ajitha, S. Biswas, and P. P. Kurur, "Metropolis algorithm for solving shortest lattice vector problem (svp)," in *Proceedings of 11th International Conference on Hybrid Intelligent Systems (HIS)*. IEEE, 2011, pp. 442–447.

[23] T. Carson, *Empirical and analytic approaches to understanding local search heuristics*. University of California, San Diego, 2001.

[24] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[25] J. Park and S. Boyd, "A semidefinite programming method for integer convex quadratic minimization," *Optimization Letters*, vol. 12, no. 3, pp. 499–518, 2018.

[26] L. Lovász, "Semidefinite programs and combinatorial optimization," in *Recent advances in algorithms and combinatorics*. Springer, 2003, pp. 137–194.

[27] L. N. Trefethen and D. Bau III, *Numerical linear algebra*. Siam, 1997, vol. 50.

[28] "The sage development team, s. reference v4.7," *Cryptography, www.sagemath.org/doc/reference/ sage/crypto/lattice.html*, 2009.